The Proof Analysis Problem for Constant-Depth Frege*

Noel Arteche† Susanna F. de Rezende‡ Erfan Khaniki§ Kilian Risse¶

Abstract

We give a parallel algorithm for the *Proof Analysis Problem* of Arteche, Atserias, de Rezende, and Khaniki (FOCS 2025). The problem asks, given a CNF formula φ on n variables and a proof π of a size-s Resolution lower bound on φ, encoded as $\neg\textsc{Ref}_s(\varphi)$, to extract a satisfying assignment for φ when one exists. The original work gave a polynomial-time extraction algorithm for Resolution refutations that enabled a formalization of the Atserias–Müller lower bound (FOCS 2019) in Extended Frege, but it appeared inherently sequential. We devise a new algorithm that performs assignment extraction in low-depth circuit classes.

Specifically, we give an assignment extraction algorithm computable by polynomial-size $\mathbf{TC}^0$ circuits and by quasipolynomial-size $\mathbf{AC}^0$ circuits, and we prove the correctness of the latter within quasipolynomial-size $\mathbf{AC}^0$-Frege. This is achieved thanks to a new formalization of the Atserias–Müller lower bound in the relativized theory $\mathsf{T}^4_2(\alpha)$ of bounded arithmetic, which corresponds to depth-4 Frege systems. One of the main technical ingredients in this formalization is a derandomization of the random restriction argument of Atserias and Müller using limited independence and tools from unconditional derandomization, combined with Jeřábek's approximate counting framework.

This yields quasipolynomial-size $\mathbf{AC}^0$-Frege proofs for a broad class of Resolution lower bounds, making progress on a question of Pudlák. A key feature of our approach is an improved parameter regime: unlike prior $\mathbf{AC}^0$-Frege formalizations of Resolution lower bounds by Li, Li, and Ren (STOC 2026), which only proved $\neg\textsc{Ref}_s(\varphi)$ formulas for $s = 2^{\Omega(n)}$, our approach applies to all s, including $s = n^{O(1)}$.

With these improved parameters we show that the Proof Analysis Problem for $\mathbf{AC}^0$-Frege is **NP**-hard under quasipolynomial-time reductions, which rules out certain Atserias–Müller-type approaches to the hardness of automating constant-depth Frege. Furthermore, automating $\mathbf{AC}^0$-Frege on Resolution lower bound formulas suffices to automate it on all formulas.

*An extended abstract of this paper will appear at the 67th IEEE Symposium on Foundations of Computer Science (FOCS 2026).

†Lund University and University of Copenhagen, noel.arteche@cs.lth.se

‡Lund University, susanna.rezende@cs.lth.se

§University of Oxford, erfan.khaniki@cs.ox.ac.uk

¶ETH Zürich, krisse@ethz.ch

Contents

1 Introduction **1**
- 1.1 Contributions . . . 2
- 1.2 Technical intermezzo: the exponential versus polynomial-size lower bound regimes . . . 5
- 1.3 Technical overview . . . 7
- 1.4 Related work . . . 13
- 1.5 Structure of the paper . . . 13

2 Preliminaries **13**
- 2.1 Proof complexity . . . 14
- 2.2 Bounded arithmetic . . . 16
- 2.3 The Ref formulas and the Proof Analysis Problem . . . 19

3 The new assignment extraction algorithm **23**
- 3.1 The Prover–Delayer game, revisited . . . 23
- 3.2 The new algorithm . . . 27

4 The restriction argument in $\mathrm{APC}_2(\alpha)$ **32**
- 4.1 Construction of designs . . . 33
- 4.2 Bad restrictions from seeds . . . 34
- 4.3 The block-width formulas . . . 35
- 4.4 Existence of good seeds . . . 36

5 The block-width lower bound in $\mathrm{T}^3_2(\alpha)$ **38**
- 5.1 Bounded formulas for Delayer-consistency and assignment extraction . . . 39
- 5.2 The block-width lower bound . . . 44

6 The Atserias–Müller lower bound in $\mathrm{T}_2(\alpha)$ **46**
- 6.1 Applying the restrictions . . . 47
- 6.2 Formalization in the polynomial-size regime: $\mathrm{T}^4_2(\alpha)$ . . . 49
- 6.3 Formalization in the exponential-size regime: $\mathrm{T}^1_2(\alpha)$ + dWPHP($\mathrm{PV}_1(\alpha)$) . . . 49

7 Propositional consequences **50**
- 7.1 Propositional fragments of the Atserias–Müller lower bound . . . 50
- 7.2 The Ref formulas are complete for automatability . . . 52

8 The Proof Analysis Problem for constant-depth Frege is NP-hard **53**
- 8.1 Haken's lower bound in the polynomial-size regime . . . 53
- 8.2 The hardness reduction . . . 55

Acknowledgements **56**

References **56**

1 Introduction

The computational complexity of automated theorem proving has been a central concern of theoretical computer science since the birth of the field and the seminal works on **NP**-completeness [Coo71; Coo75]. In the context of proof complexity, this question is captured by the notion of *automatability*, introduced by Bonet, Pitassi, and Raz [BPR00]: a proof system S for propositional logic is automatable if there exists an algorithm that finds a proof in time $\mathrm{poly}(s)$ whenever a proof of size s exists. Despite early hardness results based on cryptographic assumptions [KP98; BPR00; BDG+04], establishing the non-automatability of standard systems remained elusive for nearly two decades after the work of Bonet, Pitassi, and Raz.

In 2019, Atserias and Müller [AM20] proved that Resolution—the most extensively studied propositional proof system—cannot be automated unless $\mathbf{P} = \mathbf{NP}$, an optimal hardness result. At the heart of their argument lies a clever meta-mathematical reduction that forces Resolution to reason about itself. Their reduction maps a CNF formula φ to a new formula $\textsc{Ref}_s(\varphi)$, which asserts that φ admits a Resolution refutation of length s. When negated, this is the natural propositional encoding of a Resolution size lower bound. If φ is satisfiable, then $\textsc{Ref}_s(\varphi)$ is unsatisfiable, as it encodes a *trivial* lower bound. For such formulas, Pudlák [Pud03] had already shown how to efficiently construct a Resolution refutation from an explicit satisfying assignment for φ. In contrast, Atserias and Müller proved that if φ is unsatisfiable, then $\textsc{Ref}_s(\varphi)$ requires exponential size to be refuted in Resolution. This creates an exponential gap between the satisfiable and unsatisfiable cases, yielding the **NP**-hardness reduction.

Essential to this and similar arguments for other weak proof systems [Bel20; Gar24; GKMP20; dGN+21; IR22; Pap24] is whether propositional proof systems can prove their own non-trivial size lower bounds. Pudlák soon identified this question as "more fundamental than automatability" [Pud20], and called for a systematic study of when concrete systems can establish proof complexity lower bounds. Recently, Arteche, Atserias, de Rezende, and Khaniki [AAdRK25] approached these questions from a computational perspective by introducing the *Proof Analysis Problem*. For a fixed proof system S, the Proof Analysis Problem for S (denoted PAP_S) asks, given a CNF formula φ and an S-proof π of the Resolution lower bound statement $\neg\textsc{Ref}_s(\varphi)$, to decide whether φ is satisfiable—and hence whether π is proving a trivial lower bound. If the problem PAP_S is decidable in polynomial time, we say that the proof system S is *analyzable*. Due to the lower bound on Ref formulas, Resolution is readily seen to be analyzable: if the refutation of $\textsc{Ref}_s(\varphi)$ is short and correct, then it must be for a satisfiable φ. It is not hard to see that for any system S at least as strong as Resolution, if S is analyzable, then the system cannot be automated unless $\mathbf{P} = \mathbf{NP}$, thereby generalizing the Atserias–Müller reduction.

The Proof Analysis Problem for constant-depth Frege: easy or hard? As discussed above, if a proof system is analyzable, then automating it is **NP**-hard. Conversely, if the system is *not* analyzable, an argument in the style of Atserias and Müller does not suffice to prove the hardness of automating it. Previous work [AAdRK25] showed that the Proof Analysis Problem for Extended Frege ($\mathrm{PAP}_{\mathsf{EF}}$) is **NP**-hard, implying that new techniques are needed to attack the non-automatability of strong proof systems. However, the status of analyzability remained open for proof systems in between Resolution and Extended Frege.

Of central interest are constant-depth Frege systems, where only partial hardness results are known for automatability [BDG+04; ACG24; Pap24] and **NP**-hardness remains unknown over CNF formulas. The Atserias–Müller techniques appear insufficient to settle this, so proving that the analysis problem for these systems is hard would suggest that new ideas are needed. To prove that PAP_S is **NP**-hard it is enough for S to derive non-trivial Resolution lower bounds. For constant-depth Frege, the question of whether Resolution lower bounds are provable in polynomial size remains an open problem [Pud20, Problem 3]. A first step in this direction was recently obtained by Li, Li, and Ren [LLR26] who established some formalizations in quasipolynomial-size constant-depth Frege.

The assignment extraction problem. While the Proof Analysis Problem asks to *decide* whether the formula φ in the input is satisfiable, the natural *search problem* associated with it asks for a satisfying assignment to be extracted from the proof of $\neg\textsc{Ref}_s(\varphi)$. In [AAdRK25], it was shown that a polynomial-time extraction algorithm exists when the proofs being analyzed are themselves Resolution refutations. The design of this algorithm is intimately connected to the complexity of the Atserias–Müller lower bound. Ultimately, the extraction algorithm amounts to a new constructive proof of the lower bound on Ref formulas: given a small Resolution refutation of $\textsc{Ref}_s(\varphi)$, the algorithm always succeeds in finding a satisfying assignment for φ, implying that if φ is unsatisfiable, then no small Resolution refutation of $\textsc{Ref}_s(\varphi)$ exists.

Algorithms for the assignment extraction problem and constructive proofs of the lower bound for Ref formulas [AM20; Gar19] are thus two sides of the same coin. Arteche et al. [AAdRK25] further showed that this constructive proof can be formalized in Cook's theory PV_1 of bounded arithmetic, which captures polynomial-time reasoning. Applying Buss's witnessing theorem, one immediately recovers the extraction algorithm from the formalized PV_1 proof. Moreover, by Cook's propositional translation, it follows that Extended Frege proves the Atserias–Müller lower bound in its propositional form, yielding a series of interesting corollaries for proof complexity.

One of the main questions left open was then to determine whether the lower bound could be formalized in weaker theories and proof systems—and, consequently, whether more efficient extraction algorithms are possible. The known algorithm appeared inherently sequential in multiple aspects and difficult to parallelize or implement in sublinear space, suggesting that the extraction problem could even be complete for **FP**. Whether complete for **FP** or efficiently parallelizable, either outcome would be significant. On the one hand, if the problem turned out to be **FP**-complete, that would imply that no theory below PV_1 would prove the Atserias–Müller lower bound: if, for instance, the theory V^0 for $\mathbf{AC}^0$ reasoning proved such a statement, witnessing theorems would imply the false inclusion $\mathbf{P} \subseteq \mathbf{AC}^0$. On the other hand, the alternative outcome of a new, more efficient algorithm in, say, **NC**, **L** or **TC**, would suggest that a more efficient proof of the lower bound might be possible.

This paper. In this work, we show that the latter scenario turns out to be the right one: we devise a new algorithm capable of performing assignment extraction by polynomial-size $\mathbf{TC}^0$ circuits and quasipolynomial-size $\mathbf{AC}^0$ circuits. Inspired by this new algorithm, we obtain a new proof of the Atserias–Müller lower bound in quasipolynomial-size constant-depth Frege systems. This, in turn, gives constant-depth Frege proofs of a large class of Resolution lower bounds in all parameter regimes. Equipped with these, we can extend the **NP**-hardness reduction for $\mathrm{PAP}_{\mathsf{EF}}$ to $\mathbf{AC}^0$-Frege, under quasipolynomial-time reductions—concluding that the hardness of automating such systems requires new ideas, or at least stronger Ref formulas.

1.1 Contributions

We continue the systematic study of the Proof Analysis Problem initiated by Arteche et al. [AAdRK25] and address their two main open problems:

1. Is the assignment extraction problem for Resolution refutations computable in some subclass of **FP**? And, if so, can this be formalized in a weaker theory of bounded arithmetic than PV_1?
2. Are constant-depth Frege systems analyzable? That is, is PAP_S in **P** for $S = \mathbf{AC}^0$-Frege?

1.1.1 A new extraction algorithm for Resolution

Our first contribution is a new parallel algorithm for the assignment extraction problem for Resolution, which can be implemented in low-depth circuit classes. Recall that $\mathbf{TC}^0$ is the class of problems computable by polynomial-size constant-depth Boolean circuits with unbounded fan-in and threshold gates.

Theorem 1.1 (Assignment extraction in $\mathbf{TC}^0$, informal)**.** *The assignment extraction problem for Resolution is computable by uniform multi-output* $\mathbf{TC}^0$ *circuits of polynomial size.*

This algorithm inspects each clause of the Resolution refutation in parallel in search for a satisfying assignment, and we show that there always exists a clause in the refutation that *hides* the assignment. While extracting the assignment is highly non-trivial, the actual implementation of the algorithm is fully parallelizable. More precisely, we show that there exists a polynomial-size $\mathbf{TC}^0$ circuit E such that for every Resolution refutation π of $\textsc{Ref}_s(\varphi)$ for satisfiable φ, there exists at least one clause $C \in \pi$ such that $E(C)$ outputs a satisfying assignment. The final procedure then evaluates $E(C)$ in parallel for every $C \in \pi$.

The algorithm appears somewhat surprising for two reasons. First, the original polynomial-time algorithm consisted of two subroutines, both of which appear very hard to parallelize. Second, and more surprisingly, the natural proof of correctness of this new algorithm invokes the correctness of the original algorithm. We discuss this at length in the technical overview below (see Section 1.3.1).

Furthermore, we show that a quasipolynomial increase in size suffices to remove the threshold gates.

Theorem 1.2 (Assignment extraction in quasipolynomial $\mathbf{AC}^0$, informal)**.** *The assignment extraction problem for Resolution is computable by uniform multi-output* $\mathbf{AC}^0$ *circuits of quasipolynomial size.*

The formal statements can be found in Section 3.2.

1.1.2 Formalization of the Atserias–Müller lower bound in $\mathsf{T}^4_2(\alpha)$ and constant-depth Frege

As discussed earlier, proofs of correctness of the extraction algorithms and proofs of the Atserias–Müller lower bound are two sides of the same coin. Inspired by the new algorithms above, the next question is whether they can be proven correct in systems weaker than Extended Frege. For the sake of completeness, let us state the lower bound and the previously known formalization in PV_1, which proves a slightly weaker version of the statement.

Theorem 1.3 (The Atserias–Müller lower bound in PV_1 [AM20; AAdRK25])**.** *For every n-variate unsatisfiable CNF formula φ and every $s \in \mathbb{N}$, it holds that any Resolution refutation of the formula $\textsc{Ref}_s(\varphi)$ is of size $2^{\Omega(s/n)}$. Furthermore, the theory PV_1 proves that any such refutation is of size $2^{\Omega(s/n^2)}$.*

The target theory for a new formalization of the proof using the $\mathbf{TC}^0$ algorithm (Theorem 1.1) is the two-sorted theory VTC^0 of Cook and Nguyen [CN10]. However, the natural proof of correctness invokes the correctness of the original algorithm, only available in PV_1. We find a way around this via an alternative proof, which reasons about objects of quasipolynomial size. Due to these size bounds, the proof cannot go through in VTC^0, but can be carried out in the two-sorted theory V^0_2 corresponding to quasipolynomial $\mathbf{AC}^0$ reasoning. Effectively, this formalizes the correctness of the $\mathbf{AC}^0$ algorithm (Theorem 1.2).

While V^0_2 would be enough for quasipolynomial $\mathbf{AC}^0$ reasoning, we choose to work instead over specific levels $\mathsf{T}^i_2(\alpha)$ of the relativized hierarchy of Buss [Bus86]. This approach still translates into proofs in constant-depth Frege systems via the Paris–Wilkie translation, and it has the added benefit that we can easily control the specific depth we are translating into. Since we work in very low theories of the hierarchy, this appears more informative and worth making explicit.

In this setting, the Resolution refutations are encoded by an external relation symbol, denoted α in this introduction. Since the new algorithm analyzes each clause of the refutation in parallel to extract an assignment from one of them, we can state the result as follows. (See Theorem 6.4 for more details about the encoding of the statement.) Note that this new formalization recovers the better $2^{\Omega(s/n)}$ bound of [AM20].

Theorem 1.4 (The Atserias–Müller lower bound in $\mathsf{T}^4_2(\alpha)$, informal)**.** *The theory $\mathsf{T}^4_2(\alpha)$ proves that for every n-variate CNF formula φ, every $s \geq n^2$, and every Resolution refutation π of $\textsc{Ref}_s(\varphi)$, if none of the clauses of π encodes a satisfying assignment for φ, then $|\pi| > 2^{\Omega(s/n)}$.*

Formalizing arguments in bounded arithmetic is particularly exciting when the more constructive reasoning leads to alternative proofs of known statements. This has happened, for example, with Razborov's now-famous "encoding argument" for Håstad's switching lemma [Raz95; Bea94]. Remarkably, Razborov's new proof of the switching lemma is recognized for enabling proofs of at least two major conjectures in combinatorics [ALWZ21; PP24]. Another example where formalization in bounded arithmetic led to alternative proofs is a recent hybrid argument proving the Schwartz–Zippel lemma in S^1_2 [AT25].

Our formalization also relies on an alternative proof of the lower bound. At the core of our proof is a new derandomization of the random restriction argument used in the lower bound. While the earlier work [AAdRK25] derandomized this by a greedy algorithm, we instead rely on k-wise independent distributions within the proof system itself. The technical overview in Section 1.3.2 delves further into this.

Overall, this new formalization shows that the reasoning power needed to establish one of the most celebrated recent lower bound results of proof complexity can be made extremely efficient and constructive, a core concern for ongoing work on the reverse mathematics of computational complexity (see Section 1.4 for a discussion about related work in this direction).

Upper bounds in the propositional setting. Now, via the Paris–Wilkie translation, we can get the following propositional upper bounds. For the sake of precision, all of our statements about propositional systems use the Tait-style cedent calculus PK [Kra94] (see Section 2.1.2 for the definitions). In this setting, the depth-0 subsystem PK_0 is precisely Resolution, and we consider "half-depths", like $\mathsf{PK}_{d+1/2}$, which refers to the subsystem of PK_{d+1} where all formulas have depth $d+1$ and the connectives have unbounded fan-in but the bottom-most connectives have logarithmic fan-in in the size of the proof. In this notation, Res(log) corresponds to $\mathsf{PK}_{0+1/2}$. These "half-depths" arise naturally when translating from bounded arithmetic. As a general rule, $\mathrm{T}^{d+2}_2(\alpha)$ produces quasipolynomial-size proofs in $\mathsf{PK}_{d+1/2}$—although the final depth also depends on the complexity of the statement being proven, which does affect our translations (see Theorem 2.5 for the details and Theorem 7.2 for the complete formal statement).

Corollary 1.5 (The Atserias–Müller lower bound in constant-depth Frege, informal)**.** *For every $d \in \mathbb{N}$ and every CNF formula φ, if $\mathsf{PK}_{(d+3)+1/2}$ refutes φ in size ℓ, then $\mathsf{PK}_{(d+3)+1/2}$ refutes $\textsc{Ref}(\textsc{Ref}(\varphi))$ in size $\mathrm{quasipoly}(\ell)$.*

1.1.3 The Proof Analysis Problem for bounded-depth Frege is NP-hard

We recover the **NP**-hardness reduction used to show that the Proof Analysis Problem for Extended Frege is **NP**-hard, and extend it all the way to constant-depth Frege, albeit under quasipolynomial-time reductions. The formal statement is in Theorem 8.5. This could be done using Corollary 1.5, but we get an even better bound on depth via an alternative route based on [LLR26]. Section 8 outlines the different approaches.

Theorem 1.6 (**NP**-hardness of PAP, informal)**.** *For every propositional proof system S that p-simulates depth-2 Frege, it holds that the Proof Analysis Problem for S is* **NP***-hard under quasipolynomial-time reductions.*

This immediately implies that the basic Atserias–Müller approach of using the Ref formula for Resolution for hardness of automatability does not work for constant-depth systems. This is because any proof along those lines using the Ref formulas works by showing that the system in question is analyzable, and then argues that if a system is both analyzable and automatable, then **P** = **NP**. Of course, this result does not rule out the possibility of using $\textsc{Ref}^Q$ formulas where Q is a system stronger than Resolution. To make the existing reduction in Theorem 1.6 polynomial-time computable, it would suffice to solve Pudlák's question [Pud20, Problem 3] of formalizing Resolution lower bounds in *polynomial-size* $\mathbf{AC}^0$-Frege.

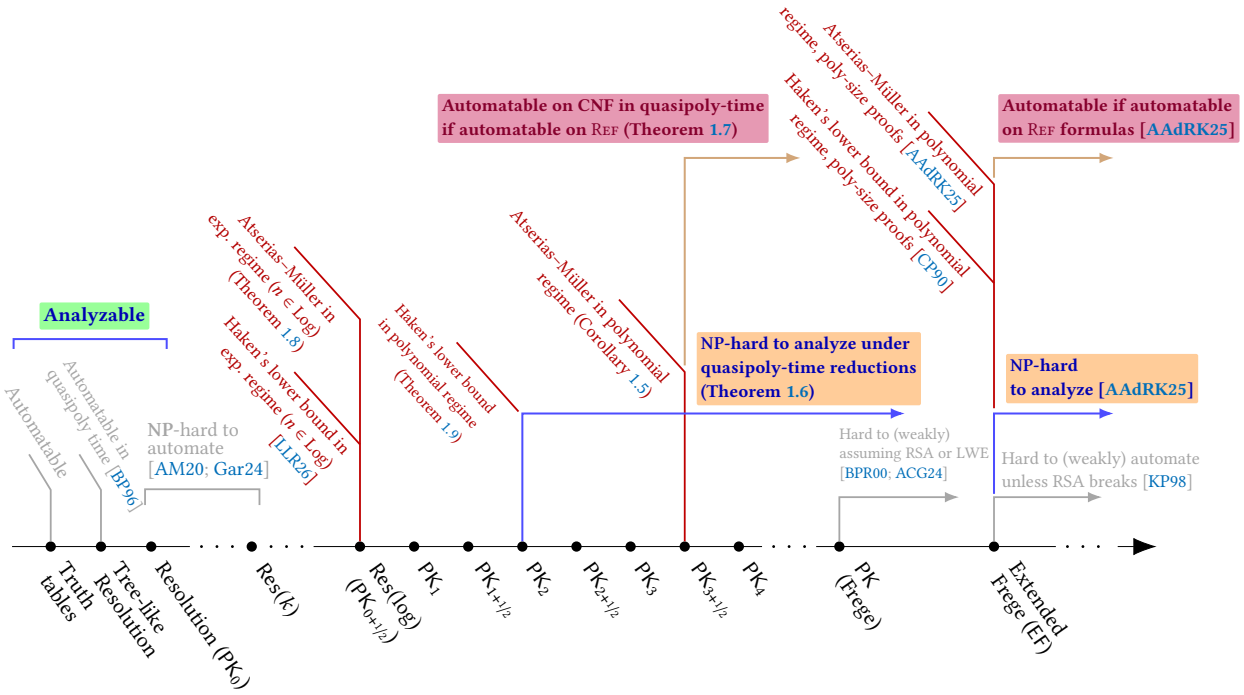


Figure 1: The landscape of automatability, analyzability and formalized Resolution lower bounds. In red, upper bounds for formalized Resolution lower bounds; in gray, upper and lower bounds on automatability; in green, analyzable proof systems; in orange, **NP**-hardness results on analyzability; in purple, results bootstrapping automatability on Ref formulas to automatability on all formulas. All results in red for constant-depth systems correspond to quasipolynomial-size proofs.

1.1.4 The Ref formulas are complete for automatability

Our last result, also a byproduct of the propositional translations, is that efficient proof search over Resolution lower bounds suffices to automate proof systems over all formulas. Arteche et al. [AAdRK25] used their formalizations in PV_1 to show this result for any system that p-simulates Extended Frege. In this work, we extend this to constant-depth systems with quasipolynomial time bounds. The formal statement is in Theorem 7.5.

Theorem 1.7 (Automatability in terms of Ref formulas, informal)**.** *For every $d \geq 3$, the proof system $\mathsf{PK}_{d+1/2}$ is automatable on Ref formulas in quasipolynomial time if, and only if, it is automatable on all CNF formulas in quasipolynomial time.*

The idea in all of these results is to leverage the propositional translation as follows: if the system S is strong enough to prove the Atserias–Müller statement, then a size-ℓ proof of $\neg\varphi$ implies a quasipoly(ℓ)-size proof of $\neg\textsc{Ref}(\textsc{Ref}(\varphi))$. We can now use the automating algorithm on this Ref formula to find a proof, and finally, use the construction of Pudlák [Pud03] to recover a proof of $\neg\varphi$ from the proof of $\neg\textsc{Ref}(\textsc{Ref}(\varphi))$. The latter step was formalized in Resolution by Arteche et al. [AAdRK25, Theorem 7.1], and only increases the size by an additive polynomial.

1.2 Technical intermezzo: the exponential versus polynomial-size lower bound regimes

The theory $\mathsf{T}^4_2(\alpha)$ and the proof system $\mathsf{PK}_{3+1/2}$ are slightly stronger than the ones used in previous formalizations of Resolution lower bounds. For example, Li, Li, and Ren [LLR26] showed, through the language

of **TFNP**, that Haken's [Hak85] exponential Resolution lower bound for the pigeonhole principle formulas (PHP_n) is provable in $\mathsf{T}^1_2(\alpha) + \mathrm{dWPHP}(\mathrm{PV}_1(\alpha)) \subseteq \mathsf{T}^2_2(\alpha)$, which corresponds to quasipolynomial-size Res(log) refutations (i.e., $\mathrm{PK}_{0+1/2}$) of the formulas $\textsc{Ref}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$ for a suitable constant ε.

Before we can state the rest of our results, a crucial technical discussion is in place. In the metalanguage, the $2^{\Omega(n)}$ lower bound proven by Haken certainly implies a weaker lower bound of, say, n^k for any fixed k—but this is not obvious at the propositional level. That is, the formula $\textsc{Ref}_{n^k}(\mathrm{PHP}_n)$ logically implies $\textsc{Ref}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$ and the system can perform this derivation, but this incurs an exponential blow-up in size if we measure the proof in terms of the starting formula $\textsc{Ref}_{n^k}(\mathrm{PHP}_n)$.

This mismatch between the size of the inner formula and the outer lower bound parameter can give unreasonable power to the proof system. In the case of PHP_n, the formula $\textsc{Ref}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$ has size $2^{O(n)}$, while the inner PHP_n formula has $O(n^2)$ variables and $O(n^3)$ clauses. In this regime, for instance, the proof system could argue that some formula over n variables is unsatisfiable simply by a brute-force argument—and this would only be a polynomial-size proof in terms of the Ref formula. Similarly, suppose the lower bound argument relies on analyzing some probabilistic event over the uniform distribution for n bits. When starting from the formula $\textsc{Ref}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$, the proof system could exactly count the probability of any such event in polynomial-size—while this kind of exact counting would be infeasible if we are proving $\neg\textsc{Ref}_{n^k}(\mathrm{PHP}_n)$ in polynomial size.

This situation causes two problems. Conceptually, one might cast doubt as to whether a formalization of the lower bound $\neg\textsc{Ref}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$ as in [LLR26] is satisfactory[1] in capturing the complexity of the argument if this cannot be easily scaled down to prove $\neg\textsc{Ref}_{n^k}(\mathrm{PHP}_n)$. Pragmatically, for many applications, like an **NP**-hardness reduction for automatability, one wants to be able to construct the $\textsc{Ref}_s(\varphi)$ formula efficiently starting from the inner formula φ. In this setting, we are interested in fixing s to be a small polynomial in the size of φ.

This mismatch between parameters also arises when working within one-sorted theories of bounded arithmetic to then obtain propositional upper bounds via translations. We distinguish two regimes.

(a) In the *polynomial-size regime*, we state the lower bound by saying: "for every n and every refutation of PHP_n of length t, if t is small, then the refutation is incorrect"; here, "small" can be a fixed polynomial, like n^k, or an exponential $2^{\varepsilon n}$, stated as $\log t > \varepsilon n$, since the exponential function 2^n is not provably total for every n.

(b) In the *exponential-size regime*, we state the lower bound by saying: "for every $n \in \mathrm{Log}$ and every refutation of PHP_n of length t, if $t < 2^{\varepsilon n}$, then the refutation is incorrect".

The term *polynomial-size regime* is perhaps a misnomer, since we can also get exponential lower bounds—it is just that the polynomial-size lower bounds are the more interesting ones from the point of view of formalization. The key difference is that in (b) we assume $n \in \mathrm{Log}$. The expression $n \in \mathrm{Log}$ is a common shorthand in bounded arithmetic for the formula $\exists x\, |x| = n$. This is often put in words by saying that "n is a *length*", since it is the length of some string x. We think of numbers in Log as small, meaning that the number 2^n exists in the theory and can be stated explicitly.

The advantage of assuming $n \in \mathrm{Log}$ is clear: 2^n exists and we can perform a great deal of exact counting. The disadvantage appears when dealing with propositional translations: if $n \in \mathrm{Log}$, then the propositional translation trivializes for $\neg\textsc{Ref}_{n^k}(\mathrm{PHP}_n)$, as the formula is artificially padded to have size 2^n.

In the exponential-size regime, our proof of the Atserias–Müller statement can be carried out in the theory $\mathsf{T}^1_2(\alpha) + \mathrm{dWPHP}(\mathrm{PV}_1(\alpha))$ used in [LLR26]. Theorem 6.5 contains the formal statement.

[1] Arguably, the exponential-size regime makes perfect sense from the point of view of Li, Li, and Ren [LLR26], since in the context of **TFNP**, analyzing purported Resolution refutations of polynomial size would be a trivial computational task, and formalizing the lower bounds is not their main priority.

Theorem 1.8 (The Atserias–Müller lower bound in $\mathsf{T}^1_2(\alpha)$ + dWPHP($\mathsf{PV}_1(\alpha)$), informal)**.** *The theory* $\mathsf{T}^1_2(\alpha)$ + dWPHP($\mathsf{PV}_1(\alpha)$) *proves that for every* $n \in \mathrm{Log}$ *and every* $s \in \mathrm{Log}$*, if* φ *is an* n*-variate CNF formula,* $s \geq n^2$*, and* π *is a Resolution refutation of* $\textsc{Ref}_s(\varphi)$ *such that none of the clauses of* π *encodes a satisfying assignment for* φ*, then* $|\pi| > 2^{\Omega(s/n)}$.

The known Res(log) upper bounds for $\textsc{Ref}(\mathrm{PHP}_n)$ coming from the formalizations of Li, Li, and Ren [LLR26] are in the exponential-size regime and cannot be directly scaled down within Res(log) to obtain fixed-polynomial lower bounds for PHP_n. Interestingly, for the particular case of these formulas, we devise a way to adapt the size parameters at the cost of increasing the depth of the proof system by a constant (recall that Res(log) is $\mathsf{PK}_{0+1/2}$).

Theorem 1.9 (Haken's lower bound in the polynomial-size regime, informal)**.** *For every* $k \in \mathbb{N}$*, there are quasipolynomial-size* PK_2*-proofs of the formulas* $\neg\textsc{Ref}_{n^k}(\mathrm{PHP}_n)$.

The idea here is to use the Res(log) refutations of $\textsc{Ref}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$ and apply a restriction sending PHP_n to an exponentially smaller subformula. This incurs an increase in depth, so we no longer stay in Res(log). This approach, however, works only for PHP and similar formulas where we can restrict to an exponentially smaller sub-instance of the same formula. In particular, this approach does not seem to work for the Ref formulas. The formal statement is given in Theorem 8.3.

1.3 Technical overview

In what follows, we present a summary of the main technical challenges and the key ideas behind the proofs.

1.3.1 The new extraction algorithm

We begin by describing the original extraction algorithm of [AAdRK25] and show how to overcome the two main obstacles to parallelizing it. The original algorithm has two steps, mirroring the two components of the lower bound for Ref formulas: a restriction argument followed by a width lower bound. This follows the standard template for Resolution size lower bounds.

In the formula $\textsc{Ref}_s(\varphi)$, variables are partitioned into s *blocks*, each encoding one of the s clauses of the purported refutation. The *block-width* of a clause C over the variables of $\textsc{Ref}_s(\varphi)$, denoted $\mathrm{bw}(C)$, is the number of distinct blocks mentioned in C, and for a refutation π, we define $\mathrm{bw}(\pi)$ as the maximum block-width over all clauses in π.

Given a Resolution refutation π of $\textsc{Ref}_s(\varphi)$, the algorithm proceeds as follows:

1. *Deterministic restriction.* The algorithm constructs a restriction ρ such that $\textsc{Ref}_s(\varphi)_{\restriction\rho} \equiv \textsc{Ref}_{s/2}(\varphi)$, and $\pi_{\restriction\rho}$ remains a valid refutation. Crucially, $\mathrm{bw}(\pi_{\restriction\rho}) \leq O\big(\sqrt{s \log |\pi|}\big)$.
2. *Prover–Delayer game.* We think of the width lower bound of [AM20] as a Prover–Delayer game [Pud00; AD08]. The Prover issues queries about specific variables or chooses to forget the value of a previously queried variable, and the Delayer responds with truth-assignments to these queries. The Prover wins when the partial assignment in the Prover's memory falsifies one of the clauses of the formula. The formula requires large width to be refuted if and only if the Delayer has a strategy that forces the Prover to remember many variables before winning.

   Now, starting from the root $\bot$, we traverse $\pi_{\restriction\rho}$ according to the Delayer's strategy used in the Prover–Delayer game that establishes the block-width lower bound of [AM20]. As shown in [AAdRK25], this traversal must eventually reach either:

   (a) a clause C with block-width $\Omega(s/n)$; or

(b) a satisfying assignment for φ.

If π is small enough—less than $2^{\varepsilon n}$ in size—, then $\mathrm{bw}(\pi_{\restriction\rho}) \leq O\left(\sqrt{s \log |\pi|}\right)$ implies we must be in case (b) and the traversal thus yields a satisfying assignment. If π is too large, then we can perform a brute-force search looking for a satisfying assignment while still running in polynomial time.

Both of these steps resist parallelization.

The challenge of parallelizing the deterministic restriction. While the original proof of Atserias and Müller used a random restriction, Arteche et al. [AAdRK25] derandomize it via a greedy procedure. At a high level, the algorithm repeatedly identifies the most frequent block in the current refutation, selects a literal that appears in many clauses involving that block, and assigns it so as to eliminate those clauses. The refutation is simplified and the process repeats. After all blocks are processed, the resulting refutation has block-width $O\left(\sqrt{s \log |\pi|}\right)$.

Although polynomial-time, this procedure is inherently sequential: the next literal can only be chosen after simplifying the formula under the current partial assignment. This dependence appears to preclude parallelization.

The challenge of parallelizing the Prover–Delayer game. The second step is also inherently sequential. The algorithm traverses the refutation while the Delayer maintains a *reservation*, a partial assignment together with some auxiliary structure encoding which blocks have been reserved and how they are arranged across n conceptual layers of the purported refutation. This layered arrangement is supposed to locally represent an embedding into the canonical exponential-size tree-like refutation of φ. In this structure, as long as the formula is unsatisfiable, the Prover can never find a mistake. The arrangement looks similar to Figure 2, except it has only n layers.

This reservation is both large (requiring linear space) and history-dependent. In particular, when φ is satisfiable, the Prover only wins once the reservation specifies a block in the final layer containing information about a satisfying assignment. Crucially, this assignment is not determined by the current clause alone, but by the entire traversal history. It is therefore unclear how to reconstruct it locally or in parallel.

A parallel quasipolynomial-time algorithm. To avoid the sequential nature of the greedy restriction procedure, we first revert to the original random restriction of Atserias and Müller [AM20]. This argument established, by the probabilistic method, that there *exists* a restriction that not only reduces block-width but actually *kills* all clauses of block-width at least $c \cdot \log |\pi|$ for some c—much better than the $c \cdot \sqrt{s \log |\pi|}$ achieved by the previously mentioned greedy construction. While we do not know what the actual restriction looks like, this stronger killing property allows us to discard all wide clauses outright. Crucially, we never actually construct the restriction—we simply look at a clause, check its block-width, and only proceed further if the clause has block-width $c \cdot \log |\pi|$.

Now, looking at some clause C of block-width $c \cdot \log |\pi|$, we consider under what circumstances the original algorithm would have found a satisfying assignment at C. This would be the case if the algorithm reached C while having in memory a reservation R that specified a width-n clause in the last layer of the purported refutation that is not a weakening of any axiom. The difficulty is that we do not have access to such a reservation R, so we will try to reconstruct it. A naive brute-force search over all possible reservations will not do: there are more than n variables per block, so even for low block-width clauses, this would take exponential time.

We instead find a valid reservation *only for the pointers* of the block. A brute-force algorithm can try all pointer reservations in time $s^{O(\mathrm{bw}(C))}$, which is quasi-polynomial if C has block-width $O(\log |\pi|)$. These

pointers determine a forest-like structure on the blocks. The next step is to determine whether this pointer reservation is consistent with C—for example, whether the variables that C enforces to be present or absent in a block are consistent with the left/right pointers leading to the block; and whether for any two blocks that are in the same connected component given by the pointers, the variables in these blocks are consistent with a valid reservation. This second stage of a consistency check can be easily implemented in parallel by polynomial-size constant-depth circuits.

After this procedure, we may construct a *partial reservation* R_0 for C: it sets all pointers in a way consistent with the clause, but we do not necessarily know what the *contents* of the blocks involved are. We will say that R_0 *encodes a satisfying assignment* if R_0 contains information about a block appearing in the last layer and the partial information available is already a (partial) satisfying assignment for φ. Furthermore, for the purpose of this discussion, we say a clause C is an *assignment-encoding clause* if there is at least one valid reservation R_0 for C and all valid reservations for C encode a satisfying assignment.

By the correctness of the original algorithm, and the fact that the Delayer cannot keep playing forever, we can conclude that there must exist a clause C in π that is assignment-encoding. It thus suffices to analyze each clause C separately and find a single valid reservation for C, if one exists: this reservation either encodes a satisfying assignment, and we are done, or it does not, and it therefore witnesses that C is not an assignment-encoding clause.

The parallel polynomial-time algorithm. The quasipolynomial algorithm described above still spends $s^{\Omega(\mathrm{bw}(C))}$ time guessing the pointer structure of a reservation. To obtain a polynomial-time algorithm, we modify the virtual grid on which the Delayer arranges the blocks of the $\textsc{Ref}(\varphi)$ formula to include $\Delta \log |\pi|$ "dummy" layers between the original n layers and the axioms of φ. Figure 2 shows this new arrangement of the blocks with the additional layers. On the first n layers the Delayer behaves as before. On each of the additional layers, however, a block is simply copied to the next layer: both of its pointers lead to the same child, carrying exactly the same width-n clause. Only at layer ℓ does the Delayer finally declare that this clause is a weakening of an axiom of φ.

The purpose of these extra layers is to isolate the final layer from the complicated branching that may occur in the first n layers. Suppose that the current clause C has block-width at most $w = c \cdot \log |\pi|$, and consider a reservation consistent with C. Since only blocks mentioned in C have their outgoing pointers fixed, a connected component determined by the reservation can contain at most w such blocks before reaching an unmentioned endpoint. It therefore cannot cross the $\Delta \log |\pi|$ dummy layers and reach the first n layers, assuming $\Delta > c$. Moreover, there is no branching inside the dummy region, since every block there has both pointers leading to the same child. This is the key simplification.

Now, when analyzing a low-block-width clause C, we do not need to reconstruct the pointers of a reservation at all. Let $b_1, \ldots, b_w$ be the blocks mentioned in C. The algorithm considers every subset B of these. Intuitively, B is a guess for the set of blocks of C lying on one of the chains that reaches the final layer. From the literals of C, we can read the partial information that falsifying C forces about the contents of each block $b \in B$. We collect all this information into a clause A_B. If A_B is non-tautological, it has a unique partial assignment α_B that falsifies all of its literals. We simply check whether α_B already satisfies every clause of φ. If it does, then every extension of α_B to the remaining variables is a satisfying assignment, and the algorithm outputs one. Crucially, there are only $2^w = 2^{O(\log |\pi|)} = |\pi|^{O(1)}$ subsets to inspect for each clause, so this procedure is polynomial once we restrict attention to clauses of block-width at most $w = \Theta(\log |\pi|)$.

The whole procedure can be implemented by polynomial-size $\mathbf{TC}^0$ circuits. Threshold gates are used to compute the block-width of a clause and thereby discard clauses wider than w; all subsets of the remaining $O(\log |\pi|)$ blocks can then be inspected in parallel. Removing the threshold gates yields quasipolynomial-size $\mathbf{AC}^0$ circuits.

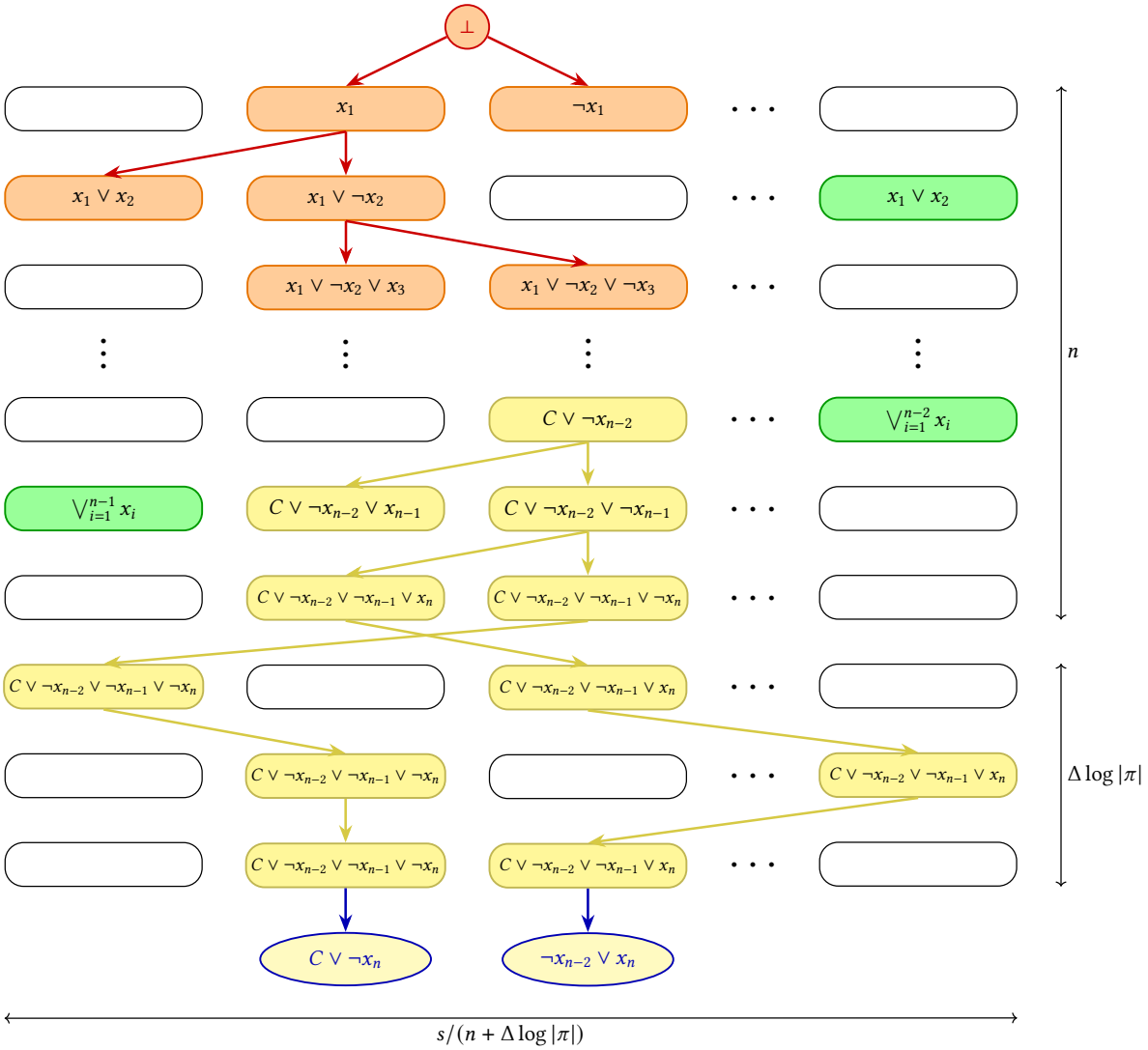


Figure 2: Snapshot of the execution of the new extraction algorithm (Algorithm 3.2), when playing over the extended grid. The Prover–Delayer game is played over the formula $\textsc{Ref}_s(\varphi)$ for s blocks, arranged in $\ell := n + \Delta \log |\pi|$ layers with s/ℓ blocks per layer. In orange and in yellow are two partial trees in which the Delayer answers consistently by locally following the canonical exponential-size tree-like refutation. The orange tree starts from the root, while the yellow one starts from some clause C of width $n - 3$ that emerged from previous queries. In green, blocks that were queried "out of context": at layer i, the Delayer answers with the clause $x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_i$ where all i literals are positive. At layer n there are width-n clauses that will be trivially weakened in the following layers until reaching layer $\ell = n + \Delta \log |\pi|$, where they must point to an axiom (in blue). For $\Delta = 0$, we have $\ell = n$ and we recover the grid used in the original algorithm.

1.3.2 Derandomizing the random restriction argument: the new proof of [AM20]

With the new algorithm in hand, we turn to proving its correctness—and hence the Atserias–Müller lower bound [AM20]—within a suitable theory of bounded arithmetic. We introduce techniques that may be of independent interest for future formalizations of complexity-theoretic arguments in bounded arithmetic.

A new derandomization of the restriction argument. Our goal is to show that there is a restriction ρ satisfying all clauses of block-width at least $w = \Theta(\log |\pi|)$. The original argument proceeds via the probabilistic method: one defines a suitable space of restrictions and shows that when sampling uniformly at random over this space, for any fixed clause C of block-width at least w, the probability that the clause survives is

$$\Pr_{\rho \in \mathcal{U}} [C_{\restriction \rho} \neq 1] \leq \left(\frac{3}{4}\right)^w. \tag{1.1}$$

A union bound over all clauses in the refutation π yields that the probability that some clause survives is at most $|\pi| \cdot (3/4)^w$, which is strictly less than 1 for our choice w—so a "good" ρ exists.

In bounded arithmetic, we work in the polynomial-size regime, meaning that the parameters n and s from the formula $\textsc{Ref}_s(\varphi)$ are not assumed to be in Log, and hence exact probability calculations are out of reach. Furthermore, in order to prove the statement in bounded arithmetic, we must first have a feasibly constructive procedure to obtain a good restriction—otherwise this would violate a corresponding witnessing theorem. This means that a naive use of the probabilistic method alone will not suffice.

Our solution is to use tools from unconditional derandomization. The key observation is that to establish the bound in (1.1), we only require *w-wise independence*, rather than full independence. We can therefore replace the uniform distribution $\mathcal{U}$ by a w-wise independent distribution $\mathcal{D}$ with much smaller support, following the classic derandomization paradigm initiated by Alon, Babai, and Itai [ABI86] and Luby [Lub86].

The only caveat is quantitative: any w-wise independent distribution over n-bit strings has support size at least $n^{\Omega(w)}$ [CGH+85; AS16, Proposition 16.2.3], so taking $w = \Theta(\log |\pi|)$ yields a quasipolynomial-size sample space. Accordingly, we formalize the quasipolynomial-size $\mathbf{AC}^0$ algorithm (Theorem 1.2), rather than the polynomial-size $\mathbf{TC}^0$ circuits from Theorem 1.1.

The observation that limited independence suffices is at the core of the proof, but we are still left with the task of choosing an explicit construction of w-wise independent distributions computable by constant-depth circuits and showing that constant-depth Frege systems can reason about them. We use the algebraic construction of combinatorial designs due to Nisan and Wigderson [NW94], based on low-degree polynomials over finite fields. Jeřábek [Jeř05] formalized a different (greedy) construction sufficient for the derandomization of $\mathbf{BPP}$ (i.e., the second construction of Nisan and Wigderson), but the parameters of that construction are inadequate in our context. As such we formalize the algebraic construction in PV_1.

We then move to the relativized setting, where an external predicate α encodes the Resolution refutation, and we work in the theory $\mathsf{APC}_2(\alpha)$ of Jeřábek, which extends $\mathsf{T}^1_2(\alpha)$ by the dual weak pigeonhole principle for $\mathbf{FP}^{\mathbf{NP}}$ functions ($\mathrm{dWPHP}(\mathsf{PV}_2(\alpha))$). In this theory we have access to Jeřábek's approximate counting machinery in the relativized setting. We use this to bound the size of the "bad event", essentially mimicking the bound in (1.1), and then simulate the union bound needed to complete the proof.

We find this derandomization particularly interesting, as it faithfully captures a probabilistic method argument within a very weak theory. More broadly, it suggests that techniques from unconditional derandomization could be systematically imported into bounded arithmetic, and may prove useful for formalizing other complexity-theoretic results.

The block-width lower bound argument. For the block-width lower bound, the workaround is simpler but a bit more subtle. Instead of appealing to the correctness of the original algorithm, we rephrase the inductive proof so that it mirrors the new algorithm. Indeed, the original proof in PV_1 worked by induction

on the length of the path constructed by the Delayer to traverse the refutation. In the relativized context, such a path would be too large to be encoded by numbers, so we cannot perform this kind of induction. Instead, our induction is over the indices of the Resolution refutation.

Let π be the Resolution refutation given in reverse order, for convenience. That is, we assume that $C_1 = \bot$ and every derived clause C_i is derived from clauses C_j and C_k for $j, k > i$. We show that for every clause $C_i \in \pi$, at least one of the following holds:

(a) there exists $j \leq i$ such that the algorithm extracts a satisfying assignment from C_j;

(b) there exists $j \leq i$ such that C_j has large block-width; or

(c) there exists $j > i$ such that C_j is *Delayer-consistent*.

The notion of *Delayer-consistency* arises naturally when analyzing the new algorithm. Intuitively, we say that a clause C is Delayer-consistent if there exists a partial reservation R consistent with C such that the Delayer could have safely continued playing starting from C with reservation R (see Section 5 for the formal definition).

The key difference with the original argument is in how the induction is structured. Instead of explicitly constructing the Delayer's path, we argue locally as follows. At stage i, for contradiction, suppose (a)-(c) all fail. Then, by induction hypothesis, C_i must be Delayer-consistent. By analyzing the local structure of the refutation and the definition of consistency, we show that one of the two clauses used to derive C_i is Delayer-consistent, contradicting our assumption. Thus, rather than building a single path, the argument moves from one Delayer-consistent clause to another, without maintaining the full traversal history. Since the refutation is finite, we cannot indefinitely move to further Delayer-consistent clauses, so as long as the refutation has low block-width, the algorithm terminates by finding a satisfying assignment.

1.3.3 The NP-hardness of the Proof Analysis Problem

The **NP**-hardness result in Theorem 1.6 follows the ideas of Arteche et al. [AAdRK25], in a slightly streamlined form. The key message of the earlier work is that if a propositional proof system S can establish even a single non-trivial Resolution lower bound, then the associated analysis problem for S is **NP**-hard. We recall this argument in a simplified form.

Suppose that S admits uniform proofs of the lower bounds encoded as $\neg\textsc{Ref}_{n^k}(\psi_n)$, where $\{\psi_n\}_{n\in\mathbb{N}}$ is a fixed family of unsatisfiable CNF formulas over n variables with $\text{poly}(n)$ clauses. For any other formula φ_n over n variables one can now efficiently construct a proof π_n of the form

$$\pi_n : S \vdash \neg\textsc{Ref}_{n^k}(\varphi_n \vee \psi_n), \tag{1.2}$$

where the disjunction is encoded in CNF using an auxiliary variable, and φ_n and ψ_n are over disjoint variable sets.

The derivation reduces the lower bound for $\varphi_n \vee \psi_n$ to that for ψ_n. Fix an arbitrary assignment α to the variables of φ_n. If α satisfies φ_n, then $\varphi_n \vee \psi_n$ is satisfied, and the corresponding lower bound is trivial. Otherwise, $\varphi_n(\alpha) = \bot$, and under this restriction the formula $\varphi_n \vee \psi_n$ simplifies to ψ_n. Provided that S proves that Resolution is closed under restrictions, it can invoke the lower bound for ψ_n to derive the desired conclusion. Crucial to this reduction is that π_n proves a fixed n^k lower bound and not an exponential one, or else we would not be able to construct π_n efficiently.

It follows that an algorithm for the analysis problem of S could be used to decide SAT: given φ_n on n variables, construct π_n as described above and run the analysis procedure on π_n. In [AAdRK25], the reduction for Extended Frege used the fact that Extended Frege can prove Resolution size lower bounds for PHP_n [CP90]. To instantiate this argument for constant-depth Frege, we may obtain the proof of a Resolution lower bound from different sources. One such source would be Corollary 1.5: fixing a formula γ

such that $\mathsf{PK}_{3+1/2}$ proves γ_n unsatisfiable, Corollary 1.5 implies that $\mathsf{PK}_{3+1/2}$ proves $\neg\textsc{Ref}_{n^k}(\textsc{Ref}_{n^3}(\gamma_n))$. Thus, in the above reduction we may take $\psi_n = \textsc{Ref}_{n^3}(\gamma_n)$. Theorem 1.9 also gives the proof of a lower bound, even at lower depth, but an even better approach, detailed in Section 8, is to rescale the formalizations of [LLR26] and then apply a restriction, to get the reduction to work as low as PK_2. It then remains to verify that constant-depth Frege proves that Resolution is closed under restrictions. This is routine, though somewhat tedious, and completes the proof.

1.4 Related work

To the best of our knowledge, the only attempts at formalizing Resolution lower bounds are the article of Cook and Pitassi [CP90] on Haken's lower bound in Extended Frege, the earlier work on the Proof Analysis Problem [AAdRK25], and the recent work of Li, Li, and Ren [LLR26] in the context of **TFNP**. The first step towards formalizing constant-depth Frege lower bounds was recently taken by Davis and Robere [DR26] in a limited parameter regime, showing that exponential lower bounds on the length of constant-depth Frege proofs of the pigeonhole principle can be proven in Res(log) in quasipolynomial size. The result presents a reduction from the weak pigeonhole principle in the style of the **TFNP** framework of [LLR26]. Accordingly, the parameters suffer from some of the same caveats as [LLR26]: the formalization works only in the exponential-size regime, and it furthermore requires a polynomial bound on the size of the lines of the purported refutations.

Similar meta-mathematical approaches to proof complexity, heavily featuring variants of the Ref formulas studied in this paper, appear in the work of Pich and Santhanam [PS19], Santhanam and Tzameret [ST21], and Lu, Santhanam, and Tzameret [LST26]. In the context of circuit complexity and communication complexity, more work has been carried out (see, for example, [Raz95; MP20; CLO24; GC25; CLOW26] and references therein). All of these works fit into the general trend of the so-called reverse mathematics of computational complexity. The literature is too vast to review here, so we refer the reader to the survey of Oliveira [Oli25]. In parallel, a growing body of work has used tools and ideas from mathematical logic to prove complexity-theoretic statements (see, e.g., [Gay21; Mül21; Kha22a; Kha22b; PS22; Nar22; PS23; Kra23; Gay23; Kha24; ACG24; AKPS24; Kra24; Nar24; Gay24; JJ25; JT26]).

1.5 Structure of the paper

The rest of the paper is structured as follows. After the preliminaries in Section 2, Section 3 introduces the new assignment extraction algorithm, proves its correctness and gives implementations as $\mathbf{TC}^0$ and $\mathbf{AC}^0$ circuits. We then turn to the task of formalizing the correctness in bounded arithmetic: Section 4 derandomizes the restriction argument via limited independence in $\mathsf{APC}_2(\alpha)$, and Section 5 formalizes the block-width lower bound. We put these pieces together in Section 6. In Section 7 we apply the Paris–Wilkie translation to get propositional upper bounds and related corollaries. Section 8 proves that the Proof Analysis Problem for constant-depth Frege is **NP**-hard.

2 Preliminaries

We review basic material from proof complexity and bounded arithmetic. For a more comprehensive treatment of proof complexity, we refer to Krajíček [Kra19]. For bounded arithmetic, the survey of Oliveira [Oli25] covers all the necessary material in a style that aligns with our work, but the reader may also look up the classic textbooks on these topics [Kra95; HP93].

2.1 Proof complexity

Following Cook and Reckhow [CR79], 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. If $S(\pi) = \varphi$, then we say π is an *S-proof* of φ. 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.

For a tautology φ and a proof system S, we denote by $\text{size}_S(\varphi) := \min_{\pi : S(\pi)=\varphi} |\pi|$ 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 $\text{size}_S(\varphi) \leq |\varphi|^c$. For a sequence $\varphi = \{\varphi_n\}_{n\in\mathbb{N}}$ of propositional tautologies, we write $S \vdash^{\text{poly}} \varphi$ or simply $S \vdash^{\text{poly}} \varphi_n$ to express that $\text{size}_S(\varphi_n) = |\varphi_n|^{O(1)}$ as n grows. For a concrete proof π achieving this, we write $\pi : S \vdash \varphi_n$. 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 $\pi : S \vdash \varphi_n$. For a sequence $\{\varphi_n\}_{n\in\mathbb{N}}$ of formulas or a sequence $\{\pi_n\}_{n\in\mathbb{N}}$ of proofs, we say that the sequence is **P**-*uniform* if we can construct the n-th element of the sequence in time $\text{poly}(n)$. We similarly use the term **QP**-*uniform* when the construction runs in quasipolynomial time. More generally, we may say $\mathbf{TIME}[t(n)]$-*uniform* for any time-constructible function $t : \mathbb{N} \to \mathbb{N}$.

A proof system S *simulates* a system Q, written $S \geq Q$, if there is a constant $c \in \mathbb{N}$ such that for every $\varphi \in \text{TAUT}$ we have $\text{size}_S(\varphi) \leq \text{size}_Q(\varphi)^c$. We say that S *p-simulates* Q and write $S \geq_{\text{p}} Q$ if there exists a polynomial-time computable function f such that for every $\varphi \in \text{TAUT}$ and every $\pi : Q \vdash \varphi$, we have $f(\pi) : S \vdash \varphi$. Two proof systems S and Q are *polynomially equivalent* if $S \geq_{\text{p}} Q$ and $Q \geq_{\text{p}} S$.

A *restriction* for a propositional formula $\varphi(x_1, \ldots, x_n)$ is a partial assignment $\rho : \{x_1, \ldots, x_n\} \to \{x_1, \ldots, x_n, 0, 1\}$ mapping every variable to either a constant or to itself. We write $\varphi_{\restriction\rho}$ for the formula $\varphi(\rho(x_1), \ldots, \rho(x_n))$, and we say that a proof system S is *closed under restrictions* if there exists a constant $d \in \mathbb{N}$ such that for every tautology φ and every restriction ρ, it holds that $\text{size}_S(\varphi_{\restriction\rho}) \leq \text{size}_S(\varphi)^d$. All the explicit proof systems dealt with in this work are closed under restrictions, and for a proof π of φ, we denote by $\pi_{\restriction\rho}$ the proof of the restricted formula $\varphi_{\restriction\rho}$ obtained by applying the restriction line by line and performing the necessary clean-up. The notion of restriction is a particular case of a *formula substitution*: a partial function mapping $\{x_1, \ldots, x_n\}$ to propositional formulas. For a substitution σ sending the variables in $\varphi(x_1, \ldots, x_n)$ to formulas $\psi_1, \ldots, \psi_n$, we write $\varphi(\psi_1, \cdots \psi_n)$ for the formula obtained by substituting ψ_i for x_i.

2.1.1 Resolution

Resolution (Res) is a refutation system for CNF formulas. Accordingly, we often write $\pi : \text{Res} \vdash \neg\varphi$ to express that π is a Resolution refutation of φ.

A *literal* is a propositional atom or its negation, a *clause* is a disjunction of literals, and we see clauses as sets of literals. We write $C \subseteq D$ 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 $i \in [s]$, the clause D_i is either (a) one of the clauses $C_1, \ldots, C_m$ of φ, or (b) 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*:

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

where we say that $A \vee B$ is obtained by *resolving* over x.

The *length* of π, denoted $\text{length}(\pi)$, is precisely s, while we denote by $|\pi|$ the length of π as binary string, where $\text{length}(\pi) \leq |\pi|$. We write $\varphi \vdash^s_{\text{Res}} \bot$ to indicate that there exists a Resolution refutation of φ in length s. Resolution is *implicationally complete*, meaning that for every pair of CNF formulas φ and ψ, if $\varphi \models \psi$, there exists a derivation of each clause of ψ from the clauses of φ. We write $\varphi \vdash_{\text{Res}} \psi$ to express that

such a collection of derivations exists. The *width* of a clause $\mathrm{w}(C)$ is the number of literals in C, and the width of a refutation π is naturally defined as $\mathrm{w}(\pi) \coloneqq \max_{C \in \pi} \mathrm{w}(C)$. To every Resolution refutation π we can associate a directed acyclic graph in a natural way, with the root labeled by the empty clause $\bot$. The number of vertices in this graph is exactly $\mathrm{length}(\pi)$.

We often see a Resolution refutation as a *Prover–Delayer game* [Pud00; AD08]. The game is played between two players, Prover and Delayer. For a fixed formula φ over n variables, in each round, the Prover either (a) asks for the value of a variable of φ and stores the answer in memory, or (b) chooses to forget the value of a previously queried variable. When a query is issued, the Delayer answers by choosing a truth-value for the variable, which is stored in memory by the Prover. The Prover wins when the partial assignment kept in memory falsifies a clause of φ. The key measure for us is the amount of memory of the Prover. Indeed, if a formula φ requires width w to be refuted in Resolution, there is a strategy for the Delayer such that by the time Prover wins, he must have $\Omega(w)$ variables in memory. The notion is readily generalized to other notions of width, like the idea of block-width we use later.

2.1.2 Frege systems

A *Frege system* [CR79] 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 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 [CR79].

A generalization of such systems are *Extended Frege* (EF) systems [CR79], in which proof lines can be succinctly written as Boolean circuits rather than formulas [Jeř05]. We often consider extensions of Extended Frege by sets of additional axioms. For a set $A \subseteq \textsc{Taut}$ of tautologies such that $A \in \mathbf{P}$, the system $\mathsf{EF} + A$ refers to Extended Frege together with the axiom schemas that allow substitution instances of any formula in A. 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, known as *constant-depth Frege systems*, in which lines are restricted to be Boolean circuits of unbounded fan-in and depth at most d.

For the sake of precision, most of our formal statements about constant-depth Frege systems are stated for a specific Tait-style calculus defined below, although they could be phrased equally well in a Gentzen-style sequent calculus formalism like LK (see [Kra19, §3.4]). We follow here the presentation of Beckmann, Pudlák, and Thapen [BPT14, §1.1], but see also other sources [Bus04; BB05; BB19].

Formulas in the PK refutation system are formed from literals together with conjunctions and disjunctions of unbounded fan-in (denoted $\bigwedge$ and $\bigvee$). In particular, negations only appear at the atomic level, and we use $\neg\varphi$ for a formula φ as an abbreviation for the corresponding formula where negations are pushed to the atoms, interchanging connectives where needed.

For such a formula φ, its *depth* is defined recursively to be $\mathrm{depth}(\ell) \coloneqq 0$ for every literal ℓ, and $\mathrm{depth}(\bigwedge_{i=1}^k \varphi_i) = \mathrm{depth}(\bigvee_{i=1}^k \varphi_i) \coloneqq 1 + \max_{i \in [k]} \mathrm{depth}(\varphi_i)$. A *cedent* is a sequence of formulas, separated by commas, interpreted as their disjunction. The system PK consists of two introduction rules and an excluded middle axiom for atoms,

$$\frac{\Gamma, \varphi \qquad \Gamma, \psi}{\Gamma, \varphi \wedge \psi}\ (\wedge\text{-Intro}) \qquad \frac{\Gamma, \varphi, \psi}{\Gamma, \varphi \vee \psi}\ (\vee\text{-Intro}) \qquad \frac{}{p, \neg p}\ (\text{Atomic LEM})$$

together with weakening and a cut rule that generalizes the Resolution rule,

$$\frac{\Gamma}{\Gamma, \Delta}\ \text{(Weak)} \qquad \frac{\Gamma, \varphi_1 \wedge \cdots \wedge \varphi_m \qquad \Gamma, \neg\varphi_1, \ldots, \neg\varphi_m}{\Gamma}\ \text{(Cut)}$$

where Γ and Δ are possibly empty cedents. A *PK-derivation* of a formula φ from a set Γ of disjunctions (interpreted as their conjunction), is a sequence of disjunctions ending in φ, such that each disjunction in the sequence is either in Γ, or follows from earlier disjunctions in the sequence by one of the rules above. A PK-*refutation* of a set of disjunctions Γ is a PK-derivation of the empty disjunction $\bot$ from Γ. We write $\mathsf{PK} \vdash \neg\varphi$ to indicate that PK refutes φ, when interpreted as a set of disjunctions.

For every $d \in \mathbb{N}$, we write PK_d for the subsystem of PK in which every formula in a cedent has depth at most d. We also define the systems PK^k_d for $k \in \mathbb{N}$ to be the systems in which the formulas have depth $d+1$, but where all gates at depth d have fan-in at most k. The system PK_0 is readily seen to be precisely Resolution (Res). For $k \geq 2$, the system PK^k_0 is denoted by $\mathrm{Res}(k)$ and known as *k-DNF Resolution* [Kra01].

Finally, for every $d \in \mathbb{N}$, we denote by $\mathsf{PK}_{d+1/2}$ the subsystem of PK in which a refutation π has size S if and only if every cedent has size at most S and the refutation has depth $d+1$ but every formula at depth $d+1$ has fan-in at most $\log S$. This follows the notation $\mathsf{LK}_{d+1/2}$ of Krajíček [Kra01]. The system $\mathsf{PK}_{0+1/2}$ is denoted $\mathrm{Res}(\log)$. We sometimes use the notation $\mathbf{AC}^0_d$-Frege to refer to PK_d.

2.1.3 Automatability

Following Bonet, Pitassi, and Raz [BPR00], for a time-constructible function $T : \mathbb{N} \to \mathbb{N}$, we say that a proof system S is *automatable in time T* if there exists an algorithm that given a propositional tautology φ, outputs an S-proof of φ in time $T(|\varphi| + \mathrm{size}_S(\varphi))$. When T is a polynomial, we simply say S is *automatable*. We say that S is *weakly automatable* [AB04] if there exists a proof system Q that is automatable and p-simulates S. This is readily seen to be equivalent to saying that there is an algorithm automating S except that proofs are output in a possibly different system S'. In this case we say that S is *weakly automated by S'*.

2.2 Bounded arithmetic

Theories of bounded arithmetic act as a uniform counterpart to propositional proof systems. Below we present the main theories used in this work.

2.2.1 Cook's theory PV_1

Cook's theory PV_1 [Coo75; KPT91] 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 **FP** [Cob65]. 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 [Kra95, Definition 5.3] for the details.

2.2.2 Buss's theories: S^i_2 and T^i_2

Let $\mathcal{L}_{\mathsf{PA}}$ denote the language of Peano Arithmetic, $\mathcal{L}_{\mathsf{PA}} := \{0, 1, +, \cdot, <\}$. 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, [Kra19, §7.4.3] for a definition), together with the *Induction Schema*

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

available for every formula φ.

Buss's theories operate over the *first-order language of bounded arithmetic*, defined by the symbols $\mathcal{L}_{\mathrm{BA}} \coloneqq \{0, 1, +, \cdot, <, |\cdot|, \lfloor \cdot/2 \rfloor, \#\}$. This extends the language of Peano Arithmetic $\mathcal{L}_{\mathrm{PA}}$ above by the symbols $|x|$, $\lfloor x/2 \rfloor$ and $x\#y$. The notation $|x|$ denotes the length of the binary encoding of the number x, defined as $\lceil \log(x+1) \rceil$, 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 $\forall x < |s|$ or $\exists x < |s|$ for some term s, then they are called *sharply bounded* quantifiers. The *hierarchy of bounded formulas* consists of the classes Σ^b_i (and Π^b_i), for $i \in \mathbb{N}$, 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_i consists of all formulas that admit an equivalent definition in both Σ^b_i and Π^b_i. In particular, the class $\Delta^b_0 = \Sigma^b_0 = \Pi^b_0$ stands for all formulas with sharply bounded quantifiers only.

The theories S^i_2 and T^i_2 of Buss [Bus86] extend Robinson's arithmetic Q by a set BASIC of simple axioms for the new function symbols (see, e.g., [Kra95, Definition 5.2.1] for the complete list). On top of this, the S^i_2 theories have the *Polynomial Induction* schema (PInd) for Σ^b_i-formulas: for every $\varphi \in \Sigma^b_i$, the theory contains the axiom

$$\varphi(0) \wedge \forall x\ (\varphi(\lfloor x/2 \rfloor) \to \varphi(x)) \to \forall x\ \varphi(x). \tag{PInd$_\varphi$}$$

The theory T^i_2 is defined analogously, except we allow instead the standard induction schema (Ind$_\varphi$) for Σ^b_i formulas. The theory T_2 is defined as $\mathsf{T}_2 \coloneqq \bigcup_{i \in \mathbb{N}} \mathsf{T}^i_2$, and similarly $\mathsf{S}_2 \coloneqq \bigcup_{i \in \mathbb{N}} \mathsf{S}^i_2$. It holds that $\mathsf{T}_2 = \mathsf{S}_2$, and $\mathsf{PV}_1 \subseteq \mathsf{S}^1_2$. The theories S^i_1 and T^i_1 are defined analogously, except for the absence of smash (#).

Unlike $\mathcal{L}_{\mathrm{PV}}$, the language $\mathcal{L}_{\mathrm{BA}}$ 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. In the rest of the paper we abuse notation and work instead over $\mathsf{T}^i_2(\mathcal{L}_{\mathrm{PV}})$ and $\mathsf{S}^i_2(\mathcal{L}_{\mathrm{PV}})$, which are the theories 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 the corresponding induction is now available for all $\Sigma^b_i(\mathrm{PV})$ formulas.

The *Log-notation* $n \in \mathrm{Log}$ is a short-hand for the formula $\exists x\ |x| = n$. A set X is called a *bounded definable set with parameter* y if there exists an arithmetic formula $\varphi(x, y)$ and some term $t(y)$ such that $X = \{x < t(y) \mid \varphi(x, y)\}$. We sometimes also adopt the standard set-theoretic notation denoting the interval $[0, a)$ directly by a. Then, a Boolean circuit $C : 2^k \to 2$ naturally defines a bounded definable set $X_C = \{x < 2^k \mid C(x) = 1\}$ with parameter C, and there exists a PV function Count which counts Log-sized initial segments of circuit-definable sets; i.e., in the standard model, $\mathrm{Count}(C, a)$ is the cardinality of the set $X_C \cap [0, |a|)$. Importantly, if $2^k \in \mathrm{Log}$, then we can *exactly count* the size of X_C when working in Buss's hierarchy. This proves crucial when carrying out different combinatorial arguments.

Following standard notation (see, for example, [HP93, pp. 324–325]), we denote by $(\cdot)_k$ the PV-function that given a number X in the theory, interpreted as a sequence, outputs the k-th element in the sequence. In this way, for $\ell \in \mathrm{Log}$ and n not necessarily in Log, we can quantify over sequences of length ℓ with elements from $[n]$: the sequence $X \in [n]^\ell$ is encoded by a number and we refer to the k-th element in the sequence by $(X)_k$. We denote by $\mathrm{len}(X)$ the PV function that outputs the number ℓ of elements in the sequence.

2.2.3 Jeřábek's theories APC_1 and APC_2

When a definable set is too large to count exactly, we can often still approximately count it using Jeřábek's framework for approximate counting [Jeř05; Jeř07]. For a class $\mathbf{C}$ of functions and $f \in \mathbf{C}$, the *dual weak pigeonhole principle for* f, denoted by $\mathrm{dWPHP}(f)$, is the formula stating that if f is a function from $[2^n]$ to $[(1+1/n)\cdot 2^n]$, then it is not surjective. More formally,

$$\mathrm{dWPHP}(f) \coloneqq \forall n \in \mathrm{Log}\ \forall z\, \exists y < (1+1/n)2^n\, \forall x < 2^n\, f(x,z) \neq y. \tag{dWPHP}$$

We write $\mathrm{dWPHP}(\mathbf{C})$ for the set of all such sentences for $f \in \mathbf{C}$.

We will overload notation and write PV_1 for the class $\mathbf{FP}$ of polynomial-time functions, and PV_2 for $\mathbf{FP}^{\mathbf{NP}}$. We can then define the theories $\mathrm{APC}_1 \coloneqq \mathrm{PV}_1 + \mathrm{dWPHP}(\mathrm{PV}_1)$, and $\mathrm{APC}_2 \coloneqq \mathsf{T}^1_2 + \mathrm{dWPHP}(\mathrm{PV}_2)$. It is known that $\mathrm{APC}_1 \subseteq \mathsf{T}^2_2$, and $\mathrm{APC}_2 \subseteq \mathsf{T}^3_2$ [PWW88; MPW02; Kra99; Jeř09].

In this framework, a map of the form $f : A \hookrightarrow B$ indicates that f is an injection, and $f : A \twoheadrightarrow B$ indicates that it is a surjection. For definable sets X and Y and $\varepsilon \leq 1$, we say that *the size of X is approximately less than the size of Y with error ε*, written as $X \preceq_\varepsilon Y$, if there exists a circuit G and $v \neq 0$ such that

$$G : v \times (Y \dot\cup \varepsilon 2^n) \twoheadrightarrow v \times X, \tag{2.1}$$

where $\dot\cup$ denotes disjoint union.

Let us state a few useful properties of the $\preceq_\varepsilon$ relation, which we invoke later.

Lemma 2.1 ([Jeř07], Lemmas 2.10.(ii)–(iii) and 2.11.(ii))**.** *Let $X, Y, Z \subseteq 2^n$ be sets definable by Boolean circuits, let $\varepsilon, \delta \leq 1$, and let $s, t \leq 2^n$. The following hold:*

(i) The theory PV_1 proves that if $X \subseteq Y$, then $X \preceq_0 Y$;

(ii) The theory PV_1 proves that if $X \preceq_\varepsilon Y$ and $Y \preceq_\delta Z$, then $X \preceq_{\varepsilon+\delta} Z$;

(iii) The theory APC_1 proves that if $s \preceq_\varepsilon X \preceq_\delta t$, then for every $\xi > 0$ with $\xi \leq 1$ and $\xi^{-1} \in \mathrm{Log}$,

$$s \leq t + (\varepsilon + \delta + \xi)2^n.$$

Using the $\preceq_\varepsilon$ relation we can simulate union bound arguments as follows.

Lemma 2.2 ([Jeř07, Proposition 2.15])**.** *Let $b \in \{1,2\}$ and let $\{X_i \mid i < m\}$ be a PV_b-definable set of subsets of 2^n, each of them defined by a sequence of circuits. Provably over APC_b, for every $\varepsilon, \xi \leq 1$ with $\xi^{-1} \in \mathrm{Log}$ and $\{s_i \mid i < m\}$ a sequence of numbers such that $X_i \preceq_\varepsilon s_i$ for every $i < m$, then*

$$\sum_{i<m} X_i \preceq_{\varepsilon+\xi} \sum_{i<m} s_i,$$

where the disjoint sum $\sum_{i<m} X_i \coloneqq \bigcup_{i<m}(X_i \times \{i\}) \subseteq 2^n \times m$ is considered a subset of $2^{n+|m|}$. Furthermore, the same holds for $\succeq$ in place of $\preceq$.

We also invoke several useful basic algebraic facts formalized by Jeřábek in PV_1. In particular, PV_1 can perform basic field arithmetic over $\mathrm{GF}(q)$ as long as $q \in \mathrm{Log}$ (see [Jeř05, §4.3.1]). Similarly, PV_1 can carry out basic linear algebra efficiently over such fields. In particular, for matrices represented as numbers, the rank of such matrices is computed by a PV function. We refer the reader to [Jeř05, §4.3.2] for more details. Below we state a couple of useful results.

Lemma 2.3 (Root upper bound in PV_1 [Jeř05, Lemma 4.3.6])**.** *Provably in PV_1, for every finite field $\mathbb{F}$ and P a non-zero polynomial of degree d over $\mathbb{F}$, it holds that P has at most d roots.*

Lemma 2.4 (Gaussian elimination in PV_1 [Jeř05, Theorem 4.3.19])**.** *There is a* PV*-function* Sol *that provably in* PV_1 *computes a solution to a system of linear equations over* $\mathbb{F}(x)$ *if one exists, and a* PV*-function* Basis *that provably in* PV_1 *computes a basis for the space of all solutions of a homogeneous linear system over* $\mathbb{F}(x)$*, where* $\mathbb{F}$ *is a finite field.*

2.2.4 Relativized theories and the Paris–Wilkie translation

We can extend the language $\mathcal{L}_{\mathrm{BA}}$ of bounded arithmetic by any countable number of relation symbols of any finite arity. We denote such a language by $\mathcal{L}_{\mathrm{BA}}(\alpha)$ where the symbols added are unspecified, and write, for instance, $\mathcal{L}_{\mathrm{BA}}(R,S)$, when two explicit symbols R and S are being added. The theories $\mathsf{S}^i_2(\alpha)$ and $\mathsf{T}^i_2(\alpha)$ are the natural extensions of S^i_2 and T^i_2 where the corresponding induction schema is allowed for relativized formulas.

By relativizing the theories we obtain a crucial connection to constant-depth Frege systems. We follow here the presentation of Krajíček [Kra19, §8.2] of the *Paris–Wilkie translation* [PW85]. Let $\phi(x_1,\dots,x_k)$ be a $\Sigma^b_i(R)$-formula in the language $\mathcal{L}_{\mathrm{BA}}(R)$, where R is a new relation symbol. (For simplicity, we assume here that R is a binary relation symbol, but it is straightforward to generalize the definition to any number of relation symbols of finite arity.) For every such $\Pi^b_i(R)\cup\Sigma^b_i(R)$-formula $\phi(x_1,\dots,x_k)$ and natural numbers $\bar{n}=n_1,\dots,n_k\in\mathbb{N}$, the translation produces the formula $\langle\phi\rangle_{\bar{n}}$ in negation normal form over the variables $p_{i,j}$ for $i,j\in[\max\{n_1,\dots,n_k\}]$, defined recursively as follows.

- If ϕ is an atomic formula like $t(\bar{x})=s(\bar{x})$ or $t(\bar{x})\le s(\bar{x})$, where s and t are terms, then $\langle\phi\rangle_{\bar{n}}\coloneqq\top$ if $\mathbb{N}\models\phi(\bar{n})$, and $\bot$ otherwise.
- If ϕ is the atomic formula $R(t(\bar{x}),s(\bar{x}))$ for terms s and t and i and j are, respectively, the values of $t(\bar{n})$ and $s(\bar{n})$ over $\mathbb{N}$, then we define $\langle\phi\rangle_{\bar{n}}\coloneqq p_{i,j}$.
- If $\phi=\neg\psi$, then $\langle\phi\rangle_{\bar{n}}\coloneqq\neg\langle\psi\rangle_{\bar{n}}$, with negations propagated to the atoms.
- If $\phi=\psi\vee\chi$, then $\langle\phi\rangle_{\bar{n}}\coloneqq\langle\psi\rangle_{\bar{n}}\vee\langle\chi\rangle_{\bar{n}}$, and if $\phi=\psi\wedge\chi$, then $\langle\phi\rangle_{\bar{n}}\coloneqq\langle\psi\rangle_{\bar{n}}\wedge\langle\chi\rangle_{\bar{n}}$.
- If $\phi(\bar{x})=\forall y\le t(\bar{x})\ \psi(\bar{x},y)$, then $\langle\phi\rangle_{\bar{n}}\coloneqq\bigwedge_{m\le t(\bar{n})}\langle\psi(\bar{x},y)\rangle_{\bar{n},m}$.
- If $\phi(\bar{x})=\exists y\le t(\bar{x})\ \psi(\bar{x},y)$, then $\langle\phi\rangle_{\bar{n}}\coloneqq\bigvee_{m\le t(\bar{n})}\langle\psi(\bar{x},y)\rangle_{\bar{n},m}$.

The connection we later use can be stated as follows, and goes back to the seminal work of Paris and Wilkie [PW85]. The version we use for translating from Buss's hierarchy can be traced back to Krajíček [Kra94, §5; Kra95, Corollary 9.1.4]. See also [Kra19, Lemma 10.5.2 and §10.6.1].

Theorem 2.5 (The Paris–Wilkie translation for $\mathsf{T}^i_2(\alpha)$)**.** *Let* $d\ge 1$*, let* $\phi(x)=\exists y\le x\,\forall z\le x\,\psi(x,y,z)$ *be a* $\Sigma^b_{d+1}(\alpha)$*-formula, where* ψ *is* $\Sigma^b_{d-1}(\alpha)$*. If* $\mathsf{T}^d_2(\alpha)\vdash\forall x\,\phi(x)$*, then there are* QP*-uniform* $\mathsf{PK}_{(d-1)+1/2}$*-refutations of the sets*

$$\left\{\left[\neg\langle\psi\rangle_{n,a,0},\dots,\neg\langle\psi\rangle_{n,a,n}\right]\ \middle|\ 0\le a\le n\right\}_{n\in\mathbb{N}},$$

in size quasipoly(n)*, where each* $[\cdots]$ *denotes a cedent of propositional translations of* $\neg\psi(x,y,z)$*, with* $x=n$ *and* $y=a$ *fixed, and* z *ranging from* 0 *to* n*. Furthermore, if the theory is* $\mathsf{T}^1_1(\alpha)$ *and the formula is* $\forall\Sigma^b_1(\alpha)$*, the translation produces polynomial-size Resolution refutations, and if the theory is* $\mathsf{T}^2_2(\alpha)$ *and the formula is* $\forall\Sigma^b_1(\alpha)$*, the translation gives quasipolynomial-size* Res(log) *refutations.*

2.3 The Ref formulas and the Proof Analysis Problem

Given a propositional CNF formula φ and a size parameter $s\in\mathbb{N}$, we define a CNF formula $\mathrm{Ref}_s(\varphi)$ stating that φ has a Resolution refutation consisting of at most $s+1$ clauses. Several alternative encodings of these formulas have appeared in the literature (see [AAdRK25, §2.4] for a discussion on the different variants),

but we stick here to the natural and robust unary encoding used in [AM20] and [AAdRK25]. The only change we introduce is that we index the blocks up to $s+1$, with $s+1$ corresponding to the root block, while in [AAdRK25] the blocks were numbered up to s. This change simplifies bookkeeping in some later proofs.

The formula $\textsc{Ref}_s(\varphi)$ consists of $s+1$ so-called *blocks* of variables, each representing a clause in the purported Resolution refutation. More precisely, blocks indexed from 1 to s represent clauses in the refutation, and block $s+1$ represents the empty clause at the end of the refutation. Each block has variables to represent the literals that appear in this block, how it was obtained (either resolved or weakened from an axiom), and it contains *pointer variables* to indicate from which blocks it was derived. Crucially, the formula includes *enabling variables* used to deactivate blocks.

The variables of $\textsc{Ref}_s(\varphi)$. We assume φ is a CNF formula over n variables $x_1, \dots, x_n$ and m clauses, and let $\mathrm{Lit}_n \coloneqq \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$. We define the following variables with their corresponding meaning,

$\mathsf{a\text{-}lit}_\ell^A$	:	literal $\ell \in \mathrm{Lit}_n$ is present in the clause $A \in [m]$ of φ;
enable^B	:	block $B \in [s+1]$ is enabled;
$\mathsf{derived}^B$	:	block $B \in [s+1]$ is obtained by a Resolution step;
weak_A^B	:	block $B \in [s+1]$ is obtained by weakening from clause $A \in [m]$ of φ;
lit_ℓ^B	:	literal $\ell \in \mathrm{Lit}_n$ is present in the block $B \in [s+1]$;
$\mathsf{res}_{x_i}^B$	:	block $B \in [s+1]$ is obtained by resolving over the variable x_i;
$\mathsf{lpoint}_{B'}^B$	:	block $B \in [s+1]$ is resolved on the left from block $B' \in [s+1]$, $B' < B$;
$\mathsf{rpoint}_{B'}^B$	:	block $B \in [s+1]$ is resolved on the right from block $B' \in [s+1]$, $B' < B$.

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

Definition 2.6 (The Ref formulas)**.** Let $n, m, s \in \mathbb{N}$, and let $\mathrm{Lit}_n \coloneqq \{x_1, \dots, x_n, \neg x_1, \dots, \neg x_n\}$ denote the set of possible literals over n variables. The $\textsc{Ref}_s(\varphi)$ formula is built from the variables defined above, together with the conjunction of the following clauses:

$$\left(\mathsf{enable}^B \wedge \mathsf{res}_{x_i}^B \wedge \mathsf{lpoint}_{B'}^B \wedge \mathsf{lit}_\ell^{B'}\right) \to \mathsf{lit}_\ell^B \quad \text{for } B, B' \in [s+1], B' < B, i \in [n], \ell \in \mathrm{Lit}_n \setminus \{x_i\}, \tag{Ref-1}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{res}_{x_i}^B \wedge \mathsf{rpoint}_{B'}^B \wedge \mathsf{lit}_\ell^{B'}\right) \to \mathsf{lit}_\ell^B \quad \text{for } B, B' \in [s+1], B' < B, i \in [n], \ell \in \mathrm{Lit}_n \setminus \{\neg x_i\}, \tag{Ref-2}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{weak}_A^B \wedge \mathsf{a\text{-}lit}_\ell^A\right) \to \mathsf{lit}_\ell^B \quad \text{for } B \in [s+1], A \in [m], \ell \in \mathrm{Lit}_n, \tag{Ref-3}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{derived}^B\right) \to \bigvee_{i \in [n]} \mathsf{res}_{x_i}^B \quad \text{for } B \in [s+1], \tag{Ref-4}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{derived}^B\right) \to \bigvee_{\substack{B' \in [s+1] \\ B' < B}} \mathsf{lpoint}_{B'}^B \quad \text{for } B \in [s+1], \tag{Ref-5}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{derived}^B\right) \to \bigvee_{\substack{B' \in [s+1] \\ B' < B}} \mathsf{rpoint}_{B'}^B \quad \text{for } B \in [s+1], \tag{Ref-6}$$

$$\left(\mathsf{enable}^B \wedge \neg\mathsf{derived}^B\right) \to \bigvee_{A \in [m]} \mathsf{weak}_A^B \quad \text{for } B \in [s+1], \tag{Ref-7}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{lpoint}_{B'}^B\right) \to \mathsf{enable}^{B'} \quad \text{for } B, B' \in [s+1], B' < B \tag{Ref-8}$$

$$\left(\mathsf{enable}^B \wedge \mathsf{rpoint}_{B'}^B\right) \to \mathsf{enable}^{B'} \quad \text{for } B, B' \in [s+1], B' < B \tag{Ref-9}$$

$\neg \mathsf{lit}_\ell^{s+1}$ for $\ell \in \mathrm{Lit}_n$, (Ref-10)

enable^{s+1}. (Ref-11)

We also consider a variant of $\mathrm{Ref}_s(\varphi)$ where the enable^B variables do not appear. This will be denoted by $\mathrm{ERef}_s(\varphi)$, and is simply $\mathrm{Ref}_s(\varphi)$ after setting $\mathsf{enable}^B \mapsto 1$ for every $B \in [s+1]$.

Remark 2.7 (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 $N = O(n^{\max\{2c, c+k\}})$. ◀

The following remark will be useful for the new extraction algorithm.

Remark 2.8 (Simulating weakenings). The weak variables only allow weakening from clauses of φ. To encode that a block B is a weakening of some other block B', one can simply set $\mathsf{lpoint}^B_{B'} \mapsto 1$, $\mathsf{rpoint}^B_{B'} \mapsto 1$, and $\mathsf{lpoint}^B_C \mapsto 0$ and $\mathsf{rpoint}^B_C \mapsto 0$ for every $C \neq B'$. Note that the clauses of Ref do not impose that the left and right pointers need to be different, so this is a valid assignment. One can then fix $\mathsf{res}^B_{x_i} \mapsto 1$ for some $i \in [n]$ and $\mathsf{res}^B_{x_j} \mapsto 0$ for every $j \in [n] \setminus \{i\}$, and note that even when we say the block was obtained by resolving over x_i, both literals x_i and $\neg x_i$ may still appear in B. ◀

Blocks and block-width. We say that a variable of $\mathrm{Ref}_s(\varphi)$ *belongs to block* B if it is indexed by B as a superscript. We also say that the variable *mentions* B. 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, without counting the root block, mentioned by the variables of the literals in C. We denote this measure by $\mathrm{bw}(C)$, and note that for every clause C, $\mathrm{bw}(C) \leq s + 1$. We generalize the measure to refutations by taking $\mathrm{bw}(\pi)$ to be the maximum block-width over all the clauses in π. We also generalize this measure to partial assignments over the variables of $\mathrm{Ref}_s(\varphi)$ in the natural way.

Encoding as first-order formulas. We will later work in the relativized theories $\mathsf{T}^i_2(\Phi, \Pi)$, where the language is the language of bounded arithmetic expanded by two new relation symbols Φ and Π. These relation symbols are intended to encode, respectively, a CNF formula φ over n variables and m clauses and a Resolution refutation π of the formula $\mathrm{Ref}_s(\varphi)$. The proof π is encoded with the variables of the formula $\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi))$. When accessing the bits of Π, we fix an encoding as follows.

There are eight types of variables in Ref, and each of them is indexed by up to two values. Let us fix an encoding by mapping $1, 2, \ldots, 8$ to a-lit, enable, derived, $\ldots$, rpoint. We also fix that the literals $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$ are represented by the numbers $1, \ldots, n, n+1, \ldots, 2n$. Then, the value of $\Pi(1, A, \ell)$ for $A \in \{1, \ldots, m\}$ and $\ell \in \{1, \ldots, 2n\}$ indicates the value of the variable $\mathsf{a\text{-}lit}^A_\ell$. Similarly, $\Pi(3, B)$ for $B \in \{1, \ldots, s+1\}$ indicates the value of $\mathsf{derived}^B$, and so on. Hence, in what follows, we abuse notation and write directly the names of the propositional variables instead of the exact indexing of Π. That is, we write $\Pi(\mathsf{derived}^B)$ instead of $\Pi(3, B)$, and so on.

The formula φ is similarly encoded by a new relation symbol Φ (see Section 2.3.1 below for details), such that $\Phi(\ell, k)$ indicates whether literal ℓ is present in clause k. From these we can define first a formula $\text{Res-Ref}(\Pi, \Phi, n, m, t)$ to be the $\Pi^b_2(\Phi, \Pi)$-formula verifying that Π encodes a Resolution refutation of the formula Φ in at most t clauses. It consists simply of the conjunction of the first-order version of the clauses in Definition 2.6.

The additional layer of complication comes from considering Resolution refutations of $\mathrm{Ref}_s(\varphi)$. To encode that Π is a Resolution refutation of $\mathrm{Ref}_s(\varphi)$ rather than of Φ, we first define a formula $\mathrm{Ref}(\Phi, n, m, s; \ell, k)$, where Φ, n, m, s are the parameters of $\mathrm{Ref}_s(\varphi)$, and for a clause $k \in [M(n, m, s)]$ and a literal $\ell \in [2N(n, m, s)]$, the formula indicates whether the literal is present in that given clause. Observe how this formula does not

involve any new quantifiers. We can now consider the composed formula

$$\text{Res-Ref}(\Pi, \text{Ref}(\Phi, n, m, s), N(n, m, s), M(n, m, s), t) \tag{2.2}$$

stating that Π is a Resolution refutation of $\text{Ref}_s(\varphi)$ in at most t clauses. The latter is still $\Pi^b_2(\Phi, \Pi)$, since Ref did not increase the complexity. We note that for every fixed φ, the formula $\langle\text{Res-Ref}\rangle_{n,m,s,t}$ obtained from the Paris–Wilkie translation on this formula is precisely $\text{Ref}_t(\text{Ref}_s(\varphi))$.

Remark 2.9 (Complexity in the exponential-size regime). If we assume $n, m, s \in \text{Log}$, then the Res-Ref formula in the definition above becomes $\Pi^b_1(\Phi, \Pi)$. It is easy to see that the big disjunctions in the clauses of Ref, which now correspond to existential quantifiers, will be sharply bounded if n, m and s are in Log. ◀

2.3.1 The Sat formula

We also have the $\text{Sat}(\varphi, \alpha)$ formula, encoding that a CNF formula φ is satisfied by an assignment α. The variables we consider are

α_i	:	value assigned by α to variable x_i;
a-lit^A_ℓ	:	literal $\ell \in \text{Lit}_n$ is present in clause $A \in [m]$;
sat^A_ℓ	:	clause $A \in [m]$ is satisfied because literal $\ell \in \text{Lit}_n$ evaluates to 1 under α.

Definition 2.10 (The Sat formulas)**.** Let $n, m \in \mathbb{N}$, let φ denote the set of variables of the form a-lit^A_ℓ as above and let α denote the set of variables α_i as above. The formula $\text{Sat}(\varphi, \alpha)$ is the CNF formula over the variables in φ, α and additionally all the variables sat^A_ℓ above consisting of the conjunction of the following clauses,

$$\neg\text{sat}^A_\ell \vee \text{a-lit}^A_\ell \qquad \text{for all } A \in [m] \text{ and } \ell \in \{x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n\}, \tag{Sat-1}$$

$$\neg\text{sat}^A_{x_i} \vee \alpha_i \qquad \text{for all } A \in [m] \text{ and } i \in [n], \tag{Sat-2}$$

$$\neg\text{sat}^A_{\neg x_i} \vee \neg\alpha_i \qquad \text{for all } A \in [m] \text{ and } i \in [n], \tag{Sat-3}$$

$$\bigvee_{i \in [n]} \text{sat}^A_{x_i} \vee \text{sat}^A_{\neg x_i} \qquad \text{for all } A \in [m]. \tag{Sat-4}$$

Resolution can convert between the Sat encoding and the "native" encoding of a formula, in the sense that for every CNF formula φ, if φ has a Resolution refutation of length s, then $\text{Sat}_{\restriction\varphi}$ has a Resolution refutation of length $O(s + nm)$, and if $\text{Sat}_{\restriction\varphi}$ has a Resolution refutation of length s, then φ has a Resolution refutation of length $O(s)$ [AAdRK25, Proposition 2.9].

Encoding as first-order formulas. Just like with the encoding of Resolution refutations into a predicate Π, in the context of first-order languages, we usually assume an additional relation symbol Φ intended to encode a formula φ, together with a predicate α encoding the assignment. If the intended variables are $x_1, \ldots, x_n$ and the formula has m clauses, then we fix again the literals $x_1, \ldots, x_n, \neg x_1, \ldots, \neg x_n$ to be encoded as $1, \ldots, n, n+1, \ldots, 2n$, we interpret $\Phi(\ell, k)$ to be true if and only if literal $\ell \in [2n]$ appears in clause $k \in [m]$.

The natural encoding of the satisfiability formula is then the $\Pi^b_2(\Phi, \alpha)$-formula defined as

$$\text{Sat}(\Phi, n, m) := \forall k \in [m]\ \exists \ell \in [2n]\ \ (\Phi(\ell, k) \wedge ((\ell \in [n] \wedge \alpha(\ell)) \vee (\ell > n \wedge \neg\alpha(\ell - n)))) \,. \tag{Sat}$$

Although the Paris–Wilkie translation of Sat is not syntactically identical to the propositional formula Sat defined earlier, the two encodings are polynomially equivalent for constant-depth Frege refutations,

up to a fixed additive increase in depth. Indeed, Sat implies the translation of Sat by polynomial-size constant-depth Frege proofs, while substituting sat_ℓ^A by the conjunction expressing that ℓ occurs in A and is satisfied by α reduces Sat to the translation of Sat. This difference is inessential for our applications.

2.3.2 The Proof Analysis Problem

We recall the definition of the *Proof Analysis Problem* [AAdRK25, Definition 3.1]. Let Q be a propositional proof system. The *Proof Analysis Problem for* Q is the language

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

where $\mathrm{Ref}_s(\varphi)$ is the formula defined in Section 2.3. We denote by $\mathrm{PAP}_Q[s(n)]$ the problem where the size parameter s is restricted to be at least $s(n)$ for n 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. We say that a propositional proof system Q is *analyzable* if there exists some constant $c > 0$ such that $\mathrm{PAP}_Q[n^c] \in \mathbf{P}$. If a propositional proof system $S \geq \mathrm{Res}$ is both automatable and analyzable, then $\mathbf{P} = \mathbf{NP}$ [AAdRK25, Proposition 3.6].

We will denote by $\textsc{Extraction}_Q$ the *search version of the Proof Analysis Problem for* Q [AAdRK25, Definition 3.5], defined as follows: given a CNF formula φ, a size parameter s in unary and a proof π such that $\pi : Q \vdash \neg \mathrm{Ref}_s(\varphi)$, output either a satisfying assignment for φ, if $\varphi \in \mathrm{SAT}$, or 0 otherwise. Similarly to $\mathrm{PAP}_Q[s(n)]$, we define $\textsc{Extraction}_Q[s(n)]$ to be the search problem where the size parameter is at least $s(n)$. (Note that in [AAdRK25], this search problem was denoted FPAP_Q.)

3 The new assignment extraction algorithm

In this section we introduce the new assignment extraction algorithm and show how to implement the procedure by polynomial-size $\mathbf{TC}^0$-circuits and quasipolynomial-size $\mathbf{AC}^0$-circuits. All of the proofs in this section are carried out in natural language, so for now we are not concerned with the feasibility of the arguments.

In the introduction (Section 1.3.1) we discussed the existence of two algorithms: one in quasipolynomial-size $\mathbf{AC}^0$ and another one in polynomial-size $\mathbf{TC}^0$. These are actually two distinct algorithms, with somewhat different approaches. However, from the $\mathbf{TC}^0$ algorithm alone, one can also recover a quasipolynomial-size $\mathbf{AC}^0$ extraction circuit just by carefully removing the threshold gates with a quasipolynomial increase in size. This is the approach we take in this section.

Section 3.1 revisits the Prover–Delayer game used for the original assignment extraction algorithm of Arteche et al. [AAdRK25] together with the setup and terminology needed to reason about it. Crucially, this section introduces a small modification to the algorithm that proves useful to reason about the new algorithm. The latter is presented in Section 3.2.

Throughout the rest of the section, we fix φ to be a CNF formula over n variables and m clauses, and let $s \in \mathbb{N}$ be an integer parameter used in the formulas $\mathrm{ERef}_s(\varphi)$ and $\mathrm{Ref}_s(\varphi)$. Recall that ERef, as defined in Definition 2.6, is just the version of Ref in which all enable variables are assigned to 1.

3.1 The Prover–Delayer game, revisited

We start by revisiting the original algorithm by Arteche et al. [AAdRK25]. The idea there was to look at the block-width lower bound of Atserias and Müller [AM20] as a Prover–Delayer game and consider the strategy of the Delayer that leads to the lower bound. Then, traversing the refutation following this strategy yields an algorithm that finds the satisfying assignment. See Section 2.1.1 for more details on

Prover–Delayer games, and Section 1.3.1 for more details on the way the original algorithm worked. Note that throughout this section we consider the formula $\mathrm{EREF}_s(\varphi)$, where the enable variables are not present (see Definition 2.6).

Like in the original algorithm, the Delayer keeps a *reservation* in memory, which embeds partial information about the purported refutation into the canonical exponential-size tree-like refutation (see Figure 2). In [AM20] and [AAdRK25], this was achieved by taking the non-root blocks of $\mathrm{EREF}_s(\varphi)$ and arranging them on a grid with n layers and s/n blocks per layer, the idea being that the i-th layer corresponds to the clauses appearing at layer i in the exponential-size tree-like refutation.

We now introduce a generalization of this. We arrange the s non-root blocks of $\mathrm{EREF}_s(\varphi)$ into $\ell := n + w_0$ layers, for a parameter $w_0 \geq 0$ to be fixed later, each containing $\lfloor s/\ell \rfloor$ blocks (with the remainder blocks left from the flooring operations collected all in the last layer), and one additional layer on top with the root block. We think of the root block as positioned at layer 0. Here w_0 is a parameter to be fixed later, and, in particular, for $w_0 = 0$, we recover the grid that the Delayer used in the original strategy. For two blocks $b, c \in [s]$, we say that a partial assignment R to the variables of $\mathrm{EREF}_s(\varphi)$

- *fixes the derived variable of b* if it sets a value for $\mathsf{derived}^b$;
- *fixes the resolved variable of b to a variable x_i* if $R(\mathsf{res}^b_i) = 1$ and $R(\mathsf{res}^b_{i'}) = 0$ for all $i' \neq i$;
- *fixes the weak pointer of b to an axiom $A \in [m]$* if $R(\mathsf{weak}^b_A) = 1$ and $R(\mathsf{weak}^b_{A'}) = 0$ for all $A' \neq A$;
- *fixes the left (respectively, right) pointer of b to c* if it sets $R(\mathsf{lpoint}^b_c) = 1$ (respectively, $R(\mathsf{rpoint}^b_c) = 1$) and $R(\mathsf{lpoint}^b_{c'}) = 0$ (respectively, $R(\mathsf{rpoint}^b_{c'}) = 0$) for all $c \neq c'$, and there is no $b' \neq b$ such that $R(\mathsf{lpoint}^{b'}_c) = 1$ or $R(\mathsf{rpoint}^{b'}_c) = 1$.

The Delayer strategy manipulates very structured partial assignments fixing such pointers called *reservations*. The formal definition follows.

Definition 3.1 (Reservations for $\mathrm{EREF}_s(\varphi)$)**.** Fix a set $B \subseteq [s+1]$ of blocks and a number $\ell \in \mathbb{N}$ of layers. A *basic reservation* R for blocks B and ℓ layers is a partial assignment for $\mathrm{EREF}_s(\varphi)$ fixing the variables of type lpoint, rpoint, weak, derived and res for all blocks in B such that the following holds. For all $b \in B$ at layer i it holds that

(i) if $i < \ell$, then R fixes the derived variable of b to 1 and the resolved variable to $x_{\min\{n,i+1\}}$;

(ii) if $i < n$, then the reservation R fixes the left and right pointers of b to blocks c and c' with $c \neq c'$ in layer $i+1$ and no other block points at blocks c, c';

(iii) if $n \leq i < \ell$, then the reservation R fixes both the left and right pointers to the same block c at layer $i+1$, and no other block points at c;

(iv) if $i = \ell$, then the reservation R fixes the weak pointer of b to some $A \in [m]$.

The block-width of R is $\mathrm{bw}(R) = |B \setminus \{s+1\}|$. A reservation R *reserves* a block $b \in [s+1]$ if either $b \in B$, or there is some $b_0 \in B$ such that R fixes the left or right pointer of b_0 to b.

A *full reservation* (or simply *reservation*, when it is clear from context) for blocks B and ℓ layers is the extension of a basic reservation that additionally contains information about the literals present in the blocks: for every block $b \in B$ at layer $i \leq \ell$, a full reservation fixes the variables $\mathsf{lit}^b_{x_k}$ to determine that b contains exactly $j := \min\{n, i\}$ literals over the variables $x_1, \ldots, x_j$ and no two literals for the same variable.

Observe that a reservation R for B reserves at most $2\,\mathrm{bw}(R) = 2|B \setminus \{s+1\}|$ blocks at any layer (except perhaps at layer 1 where the bound is $2 + \mathrm{bw}(R)$). The high-level strategy of the Delayer is to maintain a (full) reservation R that maps patches of the trivial exponential-size tree-like refutation into s blocks. The block-width of R is always bounded by the block-width of the assignment kept in memory by the Prover

and R thus reserves only few blocks. This enables the Delayer to extend the reservation R upon any query by the Prover in a consistent manner as explained next.

When playing over a grid with $\ell > n$ layers, the strategy for the Delayer when queried about a block at layer $i \in [n-1] \cup \{\ell\}$ is identical to the original strategy. However, if queried about a block at layers n through $\ell - 1$, the Delayer simply answers that the block is a "copy" of a block at the layer above, that is, a trivial weakening without altering the clause. We call this the *extended strategy*.

Recall that our ultimate goal is to extract a satisfying assignment to the CNF formula φ provided a Resolution refutation π of $\text{ERef}_s(\varphi)$. This can be achieved by traversing π according to the Delayer strategy. In the following we thus present the Delayer strategy in the form of an algorithm $\text{Delayer}_\ell(\varphi, \pi, 1^s)$ attempting to extract a satisfying assignment to φ from a refutation π of $\text{ERef}_s(\varphi)$.

Algorithm 3.1: Extended strategy for the delayer ($\text{Delayer}_\ell(\varphi, \pi, 1^s)$)

Let R denote the empty reservation. Traverse the proof dag π starting at the root $C := \bot$ according to the following instructions.

1. If C is a leaf of π, then halt and output failure.
2. If C is obtained by weakening a clause $C' \subseteq C$, then set $C := C'$ and go to Step 4.
3. If C is derived from clauses $A \vee v$ and $B \vee \neg v$ by resolving over v, then attempt the following reservations.

   (a) If v belongs to a block b on layer $0 \leq i < n$, we have two cases.

      i. If b is not reserved in R, update R to encode that b contains the clause $\bigvee_{j=1}^{i} x_j$ and is derived by resolving two unreserved blocks on layer $i+1$.
      ii. If b is reserved in R but has no children attached, then it is pointed to by a block b_0 in layer $i-1$. Let C denote the clause that b should encode: if b_0 points at b on the left and b_0 encodes C_0, then $C := C_0 \vee x_i$ and otherwise, if b_0 points at b on the right, then $C := C_0 \vee \neg x_i$. Update R to encode that b contains the clause C and is derived by resolving two unreserved blocks on layer $i+1$.

   If this update fails because there are not enough unreserved blocks on layer $i+1$, halt and output the clause C.

   (b) If v belongs to a block b on layer $n \leq i < \ell$, we distinguish two cases:

      i. If b is not reserved in R, update R to encode that b contains the clause $\bigvee_{j=1}^{n} x_j$ and fix both of its pointers to the same unreserved block on layer $i+1$.
      ii. If b is reserved in R but has no children attached, then it is pointed to by a block b_0 in layer $i-1$ encoding C_0. If $i > n$, then set $C := C_0$ to be the clause that b encodes, and point both of its pointers to the same unreserved block on layer $i+1$. If $i = n$, then C is obtained by checking whether b_0 points at b on the left or on the right, as in Step 3.(a).ii.

   If this update fails because there are not enough unreserved blocks on layer $i+1$, halt and output the clause C.

   (c) If v belongs to a block b on layer ℓ, we distinguish two cases.

      i. If b is not reserved in R, then update R so that b encodes the clause $\mathcal{A} := \bigvee_{j=1}^{n} x_j$ and it points to an axiom of φ this clause is a weakening of.
      ii. If b is reserved in R but has no children attached, then it is pointed to by a block b_0 in layer $\ell - 1$ encoding C_0. If $\ell > n$, set $\mathcal{A} := C_0$ to be the clause that b encodes, find

the first clause in φ such that $\mathcal{A}$ is a weakening of it and let R point to this clause. If $\ell = n$, then $\mathcal{A}$ is obtained by checking whether b_0 points to b on the left or on the right, as in Step 3.(a).ii.

If this update 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 reservation succeeds, if $R(v) = 1$, then update $C \coloneqq B \vee \neg v$, and otherwise update $C \coloneqq A \vee v$.

4. Clean up: remove any information from R about blocks not mentioned in C. Go to Step 1.

The following invariant is key for the analysis of the algorithm.

Definition 3.2 (Reservation invariant and Delayer-consistency)**.** A full reservation R satisfies the *reservation invariant* with respect to a clause C over the variables of $\text{EREF}_s(\varphi)$ if the following conditions hold:

(i) the reservation R falsifies C;

(ii) the reservation R does not falsify any axiom of $\text{EREF}_s(\varphi)$;

(iii) the reservation R is over blocks mentioned in C, and in particular $\text{bw}(R) = \text{bw}(C)$.

A clause C is *Delayer consistent* if there is a full reservation that satisfies the reservation invariant with respect to C.

The key property of the Delayer strategy is that if the reservation invariant holds at the beginning of an iteration and the algorithm successfully completes one more iteration, then the reservation invariant holds at the end of the iteration.

Lemma 3.3. *Let π be a Resolution refutation of $\text{EREF}_s(\varphi)$, let $\ell \in \mathbb{N}$, and consider a run of $\text{DELAYER}_\ell(\varphi, \pi, 1^s)$. Consider the state of the algorithm at the beginning of iteration i: denote by C_i the clause and by R_i the reservation considered. If R_i witnesses that C_i is Delayer consistent, then R_{i+1} witnesses that C_{i+1} is Delayer consistent, provided the algorithm does not fail in iteration i.*

Proof sketch. We proceed by induction on the number i of iterations of the algorithm. At the first iteration $C_1 = \bot$ and R_1 is the empty reservation, which readily satisfies all the conditions. It is straightforward to verify that if the invariant holds at the beginning of some later iteration and the algorithm does not fail, then the reservation update was successful and the invariant is thus satisfied again. □

Recall also the following connection between failed weakenings and satisfying assignments.

Fact 3.4. *Let φ be a 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.*

We have the following correctness result.

Theorem 3.5 (Assignment extraction)**.** *Let φ be a CNF formula over n variables and m clauses, let $s \in \mathbb{N}$, and consider a Resolution refutation π of the formula $\text{EREF}_s(\varphi)$. For every $\ell \geq n$, it holds that the procedure $\text{DELAYER}_\ell(\varphi, \pi, 1^s)$ in Algorithm 3.1 terminates in time $\text{poly}(|\pi|, s, n, m, \ell)$ and outputs either*

(a) a clause $C \in \pi$ of block-width at least $1/2(\lfloor s/\ell \rfloor - 1)$, or

(b) a satisfying assignment for φ.

Proof. At the beginning of the algorithm, the empty reservation satisfies the reservation invariant with respect to the empty clause. Given a Resolution refutation π we claim that the algorithm always halts before reaching a leaf. Indeed, suppose for the sake of contradiction that the algorithm terminates at a leaf C of π and denote by R the reservation satisfying the reservation invariant with respect to C, obtained by executing the algorithm until reaching C. By item (ii) of the invariant, R does not falsify any axiom of $\text{EREF}_s(\varphi)$. However, according to item (i) of the invariant, R falsifies C, which is a leaf of π and hence a weakening of an axiom of $\text{EREF}_s(\varphi)$. This is a contradiction and establishes the claim.

Thus the algorithm halts before reaching a leaf of π for one of the following two reasons.

(a) The algorithm attempted to reserve the children of a block at level $1 \leq i < \ell$ by finding unreserved blocks on layer $i+1$, but there were no free blocks left on layer $i+1$. This means that R already reserved at least $\lfloor s/\ell \rfloor - 1$ blocks on layer $i+1$, so $\text{bw}(R) \geq 1/2(\lfloor s/\ell \rfloor - 1)$. By item (iii) of the invariant we have that $\text{bw}(R) = \text{bw}(C)$, so putting this together we have that when outputting C we are outputting a clause of block-width at least $1/2(\lfloor s/\ell \rfloor - 1)$, as desired.

(b) The reservation R had a clause $\mathcal{A}$ encoded in a block at layer ℓ, but it failed to find an axiom of φ that $\mathcal{A}$ was a weakening of. Since the block is at layer ℓ, it encodes a width-n clause which is not the weakening of any clause of φ. By Fact 3.4, $\neg\mathcal{A}$ encodes a satisfying assignment of φ, as desired. □

3.2 The new algorithm

We have two parameters s_0 and w_0 to be fixed later. We arrange a numbering of the s blocks of $\text{REF}_s(\varphi)$ into $\ell := n + w_0$ layers, each containing $\lfloor s/\ell \rfloor$ 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. The root block lies at layer 0. Note that the following algorithm gets as input a refutation π of $\text{REF}_s(\varphi)$ and not of $\text{EREF}_s(\varphi)$.

Algorithm 3.2: Efficient assignment extraction ($\text{EXTRACTOR}_{s_0,w_0}(\varphi, \pi, 1^s)$)

If the all-zeroes assignment satisfies φ, output it and halt. If $|\pi| \geq s_0$, then perform a brute-force search to find the lexicographically-first satisfying assignment α of φ. Otherwise, if $|\pi| < s_0$, let $\pi = (C_1, \ldots, C_t)$ and proceed for $i = 1, \ldots, t$ as follows.

1. Let $w := \text{bw}(C_i)$, and denote by $b_1, \ldots, b_w \in [s]$ the non-root blocks mentioned in C_i. If $w > w_0$, move to C_{i+1}; otherwise, continue.
2. Consider every subset $B \subseteq \{b_1, \ldots, b_w\}$. For each such set B, consider, for every $b \in B$, the clauses $C_b := \{z \in \text{Lit}_n \mid \neg\text{lit}_z^b \in C_i \text{ or } \text{lit}_{\neg z}^b \in C_i\}$ and let $\mathcal{A}_B := \bigcup_{b \in B} C_b$. If $\mathcal{A}_B$ is falsifiable and the unique falsifying assignment α_B of $\mathcal{A}_B$ satisfies φ, then extend α_B to an arbitrary full assignment $\alpha \supseteq \alpha_B$ and output α.

3.2.1 Correctness

Before we can argue correctness of Algorithm 3.2, recall the restriction argument from [AM20]. This is the random restriction argument that reduces the block-width of a Resolution refutation π to $O(\log |\pi|)$. We restate it here highlighting the fact that the restriction ρ does not only *reduce* the block-width, but actually *kills* all clauses of high block-width. Let us 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 enabled, (ii) every variable

belonging to a disabled block is assigned a value, and (iii) apart from the enabling variables, no variables belonging to enabled blocks are assigned.

Lemma 3.6 (Random restriction argument [AM20; AAdRK25, Lemma 4.2]). *There exists $\delta \geq 1$ such that for every Resolution refutation π of the formula $\textsc{Ref}_s(\varphi)$, there exists an $\lceil s/2 \rceil$-disabling restriction $\rho \in \{0,1,*\}^N$ to the N variables of $\textsc{Ref}_s(\varphi)$ such that*

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

(ii) for every clause $C \in \pi$, if $\mathrm{bw}(C) > \delta \log |\pi|$, then $C_{\restriction \rho} = 1$.

Proof. Without loss of generality, let s be even, since if s is odd, then π can be turned into a refutation of $\textsc{Ref}_{s-1}(\varphi)$ by the restriction that disables and fully restricts one block. This reduces the block-width of π by at most 1.

Consider the following distribution of random restrictions: pair all s non-root blocks into $s/2$ pairs, and for each pair, with probability $1/2$, decide which block in the pair is going to be disabled. The root block indexed by $s+1$ is always enabled. Now, if a block is disabled, choose a value uniformly and set all of the remaining variables in the block to take this value. We denote by ρ a restriction obtained in this way.

It is clear that a restriction ρ sampled in this way is exactly $\lceil s/2 \rceil$-disabling and does not falsify $\textsc{Ref}_s(\varphi)$. We claim that for every Resolution derivation π of $\textsc{Ref}_s(\varphi)$, there exists a restriction that satisfies all clauses of block-width at least $\delta \log |\pi|$. Indeed, if ℓ is a literal corresponding to the variable enable_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,

$$\Pr_\rho[B_i \text{ is disabled and } \ell_{\restriction \rho} = 1\ ] = \Pr_\rho[B_i \text{ is disabled}] \cdot \Pr_\rho[\ell_{\restriction \rho} = 1 \mid B_i \text{ is disabled}] \tag{3.1}$$

$$= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}\,. \tag{3.2}$$

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

Now, if C is a clause such that $\mathrm{bw}(C) \geq w$, hence mentioning at least w non-root blocks, we have that

$$\Pr_\rho[C_{\restriction \rho} \neq 1] \leq (3/4)^{w/2}, \tag{3.3}$$

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,

$$\Pr_\rho[\text{there is } C \in \pi \text{ where } \mathrm{bw}(C) \geq w \text{ and } C_{\restriction \rho} \neq 1] \leq \mathrm{length}(\pi) \cdot (3/4)^{w/2} \leq |\pi| \cdot (3/4)^{w/2}. \tag{3.4}$$

We want to choose w such that $|\pi| \cdot (3/4)^{w/2} < 1$. This bound is met by choosing $w > 2 \log |\pi| / (\log 4/3)$, which yields the value of the constant $\delta := 2/\log(4/3) \geq 1$ in the statement. □

With the restriction argument at hand we are now ready to argue correctness of Algorithm 3.2. We prove its correctness by showing that if the new algorithm fails, then the Delayer is able to play forever following the strategy outlined in Algorithm 3.1. This contradicts the fact that the algorithm always terminates.

Theorem 3.7 (Correctness of Algorithm 3.2). *There exist $s_0 = s_0(s)$, $w_0 = w_0(\pi)$, and $n_0 \in \mathbb{N}$ such that for every satisfiable CNF formula φ over $n \geq n_0$ variables, $s \geq n^2$, and every Resolution refutation π of $\textsc{Ref}_s(\varphi)$, the procedure $\textsc{Extractor}_{s_0,w_0}(\varphi, \pi, 1^s)$ in Algorithm 3.2 outputs a satisfying assignment of φ.*

Proof. Fix $w_0 = w_0(\pi) := \lceil \delta \log |\pi| \rceil$, where δ is the constant given by Lemma 3.6. Fix $\varepsilon := 1/8\delta$ and $s_0 = s_0(s) := 2^{\varepsilon\sqrt{s}} \geq 2^{\varepsilon n}$. In the case of the brute-force search at the beginning of Algorithm 3.2, correctness is immediate, so we focus on the case $|\pi| < s_0$.

Suppose for contradiction that there exist π and φ such that Algorithm 3.2 iterates through all the clauses of π without ever finding a satisfying assignment. In the following we show that this implies that the original algorithm (Algorithm 3.1) does not always terminate thereby contradicting Theorem 3.5.

Let ρ denote the restriction as guaranteed to exist by Lemma 3.6 and consider $\pi_{\restriction\rho}$. By Lemma 3.6 it holds that $\mathrm{bw}(\pi_{\restriction\rho}) \leq \delta \log |\pi| \leq w_0$. Fix $\ell := n + 4w_0$. Since $|\pi| < s_0$ and $\varepsilon = 1/8\delta$ it holds that $\delta \log |\pi| < \delta \log s_0 = \delta\varepsilon\sqrt{s} = \frac{\sqrt{s}}{8}$, and therefore $w_0 < \frac{\sqrt{s}}{8} + 1$. Since $s \geq n^2$, we have $n \leq \sqrt{s}$, and hence

$$\ell < \sqrt{s} + 4\left(\frac{\sqrt{s}}{8} + 1\right) = \frac{3}{2}\sqrt{s} + 4 \leq \frac{5}{3}\sqrt{s}, \tag{3.5}$$

where the last inequality holds for sufficiently large s.

Up to a renaming of blocks, $\pi_{\restriction\rho}$ is a Resolution refutation of $\mathrm{ER}\textsc{ef}_{\lfloor s/2 \rfloor}(\varphi)$. We claim that the block-width $\mathrm{bw}(\pi_{\restriction\rho}) < 1/2(\lfloor s/\ell \rfloor - 1)$, that is, the block-width of $\pi_{\restriction\rho}$ is less than the block-width guaranteed in case (a) of Theorem 3.5. Indeed,

$$\frac{1}{2}\left(\left\lfloor \frac{\lfloor s/2 \rfloor}{\ell} \right\rfloor - 1\right) \geq \frac{1}{2}\left(\frac{s/2 - 1}{(5/3)\sqrt{s}} - 2\right) \tag{3.6}$$

$$= \frac{3\sqrt{s}}{20} - 1 - \frac{3}{10\sqrt{s}} \tag{3.7}$$

$$> \frac{\sqrt{s}}{8} \tag{3.8}$$

$$> \delta \log |\pi| \geq \mathrm{bw}(\pi_{\restriction\rho}), \tag{3.9}$$

where eq. (3.8) holds for sufficiently large s.

Suppose we run the original algorithm (Algorithm 3.1) on $\pi_{\restriction\rho}$. By eq. (3.6), we are guaranteed to end up in case (b) of Theorem 3.5; the algorithm outputs a satisfying assignment.

According to the proof of Theorem 3.5 it is even possible to characterize the state in which Algorithm 3.1 finds the satisfying assignment.

First, suppose that the Delayer fails in Step 3(c).i of Algorithm 3.1, when attempting to reserve a previously unreserved block in the final layer. In this case, the clause assigned to that block is $A = x_1 \vee \cdots \vee x_n$. Since the update fails, A is not a weakening of any axiom of φ. By Fact 3.4, the all-zero assignment satisfies φ. But Algorithm 3.2 explicitly checks this assignment, contradicting our assumption. Hence we may assume that the Delayer fails in Step 3(c).ii.

Denote by C the clause and by R the reservation Algorithm 3.1 considers right before outputting the satisfying assignment. Note that R satisfies the reservation invariant. The clause C mentions some block b at layer $\ell - 1$ and R encodes a width-n clause $\mathcal{B}$ appearing in this block. Furthermore, R has a block b' reserved at layer ℓ weakened from b but not mentioned in C. The algorithm attempts to extend the reservation to block b' but fails in the process. In a bit more detail, since b' is a trivial weakening of b it must also encode the width-n clause $\mathcal{B}$. In the process of extending the reservation R the algorithm attempts to find a clause A of φ that $\mathcal{B}$ is a weakening of, but fails.

Let us use the above observation of how the Delayer conceded to argue correctness of the new extraction algorithm. Let B be the set of blocks in the connected component of b in the reservation R. We claim that B consists of blocks on distinct levels ending on level $\ell - 1$ with the block b (see Figure 2). Indeed, since by item (iii) of the reservation invariant it holds that $\mathrm{bw}(R) = \mathrm{bw}(C)$, and we have that $\mathrm{bw}(C) \leq w_0$ and $\ell = n + 4w_0$, it follows that the chain starting at b is of length at most w_0. This implies in particular that B is not connected to level n; the claim follows.

Consider the clause $C' \in \pi$ such that $C = C'_{\restriction\rho}$. Since C survived the restriction it holds that $\mathrm{bw}(C') \leq w_0$.

This implies that the clause C' is evaluated by the "new" Algorithm 3.2 and the set of blocks B is considered. As we assumed that Algorithm 3.2 does not output a satisfying assignment it holds that the partial assignment α_B does not satisfy φ. Hence there exists a clause A of φ that is not satisfied by α_B. This clause A is of the form $A = A_0 \vee A_1$ with $A_0 \subseteq \mathcal{A}_B$ and $\mathcal{A}_B \cap A_1 = \emptyset$.

Consider the reservation R' that modifies R on B as follows: the topology between the blocks of B is left intact but every block in B now contains the literals $\mathcal{A}_B \cup A_1$ along with positive literals x_j for every variable x_j not mentioned in $\mathcal{A}_B \cup A_1$. By construction the reservation R' satisfies the reservation invariant with respect to C, but the Delayer no longer concedes in this round: when queried about block b' the Delayer answers that it is a weakening of A. Algorithm 3.1 thus does not fail in this iteration. This shows that Algorithm 3.1 never terminates thereby contradicting Theorem 3.5. The statement follows. □

3.2.2 Complexity of the algorithm

Theorem 3.7 shows that the new algorithm is correct, but it says nothing about its complexity. It is not hard to see that the procedure can be implemented in logarithmic space (**FL**), but we can do better. In the following we show that $\textsc{Extraction}_{\text{Res}}[n^2]$ lies in **P**-uniform $\mathbf{FTC}^0$. This puts $\textsc{Extraction}_{\text{Res}}[n^2]$ in **FP**, whereas before it was only known to be in **FZPP** ($\textsc{Extraction}_{\text{Res}}[n^3]$ was shown to be in **FP**.)

Recall that **P**-uniform $\mathbf{FTC}^0$ is the class of relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ for which there exists a Turing machine D and a constant $d \in \mathbb{N}$ such that on input 1^n, the machine $D(1^n)$ runs in time $\text{poly}(n)$ and outputs the description of a multi-output circuit C_n of depth d, unbounded fan-in and threshold gates, such that for every $x \in \{0,1\}^n$, it holds that $(x, y) \in R$ for some $y \in \{0,1\}^*$ if and only if $(x, C_n(x)) \in R$.

Theorem 3.8 ($\textsc{Extraction}_{\text{Res}}$ is $\mathbf{FTC}^0$)**.** *The search version of the Proof Analysis Problem for Resolution is solvable by* **P***-uniform multi-output circuits of polynomial size and constant depth with threshold gates whenever $s \geq n^2$. That is, it holds that* $\textsc{Extraction}_{\text{Res}}[n^2] \in$ **P***-uniform* $\mathbf{FTC}^0$.

Proof. For the sake of precision, let us fix the following natural binary encoding for $\textsc{Extraction}_{\text{Res}}$. 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 $\textsc{Ref}_s(\varphi)$ 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 $\textsc{Ref}_t(\textsc{Ref}_s(\varphi))$, 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, 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 $N = N(n, m, t, s)$. 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 **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 multi-output Boolean circuit $D_{n,m,t,s}$, where the i-th bit $D^i_{n,m,t,s}$ is defined as follows.

(a) If the interpretations of the lengths are 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-0 circuit.

(b) If the interpretation of the lengths is valid and $|\pi| \geq 2^{\varepsilon \cdot n}$, then it simply outputs the conjunction of the following two circuits. First, the CNF that checks that π is a correct Resolution refutation of $\textsc{Ref}_s(\varphi)$ in at most t clauses. Second, the standard constant-depth circuit that outputs the lexicographically-first satisfying assignment for φ. This circuit is built as follows. For every assignment $\alpha \in \{0,1\}^n$, take the CNF formula $\textsc{Sat}(\varphi, \alpha)$ and let S_α be the output of this circuit. Then, define for every $\alpha \in \{0,1\}^n$ the circuit

$$L_\alpha \coloneqq S_\alpha \wedge \bigwedge_{\beta \prec \alpha} \neg S_\beta, \tag{3.10}$$

where $\prec$ denotes the lexicographic order on $\{0,1\}^n$. The bit L_α is 1 if, and only if, α happens to be the lexicographically-first satisfying assignment of φ. Now, to output the i-th bit of α, it suffices to output the bit

$$A_i \coloneqq \bigvee_{\substack{\alpha \in \{0,1\}^n \\ \alpha_i = 1}} L_\alpha. \tag{3.11}$$

Here $D^i_{n,m,t,s} \coloneqq A_i$. This entire circuit has depth 4 and size $\mathrm{poly}(n,m) \cdot 2^{O(n)}$.

(c) If the interpretation of lengths is consistent and $|\pi| < 2^{\varepsilon \cdot n}$, we construct the following circuit, which essentially implements Algorithm 3.2. Let $w \coloneqq \lceil \delta \log |\pi| \rceil$ and fix a clause $C \in \pi$. For every $b \in [s]$, define

$$\mathrm{Present}^C_b \coloneqq \bigvee_{z \in \mathrm{Lit}(b)} \mathsf{lit}^C_z, \tag{3.12}$$

which checks whether block b is mentioned in C.

Next, suppose $\beta \in \{0,1\}^w$ and $b \in [s]$, and define

$$B^C_b(\beta) \coloneqq \mathrm{Present}^C_b \wedge \bigvee_{k \in [w]} \left(\#_k(\mathrm{Present}^C_1, \ldots, \mathrm{Present}^C_b) \wedge \beta_k \right), \tag{3.13}$$

where $\#_k(p_1, \ldots, p_n) \coloneqq \mathsf{Th}_k(p_1, \ldots, p_n) \wedge \neg \mathsf{Th}_{k+1}(p_1, \ldots, p_n)$ tests if the Hamming weight of the string consisting of bits $p_1, \ldots, p_n$ is equal to k. The formula $B^C_b(\beta)$ checks whether b belongs to the subset of blocks selected by β. More precisely, if $b_1 < \cdots < b_r$ are the blocks mentioned in C, where $r \leq w$, then $B^C_{b_k}(\beta) = \beta_k$ for every $k \in [r]$, and $B^C_b(\beta) = 0$ for all other blocks. Thus, by ranging over all $\beta \in \{0,1\}^w$, we enumerate all subsets of the blocks mentioned in C.

We can then obtain the i-th bit of the candidate assignment the algorithm obtains when inspecting clause C as

$$\alpha^C_{\beta,i} \coloneqq \bigvee_{b \in [s]} \left(B^C_b(\beta) \wedge \left(\mathsf{lit}^C_{\neg \mathsf{lit}^b_{\neg x_i}} \vee \mathsf{lit}^C_{\mathsf{lit}^b_{x_i}} \right) \right). \tag{3.14}$$

We now want to obtain the first $C \in \pi$ and the first $\beta \in \{0,1\}^w$ such that the assignment $\alpha^C_\beta \coloneqq (\alpha^C_{\beta,1}, \ldots, \alpha^C_{\beta,n})$ satisfies φ. For this, take

$$L_{\alpha^C_\beta} \coloneqq \textsc{Sat}(\varphi, \alpha^C_\beta) \wedge \bigwedge_{(C',\beta') \prec (C,\beta)} \neg \textsc{Sat}(\varphi, \alpha^{C'}_{\beta'}). \tag{3.15}$$

Analogously to (3.10), $L_{\alpha^C_\beta}$ is 1 if and only if α^C_β is the first candidate assignment to be satisfying.

Now, to recover every bit of the assignment, we simply output

$$A_i \coloneqq \bigvee_{C \in [\pi]} \bigvee_{\beta \in \{0,1\}^w} L_{\alpha^C_\beta} \wedge \alpha^C_{\beta,i}. \tag{3.16}$$

It is easy to verify that if Algorithm 3.2 found a satisfying assignment when looking at clause C and subset B of the blocks mentioned in C, then that same assignment will be considered by the construction above.

Finally, for each possible interpretation (n, m, t, s) of the lengths of the input, we obtain a circuit $D_{n,m,t,s}$. For each interpretation of the lengths there is also a circuit $\text{Correct}_{n,m,t,s}$ that verifies that the input correctly encodes an $\text{Extraction}_{\text{Res}}$ 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. We can once again repeat the construction in (b) to output the lexicographically-first assignment that passes the correctness check.

The circuit above behaves like Algorithm 3.2. Choosing the same parameters s_0 and w_0 as in Theorem 3.7, we are guaranteed to output a correct satisfying assignment when one exists. It is also clear by construction that the circuit has constant depth. It is only left to argue that the size is polynomial in the size of the input. Indeed, choosing the s_0 and w_0 as in Theorem 3.7 the construction in (b) has size $\text{poly}(n, m) \cdot 2^{O(n)}$, but since $|\pi| \geq 2^{\varepsilon \cdot n}$, we have $|\pi| \geq 2^{O(n)}$ and the overall size is $\text{poly}(n, m, |\pi|, s)$, as desired.

The construction in part (c) also has constant depth. The iteration over all possible subsets is over $2^{w_0} = 2^{\lceil \delta \log |\pi| \rceil} = \text{poly}(|\pi|)$ subsets, so it is polynomial in size. □

In the absence of threshold gates, we can still give a non-trivial circuit, at the cost of a quasipolynomial increase in size.

Theorem 3.9 ($\text{Extraction}_{\text{Res}}$ in quasipolynomial $\mathbf{FAC}^0$)**.** *The search version of the Proof Analysis Problem for Resolution is solvable by* **QP***-uniform multi-output circuits of quasipolynomial size and constant depth whenever* $s \geq n^2$.

Proof. The construction is almost identical to the one above in Theorem 3.8. The main obstacle is that in the absence of threshold gates, we cannot compute Hamming weight as in eq. (3.13). We can side-step this by considering instead all possible subsets $B \subseteq [s]$ containing exactly $w \coloneqq \lceil \delta \log |\pi| \rceil$ elements. The descriptor machine writes each such B as an indicator vector of size s, and for each $b \in [s]$, we define $B^C_b \coloneqq B_b \wedge \text{Present}^C_b$, which gives us the exact blocks that are present in both B and C. The rest of the construction proceeds as before.

This change means the big disjunctions over all $\beta \in \{0,1\}^w$ will be replaced by big disjunctions over all possible subsets $B \subseteq [s]$ containing exactly w elements. This gives us a blow-up of

$$|\pi| \cdot \binom{s}{w} = |\pi| \cdot \binom{s}{\lceil \delta \log |\pi| \rceil} \leq |\pi|^{O(1)} \cdot s^{O(\log |\pi|)}, \tag{3.17}$$

which is quasipolynomial in the size of the input. □

4 The restriction argument in $\text{APC}_2(\alpha)$

In this section we show that the random restriction argument for Ref formulas (Lemma 3.6) can be formalized in the relativized theory $\text{APC}_2(\alpha)$. In Section 4.1 we show that PV_1 (unrelativized) can prove the existence of suitable combinatorial designs giving us k-wise independent distributions with the right parameters. In Section 4.2 we employ Jeřábek's APC_1 framework (again, unrelativized) to show that for clauses of block-width at least k, we can bound the number of "bad seeds"—those for which the k-wise independent

distribution produces a restriction that does not kill the clause. Section 4.3 presents suitable relativized formulas to check that a certain clause has block-width at least k. Finally, Section 4.4 shows how we can formalize a union bound argument implying that there exists at least one "good seed" producing a restriction that kills all wide clauses in a refutation. In this latter proof we encode Resolution refutations by a new relation symbol. This final proof can be carried out in $\mathrm{APC}_2(\alpha)$.

4.1 Construction of designs

The goal is to argue in PV_1 that we can construct a combinatorial design yielding a k-wise independent distribution for suitable values of k. Our construction of designs will be the first construction of combinatorial designs given by Nisan and Wigderson [NW94, Lemma 2.5]. (Jeřábek [Jeř07, Lemma 2.4] formalized the second, greedy construction [NW94, Lemma 2.6], used to get the parameters needed for derandomizing **BPP**, but the greedy algorithm is sequential and not feasible in these theories.)

Definition 4.1 (Combinatorial designs)**.** Let $n, m, p, w \in \mathbb{N}$ with $w \le p \le m$. An (n, m, p, w)-*design* is a Boolean matrix of size $n \times m$ such that every row has exactly p ones and every two distinct rows intersect in at most w positions.

Let us construct such designs with the right parameters for our applications.

Lemma 4.2 (Existence of designs, in PV_1)**.** *There is a* PV*-function* Design *such that for every* n *and every* $2^k \in \mathrm{Log}$ *satisfying* $|n| \le 2k(2^k+1)$*, and every* $i \in [n]$*, the function* $\mathrm{Design}(n, 2^k, i)$ *outputs the* i*-th row of a fixed matrix* M*, and* PV_1 *proves that the rows given by this function form an* $(n, 2^{4k}, 2^{2k}, 2^k)$*-design. Furthermore, there is a* PV*-function that, for every set* $I \subseteq [n]$ *of indices of size* $|I| \le 2^k$*, outputs the entire submatrix* M_I *of* M *restricted to the rows indexed by* I*, and* PV_1 *proves that* $\mathrm{rank}_{\mathbb{F}_2}(M_I) = |I|$.

Proof. We fix the field $\mathbb{F} := \mathrm{GF}(2^{2k})$, where elements are binary strings of length $2k$. Since $2^k \in \mathrm{Log}$, we can operate over $\mathbb{F}$ within PV_1 (see Section 2.2.3 for details about field arithmetic over PV_1). We describe the matrix M globally; it will be clear from the construction that, given the index i of a row, its i-th row can be constructed by a PV-function.

Given $i \in [n]$, consider its binary representation. Since $|i| \le |n| \le 2k(2^k+1)$, we may pad it with leading zeroes until it has length exactly $(2^k+1)2k$. Split the resulting string into 2^k+1 segments of $2k$ bits and interpret these segments as the coefficients of a polynomial $P_i \in \mathbb{F}[X]$ of degree at most 2^k. Notice that if $i \neq i'$, then $P_i \neq P_{i'}$, since their coefficient vectors are distinct.

A position in the i-th row of M is indexed by a pair $(a, b) \in \mathbb{F}^2$, and we define $M[i, (a, b)] = 1$ if and only if $P_i(a) = b$. Thus M has dimension $n \times 2^{4k}$. Every row contains exactly 2^{2k} ones, since for every $a \in \mathbb{F}$ there is exactly one $b \in \mathbb{F}$ such that $P_i(a) = b$.

Consider now two distinct rows $i \neq i'$. A coordinate (a, b) is contained in both rows only if $P_i(a) = P_{i'}(a) = b$. Hence a is a root of the non-zero polynomial $P_i - P_{i'}$, whose degree is at most 2^k. By Lemma 2.3, available in PV_1, there are at most 2^k such values of a. Thus every two distinct rows intersect in at most 2^k positions, and M is an $(n, 2^{4k}, 2^{2k}, 2^k)$-design.

Finally, let $I \subseteq [n]$ have size $r := |I| \le 2^k$. Since $2^k \in \mathrm{Log}$, the submatrix M_I can be encoded as a number in PV_1. Suppose for contradiction that $\mathrm{rank}_{\mathbb{F}_2}(M_I) < r$. Then some row $i \in I$ is an $\mathbb{F}_2$-linear combination of the remaining rows in $I \setminus \{i\}$. Consequently, every coordinate at which row i is 1 must also be 1 in at least one of those remaining rows. The support of row i is therefore contained in the union, over $i' \in I \setminus \{i\}$, of the intersections of the supports of rows i and i'. Every such intersection has size at most 2^k, and there are at most $r - 1 \le 2^k - 1$ of them. Hence this union has size at most $2^k(r-1) \le 2^k(2^k-1) < 2^{2k}$, contradicting the fact that row i has exactly 2^{2k} ones. Thus $\mathrm{rank}_{\mathbb{F}_2}(M_I) = r$. All the sets counted above are Log-sized, so these counting arguments and the construction of M_I can be carried out in PV_1. □

4.2 Bad restrictions from seeds

Our goal is to sample a restriction that kills clauses of high block-width. We use the rows of a combinatorial design as a linear map from short seeds to the bits describing such restrictions. Recall that we are working with the $\textsc{Ref}_s(\varphi)$ formula and the restrictions we are interested in are the so-called *disabling restrictions* from Section 3.2.1. Moreover, we want to replicate the pairing argument in Lemma 3.6: we pair the s blocks into $s/2$ pairs, and we enable and disable exactly one block in each pair. Then, the variables of the disabled blocks are all set to the same value.

Let us first make precise the way of pairing blocks.

Definition 4.3 (Pair-index and pair-width)**.** Assume s is even. For every $j \in [s/2]$, we pair blocks $2j-1$ and $2j$ and define its *pair-index* to be $\mathrm{pair}(2j-1) = \mathrm{pair}(2j) := j$. We say that the block $2j-1$ is in position 0 and the block $2j$ in position 1, and we define $\mathrm{pairpos}(2j-1) := 0$, and $\mathrm{pairpos}(2j) := 1$. For a variable of $\textsc{Ref}_s(\varphi)$, its pair-index is the pair-index of the block to which the variable belongs. We denote by $\mathrm{pw}(C)$ the *pair-width* of a clause C, namely, the number of distinct pair-indices mentioned in C.

The restrictions we deal with are defined by strings of length s, where the first $s/2$ bits select the blocks to be disabled in each pair, and the last $s/2$ bits set the value of variables in the disabled blocks.

More precisely, for a pair-index $j \in [s/2]$, a choice bit $\alpha_j = 0$ means that the block in position 0 is disabled, while $\alpha_j = 1$ means that the block in position 1 is disabled. The bit β_j gives the common value assigned to all variables belonging to the disabled block.

Definition 4.4 (Restrictions from seeds)**.** Let $s, n, m \in \mathbb{N}$ be the parameters of the formula $\textsc{Ref}_s(\varphi)$, with s even. Let M be an $s \times \ell$ Boolean matrix. Suppose that a clause C has width w and pair-width w, and let its pair-indices be $\{i_1, \ldots, i_w\} \subseteq [s/2]$. Define $I(C) := \{i_1, \ldots, i_w, i_1 + s/2, \ldots, i_w + s/2\} \subseteq [s]$.

We interpret a string $\mu = \alpha_1 \cdots \alpha_w \beta_1 \cdots \beta_w \in \{0,1\}^{2w}$ as the restriction on these w pairs in which α_j determines which block of pair-index i_j is disabled and β_j gives the value assigned to all variables in that disabled block. Given a seed $x \in \{0,1\}^\ell$, we write $\rho_x^{I(C)} := M_{I(C)}x \in \{0,1\}^{2w}$ for the restriction on the pairs occurring in C generated by x.

For every clause C, we are interested in bounding the number of "bad seeds", those leading to restrictions that do not kill the clause.

Definition 4.5 (Bad seeds)**.** Let M and C be as in Definition 4.4. We denote by $\textsc{Bad}_C^M$ the set

$$\textsc{Bad}_C^M := \left\{ x \in \{0,1\}^\ell \;\middle|\; C\restriction_{\rho_x^{I(C)}} \neq 1 \right\}$$

of *bad seeds* for the clause C.

Observe that for clauses C of both width w and pair-width w, and assuming $w \in \mathrm{Log}$, the clause can be represented as a string of length $O(w(\log s + \log n + \log m))$ and hence encoded as a number. Then, given such a clause C and a seed $x \in \{0,1\}^\ell$, there is a PV-symbol that checks whether $x \in \textsc{Bad}_C^M$.

We are now ready to bound the number of bad seeds. The only property of the design needed in the following lemma is that the $2w$ rows indexed by $I(C)$ are linearly independent.

Lemma 4.6 (Restrictions at clause-level, in APC_1)**.** *The theory* APC_1 *proves the following statement. Let* $s, n, m \in \mathbb{N}$*, let* $w, \ell \in \mathrm{Log}$*, and let* C *encode a clause containing exactly* w *literals over* w *distinct pair-indices of the variables of* $\textsc{Ref}_s(\varphi)$*. Let* M *be an* $s \times \ell$ *Boolean matrix such that* $\mathrm{rank}_{\mathbb{F}_2} M_{I(C)} = 2w$*. Provably in* APC_1*,*

$$\textsc{Bad}_C^M \preceq_0 3^w 2^{\ell - 2w} = \left(\frac{3}{4}\right)^w 2^\ell.$$

Proof. Write the literals of C as $\lambda_1, \dots, \lambda_w$, with λ_j belonging to the pair of pair-index i_j. For every $j \in [w]$, let $P_j \subseteq \{0,1\}^2$ be the set of pairs (α, β) for which the restriction of pair-index i_j determined by the choice bit α and value bit β does not satisfy λ_j.

If λ_j is an enable-literal, precisely two of the four patterns are bad: the value bit β is irrelevant, and the choice bit must make the enable-literal false. If λ_j is any other literal, precisely three patterns are bad: two patterns leave its block enabled and hence the literal unset, while among the two patterns disabling its block exactly one assigns the literal the falsifying value. Thus $|P_j| \leq 3$ for every $j \in [w]$.

For each j, fix a PV-computable surjection $h_j : \{0,1,2\} \twoheadrightarrow P_j$, repeating one of the two patterns when $|P_j| = 2$. We construct a surjection

$$G : \{0,1,2\}^w \times \{0,1\}^{\ell - 2w} \twoheadrightarrow \textsc{Bad}_C^M. \tag{4.1}$$

Given $(z_1, \dots, z_w, \kappa)$ in the domain, write $h_j(z_j) = (\alpha_j, \beta_j)$ and let $\mu \coloneqq \alpha_1 \cdots \alpha_w \beta_1 \cdots \beta_w \in \{0,1\}^{2w}$.

Since $\operatorname{rank} M_{I(C)} = 2w$, the linear map $M_{I(C)} : \mathbb{F}_2^\ell \to \mathbb{F}_2^{2w}$ is surjective. Using Gaussian elimination in PV_1 (Lemma 2.4), let $\nu^\star = \mathrm{Sol}(M_{I(C)}, \mu)$ be a fixed solution of $M_{I(C)}\nu = \mu$, and let K be a matrix whose columns form a basis of $\ker M_{I(C)}$. Since the rank is $2w$, this kernel has dimension $\ell - 2w$. Define $G(z_1, \dots, z_w, \kappa) \coloneqq \nu^\star + K\kappa$.

By construction, $M_{I(C)} G(z_1, \dots, z_w, \kappa) = \mu$, and the local pattern (α_j, β_j) lies in P_j for every j. Hence none of the literals λ_j is satisfied, so the output seed belongs to $\textsc{Bad}_C^M$.

It remains to prove surjectivity. Let $x \in \textsc{Bad}_C^M$, and write $\mu = M_{I(C)} x$. Since x is bad, the two bits of μ corresponding to each pair i_j form a pattern in P_j. Choose $z_j \in \{0,1,2\}$ mapping to that pattern under h_j. Let $\nu^\star$ be the canonical solution associated with this μ. Then $x - \nu^\star \in \ker M_{I(C)}$, so there is a coefficient vector $\kappa \in \{0,1\}^{\ell - 2w}$ such that $x - \nu^\star = K\kappa$. Hence $G(z_1, \dots, z_w, \kappa) = x$.

The domain has cardinality $3^w 2^{\ell - 2w} = (3/4)^w 2^\ell$, which proves the claimed $\preceq_0$ bound. □

4.3 The block-width formulas

Until now, all arguments were in the unrelativized theories. The clauses we worked with so far had pair-width w as well as width w, for $w \in \mathrm{Log}$, and hence they could be encoded as numbers, since these are strings of length $O(w(\log s + \log n + \log m))$. In general, however, we will deal with clauses encoded as strings given by an external predicate Π (see Section 2.3 for details on how refutations are encoded in such a predicate). We will generally assume that if the refutation has t clauses, we can talk about any particular clause D_i for $i \in [t]$; this is just a figure of speech to refer to the substring defined by Π containing the information about clause D_i. In this section we define formulas checking that such a clause D_i has high enough block-width.

Observe first that given numbers s, n and m, we can determine the number of variables of the formula $\textsc{Ref}_s(\varphi)$, and there is a PV-function $\mathrm{Lit}(b, n, m, s)$ that given a block $b \in [s+1]$ indicates which are the numbers encoding the literals of $\textsc{Ref}_s(\varphi)$ belonging to block b. Note that the specifics of φ are not needed here, only the parameters s, n and m. We will abuse notation and use Lit also to refer to the set of numbers itself, although this PV function only outputs an upper and a lower bound on the numbers we have to check.

We first define a formula to check whether a certain block $b \in [s+1]$ is present in the clause D_i of Π. This is the $\Sigma_1^b(\Pi)$-formula

$$\mathrm{Present}(i, b, n, m, s) \coloneqq \exists \ell \in \mathrm{Lit}(b, n, m, s)\ \Pi(\mathrm{lit}_\ell^i). \tag{Present}$$

Next, for $w \in \mathrm{Log}$, we define the $\Sigma_1^b(\Pi)$-formula checking that the clause D_i has block-width at least w:

$$\mathrm{BW}(i, n, m, s, w) \coloneqq \exists \beta \in [s]^w \left(\forall j \in [w]\ \mathrm{Present}(i, (\beta)_j, n, m, s) \wedge \forall j < j' \in [w]\ (\beta)_j \neq (\beta)_{j'}\right). \tag{BW}$$

Here β represents a sequence of w distinct blocks present in D_i, and $(\beta)_j$ denotes its j-th element. Such a sequence is encoded using $O(w \log s)$ bits. Note that we deliberately do not count the root block towards the block width. Since $w \in \mathrm{Log}$, the quantifiers over $j, j' \in [w]$ are sharply bounded, and hence the entire formula is $\Sigma^b_1(\Pi)$.

Remark 4.7. The formula (BW) above is $\Sigma^b_1(\Pi)$, and this existential quantifier will force us to state the final restriction argument in $\mathrm{APC}_2(\alpha)$ rather than $\mathrm{APC}_1(\alpha)$. In general, as long as we use the existing encoding of Resolution refutations in Π, we cannot go below $\Sigma^b_1(\Pi)$. This is because a $\Delta^b_0(\Pi)$ formula would amount to a $\mathrm{PV}(\Pi)$-function, which corresponds to a decision tree of quasipolynomial size and polylogarithmic depth. However, it is easy to see that any decision tree deciding whether a clause of Π has block-width at least w must require linear depth. ◀

4.4 Existence of good seeds

It now remains to conclude that there exists a "good" seed that generates a restriction satisfying all wide clauses. The main technical obstacle is that the sets $\textsc{Bad}^M_C$ above were defined only for clauses C that can themselves be encoded as numbers. In the general setting, however, a Resolution refutation is encoded by the auxiliary predicate Π, and a clause D_i in the refutation may be too large to be encoded as a number.

If D_i has block-width at least w, then it mentions at least $w/2$ distinct pair-indices. We can therefore choose a subclause $C_i \subseteq D_i$ containing exactly one literal from each of $w/2$ distinct pair-indices. Thus C_i has both standard width and pair-width exactly $w/2$ and, since $w \in \mathrm{Log}$, it can be encoded as a number. Killing C_i also kills D_i. The choice of C_i will be implemented by a $\mathrm{PV}_2(\Pi)$-function. This is the point at which we need to work in $\mathrm{APC}_2(\alpha)$ rather than $\mathrm{APC}_1(\alpha)$.

Before we proceed, let us define what it means for a seed to be "good" for a clause encoded by a string. For $s \in \mathbb{N}$ and $w \in \mathrm{Log}$, let $q = q(s,w) := 2^{\lceil \log(\max\{2,w,|s|\})\rceil}$. Thus q is a power of two, $q \geq w$, $q \geq |s|$, and $q < 2 \cdot \max\{2, w, |s|\}$. In particular, $q \in \mathrm{Log}$ whenever $w \in \mathrm{Log}$. Writing $q = 2^k$, we have $|s| \leq q \leq 2k(q+1)$, so the hypothesis of Lemma 4.2 holds when the number of rows is s.

Definition 4.8 (Good seeds)**.** Assume s is even and let $q = q(s,w)$ be as above. For a seed $\sigma \in \{0,1\}^{q^4}$, write $\rho^\sigma_r := \mathrm{Design}(s,q,r) \cdot \sigma^\top$ for $r \in [s]$. Thus $\rho^\sigma_1, \ldots, \rho^\sigma_{s/2}$ are the choice bits and $\rho^\sigma_{s/2+1}, \ldots, \rho^\sigma_s$ are the corresponding value bits.

We denote by $\mathrm{Good}(s,n,m,w,\sigma,i)$ the $\Delta^b_2(\Pi)$-formula expressing the following: if the i-th clause D_i encoded by Π has block-width at least w, then there is a literal ℓ present in D_i, belonging to some block $j \in [s]$, such that one of the following holds:

(a) $\ell = \mathrm{enable}^j$, $\mathrm{pairpos}(j) = 0$, and $\rho^\sigma_{\mathrm{pair}(j)} = 1$;

(b) $\ell = \neg\mathrm{enable}^j$, $\mathrm{pairpos}(j) = 0$, and $\rho^\sigma_{\mathrm{pair}(j)} = 0$;

(c) $\ell = \mathrm{enable}^j$, $\mathrm{pairpos}(j) = 1$, and $\rho^\sigma_{\mathrm{pair}(j)} = 0$;

(d) $\ell = \neg\mathrm{enable}^j$, $\mathrm{pairpos}(j) = 1$, and $\rho^\sigma_{\mathrm{pair}(j)} = 1$;

(e) ℓ is a literal of a variable of block j different from enable^j, $\mathrm{pairpos}(j) = 0$, $\rho^\sigma_{\mathrm{pair}(j)} = 0$, and the polarity of ℓ agrees with the value $\rho^\sigma_{\mathrm{pair}(j)+s/2}$;

(f) ℓ is a literal of a variable of block j different from enable^j, $\mathrm{pairpos}(j) = 1$, $\rho^\sigma_{\mathrm{pair}(j)} = 1$, and the polarity of ℓ agrees with the value $\rho^\sigma_{\mathrm{pair}(j)+s/2}$.

We say that a seed $\sigma \in \{0,1\}^\ell$ is (ℓ, w)-*good* for Π if $\mathrm{Good}(s,n,m,w,\sigma,i)$ holds for every clause D_i encoded by Π.

We are ready to prove that good seeds exist.

Theorem 4.9 (Existence of good seeds, in $\mathrm{APC}_2(\alpha)$)**.** *The theory* $\mathrm{APC}_2(\Phi, \Pi)$ *proves the following statement. Let* $s, n, m \in \mathbb{N}$*, let* $t > 1$*, and let* $w \coloneqq 8\lceil \log t \rceil$*,* $q \coloneqq q(s, w)$*, and* $\ell \coloneqq q^4$*. If* Φ *encodes a CNF formula over* n *variables and* m *clauses and* Π *encodes a Resolution refutation of* $\mathrm{Ref}_s(\varphi)$ *of length at most* t*, then there exists an* (ℓ, w)*-good seed* $\sigma \in \{0,1\}^\ell$ *for* Π*.*

Proof. The case of odd s reduces to the even case by first applying the fixed one-block restriction used in the odd-s case of Lemma 3.6, so we may assume without loss of generality that s is even. Let $w = 8\lceil \log t \rceil$, $q = q(s, w) = 2^k$, and $\ell = q^4$. By the definition of q, we have $q \geq w$, $|s| \leq 2k(q+1)$, and $q \in \mathrm{Log}$. Hence we may invoke Lemma 4.2 to obtain an (s, q^4, q^2, q)-design M. Moreover, every submatrix formed by at most q rows has full row rank over $\mathbb{F}_2$. We refer to this matrix as M, although formally its rows are accessed through the function Design.

Put $r \coloneqq w/2$. Let $i \in [t]$ be such that D_i has block-width at least w. Since every pair contains only two blocks, D_i mentions at least r distinct pair-indices. Let N denote the number of variables of $\mathrm{Ref}_s(\varphi)$, and let $\mathrm{block}(\lambda)$ denote the PV-function returning the block containing the literal encoded by λ. For a number-coded sequence $C \in [2N]^r$, let $\mathrm{Valid}(i, C)$ be the sharply bounded formula asserting that every literal occurring in C occurs in D_i, and that the r literals of C belong to pairwise distinct pair-indices.

Since D_i has block-width at least w, some such valid C exists. Define C_i to be the lexicographically first one. Equivalently, its graph is defined by

$$\mathrm{Choice}(i, C) \coloneqq \mathrm{Valid}(i, C) \wedge \forall C' \in [2N]^r \left(C' <_{\mathrm{lex}} C \rightarrow \neg\, \mathrm{Valid}(i, C')\right). \tag{4.2}$$

Because $r \in \mathrm{Log}$, Valid is sharply bounded relative to Π, while eq. (4.2) is $\Pi^b_1(\Pi)$. Testing whether D_i has block-width at least w is $\Sigma^b_1(\Pi)$ by eq. (BW). Thus we can define in $\mathrm{PV}_2(\Pi)$ a function f which outputs this C_i whenever D_i is wide, and outputs the empty sequence otherwise.

Let $W \coloneqq \{i \in [t] : \mathrm{BW}(i, n, m, s, w)\}$ be the set collecting the clauses of block-width at least w. For every $i \in W$, the clause C_i has width and pair-width exactly r, so $|I(C_i)| = 2r = w \leq q$. By Lemma 4.2, $\mathrm{rank}_{\mathbb{F}_2} M_{I(C_i)} = w = 2r$. Therefore Lemma 4.6 gives

$$\mathrm{Bad}^M_{C_i} \preceq_0 \left(\frac{3}{4}\right)^{w/2} 2^\ell. \tag{4.3}$$

Now, if W is empty then every seed is already good. Otherwise, let i_0 be the least element of W; since eq. (BW) is $\Sigma^b_1(\Pi)$, bounded minimization is available in $\mathrm{PV}_2(\Pi)$. Fix $B \coloneqq 3^r 2^{\ell - 2r}$, and let $G_i : [B] \twoheadrightarrow \mathrm{Bad}^M_{C_i}$ denote the surjection from Lemma 4.6. Define $H(i, z) = G_i(z)$ when $i \in W$, and $H(i, z) = G_{i_0}(z)$ otherwise. Then H is a $\mathrm{PV}_2(\Pi)$-definable surjection

$$[t] \times [B] \twoheadrightarrow \bigcup_{i \in W} \mathrm{Bad}^M_{C_i}, \tag{4.4}$$

and hence $\bigcup_{i \in W} \mathrm{Bad}^M_{C_i} \preceq_0 t(3/4)^{w/2} 2^\ell$.

We claim that some seed lies outside this union. Suppose otherwise. Then $2^\ell \subseteq \bigcup_{i \in W} \mathrm{Bad}^M_{C_i}$, and by Lemma 2.1.(i), eq. (4.4), and transitivity, we get $2^\ell \preceq_0 t(3/4)^{w/2} 2^\ell$. By Lemma 2.1.(iii), taking $\xi = 1/4$, this implies $1 \leq t(3/4)^{w/2} + 1/4$. But since $w = 8\lceil \log t \rceil$, writing $h \coloneqq \lceil \log t \rceil \geq 1$, we have

$$t\left(\frac{3}{4}\right)^{w/2} \leq 2^h \left(\frac{3}{4}\right)^{4h} = \left(\frac{81}{128}\right)^h \leq \frac{81}{128}. \tag{4.5}$$

Consequently, $1 \leq 81/128 + 1/4 = 113/128 < 1$, a contradiction.

Hence there exists $\sigma \notin \bigcup_{i \in W} \mathrm{Bad}^M_{C_i}$. Consider the global restriction whose s restriction bits are

$\rho_j^\sigma = \mathrm{Design}(s, q, j) \cdot \sigma^\top$ for $j \in [s]$. For every $i \in W$, the projection of this restriction to $I(C_i)$ is $M_{I(C_i)}\sigma$, and since $\sigma \notin \mathrm{BAD}_{C_i}^M$ we have $C_i\restriction_{\rho^\sigma} = 1$. As $C_i \subseteq D_i$, it follows that $D_i\restriction_{\rho^\sigma} = 1$. Thus every clause of block-width at least w is killed by the restriction generated by σ, which is precisely to say that σ is (ℓ, w)-good. □

Remark 4.10 (Good seeds in the exponential-size regime: formalization in $\mathrm{APC}_1(\alpha)$). Suppose we want to prove Theorem 4.9 in the *exponential-size regime*, where $n, m, s \in \mathrm{Log}$. In this setting the argument goes through in $\mathrm{APC}_1(\Phi, \Pi)$. The design construction and Lemma 4.6 are already unrelativized, and the only reason the general proof requires APC_2 is the $\mathrm{PV}_2(\Pi)$-selection of the small subclauses C_i. When $s, n, m \in \mathrm{Log}$, the relevant searches through the variables and blocks of D_i are sharply bounded, so the choice of C_i can be carried out in $\mathrm{PV}(\Pi)$. Consequently, the uniform surjection used in eq. (4.4) is also available in the $\mathrm{APC}_1(\Phi, \Pi)$ setting. ◀

We finish this section with the following note on more general forms of XOR lifting.

Remark 4.11 (General XOR lifting). The restriction argument employed for Ref formulas follows the pattern of *XOR lifting*, attributed by Ben-Sasson [Ben09, §4] to Alekhnovich and Razborov around 2001, and known as *relativization* in the context of model-theoretic techniques for Resolution lower bounds [DR03; Kra11]. Roughly speaking, for a CNF formula φ, the XOR-lifted $\varphi[\oplus]$ is obtained by replacing each variable x_i of φ by $y_i \oplus z_i$, where the latter are new variables. The same restriction argument employed for Ref formulas works here, in the sense that if a formula φ requires Resolution refutations of width w, then Resolution refutations of $\varphi[\oplus]$ require exponential size $2^{\Omega(w)}$. Following the same arguments in this section, one can show that if a theory $T(\alpha) \supseteq \mathrm{APC}_2(\alpha)$ proves a width lower bound for some formula φ, then this same theory proves a size lower bound for the XOR-lifted formula. In the exponential-size regime an analogous result is implicit in the decision tree reductions of [LLR26, Theorem 6.4] and is formalizable in $\mathrm{T}_2^1(\alpha)$ + dWPHP($\mathrm{PV}_1(\alpha)$). The difference in using w-wise independence is again that we do not need to assume the parameters are in Log and we can thus derive fixed-polynomial size lower bounds. ◀

5 The block-width lower bound in $\mathrm{T}_2^3(\alpha)$

In this section, we prove within $\mathrm{T}_2^3(\alpha)$ the block-width lower bound for Ref formulas. We follow the new inductive proof explicated in Section 1.3.1. The crux of the proof is to define suitable formulas to encode the following properties: for a clause C in a Resolution refutation of $\mathrm{ERef}_s(\varphi)$,

(i) whether C *has block-width at least* w;

(ii) whether C is *Delayer consistent*, that is, whether there exists a full reservation R that satisfies the *reservation invariant* from Definition 3.2 with respect to C;

(iii) whether C *encodes a satisfying assignment to* φ: whether there is a reservation R such that C is Delayer consistent with respect to R and this reservation contains information about a satisfying assignment (a formal definition follows in Section 5.1.2).

The concept of block-width is clear: we already gave a $\Sigma_1^b(\Pi)$-formula (BW) to check block-width in Section 4.3. The concept of a clause C *encoding a satisfying assignment*, intuitively, corresponds to the quasipolynomial-time algorithm presented in Section 1.3.1 that extracts a satisfying assignment from C. The proof will be in $\mathrm{T}_2^3(\Phi, \Pi)$, where again we relativize the theory with access to two new relation symbols.

The outline of this section is as follows. In Section 5.1 we define relativized first-order formulas for the concepts listed above. In Section 5.2 we prove the block-width lower bound in $\mathrm{T}_2^3(\Phi, \Pi)$.

5.1 Bounded formulas for Delayer-consistency and assignment extraction

We assume that the relation symbol Π encodes a Resolution refutation consisting of t clauses, and that these are listed in Π as a topological ordering as $C_1, C_2, \ldots, C_t$, where C_1 is the empty clause.

5.1.1 Formula for Delayer consistency

The natural encoding of Delayer consistency has an existential quantifier guessing a reservation, but this is infeasible: full reservations contain information about the literals present in the blocks, so this requires $\mathrm{poly}(n)$ bits to represent. Instead, we observe that to check whether a clause C is Delayer consistent, it essentially suffices to check that a *basic reservation* R, as in Definition 3.1, satisfies some basic properties that allow for it to be extended into a full reservation verifying the reservation invariant.

Since information about resolved variables is fully determined by the layer at which blocks appear, a basic reservation can be represented just by the information about its pointers. As long as a basic reservation only talks about blocks B mentioned in a clause C of low block-width, this partial assignment can be encoded succinctly. In this context, we say that R is a *basic reservation for* C if R is a basic reservation that fixes the pointers of all blocks in B, and only of such blocks.

Let w be an upper bound on the number of blocks mentioned in C. A basic reservation R for C is encoded as a string over $[s+1]^{3w} \times [m]^w$, which we interpret as follows. For every block $b \in [s+1]$ mentioned in the clause C, there is an entry for it in R with a tuple of the form $\langle b, c, d, j\rangle$, where $c, d \in [s+1]$ are the blocks b points to on the left and right, respectively, in case the block b was derived; otherwise, $j \in [m]$ points to an axiom of φ that b was weakened from. Furthermore, if there is a tuple in R starting with $b \in [s+1]$, then b is mentioned in C. The string R has then size $O(w \log s + w \log m)$. We use the notation $R(b) = \langle c, d, j\rangle$ to refer to the tuple starting with b, and write $b \in \mathrm{dom}(R)$ to indicate that such a tuple exists in R. Although R is technically not indexed by values in $[s+1]$, we can always find this entry by the sharply bounded formula $\exists k \in [\mathrm{len}(R)]\ (R)_k = \langle b, c, d, j\rangle$, since $\mathrm{len}(R) \in \mathrm{Log}$ for $w \in \mathrm{Log}$. Whenever $b \in \mathrm{dom}(R)$, we write $R_{\mathrm{L}}(b)$, $R_{\mathrm{R}}(b)$, and $R_{\mathrm{A}}(b)$ for the left-pointer, right-pointer, and axiom coordinates of $R(b)$, respectively. These are just tuple projections: each can be recovered by a sharply bounded search through the entries of R. In particular, using this notation below does not introduce new bounded quantifiers.

It is straightforward to see that given R, we can write a $\Sigma^b_2(\Phi, \Pi)$-formula that checks that R is a basic reservation (Definition 3.1): its pointers only relate blocks between consecutive layers, etc. To be able to talk about layers, we have a PV-function to compute the layer of a given block, defined as

$$\mathrm{layer}(b; s, n) := \min\left\{n, \left\lfloor \frac{s-b}{\lfloor s/n \rfloor} \right\rfloor + 1\right\}, \tag{layer}$$

although throughout the rest of this section we often omit s and n as inputs, as these are clear from context.

Furthermore, we can also check with a $\Pi^b_2(\Pi)$ formula that R is a basic reservation *for* C: a block is mentioned in C if and only if there is a tuple for it in R,

$$\forall b \in [s+1]\ \left(\mathrm{Present}(i, b, n, m, s) \leftrightarrow \exists k \in [\mathrm{len}(R)]\ (R)_{k,1} = b\right). \tag{5.1}$$

We regard R as a sequence of at most w tuples, so that $\mathrm{len}(R) \le w$; in particular, the empty sequence is permitted.

Our goal is now to define a formula that checks whether there exists a full reservation R' extending R satisfying the reservation invariant of Definition 3.2. Note that the pointers of R set up a forest-like structure between the blocks. This structure restricts R' in the choice of literals in clauses. Since we cannot traverse trees in low depth, in order to reason about these constraints, we need an additional existential quantifier to

guess some extra information regarding which blocks appear in what subtrees. Let us specify the required extra information.

For every block $b \in \mathrm{dom}(R)$, we record the *set of proper descendants of b according to the pointers fixed by R*. These descendants may include blocks outside $\mathrm{dom}(R)$, namely children pointed to by R that are not mentioned in C_i. Since R contains at most w tuples, a block can have at most $2w$ proper descendants (since a binary tree with at most w internal nodes can have at most $w+1$ leaves).

We encode this collection as a sequence S of at most w tuples of the form $\langle b, c_1, \ldots, c_k \rangle$, where $k \leq 2w$. Thus S requires at most $w(2w+1)$ block entries, or $O(w^2 \log s)$ bits. We write $S(b)$ for the set of descendants recorded in the tuple beginning with b, and adopt the convention $S(b) = \emptyset$ whenever $b \notin \mathrm{dom}(R)$. In order to ensure that this collection S indeed encodes the subtrees of all the nodes in R we impose that S satisfies some basic closure properties. To state the precise conditions let us say that R *fixes a pointer of b to c* if R fixes the left or right pointer of b to c. The conditions we impose are that

(a) for every block b mentioned in R, there is exactly one tuple beginning with b in S;

(b) for every $c \in S(b)$, it holds that $\mathrm{layer}(c) > \mathrm{layer}(b)$;

(c) if $c \in S(b)$ and $\mathrm{layer}(c) = \mathrm{layer}(b) + 1$, R fixes a pointer of b to c;

(d) if R fixes a pointer of b to c, then $c \in S(b)$;

(e) for every $c \in S(b)$, we have that $S(c) \subseteq S(b)$;

(f) if $c \in S(b)$, then either R fixes a pointer of b to c, or there is some $c' \in S(b)$ such that $\mathrm{layer}(c') < \mathrm{layer}(c)$ and R fixes a pointer of c' to c.

Here, S is a sequence of exactly one tuple for each block in $\mathrm{dom}(R)$, with no additional tuples. Whenever $c \notin \mathrm{dom}(R)$, expressions involving $S(c)$ are interpreted using the convention $S(c) = \emptyset$. It is not hard to verify that S satisfies the above properties if and only if it contains exactly one tuple per block b containing all the blocks present in the subtree rooted at b. It is straightforward to check that all of the above properties are $\Pi_1^b(\Pi)$.

With R and S fixed we want to check whether there exists a full reservation R' that extends R, is defined over the same blocks B as R is, and satisfies the reservation invariant of Definition 3.2. Since R' only adds information to blocks in B it therefore satisfies item (iii) of Definition 3.2. It thus follows that in order to decide whether there is such a full reservation R' it suffices to verify (i) and (ii) of Definition 3.2. In other words, we need to check that R' falsifies C and does not falsify any axiom of $\mathrm{EREF}_s(\varphi)$. We cannot, however, check this explicitly as we do *not* have access to R'. Instead, we exhibit a small certificate that guarantees the existence of such an R'.

Note that by virtue of R being a basic reservation, axioms (Ref-4) through (Ref-11) are satisfied for all blocks B (and for other blocks these axioms are unrestricted). Hence, we need to define formulas checking whether there exists a full reservation R' extending R that satisfies condition (i) and axioms (Ref-1) to (Ref-3).

First, recall that a basic reservation fixes, for every block mentioned in C, whether the block is derived or weakened, its left and right pointers in the former case, the axiom it points to in the latter case, and the variable which it resolves over. We therefore have to check that every literal of C over variables of type derived, res, lpoint, rpoint, and weak is falsified by the corresponding value prescribed by R. It is readily verified that this can be expressed as bounded formulas directly from the encoding of R.

It remains to determine whether there is an extension R' of R assigning the lit-variables such that they simultaneously falsify C and satisfy axioms (Ref-1)–(Ref-3). The only additional constraints on the assignment of literals originate from the forest-like structure of R. With the help of S we propagate all the relevant constraints forced upon R' by R and the minimal partial assignment falsifying C. Let us formally define the formulas that check whether such an R' exists.

Denote by $i \in [t]$ the index of the clause $C = C_i$ we consider, and fix R and S. We check that a block b on layer h only contains literals over $x_1, \ldots, x_h$. Hence, for every $j > \mathrm{layer}(b)$, $\neg\mathsf{lit}^b_{x_j}$ and $\neg\mathsf{lit}^b_{\neg x_j}$ cannot be present in C_i. Formally, we check that the formula

$$\forall b \in \mathrm{dom}(R)\ \forall j \in [n]\ \Big(j > \mathrm{layer}(b) \rightarrow \neg\Pi\big(\mathsf{lit}^i_{\neg\mathsf{lit}^b_{x_j}}\big) \wedge \neg\Pi\big(\mathsf{lit}^i_{\neg\mathsf{lit}^b_{\neg x_j}}\big)\Big), \tag{5.2}$$

is satisfied.

Let us say that the clause C_i *forces* a variable to true (respectively, false) if the variable appears only negatively (respectively, only positively) in C_i. That is, the unique minimal falsifying assignment to C_i sets this variable to true (respectively, false). We say that C_i

- *forces a literal ℓ to appear in a block b* if C_i forces lit^b_ℓ to true;
- *prevents a literal ℓ from appearing in a block b* if C_i forces lit^b_ℓ to false or forces $\mathsf{lit}^b_{\neg\ell}$ to true.

Since every reserved block contains exactly one polarity of each variable up to its layer, preventing a literal from appearing also forces its opposite polarity to appear. Moreover, the axioms selected by R impose additional constraints on the contents of blocks in the last layer. To account for both sources of information, let $\mathrm{Pos}_{i,R}(c,j)$ and $\mathrm{Neg}_{i,R}(c,j)$ abbreviate

$$\mathrm{Pos}_{i,R}(c,j) \coloneqq \Pi(\mathsf{lit}^i_{\neg\mathsf{lit}^c_{x_j}}) \vee \Pi(\mathsf{lit}^i_{\mathsf{lit}^c_{\neg x_j}}) \vee (c \in \mathrm{dom}(R) \wedge \mathrm{layer}(c) = n \wedge \Phi(x_j, R_{\mathrm{A}}(c))),$$
$$\mathrm{Neg}_{i,R}(c,j) \coloneqq \Pi(\mathsf{lit}^i_{\neg\mathsf{lit}^c_{\neg x_j}}) \vee \Pi(\mathsf{lit}^i_{\mathsf{lit}^c_{x_j}}) \vee (c \in \mathrm{dom}(R) \wedge \mathrm{layer}(c) = n \wedge \Phi(\neg x_j, R_{\mathrm{A}}(c))).$$

These formulas express that x_j and $\neg x_j$, respectively, are forced to appear in block c, for $j \le \mathrm{layer}(c)$.

For a block b mentioned in C_i and its subtree $S(b)$, we need to ensure that the information forced at different blocks is consistent. In particular, for every variable x_j with $j \le \mathrm{layer}(b)$, no two blocks in $S(b) \cup \{b\}$ can be forced to contain opposite polarities of x_j. This is expressed by

$$\forall b \in \mathrm{dom}(R)\ \forall c,d \in S(b) \cup \{b\}\ \forall j \in [\mathrm{layer}(b)]\ \neg\big(\mathrm{Pos}_{i,R}(c,j) \wedge \mathrm{Neg}_{i,R}(d,j)\big). \tag{5.3}$$

On top of the propagation step above, we also check that the information forced by C_i and by the axioms of φ specified by R at final-layer blocks is consistent. In particular, the left subtree of a block resolving over x_j must contain x_j, while its right subtree must contain $\neg x_j$. Formally, we check that

$$\begin{aligned}\forall b \in \mathrm{dom}(R)\quad \Big(&\mathrm{layer}(b) < n \rightarrow \forall c \in S(R_{\mathrm{L}}(b)) \cup \{R_{\mathrm{L}}(b)\}\ \neg\,\mathrm{Neg}_{i,R}(c, \mathrm{layer}(b)+1)\\ &\wedge \forall d \in S(R_{\mathrm{R}}(b)) \cup \{R_{\mathrm{R}}(b)\}\ \neg\,\mathrm{Pos}_{i,R}(d, \mathrm{layer}(b)+1)\Big).\end{aligned} \tag{5.4}$$

where we recall that $R_{\mathrm{L}}(b)$ (respectively, $R_{\mathrm{R}}(b)$) denotes the block that R fixes the left (respectively, right) pointer of b to. We use the convention $S(c) = \emptyset$ whenever $c \notin \mathrm{dom}(R)$.

Finally, we need the analogous condition for a block weakened from an axiom of φ. Suppose that b is a block on the last layer and the basic reservation fixes the axiom $R_{\mathrm{A}}(b)$ of φ from which b is weakened.

Every literal appearing in that clause is not prevented from appearing in b. Thus we require

$$\forall b \in \mathrm{dom}(R)\ \ \mathrm{layer}(b) = n \ \rightarrow\ \forall j \in [n]\ \left(\Phi\big(x_j, R_{\mathrm{A}}(b)\big) \rightarrow \neg\Pi\Big(\mathrm{lit}^{i}_{\mathrm{lit}^{b}_{x_j}}\Big) \wedge \neg\Pi\Big(\mathrm{lit}^{i}_{\neg\mathrm{lit}^{b}_{\neg x_j}}\Big)\right) \wedge \left(\Phi\big(\neg x_j, R_{\mathrm{A}}(b)\big) \rightarrow \neg\Pi\Big(\mathrm{lit}^{i}_{\mathrm{lit}^{b}_{\neg x_j}}\Big) \wedge \neg\Pi\Big(\mathrm{lit}^{i}_{\neg\mathrm{lit}^{b}_{x_j}}\Big)\right). \tag{5.5}$$

This completes the description of the conditions we need to check. These are sufficient to enforce axioms (Ref-1)–(Ref-3).

To see this, start with the values of the non-lit variables prescribed by the basic reservation R. For each variable x_j, consider the connected components of the reserved forest restricted to layers at least j. Within each such component, the polarity of x_j must be constant. The propagation conditions guarantee that the information forced by C_i and the selected axioms is consistent within each component, while the Resolution conditions ensure compatibility with the polarities imposed by the pointers of R. We can therefore choose a consistent polarity for each component, respecting all these constraints, and choose one arbitrarily whenever neither polarity is forced.

This determines the contents of all reserved blocks. By eq. (5.2), no block is forced to contain a literal involving a variable whose index exceeds the block's layer, and the resulting assignment satisfies (Ref-1) and (Ref-2). Moreover, eq. (5.5) ensures compatibility with (Ref-3). Hence the resulting full reservation R' falsifies C_i and does not falsify any axiom of $\mathrm{ERef}_s(\varphi)$. This shows that R' satisfies the reservation invariant with respect to C_i.

Now, let $\mathrm{DC}(i, R, S, n, m, s, w)$ denote the conjunction of all formulas we have discussed so far. Note that eq. (5.1) is $\Pi^b_2(\Pi)$ and the rest are $\Sigma^b_2(\Phi, \Pi)$. By the preceding discussion, $\mathrm{DC}(i, R, S, n, m, s, w)$ holds exactly when R is a basic reservation for C_i, S correctly encodes its subtrees (by virtue of conditions (a)-(f)), and R can be extended to a full reservation R' satisfying the reservation invariant with respect to C_i. For $w \in \mathrm{Log}$, we consider the formula

$$\begin{aligned}&\text{Delayer-Cons}(i, n, m, s, w) :=\\ &\qquad \exists R \in [s+1]^{3w} \times [m]^{w}\ \exists S \in [s+1]^{w(2w+1)}\ \ \mathrm{DC}(i, R, S, n, m, s, w)\,.\end{aligned} \tag{Delayer-Cons}$$

We conclude that Delayer-Cons(i, n, m, s, w) is a $\Sigma^b_3(\Phi, \Pi)$-formula expressing that the clause C_i is Delayer consistent. If $n, m, s \in \mathrm{Log}$, then the formula is $\Sigma^b_2(\Phi, \Pi)$.

5.1.2 Formula for assignment extraction

Consider a clause C_i in Π. Recall from Section 5.1.1 that Delayer-consistency is witnessed by a basic reservation R, together with a collection S encoding its subtrees, such that R can be extended to a full reservation R' satisfying the reservation invariant. We say that C_i *encodes a satisfying assignment* if there is such a choice of R, S, and R' such that from R' we can deduce a clause whose unique falsifying assignment satisfies Φ, as formally defined below. This corresponds precisely to the quasipolynomial-time procedure described in Section 1.3.1: instead of guessing the contents of a full reservation R', we guess only its pointers, and use the information appearing in C_i, together with the connected components determined by those pointers, to reconstruct the contents of a block.

The goal of this section is to formally define a $\Sigma^b_3(\Phi, \Pi)$-formula Assignment(i, n, m, s, w) that is true if and only if the clause C_i encodes a satisfying assignment (of Φ), as explained above. To do this, suppose we have a $\Sigma^b_1(\Phi, \Pi)$-formula SatClause(k, i, R, S, b, n) encoding whether the assignment given, when inspecting block b, by propagating the information given by clause C_i, together with the reservation R and the extra

information S, satisfies the k-th clause of Φ. We can now define the formula

$$\begin{aligned}\text{Assignment}(i,n,m,s,w) \coloneqq \exists R \in [s+1]^{3w} \times [m]^w \, \exists S \in [s+1]^{w(2w+1)} \, \exists b \in [s+1] \\ \Big(\text{DC}(i,R,S,n,m,s,w) \wedge \forall k \in [m] \ \text{SatClause}(k,i,R,S,b,n)\Big).\end{aligned} \tag{Assignment}$$

Formally, we say that the clause C_i *encodes a satisfying assignment* if (Assignment) holds.

In order to define $\text{SatClause}(k,i,R,S,b,n)$, we need a $\Sigma_0^b(\Phi,\Pi)$-formula $\alpha(i,R,S,b,j)$ encoding whether the variable x_j is forced to 1 on block b by C_i, R and S. Now, given a literal $\ell \in [2n]$, where $\ell \in [n]$ represents x_ℓ and $\ell \in [2n] \setminus [n]$ represents $\neg x_{\ell-n}$, we can check whether the assignment defined above satisfies ℓ by the $\Sigma_0^b(\Pi)$-formula

$$\text{SatLit}(i,R,S,b,\ell,n) \coloneqq \big(\ell \in [n] \wedge \alpha(i,R,S,b,\ell)\big) \vee \big(\ell \in [2n] \setminus [n] \wedge \neg\alpha(i,R,S,b,\ell-n)\big), \tag{SatLit}$$

and then use this formula to define

$$\text{SatClause}(k,i,R,S,b,n) \coloneqq \exists \ell \in [2n] \Big(\Phi(\ell,k) \wedge \text{SatLit}(i,R,S,b,\ell,n)\Big). \tag{SatClause}$$

Thus the candidate assignment satisfies the whole formula precisely when $\forall k \in [m]\ \text{SatClause}(k,i,R,S,b,n)$. As in (Sat), this is a $\Pi_2^b(\Phi,\Pi)$ condition.

It remains to define $\alpha(i,R,S,b,j)$. For this, use the following two formulas:

$$\text{Below}_{R,S}(b,c) \coloneqq (b=c) \vee \big(c \in \text{dom}(R) \wedge b \in S(c)\big), \tag{5.6}$$

encoding that block b is either identical to c or it is a descendant of c according to the pointers fixed by R; and for $j \in [n]$,

$$\text{Same}_{R,S}(b,d,j) \coloneqq \exists c \in \text{dom}(R) \Big(j \le \text{layer}(c) \wedge \text{Below}_{R,S}(b,c) \wedge \text{Below}_{R,S}(d,c)\Big). \tag{5.7}$$

The point of the condition $j \le \text{layer}(c)$ is that every Resolution step strictly below c resolves over a variable with index larger than j. Consequently, all blocks below c contain the same polarity of x_j.

We now reconstruct one particular completion of the contents of b. For $j \in [n]$, let $\alpha(i,R,S,b,j)$ express that the assignment obtained by falsifying this clause sets x_j to 1. There are two ways in which this can be forced. First, C_i may explicitly determine the polarity of x_j at some block d whose x_j-content must agree with that of b. Since the reservation falsifies C_i, either a positive occurrence of $\text{lit}_{x_j}^d$ or a negative occurrence of $\text{lit}_{\neg x_j}^d$ forces $\neg x_j$ to be present in d, and hence also in b. Second, the pointers themselves may force b to lie below the right child of a block resolving over x_j, which again forces $\neg x_j$ to be present. We therefore define

$$\begin{aligned}\alpha(i,R,S,b,j) \coloneqq \exists d \in \text{dom}(R) \Big(\text{Same}_{R,S}(b,d,j) \wedge \text{Neg}_{i,R}(d,j)\Big) \\ \vee \exists c \in \text{dom}(R) \Big(j = \text{layer}(c)+1 \wedge \text{Below}_{R,S}(b,R_R(c))\Big).\end{aligned} \tag{5.8}$$

All the quantifiers ranging over $\text{dom}(R)$ or over the entries of a subtree $S(c)$ are sharply bounded, and hence the formula α above is $\Sigma_0^b(\Phi,\Pi)$. Let us justify the definition. The first disjunct accounts for the information forced by C_i or by an axiom selected in the reservation. In either case, $\text{Neg}_{i,R}(d,j)$ forces $\neg x_j$ to appear in d, and $\text{Same}_{R,S}(b,d,j)$ guarantees that the same polarity must appear in b. The second disjunct accounts for the information imposed by the Resolution pointers: a block at layer $j-1$ resolves over x_j, and its right child must contain $\neg x_j$. Hence, if b lies below this right child, then $\neg x_j$ must appear in b. If neither condition holds, we complete the contents of b by taking x_j. By the consistency conditions included in DC,

this completion does not contradict any information forced by C_i, the selected axioms, or the pointers of R.

This completes the definition of (Assignment).

5.2 The block-width lower bound

With the formal definitions in place, we are ready to prove the block-width lower bound. We start by recalling the following simple fact.

Lemma 5.1. *The theory $\mathsf{T}^1_1(\Phi, C)$ proves the following statement. For every n and m, if Φ encodes a CNF formula φ over n variables and m clauses and C encodes a non-tautological width-n clause that is not the weakening of any clause of φ, then the formula φ is satisfied by the unique assignment that falsifies C.*

Proof sketch. The proof does not use any induction. The definition of the assignment and the satisfiability condition is encoded analogously to (SatClause) (see Section 2.3.1 for details). Suppose for contradiction that the assignment does not satisfy φ, meaning that there is a clause C_i of φ, with $i \in [m]$, such that C_i is falsified by our assignment. By definition of the assignment, this means C_i is a subclause of C, so in fact C is a weakening of C_i. Contradiction. □

We now move to the actual inductive argument.

Lemma 5.2. *The theory $\mathsf{T}^3_2(\Phi, \Pi)$ proves the following $\forall\Sigma^b_3(\Phi, \Pi)$ statement. For every n, m, s and t and every $w \in \mathrm{Log}$ with $w \leq 1/2(\lfloor s/n \rfloor - 1)$, suppose Φ encodes a CNF formula φ over n variables and m clauses, suppose Π encodes a length-t Resolution refutation of $\mathrm{ERef}_s(\varphi)$ and suppose that for every enabled clause $i \in [t]$,*

(a) the clause C_i has block-width less than w; and

(b) C_i does not encode a satisfying assignment.

Then, for every $i \in [t]$, it holds that there exists $j > i$ such that $j \leq t$, the clause C_j is enabled and C_j is Delayer consistent.

Proof. Let us first check that the complexity of the statement is $\forall\Sigma^b_3(\Phi, \Pi)$. Formally, the statement is that for all n, m, s, t, for every $w \in \mathrm{Log}$ and for every $i \in [t]$, either Φ does not encode a CNF formula, or Π does not encode a correct Resolution refutation of $\mathrm{Ref}_s(\varphi)$, or there exists $i' \in [t]$ such that (a) or (b) are false, or there exists $j \in [t]$ such that $j > i$ and C_j is Delayer consistent.

The formulas checking that Π encodes a correct Resolution refutation of $\mathrm{Ref}_s(\varphi)$ and that Φ encodes a CNF formula are both $\Pi^b_2(\Phi, \Pi)$, so their negation is $\Sigma^b_2(\Phi, \Pi)$. Since the formula (BW) encoding that C_i has block-width at least w is $\Sigma^b_1(\Phi, \Pi)$, and the formula (Assignment) is a $\Sigma^b_3(\Phi, \Pi)$-formula, then we have that the formula

$$\exists i' \in [t] \ (\mathrm{BW}(i', n, m, s, w) \vee \mathrm{Assignment}(i', n, m, s, w)) \ , \tag{5.9}$$

is $\Sigma^b_3(\Phi, \Pi)$. Finally, the formula encoding Delayer consistency (Delayer-Cons) is $\Sigma^b_3(\Phi, \Pi)$, and therefore the formula

$$\exists j \in [t] \ (j > i \wedge \text{Delayer-Cons}(j, n, m, s, w)) \tag{5.10}$$

is also $\Sigma^b_3(\Phi, \Pi)$.

We can therefore proceed by $\Sigma^b_3(\Phi, \Pi)$-induction on i, available in $\mathsf{T}^3_2(\Phi, \Pi)$. Consider the refutation $\Pi = (C_1, \ldots, C_t)$ in a top-down ordering, that is, where $C_1 = \bot$ and every clause used to derive C_i has index strictly larger than i. We prove the statement by induction on $i = 1, \ldots, t$. Throughout this proof, all references to clauses of Π are understood to range over enabled clauses. Since Π is a valid refutation, every enabled derived clause has enabled premises.

Before setting up the induction, note that it is straightforward to verify that the update of a reservation according to the Delayer strategy $\mathrm{Delayer}_n(\Phi, \Pi, 1^s)$ as in Algorithm 3.1 can be expressed by a bounded

formula. More precisely, given the basic reservation R and the tree-structure S as in (Delayer-Cons) witnessing that a clause C_i is Delayer consistent, it is simple to define an updated reservation R' and S' following Algorithm 3.1 using only $\Sigma^b_3(\Phi, \Pi)$ formulas. We implicitly use this observation throughout the inductive argument.

Now, for the base case $i = 1$, consider the clause $C_1 = \bot$ and the empty reservation. If this is a Resolution step, the required reservation update following Algorithm 3.1 is possible: no block is reserved, and hence the update cannot fail because of a shortage of free blocks. The resulting reservation witnesses that the chosen child is Delayer consistent, establishing item (5.10).

Suppose now that the claim holds for i and let us prove it for $i + 1$. If eq. (5.9) holds, there is nothing to prove. Otherwise, the induction hypothesis yields an index j with $i < j \leq t$ such that C_j is Delayer consistent. If $j > i + 1$, then the same C_j witnesses the final statement for C_{i+1}. Hence it only remains to consider the case $j = i + 1$.

Suppose C_{i+1} is Delayer consistent and does not encode a satisfying assignment. Fix witnesses R and S to its Delayer-consistency, so that $\mathrm{DC}(i + 1, R, S, n, m, s, w)$ holds. In particular, R can be extended to a full reservation R' satisfying the reservation invariant with respect to C_{i+1}. Let us now do a case distinction on how C_{i+1} is derived.

- If C_{i+1} is obtained by weakening a clause $C_k \subseteq C_{i+1}$, where $k > i + 1$, then the same full reservation falsifies C_k and does not falsify any axiom of $\mathrm{ERef}_s(\varphi)$. After deleting from the reservation any information about blocks not mentioned in C_k, we obtain a reservation satisfying the reservation invariant with respect to C_k. Hence C_k is Delayer consistent and since $k > i + 1$ we are done.
- Suppose that C_{i+1} is derived by a Resolution step. Starting from the full reservation R' extending R, choose the child C_k to which the Delayer moves according to Algorithm 3.1. If the corresponding reservation update succeeds, it is easy to show that the updated reservation witnesses that C_k is Delayer consistent. We therefore only need to rule out the two ways in which this update can fail.

  – Suppose first that the update fails because the Delayer needs to attach fresh children to a block at some layer $h < n$, but there are not enough unreserved blocks on layer $h + 1$. Exactly as in the block-counting argument for the Delayer strategy, this implies that at least $\lfloor s/n \rfloor - 1$ blocks on that layer are already reserved. Since each block mentioned in the current clause can account for at most two reserved blocks in this update, we obtain

$$\mathrm{bw}(C_{i+1}) \geq \frac{1}{2}\left(\left\lfloor \frac{s}{n} \right\rfloor - 1\right) \geq w. \tag{5.11}$$

  This implies C_{i+1} has large block-width, contradicting our assumption that eq. (5.9) does not hold.

  – If the update concerns a previously unreserved block b' in the final layer, then the Delayer attempts to assign to it the clause $A = x_1 \vee \cdots \vee x_n$. Failure means that A is not a weakening of any axiom of Φ. Consequently, the all-zero assignment satisfies Φ, by Lemma 5.1.

  Moreover, since b' is unreserved, it is neither in $\mathrm{dom}(R)$ nor pointed to by R. Thus, for every $q \in [n]$, the formula $\alpha(i+1, R, S, b', q)$ is false. Hence R, S, b' witness $\mathrm{Assignment}(i+1, n, m, s, w)$, contrary to our assumption that (5.9) does not hold.

  We may therefore assume that the block b' was already reserved as a child of another block. The remaining possibility is the final-layer case. That is, C_{i+1} mentions a block b at layer $n - 1$, the reservation R points from b to a reserved block b' at layer n, and when moving to C_k, the block b' becomes relevant. This would imply that extending the reservation requires assigning b' to an axiom of Φ. The only way the update of the reservation can fail is if no such extension is possible.

Consider the width-n clause $\mathcal{B} = \mathcal{B}_{R,S,b'}$ whose contents are reconstructed from C_{i+1}, R, and S by the definition of $\alpha(i+1, R, S, b', \cdot)$ in eq. (5.8). More explicitly, for every $q \in [n]$, the clause $\mathcal{B}$ contains $\neg x_q$ precisely when $\alpha(i+1, R, S, b', q)$ holds, and contains x_q otherwise. By (Delayer-Cons) and the consistency conditions comprising DC, this is a well-defined non-tautological width-n clause.

We claim that $\mathcal{B}$ is not a weakening of any clause of Φ. Indeed, if $\mathcal{B}$ were a weakening of some axiom A of Φ, then we could extend R by declaring that b' is weakened from A and by assigning to b' the contents $\mathcal{B}$. Because of the definition of DC, this would give a full reservation satisfying the reservation invariant after the update, which we assumed failed.

Hence $\mathcal{B}$ is a non-tautological width-n clause that is not a weakening of any clause of Φ. By Lemma 5.1, its unique falsifying assignment satisfies Φ. But this falsifying assignment is exactly the assignment represented by $\alpha(i+1, R, S, b', \cdot)$. Moreover, R and S satisfy DC. Therefore the same witnesses R, S, b' satisfy every conjunct in eq. (Assignment). Consequently C_{i+1} encodes a satisfying assignment, contradicting our assumption that eq. (5.9) does not hold.

Both possible failures of the Delayer update are therefore impossible. The chosen child C_k, with $k > i+1$, is Delayer consistent. This completes the induction. □

The universal statement above now implies an existential statement: there is either a wide clause or a satisfying assignment.

Theorem 5.3 (Assignment or wide clause, in $\mathsf{T}^3_2(\alpha)$)**.** *The theory $\mathsf{T}^3_2(\Phi, \Pi)$ proves the following $\forall\Sigma^b_3(\Phi, \Pi)$ statement. For every n, m, s and t and every $w \in \mathrm{Log}$ with $w \leq 1/2(\lfloor s/n \rfloor - 1)$, suppose Φ encodes a CNF formula over n variables and m clauses, and suppose Π encodes a length-t Resolution refutation of $\mathrm{ERef}_s(\varphi)$. Then, there exists $i \in [t]$ such that either*

(a) the clause C_i is enabled and encodes a satisfying assignment to φ, or else

(b) the clause C_i is enabled and the block-width of C_i is at least w.

Proof. The complexity of the statement is indeed $\forall\Sigma^b_3(\Phi, \Pi)$, as already discussed in Lemma 5.2. To complete the proof, working within the theory, suppose for contradiction that for every $i \in [t]$, neither (a) nor (b) holds. Then, by Lemma 5.2, for every $i \in [t]$ there exists j such that $i < j \leq t$ and C_j is Delayer consistent. But then for the case $i = t$ we get that there is a j such that $t < j \leq t$, which is absurd. □

Remark 5.4 (The block-width lower bound in the exponential-size regime: formalization in $\mathsf{T}^1_2(\alpha)$). Suppose we wanted to prove Theorem 5.3 above in the so-called exponential-size regime, where n, m and s are assumed to be in Log. In this setting, the proof goes through in $\mathsf{T}^1_2(\Phi, \Pi)$. This is because when assuming $s, n, m \in \mathrm{Log}$, the universal quantifiers in the definition of Delayer-consistency in Section 5.1.1 are now sharply bounded, making that overall formula $\Sigma^b_1(\Pi)$. The same happens with the assignment extraction formula (Assignment), where the universal quantifiers become sharply bounded. This overall makes Lemma 5.2 available in $\mathsf{T}^1_2(\Phi, \Pi)$. ◀

6 The Atserias–Müller lower bound in $\mathsf{T}_2(\alpha)$

With both the restriction argument and the block-width lower bound ready, we combine them into the final statement of the lower bound. As previously discussed, depending on the regime we find ourselves in, we can slightly vary the power of the theory. In the exponential-size regime we can work in $\mathsf{T}^1_2(\alpha) + \mathrm{dWPHP}(\mathsf{PV}_1(\alpha))$, while in the polynomial-size regime we are forced to go to $\mathsf{T}^4_2(\alpha)$. Section 6.1 shows how to manipulate restrictions within these theories, and Section 6.2 and Section 6.3 complete the arguments for the final lower bound.

6.1 Applying the restrictions

In the next proposition, ρ is a new relation symbol encoding a restriction in the natural way and we show that the fact that Resolution is closed under restrictions is easily derivable already in $\mathsf{T}^1_1(\alpha)$. The idea is that from Π and ρ we can define a relation Π' encoding $\Pi_{\restriction\rho}$.

Proposition 6.1. *The theory $\mathsf{T}^1_1(\Phi, \Pi, \Phi', \Pi', \rho)$ proves that Resolution is closed under restrictions: for every $n \geq 1$, m and t, if Φ encodes a CNF formula over n variables and m clauses and Π encodes a Resolution refutation of Φ of length t, Φ' encodes $\Phi_{\restriction\rho}$ and Π' encodes $\Pi_{\restriction\rho}$, then Π' is a length-t Resolution refutation of Φ'. Furthermore, the formulas relating Φ and Π to Φ' and Π' are $\Delta^b_2(\Phi, \Pi, \Phi', \Pi', \rho)$.*

Proof. It suffices to see that there are $\Delta^b_2(\Phi, \Pi, \Phi', \Pi', \rho)$ formulas defining the new formula and refutation after the restriction. For a clause A of Φ, let

$$\mathrm{Killed}_\rho(A) \coloneqq \exists \ell \in [2n] \left(\Phi(\ell, A) \wedge \rho(\ell) = 1\right). \tag{6.1}$$

Since our encoding has a fixed number of clauses, we represent a clause that is satisfied by ρ by the fixed tautology $x_1 \vee \neg x_1$. Thus we define Φ' by

$$\begin{aligned}\Phi'(\ell, A) \longleftrightarrow \big(&\neg\, \mathrm{Killed}_\rho(A) \wedge \Phi(\ell, A) \wedge \rho(\ell) = \star\big) \\ &\vee \big(\mathrm{Killed}_\rho(A) \wedge (\ell = x_1 \vee \ell = \neg x_1)\big).\end{aligned} \tag{6.2}$$

This convention is semantically equivalent to deleting the satisfied clauses, while keeping the clause indices unchanged.

To illustrate the definition of Π', let us first note that a proof line B is killed if either it was already disabled or some literal in it is set to true by ρ. The remaining proof variables can be defined by

$$\Pi'(\mathsf{enable}^B) \longleftrightarrow \Pi(\mathsf{enable}^B) \wedge \forall \ell \in [2n] \ \left(\Pi(\mathsf{lit}^B_\ell) \to \rho(\ell) \neq 1\right) \tag{6.3}$$

$$\Pi'(\mathsf{a\text{-}lit}^A_\ell) \longleftrightarrow \Phi'(\ell, A) \tag{6.4}$$

$$\Pi'(\mathsf{derived}^B) \longleftrightarrow \Pi(\mathsf{derived}^B) \tag{6.5}$$

$$\Pi'(\mathsf{weak}^B_A) \longleftrightarrow \Pi(\mathsf{weak}^B_A) \wedge \neg\, \mathrm{Killed}_\rho(A) \tag{6.6}$$

$$\Pi'(\mathsf{lit}^B_\ell) \longleftrightarrow \Pi(\mathsf{lit}^B_\ell) \wedge \rho(\ell) = \star \tag{6.7}$$

$$\Pi'(\mathsf{res}^B_{x_i}) \longleftrightarrow \Pi(\mathsf{res}^B_{x_i}) \tag{6.8}$$

$$\begin{aligned}\Pi'(\mathsf{lpoint}^B_{B'}) \longleftrightarrow &\left(\Pi(\mathsf{lpoint}^B_{B'}) \wedge \forall \ell \in [2n] \ \left(\Pi(\mathsf{lit}^{B'}_\ell) \to \rho(\ell) \neq 1\right)\right) \\ &\vee \left(\Pi(\mathsf{rpoint}^B_{B'}) \wedge \exists i \in [n] \ (\Pi(\mathsf{res}^B_{x_i}) \wedge \rho(x_i) = 1)\right)\end{aligned} \tag{6.9}$$

$$\begin{aligned}\Pi'(\mathsf{rpoint}^B_{B'}) \longleftrightarrow &\left(\Pi(\mathsf{rpoint}^B_{B'}) \wedge \forall \ell \in [2n] \ \left(\Pi(\mathsf{lit}^{B'}_\ell) \to \rho(\ell) \neq 1\right)\right) \\ &\vee \left(\Pi(\mathsf{lpoint}^B_{B'}) \wedge \exists i \in [n] \ (\Pi(\mathsf{res}^B_{x_i}) \wedge \rho(x_i) = 0)\right).\end{aligned} \tag{6.10}$$

The right-hand side formulas are all $\Delta^b_2(\Phi, \Pi, \Phi', \Pi', \rho)$, so the entire conjunction of double implications has the same complexity.

Once the definitions are in place, it is easy to verify that the statement follows. If the resolved variable is unset, a surviving Resolution inference remains a Resolution inference. If the resolved variable is assigned, exactly the premise falsifying that value survives, and the two new pointers are both redirected to that premise; as observed in Remark 2.8, this is a valid way to encode a weakening. Finally, if an axiom is killed,

every proof line weakened from it is killed as well. Hence every enabled line of Π' is a valid Resolution or weakening step from earlier enabled lines, and Π' is a Resolution refutation of Φ'. □

The following is the propositional version of the previous proposition, which will be useful later.

Proposition 6.2 (Resolution proves that Resolution is closed under restrictions)**.** *There are polynomial-size* **P***-uniform Resolution derivations of the form*

$$\textsc{Ref}_t(\varphi,\pi) \wedge (\pi' = \pi_{\restriction\rho}) \wedge (\varphi' = \varphi_{\restriction\rho}) \vdash_{\mathrm{Res}} \textsc{Ref}_t(\varphi',\pi'),$$

where, once ρ is fixed, $(\pi' = \pi_{\restriction\rho}) \wedge (\varphi' = \varphi_{\restriction\rho})$ are depth-2 formulas.

Proof. Immediate from the Paris–Wilkie translation in Theorem 2.5. □

We next record explicitly the form of the disabling restriction that will be used below. Put $s_{\mathrm{e}} := 2\lfloor s/2\rfloor$ and $s' := \lfloor s/2\rfloor$. We pair the first s_{e} blocks consecutively. If s is odd, the remaining block is disabled deterministically. For $w \in \mathrm{Log}$, let $q := q(s_{\mathrm{e}}, w)$ be the power of two from Theorem 4.9, and let $\ell := q^4$. Given a seed $\sigma \in \{0,1\}^\ell$, write $\eta_r := \mathrm{Design}(s_{\mathrm{e}}, q, r) \cdot \sigma^\top$ for $r \in [s_{\mathrm{e}}]$. For every $j \in [s']$, the bit η_j chooses the disabled block in pair j, and $\eta_{j+s'}$ gives the common value assigned to the remaining variables of that disabled block. The root is always enabled.

Let ρ_σ denote this disabling restriction. We further extend it to a restriction $\widehat{\rho}_\sigma$ by setting to 0 every pointer variable whose source block survives but whose target block is disabled. This extra clean-up can only satisfy additional literals and never revives a killed clause. If $\mathrm{surv}_\sigma(j) := 2j - \eta_j$ denotes the surviving block from pair j, then, after renaming the surviving block $\mathrm{surv}_\sigma(j)$ to j and the old root to $s' + 1$, the restricted formula $\textsc{Ref}_s(\varphi)_{\restriction\widehat{\rho}_\sigma}$ is precisely the enable-free formula $\textsc{ERef}_{s'}(\varphi)$, up to clauses already satisfied by the restriction.

Lemma 6.3 (Applying a good seed)**.** *The theory* $\mathrm{APC}_2(\Phi,\Pi)$ *proves the following. Let* $w := 8\lceil \log t\rceil$*, let* $s_{\mathrm{e}} = 2\lfloor s/2\rfloor$*, let* $q = q(s_{\mathrm{e}}, w)$*, and let* $\ell = q^4$*. If* Π *encodes a length-*t *Resolution refutation of* $\textsc{Ref}_s(\varphi)$*, then there is a seed* $\sigma \in \{0,1\}^\ell$ *and a* $\Delta^b_2(\Phi,\Pi)$*-definable relation* $\widehat{\Pi}^\sigma$ *such that:*

(i) $\widehat{\Pi}^\sigma$ *encodes a length-*t *Resolution refutation of* $\textsc{ERef}_{s'}(\varphi)$*, where* $s' = \lfloor s/2\rfloor$*;*

(ii) every enabled clause of $\widehat{\Pi}^\sigma$ *has block-width at most* $w - 1$*.*

If $n, m, s \in \mathrm{Log}$*, the same statement is available in* $\mathrm{APC}_1(\Phi,\Pi)$ *and the definition of the relation* $\widehat{\Pi}^\sigma$ *is sharply bounded relative to* Φ *and* Π*.*

Proof. If s is even, take the good seed given by Theorem 4.9. If s is odd, leave the last block unpaired and disable it deterministically, and repeat the proof of Theorem 4.9 on the remaining $s - 1$ blocks. This does not change the estimate: since w is even, every clause mentioning at least w blocks mentions at least $w/2$ distinct pair-indices among the paired blocks, so the same width-$w/2$ subclause and the same bad-seed bound can be used.

For this structured formula we use the same line-by-line restriction clean-up as in Proposition 6.1, but we omit clauses of the inner formula that become true rather than representing them by tautological placeholders. After applying $\widehat{\rho}_\sigma$ and renaming the variables, the unsatisfied clauses are exactly the clauses of $\textsc{ERef}_{s'}(\varphi)$. The resulting proof relation $\widehat{\Pi}^\sigma$ is still $\Delta^b_2(\Phi,\Pi)$-definable, since this is the same local clean-up and renaming as in Proposition 6.1. If an original clause has block-width at least w, the good-seed property already kills it under ρ_σ, and hence also under the stronger restriction $\widehat{\rho}_\sigma$. Restriction and renaming cannot increase block-width, so every enabled clause in the resulting refutation has block-width at most $w - 1$.

In the exponential-size regime all ranges occurring in the restriction and clean-up definitions are in Log, so the defining relation becomes sharply bounded and Remark 4.10 gives the result in $\mathrm{APC}_1(\Phi,\Pi)$. □

Finally, for a seed σ, let Ren_σ be the PV-function sending a literal over the variables of $\mathrm{EREF}_{s'}(\varphi)$ to the corresponding literal over the surviving blocks of $\mathrm{REF}_s(\varphi)$. For a clause index i, define the restricted-clause predicate by

$$\Pi^\sigma_{\mathrm{cl}}(\mathrm{lit}^i_L) \coloneqq \Pi\big(\mathrm{lit}^i_{\mathrm{Ren}_\sigma(L)}\big) \wedge \widehat{\rho}_\sigma\big(\mathrm{var}(\mathrm{Ren}_\sigma(L))\big) = \star. \tag{6.11}$$

We also let $\mathrm{Survives}(i,\sigma)$ express that no literal of C_i is set to true by $\widehat{\rho}_\sigma$. Now let $\mathrm{RAssignment}(i,\sigma,n,m,s,w)$ denote the formula obtained from eq. (Assignment), with the block parameter replaced by $s' = \lfloor s/2 \rfloor$ and every query to the clause predicate Π replaced by the sharply bounded predicate Π^σ_{cl}, together with the conjunct $\mathrm{Survives}(i,\sigma)$.

Because eq. (6.11) is sharply bounded relative to Π, while $\mathrm{Survives}(i,\sigma)$ is $\Pi^b_1(\Pi)$, and RAssignment is $\Sigma^b_3(\Phi,\Pi)$. If $n, m, s \in \mathrm{Log}$, RAssignment collapses to $\Sigma^b_1(\Phi,\Pi)$.

6.2 Formalization in the polynomial-size regime: $\mathsf{T}^4_2(\alpha)$

Our first statement is for the most general setting, where we deal with any CNF formula without any bounds on the width, and we are in the polynomial-size regime. The name "polynomial-size regime" might look like a misnomer, since in the following statement we prove instead an exponential lower bound on the length t of the refutations. The crucial point here is that we do not assume that either s or n is in Log, so we cannot feasibly express 2^n. In the following statements, we say that *C_i encodes a satisfying assignment* to mean that RAssignment is true for some seed σ.

Theorem 6.4 (Atserias–Müller in $\mathsf{T}^4_2(\alpha)$, polynomial-size regime)**.** *There exists a positive constant $\varepsilon \in \mathbb{Q}$ and $n_0 \in \mathbb{N}$ such that the theory $\mathsf{T}^4_2(\Phi,\Pi)$ proves the following $\forall\Sigma^b_3(\Phi,\Pi)$ statement. For every n and m, for every s and every $t > 1$, suppose Φ encodes a CNF formula over $n \geq n_0$ variables and m clauses, and suppose Π encodes a length-t Resolution refutation of $\mathrm{REF}_s(\varphi)$. If $|t| < \varepsilon \cdot s/n$, then there exists $i \in [t]$ such that C_i encodes a satisfying assignment.*

Proof. Fix $\varepsilon = 1/64$ and $n_0 = 1$. Let $w = 8\lceil \log_2 t \rceil$, $s' = \lfloor s/2 \rfloor$, $q = q(2s', w)$, and $\ell = q^4$. Since $\lceil \log_2 t \rceil \leq |t|$, the hypothesis gives $w \leq 8|t| \leq \frac{s}{8n}$. Moreover, since n is a positive integer, $\lfloor s'/n \rfloor = \lfloor s/2n \rfloor \geq s/2n - 1$. Consequently,

$$\frac{1}{2}\left(\left\lfloor \frac{s'}{n} \right\rfloor - 1\right) \geq \frac{s}{4n} - 1 \geq w. \tag{6.12}$$

By Lemma 6.3, working in $\mathrm{APC}_2(\Phi,\Pi)$, we obtain a seed $\sigma \in \{0,1\}^\ell$ and a $\Delta^b_2(\Phi,\Pi)$-definable length-t Resolution refutation $\widehat{\Pi}^\sigma$ of $\mathrm{EREF}_{s'}(\varphi)$ in which every enabled clause has block-width at most $w-1$.

By eq. (6.12), we may apply Theorem 5.3 to $\widehat{\Pi}^\sigma$ with width parameter w. It follows that some enabled clause of $\widehat{\Pi}^\sigma$ encodes a satisfying assignment to Φ. Let $i \in [t]$ be its index in the original refutation. By construction, C_i survives the restriction and the clause analyzed in $\widehat{\Pi}^\sigma$ is its renamed restriction. Thus $\mathrm{RAssignment}(i,\sigma,n,m,s,w)$ holds.

Note that the restriction argument is available in $\mathrm{APC}_2(\Phi,\Pi) \subseteq \mathsf{T}^3_2(\Phi,\Pi)$. Applying the block-width lower bound to the $\Delta^b_2(\Phi,\Pi)$-definable relation $\widehat{\Pi}^\sigma$ raises the induction level from T^3_2 to T^4_2. The conclusion remains $\Sigma^b_3(\Phi,\Pi)$, since RAssignment has this complexity and the existential quantifiers over the number-coded seed σ and the index i can be merged with its leading existential block. □

6.3 Formalization in the exponential-size regime: $\mathsf{T}^1_2(\alpha)$ + dWPHP$(\mathrm{PV}_1(\alpha))$

In the so-called exponential-size regime we assume the initial parameters of the REF formula are in Log. This means many of the quantifiers in our definitions become sharply bounded, leading to the following result.

Theorem 6.5 (Atserias–Müller for all formulas in $\mathsf{T}^1_2(\alpha)$ + dWPHP($\mathsf{PV}_1(\alpha)$), exponential-size regime)**.** *There exists a positive constant $\varepsilon \in \mathbb{Q}$ and $n_0 \in \mathbb{N}$ such that the theory $\mathsf{T}^1_2(\alpha)$ + dWPHP($\mathsf{PV}_1(\alpha)$) proves the following $\forall\Sigma^b_1(\Phi,\Pi)$ statement. For every $n \in \mathrm{Log}$, $m \in \mathrm{Log}$ and $s \in \mathrm{Log}$, and every $t > 1$, suppose Φ encodes a CNF formula over $n \geq n_0$ variables and m clauses, and suppose Π encodes a length-t Resolution refutation of $\textsc{Ref}_s(\varphi)$. If $t < 2^{\varepsilon \cdot s/n}$, then there exists $i \in [t]$ such that C_i encodes a satisfying assignment.*

Proof. The parameters are in the proof of Theorem 6.4. Because $n, m, s \in \mathrm{Log}$, the restriction in Lemma 6.3 is available already in $\mathsf{APC}_1(\Phi,\Pi)$ as discussed in Remark 4.10, and the block-width lower bound is available in $\mathsf{T}^1_2(\Phi,\Pi)$ as in Remark 5.4; substituting the sharply bounded definition of $\widehat{\Pi}^\sigma$ does not increase the complexity of the induction. □

7 Propositional consequences

In this section we show how the new formalizations extend some of the propositional corollaries of Arteche et al. [AAdRK25] from Extended Frege to constant-depth systems, with a quasi-polynomial blow-up in parameters. Section 7.1 shows how to apply the Paris–Wilkie translations to get Resolution lower bounds formalized in the propositional setting. Section 7.2 applies these to argue that automating on Ref formulas suffices to automate on all formulas.

7.1 Propositional fragments of the Atserias–Müller lower bound

We start by showing that if a proof system can refute some formula efficiently, then it can also refute a lower bound on its corresponding Ref formula. Before we state this formally, we prove a simple technical lemma showing how the depth of a propositional proof increases when substituting formulas of bounded depth.

Lemma 7.1. *Let $d \in \mathbb{N}$, and let π be a PK_{d+1}-refutation of φ where every connective in π at depth $d+1$ has arity at most r. Let ψ be a formula of depth $k \geq 1$, whose top connective has arity ℓ, and whose bottom connectives have arity at most b. It holds that the substitution of ψ into π, denoted by $\pi(\psi)$, is a PK_{d+k}-refutation of $\varphi(\psi)$ of size $|\pi(\psi)| = O\left(|\pi| \cdot |\psi| \cdot \ell^r\right)$,where the connectives at depth $d+k$ have arity at most b if $k \geq 3$, at most rb if $k = 2$, and at most $r\ell$ if $k = 1$.*

Proof. Since every line of π has depth at most $d+1$, substitution of a depth-k formula for atoms initially increases the depth to at most $d+k+1$. To reduce the depth, consider the original bottom-layer gates of π, each of arity at most r, whose inputs are replaced by copies of ψ. If the bottom gate of π matches the top gate of ψ, we merge the two gates, reducing the depth to at most $d+k$. The bottom fan-in remains at most b when $k \geq 2$, while for $k = 1$ the merged gate has arity at most $r\ell$. Otherwise, we distribute the bottom gate of π over the top gates of the substituted formulas. This produces at most ℓ^r terms and reduces the depth to at most $d+k$. If $k \geq 3$, the bottom gates of ψ remain unchanged and have arity at most b. If $k = 2$, the distribution may merge up to r bottom gates of ψ, giving arity at most rb. Finally, if $k = 1$, the resulting bottom gates have arity at most r. Combining the two cases, the bottom fan-in is bounded by b for $k \geq 3$, by rb for $k = 2$, and by $r\ell$ for $k = 1$. Each line γ of π increases in size by at most a factor $O(|\psi|\ell^r)$. Summing over all lines yields $|\pi(\psi)| = O\left(|\pi| \cdot |\psi| \cdot \ell^r\right)$. □

We are ready to apply the propositional translations to our formalizations.

Theorem 7.2 (Propositional fragments of Atserias–Müller)**.** *Let $d \in \mathbb{N}$, let φ be a CNF formula over a large enough number n of variables and m clauses, let $s \geq n^2$, and let $t > 1$ satisfy $|t| < \varepsilon s/n$, where $\varepsilon > 0$ is the constant from Theorem 6.4. The following hold:*

(i) if $\mathsf{PK}_d \vdash \neg\varphi$ *in size* ℓ*, then* $\mathsf{PK}_{\max\{d+1/2,\ 3+1/2\}} \vdash \neg\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi))$ *in size* $\mathrm{quasipoly}(n, m, s, t, \ell)$*;*

(ii) if $\mathsf{PK}_{(d+3)+1/2} \vdash \neg\varphi$ *in size* ℓ*, then* $\mathsf{PK}_{(d+3)+1/2} \vdash \neg\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi))$ *in size* $\mathrm{quasipoly}(n, m, s, t, \ell)$*;*

Furthermore, in both cases, given the proof of $\neg\varphi$ *in the left-hand side, one can obtain a proof of* $\neg\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi))$ *in quasipolynomial time.*

Proof. In Theorem 6.4 we showed that $\mathsf{T}^4_2(\Phi, \Pi)$ proves the Atserias–Müller lower bound for CNF formulas of arbitrary width. Since the complexity of the statement is $\forall\Sigma^b_3(\Phi, \Pi)$, we can apply the Paris–Wilkie translation (Theorem 2.5) to get that there are quasipolynomial-size derivations of the form

$$\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi), \pi) \vdash_{\mathsf{PK}_{3+1/2}} \mathrm{Sat}_1(\varphi, \alpha_1(\pi)), \ldots, \mathrm{Sat}_T(\varphi, \alpha_T(\pi)). \tag{7.1}$$

Here, $T = \mathrm{quasipoly}(n, m, s, t)$ is the upper bound of the total number of possible candidate assignments we try. That is, the Paris–Wilkie translation would produce a big disjunction when translating the existential quantifiers in (Assignment), which we have represented here as a cedent. For each candidate assignment we have a small subformula $\alpha_i(\pi)$ which is just the polylogarithmic-width disjunct coming from the translation of (5.8). Note that we also see $\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi), \pi)$ as a set of formulas, rather than as a syntactic conjunction.

Suppose now there is a PK_d-refutation of φ in size ℓ. Then, with at most a linear blow-up, there is a PK_d refutation of $\mathrm{Sat}_{\restriction\varphi}(\alpha)$ (see Section 2.3.1). For each $i \in [T]$, substitute for α the clauses given by $\alpha_i(\pi)$ extracting a candidate assignment. This is a clause of width $\mathrm{polylog}(n, m, s)$, so it follows that we now have a derivation of the form $\mathsf{PK}_{d+1/2} \vdash \neg\mathrm{Sat}_{\restriction\varphi}(\alpha_i)$ in size $O(\ell)$ for every $i \in [T]$. By successively cutting T times with (7.1), we get a $\mathsf{PK}_{\max\{d+1/2,\ 3+1/2\}}$-refutation of $\mathrm{Ref}_t(\mathrm{Ref}_s(\varphi), \pi)$ in size $\mathrm{quasipoly}(n, m, s, t, \ell)$. This proves the first item.

To prove the second item, repeat the same process using Lemma 7.1 to argue about the depth increase after substitution. We have that the proof of $\neg\varphi$ has depth $d + 4$ and polylogarithmic bottom fan-in, and the α_i circuits have depth 1 with polylogarithmic bottom fan-in, so we get a depth-$(d + 4)$ proof with polylogarithmic fan-in at the bottom layer, which is precisely $\mathsf{PK}_{(d+3)+1/2}$. The size blow-up, as accounted for by Lemma 7.1, is precisely quasipolynomial. □

We also obtain a converse of this statement building on Pudlák's upper bound construction [Pud03], which was already formalized in Resolution.

Theorem 7.3 (Pudlák's upper bound in Resolution [AAdRK25, Theorem 7.1])**.** *For every* $n, m, s \in \mathbb{N}$*, there is some* $\tau = \Theta(s(m + sn^2))$ *such that for* $t \geq \tau$ *there exist uniform polynomial-size Resolution derivations of the form*

$$\mathrm{Sat}(\varphi, \alpha) \wedge \pi = P(\varphi, \alpha, s) \vdash_{\mathsf{Res}} \mathrm{Ref}_t(\mathrm{Ref}_s(\varphi), \pi)\,,$$

where the Sat *formula is for CNF formulas with* n *variables and* m *clauses.*

In the statement above, $\pi = P(\varphi, \alpha, s)$ stands for the conjunction of clauses describing the upper bound construction of Pudlák. As shown in [AAdRK25], this circuit P is actually of very small depth. In fact, once φ is fixed, the π variables are simply 2-extensions over the variables of α. In other words, for every fixed φ, the formula $\pi = P(\varphi, \alpha, s)$ amounts to the set of all $O(n^2)$ possible extensions for two literals over n variables. This leads to the following converse statement to Theorem 7.2.

Theorem 7.4. *There is a function* $p(n, m, s) = \mathrm{poly}(n, m, s)$ *such that for every CNF formula* φ *over* n *variables and* m *clauses, every* $s \in \mathbb{N}$*, and* $d \in \mathbb{N}$*,*

(i) if $\mathsf{PK}_d \vdash \neg\mathrm{Ref}_{p(n,m,s)}(\mathrm{Ref}_s(\varphi))$ *in size* ℓ*, then* $\mathsf{PK}^2_{d+1} \vdash \neg\varphi$ *in size* $\mathrm{poly}(n, m, s, \ell)$*;*

(ii) if $\mathsf{PK}_{d+1/2} \vdash \neg\mathrm{Ref}_{p(n,m,s)}(\mathrm{Ref}_s(\varphi))$ *in size* ℓ*, then* $\mathsf{PK}_{d+1/2} \vdash \neg\varphi$ *in size* $\mathrm{poly}(n, m, s, \ell)$*.*

Furthermore, in both cases, given a proof of the left-hand side statement, one can obtain a proof of the right-hand side statement in polynomial time.

Proof. To prove the first item, we use the fact that from Theorem 7.3 we have Resolution derivations of the form

$$\{\textsc{Sat}(\varphi, \alpha)\} \cup \{\pi = P(\varphi, \alpha, s)\} \vdash_{\mathsf{Res}} \{\textsc{Ref}_t(\textsc{Ref}_s(\varphi), \pi)\}, \tag{7.2}$$

for t the parameter τ in the statement of Theorem 7.3, which will also be our polynomial p in this statement. We write the premises and conclusion in set brackets to emphasize that these are sets of formulas interpreted as their conjunction, but without this explicitly counting towards the depth.

For φ fixed, Resolution can go from the formula $\textsc{Sat}$ to the native encoding of φ with a linear blow-up (see Section 2.3.1). Thus, chaining this derivation with the refutation given by assumption in PK_d, we have

$$\{\varphi\} \cup \{\pi = P(\varphi, \alpha, s)\} \vdash_{\mathsf{Res}} \{\textsc{Ref}_t(\textsc{Ref}_s(\varphi), \pi)\} \vdash_{\mathsf{PK}_d} \bot, \tag{7.3}$$

in size $\mathrm{poly}(n, m, s, \ell)$ for t the aforementioned polynomial $p(n, m, s)$.

Since once φ is fixed the conjunct $\pi = P(\varphi, \alpha, s)$ is simply defining every variable in π in terms of a conjunction of two literals, we can simply substitute these small conjunctions in the proof, getting a refutation in PK^2_{d+1}.

The proof of the second item is analogous, except in this last step, when substituting the extensions given by π, we start from a derivation in $\mathsf{PK}_{d+1/2}$ of size $S := \mathrm{poly}(n, m, s, \ell)$, and Lemma 7.1 now guarantees that after substitution we stay in $\mathsf{PK}_{d+1/2}$ with a blow-up in size of $O(2^{\log S}) = O(S)$. □

7.2 The Ref formulas are complete for automatability

Arteche et al. [AAdRK25] proved that Resolution lower bound statements are complete for the automatability and weak automatability of strong propositional proof systems. In particular, this was proven for any so-called *reasonably strong* system: extensions of Extended Frege by some polynomial-time recognizable set of axioms.

In what follows, we shall say that a proof system S is *automatable in time T on* Ref *formulas* if there is an algorithm such that whenever there is a size-ℓ S-refutation of $\textsc{Ref}_s(\varphi)$, then the algorithm finds an S-refutation in time $T(\ell)$. For $\mathsf{PK}_{d+1/2}$ for large enough d, this immediately implies automatability on all CNF formulas.

Theorem 7.5 (Automatability on Ref formulas)**.** *Let $d \in \mathbb{N}$ and let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. If the proof system $\mathsf{PK}_{(d+3)+1/2}$ is automatable on* Ref *formulas in time T, then $\mathsf{PK}_{(d+3)+1/2}$ is automatable in time $\mathrm{poly}(T(\mathrm{quasipoly}(\cdot)))$ on all CNF formulas.*

Proof. We assume there is an automating algorithm on Ref formulas and describe the algorithm that extends this to all formulas. If φ is a CNF formula and has $\mathsf{PK}_{(d+3)+1/2}$-refutations in size ℓ, then by Theorem 7.2.(ii), we have that there exists a $\mathsf{PK}_{(d+3)+1/2}$-refutation of $\textsc{Ref}_{p(n,m,s)}(\textsc{Ref}_{n^3}(\varphi))$ in size $\ell' := \mathrm{quasipoly}(n, m, s, t, \ell)$, where p is the polynomial in Theorem 7.4. We use the automating algorithm on Ref formulas to find a refutation of this formula in time $T(\ell')$. Then, by Theorem 7.4.(ii), we can obtain a $\mathsf{PK}_{(d+3)+1/2}$-refutation of φ in time $\mathrm{poly}(T(\ell'))$. □

For T a function of quasipolynomial growth-rate, we can give a cleaner statement.

Corollary 7.6. *Let $d \in \mathbb{N}$. The proof system $\mathsf{PK}_{(d+3)+1/2}$ is automatable in quasipolynomial time on all CNF formulas if, and only if, it is automatable on* Ref *formulas in quasipolynomial time.*

If instead of automatability we are interested in *weak automatability*, the situation is a bit more general.

Theorem 7.7 (Weak automatability on Ref formulas). *Let $d \in \mathbb{N}$, let $T : \mathbb{N} \to \mathbb{N}$ be a time-constructible function and let S be a propositional proof system closed under restrictions, modus ponens and substitutions of polylogarithmic-width clauses. Suppose that S simulates $\mathsf{PK}_{(d+3)+1/2}$. If S is weakly automatable on* Ref *formulas in time T, then S is weakly automatable on all CNF formulas in time* $\mathrm{poly}(T(\mathrm{quasipoly}(\cdot)))$.

Proof. Suppose S is p-simulated by Q and Q is automatable on Ref formulas. Now, if φ is a CNF formula and has S-refutations in size ℓ, then by Theorem 7.2.(ii) and the closure and substitutions properties, we have that there exists an S-refutation of $\mathrm{Ref}_{p(n,m,s)}(\mathrm{Ref}_{n^3}(\varphi))$ in size $\ell' := \mathrm{quasipoly}(n, m, s, t, \ell)$, where p is the polynomial in Theorem 7.4. We use the automating algorithm on Ref formulas for Q to find a refutation of this formula in time $T(\ell')$ in Q. Then, we turn this into a refutation in $Q' := \mathsf{EF} + \mathrm{Refl}^{Q} \geq_{\mathrm{p}} Q$ [Kra19, Theorem 8.4.3], where Theorem 7.4 is available and we have all the necessary closure properties. We can now obtain a Q'-refutation of φ in time $\mathrm{poly}(T(\ell'))$, so Q' weakly automates S. □

8 The Proof Analysis Problem for constant-depth Frege is NP-hard

In this section, we show that the Proof Analysis Problem for constant-depth Frege is **NP**-hard. Recall that a proof system S is said to be *analyzable* if given an S-proof of a Resolution lower bound on φ, we can decide in polynomial time whether φ is satisfiable. Formally, this is the decision problem PAP_S from Section 2.3.2. If a proof system is both automatable and analyzable, then **P** = **NP**; this makes analyzability a generalization of the Atserias–Müller strategy for proving **NP**-hardness of automatability. As discussed in Section 1.1.3, showing that a proof system is hard to analyze suggests that the Atserias–Müller approach of using Ref formulas to show **NP**-hardness of automatability cannot work for $\mathbf{AC}^0$-Frege (as long as Ref is the formula for Resolution lower bounds). To show that a proof system is *not* analyzable, we first show that it is able to prove fixed polynomial-size Resolution lower bounds.

In Section 8.1, we prove that for the particular case of the pigeonhole principle formulas, lower bounds in the exponential-size regime imply lower bounds in the polynomial-size regime. In Section 8.2 we show that the Proof Analysis Problem for constant-depth Frege is **NP**-hard.

8.1 Haken's lower bound in the polynomial-size regime

Building on the formalizations of Li, Li, and Ren [LLR26] in $\mathsf{T}^1_2(\alpha)$ + $\mathrm{dWPHP}(\mathsf{PV}_1(\alpha))$, we show how for the particular case of the pigeonhole principle formulas, exponential lower bounds can be turned into fixed polynomial lower bounds with a mild increase in the depth of the proof.

Let us recall the definition of the CNF formulas encoding 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

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

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

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

Haken [Hak85] proved that every Resolution refutation of PHP_n requires size $2^{\Omega(n)}$. The result can be formalized in $\mathsf{T}^1_2(\alpha)$ + $\mathrm{dWPHP}(\mathsf{PV}_1(\alpha))$, assuming $n \in \mathrm{Log}$.

Theorem 8.1 (Haken's lower bound in the exponential-size regime [LLR26, Theorem 3.17]). *There exists a positive $\varepsilon \in \mathbb{Q}$ such that $\mathsf{T}^1_2(\Pi)$ + $\mathrm{dWPHP}(\mathsf{PV}_1(\Pi))$ proves the following statement: for every t and every $n \in \mathrm{Log}$, if Π encodes a length-t Resolution refutation of PHP_n, then $t > 2^{\varepsilon \cdot n}$.*

By applying the Paris–Wilkie translation (Theorem 2.5) to $\mathsf{T}^2_2(\Pi) \supseteq \mathsf{T}^1_2(\Pi) + \mathrm{dWPHP}(\mathrm{PV}_1(\Pi))$, we immediately get the following upper bound in Res(log).

Corollary 8.2. *There exists a positive $\varepsilon \in \mathbb{Q}$ such that the sequence of formulas $\{\mathrm{REF}_{2^{\varepsilon n}}(\mathrm{PHP}_n)\}_{n\in\mathbb{N}}$ admits* QP*-uniform quasipolynomial-size refutations in* Res(log).

The formulas $\mathrm{REF}_{2^{\varepsilon n}}(\mathrm{PHP}_n)$ have size exponential in n, so it is not obvious how one could derive directly a formula of the form $\mathrm{REF}_{n^k}(\mathrm{PHP}_n)$. The following theorem shows this is possible for the case of PHP_n if we go slightly above Res(log).

Theorem 8.3. *For every $k \in \mathbb{N}$, the formulas $\{\mathrm{REF}_{n^k}(\mathrm{PHP}_n)\}_{n\in\mathbb{N}}$ admit* QP*-uniform PK_2-refutations in quasipolynomial size.*

Proof. Fix $k \in \mathbb{N}$ and let $\varepsilon \in \mathbb{Q}$ be the constant in Corollary 8.2. Let $N := \lceil \log n^{k/\varepsilon} \rceil$ and consider a partial matching ρ from $[n+1]$ to $[n]$, so that as a restriction over the variables of PHP_n it leaves $N+1$ pigeons and N holes unassigned. That is, $\mathrm{PHP}_n\restriction_\rho \equiv \mathrm{PHP}_N$. The goal is to derive the same fact inside the proof system.

Since $n^k \leq 2^{\varepsilon N}$, we have that by Proposition 6.2,

$$\mathrm{REF}_{n^k}(\mathrm{PHP}_n, \pi) \wedge (\pi' = \pi_{\restriction\rho}) \vdash^{n^{O(1)}}_{\mathsf{Res}} \mathrm{REF}_{2^{\varepsilon N}}(\mathrm{PHP}_N, \pi'), \tag{8.1}$$

and by Corollary 8.2,

$$\mathrm{REF}_{2^{\varepsilon N}}(\mathrm{PHP}_N, \pi') \vdash^{n^{(\log n)^{O(1)}}}_{\mathsf{Res}(\log)} \bot. \tag{8.2}$$

Together, these imply

$$\mathrm{REF}_{n^k}(\mathrm{PHP}_n, \pi) \wedge (\pi' = \pi_{\restriction\rho}) \vdash^{n^{(\log n)^{O(1)}}}_{\mathsf{Res}(\log)} \bot. \tag{8.3}$$

In order to get rid of the π' variables we need to substitute them in the proof. Since these come from the $\Delta^b_2(\Phi,\Pi)$-definition in Proposition 6.1 and ρ is fixed, we get that the conjunct $(\pi' = \pi_{\restriction\rho})$ amounts to a depth-2 formula of polynomial size. By Lemma 7.1, when substituting this into a Res(log) refutation, we get a PK_2-refutation in quasipolynomial size. □

This restriction trick for PHP_n works because we can send the formula to an exponentially smaller subinstance of itself. This appears problematic when dealing with the REF formulas, as we cannot control the innermost formula φ. When looking at $\mathrm{REF}_s(\varphi)$, we can disable blocks to send the formula to an exponentially smaller subformula $\mathrm{REF}_{\log s}(\varphi)$, but we have no way of restricting φ.

More formally, let us suppose we want to obtain fixed-polynomial lower bounds, of the form encoded by $\neg\mathrm{REF}_{(s/n)^k}(\mathrm{REF}_s(\varphi))$. Then, let $\sigma := n\log(s/n)^{k/\varepsilon}$, so that $(s/n)^k = 2^{\varepsilon\sigma/n}$. By a restriction argument analogous to eq. (8.1), we can start from $\mathrm{REF}_{(s/n)^k}(\mathrm{REF}_s(\varphi))$ and derive $\mathrm{REF}_{2^{\varepsilon\sigma/n}}(\mathrm{REF}_\sigma(\varphi))$. To complete the proof we need a refutation of this last formula. Theorem 6.5 yields such a refutation in quasipolynomial-size Res(log), but this is no simplification: the formalization in the exponential-size regime uses the exact same proof as Theorem 6.4 without really exploiting the fact that the formula parameters are in Log (that is, we still use limited independence and the more sophisticated restriction argument in Section 4, so we can derive fixed-polynomial lower bounds directly from this argument without the restriction trick).

For the trick in eq. (8.1) to constitute a simplification, we would need to obtain a simpler formalization of the restriction argument in Lemma 3.6 exploiting the fact that the parameters n and s are in Log. The advantage in this regime is that we can carry out the counting arguments exactly. Since the disabling restrictions in Lemma 3.6 are strings of the form $\rho \in \{0,1\}^{2s}$, the space of restrictions is of size 2^{2s}. For this to be quasipolynomial in the size of the formula $\mathrm{REF}_{2^{\varepsilon s/n}}(\mathrm{REF}_s(\varphi))$, we need $2^{2s} = \mathrm{quasipoly}(2^{s/n})$. Unfortunately, this latter requirement makes the restriction trick in Theorem 8.3 impossible: we would

restrict to a formula where s becomes $\sigma = n \log(s/n)^{k/\varepsilon}$. If s is, say, $n^{O(1)}$, then this gives $\sigma = \Theta(n \log n)$, and the requirement becomes $\Theta(n \log n) \leq \mathrm{poly}(\log n)$, which is impossible.

8.2 The hardness reduction

The **NP**-hardness proof follows the ideas of Arteche et al. [AAdRK25] and the streamlined presentation sketched in Section 1.3.3. Formally, this is captured by the following methodology theorem. The statement below is a slight modification of the original one in [AAdRK25], tailored to constant-depth subsystems of PK. For the sake of completeness, we give a formal proof here as well.

Theorem 8.4 (General **NP**-hardness of PAP [AAdRK25, Corollary 5.7])**.** *Let* $\mathrm{Res} \leq_{\mathrm{p}} S_1 \leq_{\mathrm{p}} S_2$ *be propositional proof systems such that substituting polynomial-size depth-2 formulas in an* S_1*-proof results in an* S_2*-proof with at most a polynomial blow-up. Suppose there exists a constant* $c > 0$*, a time-constructible function* $t : \mathbb{N} \to \mathbb{N}$ *and a* $\mathbf{TIME}[t(n)]$*-uniform sequence of unsatisfiable formulas* $\{\psi_n\}_{n\in\mathbb{N}}$ *such that* ψ_n *has at most* n *variables and* $\mathrm{poly}(n)$ *clauses and there exists a* $\mathbf{TIME}[t(n)]$*-uniform sequence of* S_1*-proofs* $\{\pi_n\}_{n\in\mathbb{N}}$ *such that* $\pi_n : S_1 \vdash \neg\mathrm{Ref}_{n^{c+1}}(\psi_n)$*. Then, the problem* $\mathrm{PAP}_{S_2}[n^c]$ *is* **NP***-hard under many-one reductions computable in time* $\mathrm{poly}(t(n))$*.*

Proof. For every two CNF formulas $\varphi = C_1 \wedge \cdots \wedge C_k$ and $\psi = D_1 \wedge \cdots \wedge D_m$ over disjoint variables, let us denote by $\mathrm{disj}(\varphi, \psi)$ the formula

$$\mathrm{disj}(\varphi, \psi) \coloneqq \bigwedge_{i=1}^{k} (\neg y \vee C_i) \wedge \bigwedge_{i=1}^{m} (y \vee D_i), \tag{8.4}$$

where y is a fresh variable not present in φ or ψ.

By Proposition 6.2 and the assumption that S_1 proves Resolution lower bounds on ψ we have that for every CNF formula φ over n variables,

$$\mathrm{Ref}_{n^{c+1}}(\mathrm{disj}(\varphi, \psi_n), \pi) \wedge (\pi' = \pi_{\restriction\rho}) \vdash_{\mathrm{Res}} \mathrm{Ref}_{n^{c+1}}(\psi_n, \pi') \vdash_{S_1} \bot, \tag{8.5}$$

where ρ is fixed to be the restriction that sets $\rho : y \mapsto 0$ and sets all variables of φ to 1. Note that the first derivation has size $\mathrm{poly}(n)$ and the second derivation has size $O(t(n))$.

Since the conjunct $(\pi' = \pi_{\restriction\rho})$ comes from the $\Delta_2^b(\Phi, \Pi)$-definition in Proposition 6.1 and ρ is fixed, we get that this amounts to a depth-2 formula of polynomial size, which can be substituted to get an S_2-refutation of $\mathrm{Ref}_{n^{c+1}}(\mathrm{disj}(\varphi, \psi_n), \pi)$ in size $\mathrm{poly}(t(n))$.

The **NP**-hardness reduction is now from SAT and maps a CNF formula over n variables to the instance $(\mathrm{disj}(\varphi, \psi_n), \tau_n, 1^{N^c})$ of $\mathrm{PAP}_{S_2}[N^c]$, where $\tau_n : S_2 \vdash \neg\mathrm{Ref}_{N^c}(\mathrm{disj}(\varphi, \psi_n), \pi)$, N is the number of variables of $\mathrm{disj}(\varphi, \psi_n)$. Note that for large enough n, we have $N^c = (2n+1)^c \leq n^{c+1}$, so the n^{c+1} Resolution lower bound at hand is good enough. It is easy to see that φ is satisfiable if and only if $\mathrm{disj}(\varphi, \psi_n)$ is, and τ_n is always a correct S_2-proof. This completes the proof. □

This yields the desired **NP**-hardness reduction.

Theorem 8.5. *Let* Q *be a propositional proof system that p-simulates* PK_2*. For every* $c > 0$*, the problem* $\mathrm{PAP}_Q[n^c]$ *is* **NP***-hard under quasipolynomial-time many-one reductions.*

Proof. Consider $m \coloneqq \lceil {}^{c+1}/_{\varepsilon} \log n \rceil$ and the formulas PHP_m. By Corollary 8.2, we have Res(log) refutations of $\mathrm{Ref}_{2^{\varepsilon m}}(\mathrm{PHP}_m)$, which is precisely $\mathrm{Ref}_{n^{c+1}}(\mathrm{PHP}_m)$.

Since Lemma 7.1 guarantees that substituting depth-2 formulas in $\mathrm{PK}_{0+1/2}$-proofs results in PK_2-proofs with a quasipolynomial increase in size, we can take $S_1 \coloneqq \mathrm{PK}_{0+1/2}$ and $S_2 \coloneqq \mathrm{PK}_2$ and invoke Theorem 8.4 to conclude that $\mathrm{PAP}_{\mathrm{PK}_2}[n^c]$ is **NP**-hard under quasipolynomial-time many-one reductions. Since $Q \geq_{\mathrm{p}} \mathrm{PK}_2$, we immediately extend the **NP**-hardness reduction from $\mathrm{PAP}_{\mathrm{PK}_2}[n^c]$ to $\mathrm{PAP}_Q[n^c]$. □

Acknowledgements

We are grateful to Albert Atserias for useful initial discussions, and we thank Jonas Conneryd for helpful comments and pointers. We thank Sam Buss and Jan Krajíček for kindly answering technical questions about details of the Paris–Wilkie translation. We also thank the anonymous FOCS reviewers for helpful comments that improved the clarity of the presentation.

Noel Arteche was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. 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". Kilian Risse was supported by the Swiss National Science Foundation Postdoc.Mobility fellowship P500-2_235298. We also gratefully acknowledge that we have benefited greatly from being part of the Basic Algorithms Research Copenhagen (BARC) center financed by the Villum Investigator grant 54451.

AI Disclosure

The authors used GPT 5.6-Sol only to assist with typesetting and generating initial versions of Figure 1 and Figure 2, as well as polishing the presentation of some of the proofs and finding small errors.

References

[AAdRK25] N. Arteche, A. Atserias, S. F. de Rezende, and E. Khaniki, "The proof analysis problem," in *2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS)*, Full version available at https://arxiv.org/abs/2506.16956, 2025, pp. 2558–2579.

[AB04] 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.

[ABI86] N. Alon, L. Babai, and A. Itai, "A fast and simple randomized parallel algorithm for the maximal independent set problem," *J. Algorithms*, vol. 7, no. 4, pp. 567–583, 1986.

[ACG24] 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, 15:1–15:25.

[AD08] A. Atserias and V. Dalmau, "A combinatorial characterization of resolution width," *J. Comput. System Sci.*, vol. 74, no. 3, pp. 323–334, 2008.

[AKPS24] 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, Art. No. 12, 20.

[ALWZ21] 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.

[AM20] A. Atserias and M. Müller, "Automating resolution is **NP**-hard," *Journal of the ACM (JACM)*, vol. 67, no. 5, pp. 1–17, 2020.

[AS16] N. Alon and J. H. Spencer, *The probabilistic method* (Wiley Series in Discrete Mathematics and Optimization), Fourth. John Wiley & Sons, Inc., Hoboken, NJ, 2016.

[AT25] A. Atserias and I. Tzameret, "Feasibly constructive proof of Schwartz-Zippel lemma and the complexity of finding hitting sets," in *STOC'25—Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, ACM, New York, 2025, pp. 1096–1107.

[BB05] A. Beckmann and S. R. Buss, "Separation results for the size of constant-depth propositional proofs," *Ann. Pure Appl. Logic*, vol. 136, no. 1-2, pp. 30–55, 2005.

[BB19] A. Beckmann and S. Buss, "On transformations of constant depth propositional proofs," *Ann. Pure Appl. Logic*, vol. 170, no. 10, pp. 1176–1187, 2019.

[BDG+04] 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.

[Bea94] P. Beame, "A switching lemma primer," Department of Computer Science and Engineering, University of Washington, Tech. Rep., 1994.

[Bel20] Z. Bell, "Automating regular or ordered resolution is NP-hard," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR20-105, 2020.

[Ben09] E. Ben-Sasson, "Size-space tradeoffs for resolution," *SIAM J. Comput.*, vol. 38, no. 6, pp. 2511–2525, 2009.

[BP96] 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.

[BPR00] 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.

[BPT14] A. Beckmann, P. Pudlák, and N. Thapen, "Parity games and propositional proofs," *ACM Trans. Comput. Logic*, vol. 15, no. 2, May 2014.

[Bus04] S. R. Buss, "Bounded arithmetic and constant depth Frege proofs," in *Complexity of computations and proofs*, ser. Quad. Mat. Vol. 13, Dept. Math., Seconda Univ. Napoli, Caserta, 2004, pp. 153–174.

[Bus86] S. R. Buss, *Bounded arithmetic*. Bibliopolis, Naples, 1986.

[CGH+85] B. Chor, O. Goldreich, J. Håstad, J. Freidmann, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *26th Annual Symposium on Foundations of Computer Science (SFCS 1985)*, 1985, pp. 396–407.

[CLO24] L. Chen, J. Li, and I. C. Oliveira, "Reverse mathematics of complexity lower bounds," in *2024 IEEE 65th Annual Symposium on Foundations of Computer Science—FOCS 2024*, 2024, pp. 505–527.

[CLOW26] L. Chen, J. Li, I. C. Oliveira, and R. Williams, "A theory for probabilistic polynomial-time reasoning," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR26-015, 2026.

[CN10] S. Cook and P. Nguyen, *Logical foundations of proof complexity*. Cambridge University Press, 2010.

[Cob65] 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.

[Coo71] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71, Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158, ISBN: 9781450374644.

[Coo75] 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.

[CP90] S. Cook and T. Pitassi, "A feasibly constructive lower bound for resolution proofs," *Information Processing Letters*, vol. 34, no. 2, pp. 81–85, 1990.

[CR79] 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.

[dGN+21] S. F. de Rezende, M. Göös, J. Nordström, T. Pitassi, R. Robere, and D. Sokolov, "Automating algebraic proof systems is **NP**-hard," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 209–222.

[DR03] S. Dantchev and S. Riis, "On relativisation and complexity gap for resolution-based proof systems," in *Computer science logic*, ser. Lecture Notes in Comput. Sci. Vol. 2803, Springer, Berlin, 2003, pp. 142–154.

[DR26] B. Davis and R. Robere, "Res(log) proves bounded-depth frege lower bounds," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR26-055, 2026.

[Gar19] M. Garlík, "Resolution lower bounds for refutation statements," in *44th International Symposium on Mathematical Foundations of Computer Science*, 2019, Art. No. 37, 13.

[Gar24] M. Garlík, "Failure of feasible disjunction property for k-DNF resolution and **NP**-hardness of automating it," in *39th Computational Complexity Conference, CCC 2024, July 22-25, 2024, Ann Arbor, MI, USA*, vol. 300, 2024, 33:1–33:23.

[Gay21] A. Gaysin, "H-coloring dichotomy in proof complexity," *Journal of Logic and Computation*, vol. 31, no. 5, pp. 1206–1225, Apr. 2021.

[Gay23] A. Gaysin, *Proof complexity of CSP*, 2023. arXiv: 2201.00913.

[Gay24] A. Gaysin, *Proof complexity of universal algebra in a CSP dichotomy proof*, 2024. arXiv: 2403.06704.

[GC25] S. Grosser and M. Carmosino, "Student-teacher constructive separations and (un)provability in bounded arithmetic: Witnessing the gap," in *STOC'25—Proceedings of the 57th Annual ACM Symposium on Theory of Computing*, 2025, pp. 1341–1347.

[GKMP20] M. Göös, S. Koroth, I. Mertz, and T. Pitassi, "Automating cutting planes is **NP**-hard," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 68–77.

[Hak85] A. Haken, "The intractability of resolution," *Theoret. Comput. Sci.*, vol. 39, no. 2-3, pp. 297–308, 1985.

[HP93] P. Hájek and P. Pudlák, *Metamathematics of first-order arithmetic*. Springer-Verlag, Berlin, 1993.

[IR22] D. Itsykson and A. Riazanov, "Automating OBDD proofs is **NP**-hard," in *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, vol. 241, 2022, 59:1–59:15.

[Jeř05] E. Jeřábek, "Weak pigeonhole principle, and randomized computation," Ph.D. dissertation, Faculty of Mathematics and Physics, Charles University, Prague, 2005.

[Jeř07] E. Jeřábek, "Approximate counting in bounded arithmetic," *Journal of Symbolic Logic*, vol. 72, no. 3, pp. 959–993, 2007.

[Jeř09] E. Jeřábek, "Approximate counting by hashing in bounded arithmetic," *J. Symbolic Logic*, vol. 74, no. 3, pp. 829–860, 2009.

[JJ25] R. Jalali and O. Ježil, "Feasibility of primality in bounded arithmetic," 2025. arXiv: 2504.17041.

[JT26] O. Ježil and D. Tsintsilidas, *Parallelism and adaptivity in Student-Teacher witnessing*, 2026. arXiv: 2602.19934.

[Kha22a] 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.

[Kha22b] E. Khaniki, "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, 17:1–17:15.

[Kha24] 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.

[KP98] 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.

[KPT91] 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.

[Kra01] J. Krajíček, "On the weak pigeonhole principle," *Fundamenta Mathematicae*, vol. 170, no. 1-2, pp. 123–140, 2001.

[Kra11] J. Krajíček, "A note on propositional proof complexity of some Ramsey-type statements," *Arch. Math. Logic*, vol. 50, no. 1-2, pp. 245–255, 2011.

[Kra19] J. Krajíček, *Proof complexity*. Cambridge University Press, Cambridge, 2019.

[Kra23] J. Krajíček, "A proof complexity conjecture and the incompleteness theorem," *The Journal of Symbolic Logic*, pp. 1–5, 2023.

[Kra24] J. Krajíček, "Extended Nullstellensatz proof systems," *Proceedings of the American Mathematical Society*, vol. 152, pp. 4881–4892, 2024.

[Kra94] J. Krajíček, "Lower bounds to the size of constant-depth propositional proofs," *J. Symbolic Logic*, vol. 59, no. 1, pp. 73–86, 1994.

[Kra95] J. Krajíček, *Bounded arithmetic, propositional logic, and complexity theory*. Cambridge University Press, Cambridge, 1995.

[Kra99] J. Krajíček, *Seminar notes on Maciel–Pitassi–Woods*, https://www.karlin.mff.cuni.cz/~krajicek/mpw.ps, Seminar notes (15.11.1999). Accessed: 2026-03-24, 1999.

[LLR26] J. Li, Y. Li, and H. Ren, "Finding bugs in short proofs: The metamathematics of resolution lower bounds," in *STOC'26—Proceedings of the 58th Annual ACM Symposium on Theory of Computing*, ACM, New York, [2026] ©2026, pp. 788–798, ISBN: 979-8-4007-2536-4.

[LST26] J. Lu, R. Santhanam, and I. Tzameret, "$AC^0[p]$-Frege cannot efficiently prove that constant-depth algebraic circuit lower bounds are hard," in *17th Innovations in Theoretical Computer Science Conference*, ser. LIPIcs. Leibniz Int. Proc. Inform. Vol. 362, 2026, Art. No. 99, 25.

[Lub86] M. Luby, "A simple parallel algorithm for the maximal independent set problem," *SIAM J. Comput.*, vol. 15, no. 4, pp. 1036–1053, 1986.

[MP20] M. Müller and J. Pich, "Feasibly constructive proofs of succinct weak circuit lower bounds," *Annals of Pure and Applied Logic*, vol. 171, no. 2, p. 102 735, 2020.

[MPW02] A. Maciel, T. Pitassi, and A. R. Woods, "A new proof of the weak pigeonhole principle," *J. Comput. System Sci.*, vol. 64, no. 4, pp. 843–872, 2002, Special issue on STOC 2000 (Portland, OR).

[Mül21] M. Müller, "Typical forcings, NP search problems and an extension of a theorem of Riis," *Annals of Pure and Applied Logic*, vol. 172, no. 4, p. 102 930, 2021.

[Nar22] M. Narusevych, *Models of bounded arithmetic and variants of pigeonhole principle*, 2022. arXiv: 2208.14713.

[Nar24] M. Narusevych, *An independence of the MIN principle from the PHP principle*, 2024. arXiv: 2406.14930.

[NW94] N. Nisan and A. Wigderson, "Hardness vs randomness," *Journal of Computer and System Sciences*, vol. 49, no. 2, pp. 149–167, 1994.

[Oli25] I. C. Oliveira, "Meta-mathematics of computational complexity theory," *Electronic Colloquium on Computational Complexity (ECCC)*, no. TR25-041, 2025.

[Pap24] T. Papamakarios, "Depth-d Frege systems are not automatable unless **P** = **NP**," in *39th Computational Complexity Conference (CCC 2024)*, vol. 300, 2024, 22:1–22:17.

[PP24] J. Park and H. T. Pham, "A proof of the Kahn–Kalai conjecture," *J. Amer. Math. Soc.*, vol. 37, pp. 235–243, 2024.

[PS19] J. Pich and R. Santhanam, "Why are proof complexity lower bounds hard?" In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science*, IEEE Comput. Soc. Press, Los Alamitos, CA, 2019, pp. 1305–1324.

[PS22] 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, 101:1–101:20.

[PS23] J. Pich and R. Santhanam, *Towards* **P** $\neq$ **NP** *from Extended Frege lower bounds*, 2023. arXiv: 2312.08163.

[Pud00] P. Pudlák, "Proofs as games," *Amer. Math. Monthly*, vol. 107, no. 6, pp. 541–550, 2000.

[Pud03] P. Pudlák, "On reducibility and symmetry of disjoint **NP** pairs," *Theoretical Computer Science*, vol. 295, no. 1-3, pp. 323–339, 2003.

[Pud20] P. Pudlák, *Reflection principles, propositional proof systems, and theories*, 2020. arXiv: 2007.14835.

[PW85] J. Paris and A. Wilkie, "Counting problems in bounded arithmetic," in *Methods in mathematical logic (Caracas, 1983)*, ser. Lecture Notes in Math. Vol. 1130, Springer, Berlin, 1985, pp. 317–340.

[PWW88] J. B. Paris, A. J. Wilkie, and A. R. Woods, "Provability of the pigeonhole principle and the existence of infinitely many primes," *J. Symbolic Logic*, vol. 53, no. 4, pp. 1235–1244, 1988, ISSN: 0022-4812,1943-5886.

[Raz95] A. A. Razborov, "Bounded arithmetic and lower bounds in Boolean complexity," in *Feasible mathematics, II (Ithaca, NY, 1992)*, ser. Progr. Comput. Sci. Appl. Logic, vol. 13, Birkhäuser Boston, Boston, MA, 1995, pp. 344–386.

[ST21] 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.