Provable Reductions in TFNP

Noah Fleming *Lund & Columbia*
Stefan Grosser *McGill*
Toniann Pitassi *Columbia*
Robert Robere *McGill*

June 26, 2026

Abstract

We introduce a new family of propositional proof systems, denoted $\langle EF, R\rangle$, for an arbitrary TFNP search problem R. Informally, a refutation of a CNF formula F in $\langle EF, R\rangle$ is given by a polynomial-time mapping reduction from the *false-clause search problem* Search_F to R, combined with an Extended Frege proof that the reduction is correct. These new systems are naturally motivated in two ways:

1. They are the propositional translations of *witnessing theorems* in bounded arithmetic, by which proofs of $\forall\Sigma^b_1$ formulas ϕ in a theory T imply algorithms solving the search problem for ϕ in a TFNP subclass corresponding to T [Bus86, KP90, BK94, KST07, BB09].
2. They form a white-box analogue of the recent characterizations of proof systems using *decision tree reductions* to black-box TFNP problems [GHJ+22b, BIK+94, DR23, LPR24, FGJ+26, FIM25].

We consider the proof system $\langle EF, \textsc{Iter}\rangle$, where ITER is the complete problem for the classical TFNP subclass PLS. We prove that $\langle EF, \textsc{Iter}\rangle$ is polynomially equivalent to the quantified boolean sequent calculus G_1, and also to the implicit Resolution proof system $[EF, \mathrm{Resolution}]$ introduced by Krajíček [Kra04]. Hence G_1 and $[EF, \mathrm{Resolution}]$ are polynomially equivalent, which is the first natural characterization of an implicit proof system by a classical propositional proof system beyond the work of Wang [Wan13]. We further observe our characterization can be extended to capture G_i via the *game induction principles* of [ST07].

We also calibrate the strength of $\langle EF, R\rangle$ for general TFNP relations R. We observe that if Extended Frege can prove that a search problem R is in FP, then $\langle EF, R\rangle$ is polynomially equivalent to EF. This contrasts to our above result, which shows that Extended-Frege provable reductions to ITER, a problem widely believed *not* to be in FP, yields a proof system (G_1) that is believed to be stronger than Extended Frege.

Finally, and somewhat paradoxically, we show that for *any* proof system P which is sufficiently strong, there is a polynomial-time computable search problem $R_P \in$ FP such that $\langle EF, R_P\rangle$ is polynomially equivalent to P. Letting $P = [EF, \mathrm{Resolution}]$ and combining our two results shows that $\langle EF, \textsc{Iter}\rangle$ is polynomially equivalent to $\langle EF, R_{[EF,\mathrm{Resolution}]}\rangle$. Hence, despite the widely-believed conjecture that ITER $\notin$ FP, the problems which EF-provably reduce to ITER are exactly the problems which EF-provably reduce to a fixed polynomial-time computable set.

Contents

1 Introduction

A central goal in propositional proof complexity is to understand the relative power of proof systems and, ultimately, to separate NP from coNP by proving superpolynomial lower bounds on proof lengths. Despite decades of progress on concrete systems such as Resolution, bounded-depth Frege, and Cutting Planes, our understanding remains fragmented: the proof systems for which we have unconditional lower bounds captures a fairly narrow class of relatively weak proof systems, and extending these lower bounds even for AC^0_p-Frege systems has been a longstanding open problem.

More fundamentally, even if we managed to develop techniques for proving lower bounds for stronger proof systems (such as Frege and Extended Frege (EF) systems), it is conjectured that there is no single optimal proof system. On the other hand, despite much effort, there is still no evidence of a family of concrete tautologies (beyond reflection principles) that are believed to be hard for EF but easy for a stronger proof system.

In a beautiful paper [Kra04] (see also [Kra01a]), Krajíček proposed a method to create stronger propositional proof systems from weaker ones by encoding proofs in a weaker proof system implicitly as succinct circuits, together with short certificates of their correctness. Concretely, given a Cook-Reckhow propositional proof system P, the *implicitization operator* applied to P gives a new *implicit* proof system, denoted iP (cf. Section 2). An iP proof is a pair (Π, C) such that C is a succinct representation of a P proof, and Π is a P-proof that C encodes a valid P-proof. By analogy to jump operators in classical logic, Krajicek conjectured that implicit proofs may give a way to create a *strictly* stronger proof system from a weak one. Iterating this construction generates a natural hierarchy of increasingly powerful systems which closely mirror the reflection or consistency hierarchies in bounded arithmetic. Therefore, implicit proofs can be seen as a concrete framework for studying how much additional strength we get by augmenting a proof system with the ability to reason about its own proofs.

Viewed from the lens of complexity theory, the idea behind implicit proofs is analogous to the definitions of succinct complexity classes. Formally, if L is a language, then we can define the *succinct* version of L as follows: given a polynomial-size circuit C as input, decide if $tt(C) \in L$, where $tt(C)$ is the truth table of C obtained by evaluating C on all possible inputs. For example, it is well-known that given any NP-complete problem, its succinct version is NEXP-complete, and similarly for any problem that is complete for P under projections, its succinct version is EXP-complete. In both cases, compression is used to "jump" from a lower complexity class or weaker proof system to a stronger complexity class or proof system. Notably, compressed/succinct versions of complexity classes are known to be provably stronger than the base class, via time hierarchy theorems, whereas in contrast, it is a major open problem whether the succinct versions of proof systems are generally stronger than the original proof systems.

Despite the intrinsic appeal and motivation behind the implicit proof framework, very little is currently known about their strength, and how they relate to more standard proof systems. At the high end (i.e. for strong proof systems that can simulate EF), Pudlák [Pud20]

recently conjectured a striking connection between implicit Extended Frege proofs and the consistency extension of Buss's theory S^1_2 of bounded arithmetic: iterated implicit operations on EF, i^kEF, capture the strength of $S^1_2 + \mathrm{Con}(S^1_2)$. Specifically, Pudlák suggests that the provable $\forall\Sigma^b_1$-formulas in $S^1_2 + \mathrm{Con}(S^1_2)$ are exactly those which have propositional proofs in i^kEF, for some $k \geq 0$. Other work has studied the strength of iEF as a tool for formalizing results in complexity theory. Khaniki [Kha24] showed that iEF is strong enough to formalize the soundness of the sum-check protocol. This was further used [AKPS24] to show that if iEF proves strong derandomization assumptions, then iEF lower bounds imply $\#\mathsf{P} \not\subset \mathsf{FP}/\mathrm{poly}$. Given these results and the jump operator origins of Krajíček, it is clearly important to understand the strength of the implicit operator and what implicit proofs can formalize.

At the low end – for proof systems seemingly weaker than Extended Frege, much less is known or conjectured. Wang [Wan13] proved that implicit tree-like Resolution is polynomial-equivalent to EF[1] However, analogous characterizations have not yet been shown for other standard propositional proof systems. For example, for well-studied propositional proof systems such as Resolution or Frege, what can we say about their implicit versions, $i\mathrm{Res}$ and $i\mathrm{Frege}$? Can they be characterized by a natural proof system? How strong are they?

1.1 Main Results

We obtain characterizations of implicit propositional proof systems at the low end, for a variety of natural proof systems. As a concrete example, we prove the following theorem, and will use it as a running example. Recall that G is a proof system for reasoning about *quantified* boolean formulas. Formally, G is a sequent calculus where each individual formula is a quantified boolean formula, and G_i is the fragment of G in which cuts are restricted to formulas of the form $\exists\vec{x}_1\forall\vec{x}_2\cdots Q_i\vec{x}_i F(\vec{x}_1,\ldots,\vec{x}_i,\vec{y})$, where Q_i is $\exists$ if i is odd and $\forall$ if i is even. For proving ordinary propositional statements, G_1 can be viewed as an extension of Extended Frege where proofs can reason with the more general family of quantified Boolean formulas of the form $\exists\overline{y}F(\overline{x},\overline{y})$, where F is a propositional formula. We also note that Krajíček [Kra01a] observed that Implicit Resolution is equivalent to the (seemingly stronger) system $[EF, \mathrm{Resolution}]$, in which the proof of correctness of the succinct Resolution proof C is in the system EF, rather than Resolution. Our first main theorem is the following characterization of Implicit Resolution.

Theorem 1.1 (cf. Theorem 4.1)**.** *Implicit Resolution, and equivalently* $[EF, \mathrm{Resolution}]$*, is polynomially-equivalent to* G_1*.*

This generalizes Wang's result for tree-like Resolution to general (dag-like) Resolution.

Methodology. The proof of our theorem crucially relies on new tools that we develop using the theory of *total functions in* NP (TFNP), which has had an expanding role on propo-

[1] Wang refers to implicit tree-like Resolution as "implicit Resolution".

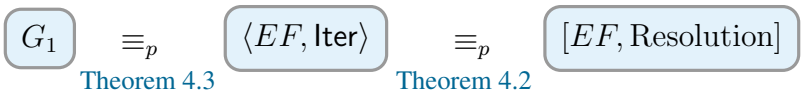
sitional proof complexity in recent years [GHJ+22b, GMRS25, LPR24, BFI23, GKRS19, DR23, HKT24, FIM25, FGJ+26, FGIM26, FGPR24, Kam19]. In particular, we take inspiration from a recent line of work (e.g., [GKRS19, BFI23]) showing that black-box (low-depth decision tree) reductions to black-box TFNP problem are equivalent to short proofs in a corresponding propositional proof system. Formally speaking, if $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ is an unsatisfiable CNF formula, we can associate with F the following *false-clause search problem* Search_F: given an assignment x to the variables of F, output the index i of a false clause $C_i(x) = 0$. It has long been known that efficient decision trees for Search_F correspond directly to tree-like Resolution refutations for F. Indeed, the *totality* of the search problem Search_F (for every input x, there is a valid output i such that $C_i(x) = 0$) is equivalent to the fact that F is an unsatisfiable formula. Recent work has leveraged this observation to show that other classical propositional proof systems can be also captured, now by giving efficient decision-tree *reductions* from Search_F to other search problems which are known to be total. For instance, the following theorem of Kamath shows how to capture low-width Resolution refutations in this way via reductions to the *black-box* ITERATION *problem* (see Section 2.4 for background on TFNP).

Theorem 1.2 (cf. [Kam19])**.** *Let F be an unsatisfiable CNF formula. If there is a width-w Resolution refutation of F, then there is a depth $O(w)$ reduction from Search_F to $\textsc{Iter}^{dt}$. Conversely, if there is a depth-w reduction from Search_F to $\textsc{Iter}^{dt}$, then there is a width $O(w)$ Resolution refutation of F.*

Our new characterizations of implicit proof systems proceed by considering a natural extension of the above idea. Instead of considering reductions from Search_F to other *black-box* TFNP problems, we will consider reductions from Search_F to classical *white-box* TFNP problems using normal polynomial-time algorithms. Unlike a decision tree reduction, we cannot explicitly verify that the reduction is correct, so we augment the proof system by adding an Extended Frege proof of correctness. Formally, for *any* TFNP relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$, we introduce a new propositional proof system $\langle EF, R\rangle$ defined as follows.

Definition 1.3 (Informal, see Definition 3.5)**.** Let $F(x_1, \ldots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an unsatisfiable CNF formula, and let $R \in$ TFNP be any total NP search problem. An $\langle EF, R\rangle$ *refutation* of F is a tuple (C, D, Π), where C and D are polynomial-size circuits encoding a mapping reduction from Search_F to R, and Π is an Extended Frege proof that C and D are a correct mapping reduction. We call (C, D, Π) an *EF-provable mapping reduction* from Search_F to R.

We argue that this definition is the correct "white-box analogue" of the decision-tree reductions used in the black-box setting. (Indeed, the previous definition can be viewed as an implicit version of the proof systems that can be characterized by black-box decision tree reductions, as studied by [BFI23].) In Section 4, we prove Theorem 1.1 in two steps, using $\langle EF, R\rangle$ systems as a crucial intermediate step. Below, the problem ITER is the canonical ITERATION problem, which is known to be complete for the TFNP class PLS.



A fruitful way of interpreting the new proof system $\langle EF, R\rangle$ is as a propositional translation of the famous and central *witnessing theorems* in bounded arithmetic[2]. Informally, a witnessing theorem shows that for a bounded arithmetic theory T, a T-proof of a tautology of the form $\forall x \exists y \leq t : \phi(x, y)$ (these are called $\forall\Sigma^b_1$-statements) implies the existence of an algorithm from the corresponding TFNP class which on input x, outputs a y such that $\phi(x, y)$ holds. In this paper, we exploit a witnessing theorem due to Buss and Krajíček [BK94], who showed that the $\forall\Sigma^b_1$ statements provable in the bounded arithmetic theory T^1_2 are witnessed by reductions to the TFNP class PLS. A follow-up work of Beckmann and Buss [BB09] generalized this theorem to higher T^i_2 classes, and showed that these theorems can themselves be formalized in S^1_2. The first step above (see cf. Theorem 4.3) follows by known theorems, and is essentially obtained by combining a propositional translation of the provable witnessing due to [BB09] of T^1_2 by PLS algorithms (and hence by reductions to ITER), and combining it with the known relationships between G_1 proofs and tautologies in T^1_2.

The second step (cf. Theorem 4.2) we view as one of our main contributions, and requires new ideas. To prove this, we take direct inspiration from Theorem 1.2 characterizing low-width Resolution by black-box reductions to ITER. For the reader familiar with bounded arithmetic, one intuitive way think of Theorem 4.2 is by formalizing the proof of Theorem 1.2 in the *relativized* theory $S^1_2(\alpha)$, where the oracle α will code a Resolution proof or decision tree reduction, depending on the direction of the equivalence. We can then substitute α for a circuit everywhere in the proof and take a propositional translation, using the fact that S^1_2 proofs propositionally translate to Extended Frege proofs [Bus86].

While the proof can be formalized in this way, we opt to work directly instead, as it is more illuminating. In one direction, from an implicit Resolution refutation (C, Π) of a formula F, we will directly create a mapping reduction from Search_F to ITER by mimicking the standard *Prover-Delayer game* for Resolution [Pud00]. For each line in the implicit proof encoded by C, we create a node in our ITER instance, and given an assignment x which falsifies the line, we define the successor to be the clause used to derive that line which is also falsified. In this way, solutions of the ITER instance will correspond directly to false clauses of F, and we can use the Extended Frege proof Π of the correctness of the encoding of C to argue that this is a provably correct reduction from Search_F to ITER.

The converse direction is proved similarly by following the converse direction of the proof of Theorem 1.2. Now, from a provable reduction from Search_F to ITER we must construct an implicit Resolution proof (C, Π). From the reduction we explicitly construct

[2]In the introduction we discuss the usual, single-sorted theories in bounded arithmetic. However, we warn the reader that, later in the paper, we have found it more convenient to do our formalizations in the two-sorted setting à la Cook-Nguyen [CN10].

this proof, but we note that a remarkable feature is that the proof we construct is *highly* redundant: the Resolution proof we build will have size $2^{\mathsf{poly}(n)} \gg 2^n$, and much of it looks like re-wiring various redundant clauses to one-another. We refer the reader to Section 4 for more details.

Characterizing G_i, for $i > 1$. Using the same proof skeleton as Theorem 4.3 (using an S^1_2-provable witnessing theorem for T^i_2 [BB09]), we are able to obtain similar characterizations of implicit depth-d Frege systems. We use the *game induction principles* GI_d of Skelley and Thapen [ST07], which are the black-box TFNP problems characterizing depth-d Frege.

Theorem 1.4. *For all $d > 0$, $\langle EF, \mathsf{GI}_d \rangle$ is polynomially equivalent to G_d.*

We note that this does not directly imply that $G_d = [EF, \mathsf{GI}_d]$, without a generalization of Theorem 4.2.

Problem 1.5. *Let Frege_d denote the fragment of Frege proofs in which each line has alternation depth d. Is $\langle EF, \mathsf{GI}_d \rangle \equiv_p [EF, \mathrm{Frege}_d]$?*

We leave this problem to follow-up work, due to its considerable technicality in comparison to Theorem 4.2, as well as its independent interest.

These theorems can be viewed as supporting evidence for a main message of this work: *well-studied strong proof systems for propositional reasoning are compressible instances of a corresponding natural weak proof system.*

Generalizing Theorem 4.2. Our second main result (cf. Section 5) shows that the connection proved in Theorem 4.2 holds generally. Our work here is inspired by recent work of Buss, Fleming, and Impagliazzo [BFI23], who obtained a general characterization of black-box TFNP problems and propositional proofs. They show that for every (sufficiently uniform) black-box problem $R \in \mathsf{TFNP}^{dt}$, there is a corresponding propositional proof system P such that a TFNP^{dt} problem Q is decision-tree reducible to R if and only if P can prove that Q is total. In Section 5 we prove a similar relationship between strong proof systems P (those that can simulate EF) and provable mapping reductions. In particular, we view the next theorem as the correct "white-box" analogue of the known connections between proof systems and black-box reductions [BFI23].

Theorem 1.6 (Informal, cf. Theorem 5.2)**.** *Let $P \geq_p EF$ be any proof system which has polynomial-size proofs of its own reflection principle. Then P is polynomially-equivalent to $\langle EF, \mathrm{WrongProof}_P \rangle$.*

In the above theorem, $\mathrm{WrongProof}_P$ is the following total search problem: given a CNF formula F, an assignment x to the variables of F, and a (proposed) P-refutation Π of F, either output a false clause of F under x or find an error in the proof Π. By applying the above theorem when $P = [EF, \mathrm{Resolution}]$, we therefore obtain

$$G_1 \equiv_p \langle EF, \textsc{Iter} \rangle \equiv_p [EF, \mathrm{Resolution}] \equiv_p \langle EF, \mathrm{WrongProof}_{[EF, \mathrm{Resolution}]} \rangle.$$

We believe the characterization $\langle EF, \text{ITER}\rangle \equiv_p \langle EF, \text{WrongProof}_{[EF,\text{Resolution}]}\rangle$ is quite surprising in its own right. The ITER problem is complete for PLS, which is widely believed to be different from FP. However, for *any* proof system P, $\text{WrongProof}_P \in \mathsf{FP}$ (!), since given F, x and Π, we can easily verify in polynomial time if x falsifies a clause of F or if Π contains an incorrect proof step. Hence, even though ITER and $\text{WrongProof}_{[EF,\text{Resolution}]}$ likely have very different complexity as *search problems* (with respect to unrestricted polynomial-time reductions), if we restrict ourselves to EF*-provable* reductions they capture the same set of problems.

Provably Correct Algorithms. As far as we are aware, the only known example in the literature of a polynomial time algorithm whose "proof of correctness" is not in S^1_2 is the AKS primality testing algorithm [AKS04]. This follows by witnessing argument that if S^1_2 proves "AKS(p) = 1 if and only if p is prime", then factoring is in FP [CN10]. The best upper bound for formalizing the correctness of AKS is the theory VTC^0_2, which corresponds to quasipolynomial size TC^0 reasoning, and is incomparable with the full Buss Hierarchy $S_2 = \bigcup_{i=1}^{\infty} S^i_2$ [JJ26].

Taken from the lens of bounded arithmetic, Theorem 5.2 gives a suite of polynomial time algorithms whose *correctness* is arbitrarily hard to prove. Consider the equivalence $G_1 \equiv_p \langle EF, \text{WrongProof}_{G_1}\rangle$, with $\text{WrongProof}_{G_1} \in \mathsf{FP}$. Notice that any polynomial time algorithm f solving WrongProof_{G_1} cannot *provably* do so in S^1_2, unless T^1_2 has the same Π^b_1-consequences as S^1_2. This holds not only for G_1, but for any proof system $P \geq_p EF$ that proves it own reflection principle and corresponds to a bounded arithmetic theory $\mathcal{T}Pe$. If $\mathcal{T}_P \supseteq \mathsf{S}^1_2$, then WrongProof_P is provably in FP if and only if $\mathcal{T}_P$ is Π^b_1-conservative over S^1_2.

1.2 Related Work

In recent independent work of Pudlák and Thapen [PT26], they also prove that Implicit Resolution is equivalent to G_1. Their proof is related to our own, using the known connections between the theory T^1_2, G_1, and PLS. They additionally show, using different techniques based on cut elimination, a p-equivalence between G_i and *narrow* implicit G_{i-1}, denoted as $[EF, G_{i-1}]^m$ by Krajíček [Kra01a], which indicates that the lines of the exponentially large G_{i-1} proof are all polynomial size.

Where this work differs to [PT26] is in our heavy reliance on TFNP and our new definition of the $\langle EF, R\rangle$ proof systems. We believe that our techniques paint a simple and clear picture of the implicit proof landscape, and explain how the implicit resolution characterization of ours and Pudlák-Thapen can be generalized to many other settings. Additionally, our white-box analogue of Buss, Fleming, and Impagliazzo [BFI23] is unique to this work.

1.3 Open Problems

This work raises several questions. First, can we give implicit characterizations of algebraic and semi-algebraic proof systems, such as Nullstellensatz (NS), Cutting Planes, and SOS?

In particular is there a natural system characterizing $[EF, NS]$? Similarly, the characterization of implicit Resolution by G_1 gives rise to natural candidate hard problems for EF. Are there similar natural (e.g., combinatorial or algebraic) candidate hard tautologies for other well-studied propositional proof systems?

Secondly, it is important to point out that our results connecting implicit black-box TFNP to provable white-box TFNP classes are sensitive to the particular complete problem for the class. In particular, the equivalence holds with respect to complete problems where the input is a succinct circuit representation of an exponentially large object. Notably, for complete problems (such as Nash) that are not defined as succinct black-box problems, our characterization does not apply and more generally the connection between white and black box problems in this context is far from clear.

Lastly we note the importance of metamathematical upper bounds in our work. Our characterization of implicit propositional proof systems hinge on the fact that witnessing theorems can be formalized in S^1_2, together with known connections between bounded arithmetic proofs (of Σ^b_1 statements) and their propositional translations. Similarly, Theorem 4.3 hinges on formalizing the Buss–Fleming–Impagliazzo characterization of *every* black-box TFNP problem in S^1_2. This adds to a growing body of work on the metamathematics of complexity theory that are based on the metamathematics of bounded arithmetic itself (e.g. [Jer04, Jer07, BKKK20, Raz93]). Here we mention a few other formalizability problems that are likely to have significant implications: (1) characterizing the minimal theory that can prove the PCP theorem; (2) prove (or disprove) the existence of short EF proofs of other higher order witnessing theorems; (3) determine the minimal theory necessary to prove correctness of nonconstructive or nonuniform polynomial-time algorithms.

Organization

Section 2 contains preliminaries including definitions of standard proof systems, and implicit proofs. In Section 3 we define our new TFNP-based implicit proof systems, $\langle EF, R\rangle$, where $R \in \mathsf{TFNP}$. Section 4 proves Theorem 4.1, and in Section 5 we prove the generalization for arbitrary TFNP problems (Theorem 5.2). Finally, formal definitions of the proof systems that we work with, bounded arithmetic theories, as well as witnessing theorems are supplied in Appendix A.

2 Preliminaries

2.1 Proof Complexity

In this section we outline the necessary background on relevant topics in proof complexity. We refer the reader to Krajíček's monograph for further details [Kra19].

Definition 2.1. A *propositional proof system* P is given by a polynomial-time algorithm $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$, known as its verifier, such that for every CNF formula F,

$$\exists\, \Pi\colon\ V(F, \Pi) = 1 \iff F \text{ is unsatisfiable.}$$

Given two propositional proof systems P, Q, with verifiers V_P and V_Q, we say that P *polynomially-simulates* Q, written $P \geq_p Q$, if there is a polynomial-time computable function f such that $V_Q(F, \Pi) \implies V_P(F, f(F, \Pi))$ for all F and Π.

A standard example is the *Resolution* system.

Definition 2.2. Let $C_1, \ldots C_m, D$ be a collection of clauses. A *Resolution* proof of D from $C_1, \ldots, C_m$ is a sequence of clauses

$$\Pi = (D_1, D_2, \ldots, D_s)$$

where $D_s = D$ and, for each $i \leq s$, either $D_i = C_j$ for some j, or D_i is derived from earlier clauses in the sequence by one of the following two rules:

- *Resolution* $C \vee x, D \vee \neg x \vdash C \vee D$
- *Weakening* $C \vdash C \vee D$

The proof is a *refutation* if $D = \bot$ (the empty clause).

Frege and Extended Frege. Frege systems are a broad class of propositional proofs systems which are polynomially equivalent to Gentzen's propositional calculus, PK, which we define formally in Definition A.1 in the appendix . Briefly, a Frege system consists of a finite set of *inference schemas* $A_1, \ldots, A_k \to B$ which is both sound and implicationally complete. Proofs in a Frege system are a sequence of formulas $\Pi = L_1, \ldots, L_t$, where each successive formula is either an axiom, or is derived from previous formulas by application of (substitution into) one of the rule schemas.

Extended Frege strengthens Frege by allowing for the introduction of *extension variables*.

Definition 2.3. Let $\Pi = L_1, \ldots, L_t$ be a Frege-derivation using variables $x_1, \ldots, x_n$. The *extension rule* allows one to add the line $L_{t+1} := e \leftrightarrow A$, where A is any formula over the variables $x_1, \ldots, x_n$, and the *extension variable* e is not among these variables.

Extended Frege (EF) is the propositional proof system of PK, with the addition of the extension rule.

By allowing the lines in our Frege proof to include existential and universal quantifiers, we move from EF to G, the quantified propositional calculus. The fragment of G we will deal with the most is G_1, which restricts lines to contain only universal or existential quantifiers. More generally, the lines in the G_d proof system may have d alternations of quantifiers. These are defined formally in Definition A.3 in the appendix.

2.2 Reflection Principles

For a propositional proof system P with verifier V, let $\{V_{n,m,s}\}$ be a family of polynomial-size circuits computing V on each input length. Here, n is the number of variables of the

CNF formula F being proven, m is the number of clauses of F, and s is the size of the proof Π being verified. A *reflection principle* is a propositional formula which captures the soundness of a proof system, and will be central to our later characterizations.

Definition 2.4. For a proof system P with verifier V, its *reflection principle* is the family of tautologies

$$\mathrm{Refl}_{P,n,m,s}(F, \Pi, x) := \neg \mathrm{Proof}_{n,m,s}(F, \Pi) \vee \mathrm{SAT}_{n,m}(F, x)$$

where $\mathrm{Proof}_{n,m,s}$ is a formula (using the definition of $V_{n,m,s}$) asserting that Π is a P-proof that F is a tautology, and $\mathrm{SAT}_{n,m}$ is the CNF formula claiming that x is a satisfying assignment to F. This captures the soundness of the proof system P: it asserts that if Π is a P-proof of F then every assignment satisfies F. If the parameters n, m, s are clear from context, then we will suppress the subscripts and call the formula Refl_P.

A Σ_1^q-formula is a quantified boolean formula of the form $\exists \vec{y} G(x, y)$, where G is unquantified. The reflection principle for Σ_1^q-formulas is defined similarly by changing $\mathrm{SAT}(F, x)$ to $\mathrm{SAT}_{\Sigma_1^q}(F, x)$, asserting that F, a Σ_1^q-formula, is satisfied by assignment x. The formula $\mathrm{SAT}_{\Sigma_1^q}(F, x)$ may itself be written as a Σ_1^q-formula in the natural way. We will denote this version of reflection as Σ_1^q-Refl_P.

We note that reflection principles are more easily described in the first order setting of bounded arithmetic—we do so in Appendix A.2.

2.3 Implicit Proof Systems

As briefly mentioned in the Introduction, [Kra01a,Kra04] introduced implicit proof systems as a way to create stronger proof systems from weaker ones by compression. We now briefly review the definitions of these systems.

We start with some complexity theoretic motivation behind Krajicek's definition, by viewing implicit proofs as succinct versions of existing proof systems, similar to the development of succinct versions of complexity classes. Recall that for a string $x \in \{0,1\}^N$ (viewed as a Boolean function on $\log N$ input variables), the natural succinct encoding of x is a boolean circuit S such that $tt(S) = x$, where $tt(\cdot)$ outputs the truth table of the given circuit. For a language $L \subseteq \{0,1\}^*$, the succinct version of L consists of all polynomial-size circuits S such that $tt(S) \in L$. This "succinct operator" compresses a language, and it is well-known that if L is NP-complete, then the succinct version is NEXP-complete.

The same idea can be applied to proofs, giving rise to the implicit operator i. For a proof system P and a P-proof $\pi \in \{0,1\}^*$, we can consider a succinct encoding of it by circuit S such that $tt(S) = \pi$. Of course, such a compressed proof may not be polynomial-time verifiable since we need to decompress the circuit first. To circumvent this, an $[EF, P]$ proof consists of both the circuit S, along with an Extended Frege proof that $tt(S)$ is a P-proof. We describe this concretely in Section 4.1.1.

We now give a more detailed definition of implicit proof systems, including a rough overview of the encodings of formulas. (As Krajíček notes [Kra04], the actual encoding is not important as long as it satisfies some basic properties.)

We start by fixing a description of the implicit Resolution system $[EF, \text{Resolution}]$. If C is a clause over variables $x_1, \ldots, x_n$, we encode C by two strings $(P^{(C)}, N^{(C)}) \in \{0,1\}^n \times \{0,1\}^n$, where $P_i^{(C)} = 1$ iff x_i occurs in the clause and $N_i^{(C)} = 1$ iff $\neg x_i$ occurs in the clause. In this way, all strings $(P, N) \in \{0,1\}^{2n}$ encode a clause.

Next, suppose we have a Resolution refutation of a CNF formula $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$, defined by a sequence of clauses $D_1, D_2, \ldots, D_s$ where $D_s = \bot$. We can encode this refutation as a string as follows. Each clause D_i in the proof is encoded by a string of the form $P^{(D_i)} \circ N^{(D_i)} \circ t^{(i)} \circ L^{(i)} \circ R^{(i)}$, where $\circ$ denotes string concatenation. The components of the string are:

- $P^{(D_i)}, N^{(D_i)} \in \{0,1\}^n$ together encode the literals in the clause D_i, as described above. If $i \leq m$ note that $D_i = C_i$.
- $t^{(i)} \in \{0,1\}^2$ is a short string encoding how D_i was derived in the proof. The tag only has meaning if $i > m$, and depending on its value, $L^{(i)}$ and $R^{(i)}$ will take on different meanings. Namely:
  - If $t^{(i)} = 00$ then D_i is derived by weakening, and the string $L^{(i)}$ will index into the clause D_j with $j < i$ such that D_i is obtained by weakening j.
  - If $t^{(i)} = 01$ then D_i is derived from earlier clauses by Resolution, and the string $L^{(i)}, R^{(i)}$ are the indices of the two clauses used to derive D_i.
  - If $t^{(i)} = 10$, then D_i is a weakening of a clause from F.
  - If $t^{(i)} = 11$ then this line is disabled, and is not used by the proof.

The encoding of the proof is obtained by concatenating the encoding of each line of the proof, as described above.

Now that we have described how to encode a Resolution proof by a string, we can talk about how to implicitly represent a proof. A circuit

$$C : \{0,1\}^s \to \{0,1\}^2 \times \{0,1\}^{2n} \times \{0,1\}^{2s}$$

is said to *implicitly encode a Resolution refutation* of F if the truth-table of C

$$tt(C) := C(0^s) \circ C(0^{s-1}1) \circ \cdots \circ C(1^s)$$

implicitly encodes a Resolution refutation of F. Note that the length of the refutation may be as long as 2^s.

It is possible to formalize propositionally the notion of a correct implicit resolution proof in a CNF we denote $\text{ImpProof}_{\text{Res}}(n, m, s, C, F)$. We carefully do this in Section 4.1.1 for the first-order setting—informally, $\text{ImpProof}_{\text{Res}}$ encodes that a circuit C is a valid circuit, and that its truth table encodes the lines of a resolution proof, with each line being either an axiom of F, deduced by weakening, or deduced by a resolution rule.

Definition 2.5. The proof system $[EF, \text{Resolution}]$ is defined as follows. If $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ is an unsatisfiable CNF formula, then a refutation in $[EF, \text{Resolution}]$ of F is given by a pair (C, Π), where C is an implicit encoding of a Resolution refutation of F, and Π is an Extended Frege proof that C is correctly encoded.

This definition may be generalized to $[EF, P]$, for any propositional proof system P. Now, C encodes a succinct P-refutation of F, and Π similarly is an EF proof of correctness of C. We remark that for static proof systems like Nullstellensatz or Sum-of-Squares, it is not immediately clear how to similarly encode such proofs. However, this can be done through ad-hoc modifications of proof encodings, or more generally through succinctly encoding the computation history of the *verifier* V_P.

2.4 TFNP Preliminaries

In this section, we introduce some of the background on total NP search problems that we regularly use throughout the paper. A recurring theme will be systematic connections between the classical *white-box* theory of TFNP, and the more recently studied *black-box* theory of TFNP (e.g., [MP91, Pap94, GKRS18, BCE+98, GHJ+22a, BFI23, GHJ+22b, KNY19]). We start by recalling the definition of an NP search problem.

Definition 2.6. A relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is

- *Total*, if for all $x \in \{0,1\}^*$ there is a $y \in \{0,1\}^*$ such that $(x,y) \in R$.
- *Polynomially bounded*, if there is a polynomial $\ell(n)$ such that for all $(x,y) \in R$, $|y| \leq \ell(n)$.

An NP *search problem* is a polynomial-time computable, polynomially bounded relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$. The class of all total NP search problems is denoted TFNP.

We think of $R \subseteq \{0,1\}^* \times \{0,1\}^*$ as a search problem in the usual sense: on receiving an input $x \in \{0,1\}^*$, the goal is to output a $y \in \{0,1\}^*$ such that $(x,y) \in R$. We also often use the predicative notation $R(x,y)$ to denote the relation "$(x,y) \in R$". For a TFNP relation R, we will reserve ℓ to denote an upper bound on the length of the certificate of R as a function of n.

Since R is a polynomial-time computable predicate, we can encode the execution of R on an input by a polynomial-size circuit family V_R. We record this next, as it will be useful later.

Definition 2.7. Let $R \in$ TFNP and let n, ℓ be positive integers. Define $V_{R,n,\ell}$ to be a polynomial-size circuit such that $V_{R,n,\ell}(x,y) = R(x,y)$. If the parameters n and ℓ are clear from context, then we will omit them and write $V_R(x,y)$.

As usual, there are notions of reductions between TFNP problems. We record the standard notion of reductions for now.

Definition 2.8. Let $R, S \in$ TFNP. A *mapping reduction*[3] from R to S is given by two polynomial-time computable functions f and g such that for all $x, y \in \{0,1\}^*$,

$$S(f(x), y) \implies R(x, g(y)).$$

If R is mapping reducible to S we write $R \leq_m S$.

[3] In this paper, all mapping reductions will be polynomial-time computable.

The above notion is natural as if $S \in \mathsf{FP}$ and $R \leq_m S$ then $R \in \mathsf{FP}$: given x, we run the polynomial-time algorithm A solving S on $f(x)$, receive an output y, and then output $g(y)$. A canonical example of an interesting TFNP problem is the ITERATION problem, which we define next.

Definition 2.9. The ITERATION problem, also denoted ITER, is defined as follows. For any positive integer n, let $\ell(n) = n^{O(1)}$ be a polynomially-bounded input parameter. The input to the problem is a size parameter 1^n and a circuit $S : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$ of size polynomial in n. Given n and S, a string $y \in \{0,1\}^\ell$ is a valid solution if either

- $y = 0^\ell$, if $S(x, 0^\ell) = 0^\ell$, or
- $y \neq 0^\ell$, if $S(x, y) <_{lex} y$, or
- $y \neq 0^\ell$, if $S(x, y) >_{lex} y$ and $S(x, S(x, y)) = S(x, y)$.

Above $<_{lex}$ denotes the usual lexicographic ordering on strings. The class of all search problems mapping reducible to ITERATION is denoted by PLS.

Black-Box TFNP. In black-box TFNP we are provided the input by query access, as it is now thought of as an exponentially-long string, and we are interested in performing *reductions* via black-box algorithms, where the primary complexity measure is the number of queries made by the reductions. One of the main themes of this paper is the link between white-box and black-box TFNP, and the central example of a total search problem in black-box TFNP is the *false-clause search problem* associated with an unsatisfiable CNF formula $F(x_1, \ldots, x_n) := C_1 \wedge \cdots \wedge C_m$. For this reason, we will briefly review these connections, although we will not need the tools from this section in this paper beyond the definition of the false-clause search problem below.

The false-clause search problem is the following: given an assignment $x \in \{0,1\}^n$ to the input variables $x_1, \ldots, x_n$, output the index i of any falsified clause $C_i(x) = 0$.

Definition 2.10. If $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ is an unsatisfiable CNF formula, the *false-clause search problem* Search_F associated with F is the total relation

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

More generally, in the theory of black-box TFNP we are interested in *black-box total search problems*.

Definition 2.11. Let n be a positive integer. A *black-box search problem* is any total relation

$$R_n^{dt} \subseteq \{0,1\}^n \times O_n,$$

where O_n is a polynomial-sized set of feasible solutions. The class TFNP^{dt} contains all sequences $R^{dt} = \{R_n^{dt}\}_n$ of black-box search problems such that the following holds: there is a universal constant c such that for each input length n, and each $o \in O_n$, there is a $O((\log n)^c)$-depth decision tree T_o satisfying $T_o(x) = R_n^{dt}(x, o)$.

Following standard convention, we will usually think of n as a parameter and consider problems on $\mathsf{poly}(n)$ input bits. If we have a sequence of unsatisfiable CNF formulas $F = \{F_n\}$, each with $\mathsf{poly}(n)$ variables and $\mathsf{poly}(\log n)$ width, then we abuse notation and write $\mathrm{Search}_F := \{\mathrm{Search}_{F_n}\}_n$ to mean the associated sequence of search problems. We note that any such sequence lies in TFNP^{dt}, since each feasible solution of F_n is a $\mathsf{poly}(\log n)$-width clause which can be easily verified by a $\mathsf{poly}(\log n)$-depth decision tree.

Next, we introduce a notion of reduction between black-box search problems, encoded by polynomial-size decision trees. If T is a decision tree then let $|T|$ be the number of leaves of T.

Definition 2.12. Let $S^{dt}_m \subseteq \{0,1\}^m \times O_S, R^{dt}_n \subseteq \{0,1\}^n \times O_S$ be black-box search problems. A *decision-tree reduction* from S^{dt}_m to R^{dt}_n is given by functions $f = \{f_i : \{0,1\}^m \to \{0,1\}\}_{i\in[n]}$ and $g = \{g_o : \{0,1\}^n \to O_S\}_{o\in O_R}$ each computable by decision trees T_{f_i}, T_{g_o}, such that for every $x \in \{0,1\}^m$ and $o \in O_R$,

$$(T_f(x), o) \in R^{dt} \implies (x, T_{g_o}(x)) \in S^{dt}.$$

where $T_f(x)$ denotes the n-bit string obtained by evaluating $T_{f_1}(x), \ldots, T_{f_n}(x)$. The *depth* of the reduction is the maximum depth of any of the decision trees $\{T_{f_i}\}_i \cup \{T_{g_o}\}_o$. The *complexity* of the reduction to be:

$$\sum_{i=1}^{n} |T_{f_i}| + \sum_{o\in O_S} |T_{g_o}|.$$

We extend this definition to sequences as follows. If $\mathsf{R}^{dt} = \{R^{dt}_n\}$ is a sequence of search problems, then $R^{dt}(S^{dt}_m)$ is the minimum complexity of a reduction from S^{dt}_m to any problem in R^{dt}. If $S^{dt} = \{S^{dt}_m\}$ is also a sequence of search problems, then we write $S^{dt} \leq^{dt}_m \mathsf{R}^{dt}$ if $\mathsf{R}^{dt}(S^{dt}_m)$ is polynomially-bounded in m.

A few remarks on this definition are needed. First, the definition above slightly differs from the usual definition of a decision-tree reduction, as seen in e.g. [GHJ+22b]. The usual notion defines the complexity of a reduction to be the maximum depth of any decision tree, plus the logarithm of the size of all decision trees. This notion is useful as it captures the query complexity of reductions in TFNP. For us, the above notion, capturing *just* size, will be more useful as it captures the lengths of proofs in a natural way.

Second, when actually defining reductions it will be more convenient for us to think of the decision trees f as outputting more general objects, like elements in some range $[n]$, rather than individual bits or restrictions. We will abuse notation and do this liberally, with the understood convention that whenever we output or query an element $r \in [n]$, this would be formalized by replacing r with a $\log n$-length bit string encoding r.

To continue our motivating example, just as in the classical setting, the ITERATION search problem has a natural black-box counterpart denoted $\textsc{Iteration}^{dt}$.

Definition 2.13. Let $N = 2^k$ be a positive integer. The problem $\textsc{Iter}^{dt}_N$ is defined as follows. As input, for each element $u \in [N]$, we are given a *successor* $s_u \in [N]$, interpreted

as naming the successor node of u, and which we can think of as being encoded by a string of k bits. The goal of the search problem is to output any of the following

1. 1 if $s_1 = 1$ *(inactive distinguished source)*
2. $u \neq 1$, if $s_u < u$, *(decreasing successor)*
3. u, if u is active and s_u is a proper sink. *(proper sink)*

For ITER^{dt}, it is helpful to think of the successors s_u as describing a fan-out 1 dag on a line of N nodes. Active nodes are those which have some edge to a node Then, if we require that 1 is active, then there must be an active node which points backwards or an active node which points at an inactive node. This can be viewed as a black-box version of the ITERATION problem described earlier, as in the ITERATION problem the successor values S_u were described by a circuit, as opposed to being described directly by query access.

3 Extended-Frege-Provable TFNP reductions.

As we have seen above, if $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ is an unsatisfiable CNF formula, then Search_F is a total search problem, because every input assignment $x \in \{0,1\}^n$ will falsify some clause. This means that if we have a decision-tree reduction from Search_F to some search problem R^{dt} which is *a priori* known to be total, then the decision-tree reduction is itself a proof that F is unsatisfiable. In fact, it has been shown that highly efficient decision tree reductions from Search_F to certain fixed total search problems actually *completely characterize* efficient proofs in certain proof systems [GKRS19,Kam19,GHJ+22b,DR23], and moreover, any proof system satisfying a few mild conditions can be characterized in this way [BFI23].

In this section, we consider reductions from Search_F directly to classical TFNP problems, such as ITER (instead of ITER^{dt}). Of course, now it no longer makes sense to use black-box reductions, so instead we will follow the approach of classical mapping reductions and use boolean circuits instead. First, we introduce an auxiliary formula which will help us define our reductions.

Definition 3.1. Let $F(x_1, \ldots, x_n) = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be a CNF formula. The formula $\text{UNSAT}_F(x_1, \ldots, x_n, z_1, \ldots, z_m)$ is defined to be

$$\text{UNSAT}_F(x, z) := \left(\bigvee_{i=1}^{m} z_i \right) \wedge \bigwedge_{i=1}^{m} (\neg z_i \vee \neg C_i).$$

If $\mathcal{F} = \{F_i\}_i$ is a sequence of unsatisfiable CNF formulas, then let $\text{UNSAT}_{\mathcal{F}} = \{\text{UNSAT}_{F_i}\}_i$ be the corresponding sequence of formulas.

When F is unsatisfiable, the UNSAT_F formula can be viewed as a formalization of the search problem Search_F: given an assignment $x \in \{0,1\}^n$, output any z such that

$\mathrm{UNSAT}_F(x, z)$ is satisfied. Since F is unsatisfiable, such a z will always exist, and hence $\forall x \exists z \mathrm{UNSAT}_F(x, z)$ is itself a tautology. First we observe that these are easily computable as search problems.

Proposition 3.2. *If $\mathcal{F}$ is a polynomial-time uniform sequence of unsatisfiable CNF formulas, then $\mathrm{UNSAT}_{\mathcal{F}}$, interpreted as a total search problem, is in* FP.

Proof. Write $\mathcal{F} = \{F_i\}_i$, and let A be the polynomial-time algorithm such that $A(1^i)$ outputs the description of F_i. The polynomial-time algorithm for $\mathrm{UNSAT}_{\mathcal{F}}$ is simple: given x as input, run $A(1^{|x|})$ to recover $F_{|x|}$. Evaluate $F_{|x|}(x)$ and, for each clause C_i of $F_{|x|}$, let $z_i = C_i(x)$, and output the string z. □

A crucial distinction must be made here. The family $\mathcal{F}$ is polynomial-time uniform, and $\mathrm{UNSAT}_{\mathcal{F}}$ is computable in polynomial time (when considered as a search problem), but both of these facts should not be interpreted as saying there are polynomial-time verifiable proofs that the families are unsatisfiable.

Next, we define a notion of non-uniform mapping reduction from Search_F to an arbitrary TFNP relation R using the UNSAT_F formulas.

Definition 3.3. Let $R \in$ TFNP and let $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ be an unsatisfiable CNF formula. Let $\ell(n)$ denote the polynomial upper bound on the length of any solution to R on inputs of length n. A *non-uniform mapping reduction* from Search_F to R is given by two circuits $C : \{0,1\}^n \to \{0,1\}^s, D : \{0,1\}^n \times \{0,1\}^{\ell(s)} \to \{0,1\}^m$, such that

$$R(C(x), y) \implies \mathrm{UNSAT}_F(x, D(x, y)). \tag{1}$$

Observe that we have no efficient way of verifying that two circuits (C, D) actually satisfy the previous definition. For example, given Search_F, one could consider the following "trivial" polynomial-time algorithm that solves it: given $x \in \{0,1\}^n$, test the clauses of F one-by-one on x, and output the first one that is false. If no clause is false, then loop forever. If F is unsatisfiable, then we are guaranteed that this algorithm will halt in polynomial time. However, *proving* that this algorithm halts is equivalent to proving that F is unsatisfiable in the first place. To turn this notion into a propositional proof system, we will supply an Extended Frege proof that the mapping reduction is correct. In order to do so, we will need to introduce a propositional formalization of Equation (1).

Recall (Definition 2.7) that if $R \in$ TFNP then $V_{R,n,\ell}$ is a polynomial-size circuit which is equivalent to R on inputs of length n and ℓ. First, we define a formula $\mathrm{Reduction}_R$ which captures the truth of (1).

Definition 3.4. Let n, m be positive integers, let $R \in$ TFNP, and let $F(x_1, \ldots, x_n) = \wedge_{i=1}^m C_i$ be a CNF formula. Let ℓ be a polynomial bound on the certificate length of R. If $R \in$ TFNP and $F(x_1, \ldots, x_n) = \wedge_{i=1}^m C_i$, then $\mathrm{Reduction}_{R,n,m,S,s}(C, D, F)$ is the boolean formula defined as follows. We introduce several helper formulas which can easily be represented in CNF.

– $\mathrm{isCkt}_S(C)$ is satisfied iff the variables in C encode a valid boolean circuit with S gates.
– $\mathrm{Eval}_{n,S,s}(C, x, t)$ is satisfied iff C encodes a boolean circuit with n inputs, S gates, and s outputs, and $t \in \{0,1\}^s$ is the output of C on input $x \in \{0,1\}^n$.

Then $\mathrm{Reduction}_{R,n,m,S,s}(C, D, F) :=$

$$\mathrm{isCkt}(C) \wedge \mathrm{isCkt}(D) \wedge \mathrm{Eval}(C, x, x') \wedge \mathrm{Eval}(D, (x, y), z) \wedge \mathrm{Eval}(V_R, x', y) \rightarrow \mathrm{UNSAT}_F(x, z).$$

For readability, we have omitted parameter subscripts above, and we will continue to do so when those parameters are clear from context. The circuit C has n inputs, S gates, and s outputs, and the circuit D has $n + \ell(s)$ inputs, S gates, and m outputs.

By definition, for any circuits C and D and any CNF formula F, we have that (C, D) form a non-uniform mapping reduction from Search_F to R if and only if $\mathrm{Reduction}_R(C, D, F)$ is a tautology. So, with this definition in hand, we can define a *provable* mapping reduction from Search_F to R. Now, along with the circuits C, D, defining a mapping reduction, we supply an Extended Frege proof Π of $\mathrm{Reduction}_R(C, D, \Pi)$.

Definition 3.5. (Defining $\langle EF, R\rangle$) For any $R \in \mathsf{TFNP}$, the propositional proof system $\langle EF, R\rangle$ is defined as follows. For any unsatisfiable CNF formula $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$, a refutation of F in $\langle EF, R\rangle$ is given by an *EF-provable reduction from* Search_F *to* R. Formally, such a reduction is a tuple (C, D, Π), where (C, D) are circuits computing a non-uniform mapping reduction from Search_F to R, and Π is an Extended Frege proof of a boolean formula $\mathrm{Reduction}_R(C, D, F)$.

Observe that an $\langle EF, R\rangle$ proof can be easily verified in polynomial time, since we just need to verify the Extended Frege proof Π of $\mathrm{Reduction}_R(C, D, F)$. Since R is a total search problem, if (C, D) are a valid mapping reduction then Π proves that Search_F is total, and hence F must be unsatisfiable. (Note that Extended Frege itself may *not* be able to prove that R is total, and indeed later we will be interested in relations R where it seems this is not the case.) To see that $\langle EF, R\rangle$ is a complete proof system we show that it polynomially simulates Extended Frege.

Proposition 3.6. *For any $R \in \mathsf{TFNP}$, $\langle EF, R\rangle \geq_p EF$.*

Proof. We give a sketch of a proof that can be easily formalized in Extended Frege. Let Π be an Extended Frege refutation of $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$. Let C be a circuit which, given x, ignores it and hard-codes an instance of R with a fixed valid solution o. Let D be the circuit which, given (x, o), ignores o, evaluates $F(x)$, and outputs the string $z \in \{0,1\}^m$ defined by $z_i = 1$ iff $C_i(x) = 0$. From the definition of D, it is immediate that $\neg z_i \vee \neg C_i(x)$ holds for all $i \in [m]$, and so we prove that for all x, $D(x, o) \neq 0^m$ by contradiction. If $D(x, o) = 0^m$, then by the definition of D, we would have $C_i(x) = 1$ for all $i \in [m]$, and hence we can prove $D(x, o) = 0^m$ implies F is satisfiable. But Π is an Extended Frege refutation of F, which leads to a final contradiction. □

As a second calibration for its strength, it turns out that a converse holds provided Extended Frege can prove that R is inside of FP.

Definition 3.7. If $R \in \mathsf{TFNP}$, then we say R is *EF-provably inside of* FP if for every input length n, there is a polynomial-size circuit A and a polynomial-size Extended Frege proof of the formula $\mathrm{Eval}(A, x, y) \wedge \mathrm{Eval}(V_R, (x, y), b) \wedge b$, where V_R is understood to encode the polynomial-time verifier for the relation R.

Proposition 3.8. *If $R \in \mathsf{TFNP}$ is EF-provably in FP, then $\langle EF, R\rangle \equiv_p EF$.*

Proof. We again sketch the proof and note it can be easily formalized. From the previous proposition, we have that $\langle EF, R\rangle \leq_p EF$. For the converse, consider an $\langle EF, R\rangle$ proof (C, D, Π) of F. Let A be a polynomial-size circuit solving R such that Ψ is an Extended Frege proof of $\mathrm{Eval}(A, x, y) \wedge \mathrm{Eval}(V_R, (x, y), b) \wedge b$, which is equivalent to $V_R(x, A(x))$. Then Π proves the statement

$$V_R(C(x), y) \to \mathrm{UNSAT}_F(x, D(x, y)),$$

so by substituting C into Ψ, we can prove the statement $V_R(C(x), A(C(x)))$ in Extended Frege, and hence also $\mathrm{UNSAT}_F(x, D(x, A(C(x))))$. From this we can derive $\neg F$ in Extended Frege directly. □

As we have mentioned in the introduction, later we will see examples of total search problems $R \in \mathsf{FP}$ but for which if R was *EF-provably* in FP, then surprising consequences will hold: EF can polynomially simulate G_1 over propositional tautologies.

4 Implicit Resolution and G_1

In this section we will use provable mapping reductions in order to establish the following theorem.

Theorem 4.1. *$[EF, \mathrm{Resolution}]$ and G_1 are polynomially equivalent.*

This theorem will be proven in two parts over the next subsections:

Theorem 4.2. $\langle EF, \textsc{Iter}\rangle \equiv_p [EF, \mathrm{Resolution}]$.

Theorem 4.3. $G_1 \equiv_p \langle EF, \mathsf{Iter}\rangle$.

4.1 $\langle EF, \textsc{Iter}\rangle$ is equivalent to $[EF, \mathrm{Resolution}]$

In this section we prove Theorem 4.2, showing that $\langle EF, \textsc{Iter}\rangle$ is polynomially-equivalent to the implicit Resolution system $[EF, \mathrm{Resolution}]$. The intuition for this proof is inspired by the known connection between Resolution proofs and reductions to $\textsc{Iteration}^{dt}$. This connection, while having roots in results in bounded arithmetic, is perhaps best exemplified by the result of Kamath [Kam19] (cf. Theorem 1.2), who proved that low-depth decision-tree reductions from Search_F to $\textsc{Iter}^{dt}$ correspond directly to low-width Resolution refutations of F. In some sense, our proof of Theorem 4.2 can be viewed as a formalization in bounded arithmetic of Theorem 1.2.

4.1.1 Setup for Proof of Theorem 4.2

In order to prove Theorem 4.2 we need to show how to simulate $\langle EF, \text{ITER}\rangle$ proofs by $[EF, \text{Resolution}]$ proofs and *vice-versa*. This will require formalizing proofs in Extended Frege, which can be quite technical, so instead we opt to formalize these results in the bounded arithmetic theory V^1 (actually, $\mathsf{V}^1(\mathsf{VPV})$), and then use the known propositional translations to convert V^1 proofs into Extended Frege proofs. See Appendix A.2 for more details on the theories V^1 and $\mathsf{V}^1(VPV)$.

First, we introduce some formulas in the language of $V^1(\mathsf{VPV})$ which capture the statements $\text{Reduction}_{\text{ITER}}$ and $\text{ImpProof}_{\text{Res}}$. We recall that $\mathsf{V}^1(\mathsf{VPV})$ has variables of two sorts: *number sort* variables, represented by lower-case letters $x, y, z, \ldots$, and *string sort* variables, represented by uppercase letters $X, Y, Z, \ldots$. String sort variables X can be indexed into by number sort variables i, where $X(i)$ is true if the ith entry of X is 1, and the first entry of the string is 0. We note that the string sort variables represent *arbitrary* finite-length strings, and so $|X|$, which is the length of X, is defined to be $\max\{i : X(i) = 1\}+1$. Hence we can also index into X beyond $|X|$, but the response is always 0. Finally, $\mathsf{V}^1(\mathsf{VPV})$ has a function symbol $A(\vec{x}, \vec{X})$ for every polynomial-time string algorithm A. We refer to Appendix A.2 or the monograph [CN10] for more technical details regarding two-sorted theories of bounded arithmetic.

Circuit Encodings. We encode a circuit C with s gates on n inputs by a pair (n, C), where n is the number of inputs and where the gates are numbered from $0, 1, \ldots, s + n + 1$ To simplify our encoding, we express our circuits over the $\{\wedge, \neg\}$ basis, writing $\vee$ as $\neg \wedge \neg$. Let $\beta(n, m)$ denote the usual pairing function[4] [CN10]. The string C has length $\beta(2(s+n), \beta((s+n), (s+n)))$, where the substring $C^{[0]}$ is of length $2(s+n)$ and encodes the gate sequence of the circuit, and the string $C^{[1]}$ is a two-dimensional string of length $\beta(s + n, s + n)$ and encodes the edge relation of the circuit. The $s + n$ gate labels are encoded by two bits each, where the first n gates encode the input variables, and the next s gates are either $\wedge$ gates, $\vee$ gates, or the constants 0 or 1. More formally:

- For $i = 0, \ldots, n - 1$ we set $C^{[0]}(2i)C^{[0]}(2i + 1) = 00$, representing that this is an input gate.
- For $i = n, \ldots, s + n - 1$, we break into cases:
  - if $C^{[0]}(2i)C^{[0]}(2i + 1) = 00$, then this gate is the constant 0.
  - If $C^{[0]}(2i)C^{[0]}(2i + 1) = 01$, then this gate is the constant 1.
  - If $C^{[0]}(2i)C^{[0]}(2i + 1) = 10$, then this is an $\wedge$ gate.
  - If $C^{[0]}(2i)C^{[0]}(2i + 1) = 11$, then this a $\neg$ gate.

Finally, for each i and j, $C^{[1]}(i, j) = 1$ iff there is a wire connecting the output of gate i into the input of gate j. Of course, the input variables and constants $0, 1$ do not receive any inputs. For the sake of brevity, we will often write C instead of the tuple (n, C). Let

[4] In [CN10] the notation $\langle n, m\rangle$ is used. Since we use $\langle\rangle$ for provable mapping reductions, we opt to use β instead.

$\mathrm{Size}(n, C) = |C^{[0]}|/2$ denote the number of gates in the circuit C, including the number of input variables. Let $\mathrm{isCkt}(n, C)$ denote the relation that is true iff the input properly encodes a circuit with $s = \mathrm{Size}(n, C)$ gates in total and n input gates, and note that this relation is definable in V^0. If C is a circuit of size s encoded as (n, C), then for $o < \mathrm{Size}(C)$ we let $\mathrm{Eval}(n, o, C, X)$ denote the polynomial-time function which evaluates C on the string X and outputs the values of the final o gates as a single string.

Encoding Resolution Proofs. Let $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$ be an unsatisfiable CNF formula, encoded by strings $P, N \in \{0,1\}^{m \times n}$ as described above. The validity of an implicit Resolution refutation of F is encoded by a formula

$$\mathrm{ImpProof}_{\mathrm{Res}}(n, m, s, C, P, N)$$

which we define now. The string C will encode a circuit that implicitly encodes a Resolution refutation of F, following the definition given in Section 2.3. Each line of the proof is described by a string of length $\ell^* := 2n + 2 + 2s$, which together encode a clause, how the line was derived, and pointers to earlier clauses. As a function, $C : \{0,1\}^s \to \{0,1\}^{\ell^*}$, taking a string as input which indexes a line of the proof, and outputs a string encoding the entire line.

We represent $\mathrm{Proof}_{\mathrm{Res}}$ as a conjunction of polynomial-time computable predicates, each of which is a symbol in the language of $\mathsf{V}^1(\mathsf{VPV})$. Below, the formula F is represented by the pair of strings (P, N). We say a line is *active* if the line is not disabled in the proof. Formally, we consider the polynomial-time predicate symbols:

- $\mathrm{Active}(n, s, C, Y)$ is true if the clause at line $C(Y)$ is active.
- $\mathrm{Empty}(n, s, C, Y)$ is true if the clause at line $C(Y)$ is empty.
- $\mathrm{isWeaken}(n, s, C, Y)$ is true if the line $C(Y)$ of the proof is a valid weakening of an earlier, active line.
- $\mathrm{isResolve}(n, s, C, Y)$ is true if the line $C(Y)$ is a valid Resolution of earlier active lines.
- $\mathrm{isAxiomW}(n, m, s, i, P, N, C, Y)$ is true if the clause at line $C(Y)$ is a weakening of the ith clause of F.

If m is a number term then we let $\mathrm{Dyad}(m)$ denote the string encoding m in dyadic notation, and note that $\mathrm{Dyad}(m)$ is a polynomial-time computable function and its basic properties can be formalized in V^1 [CN10].

Definition 4.4. The formula $\mathrm{ImpProof}_{\mathrm{Res}}(n, m, s, C, P, N)$ is the conjunction of the following formulas:

- $\mathrm{isCkt}(n, C)$
- $\mathrm{Active}(n, s, C, 1^s) \wedge \mathrm{Empty}(n, s, C, 1^s)$
- $\forall Y < s : \mathrm{Active}(n, s, C, Y) \to \mathrm{isWeaken}(n, s, C, Y) \vee \mathrm{isResolve}(n, s, C, Y) \vee \exists i < m : \mathrm{isAxiomW}(n, m, s, i, P, N, C, Y)$.

Encoding Mapping Reductions. We now define a formula

$$\mathrm{Reduction}_{\textsc{Iter}}(n, m, s, C, D, P, N)$$

in the language of $\mathsf{V}^1(\mathsf{VPV})$ which verifies if the circuits C and D are a non-uniform mapping reduction from Search_F to ITER. For this, we introduce a new formula

$$\mathrm{UNSAT}(n, m, P, N, X, Z),$$

which can be viewed as the Δ_0^B-formula encoding UNSAT_F from Definition 3.1. This is indeed closely related to the propositional formula $\mathrm{UNSAT}(C, D, F)$ we introduced in Definition 3.1, albeit with a slight change in the encoding for the formula F. In this formula, (P, N) are strings of length $\{0,1\}^{m\times n}$ encoding a CNF formula F with n variables and m clauses, X is a string encoding an input to F, and Z is a string encoding indices to false clauses of F on input X. In particular, applying the propositional translation $||\mathrm{UNSAT}(n, m, P, N, X, Z)||_{n,m}$ will give essentially the same formula as $\mathrm{UNSAT}(C, D, F)$, albeit with the formula F encoded by lists of free variables P and N. Formally,

$$\begin{aligned}\mathrm{UNSAT}(n, m, P, N, X, Z) := (\exists i < m : Z(i)) \wedge \\ \forall i < m : Z(i) \to (\forall j < n : (P^{[i]}(j) \to \neg X(j)) \wedge (N^{[i]}(j) \to X(j)))\end{aligned}$$

Using this formula we can define the reduction formula.

Definition 4.5. Let ITER denote the polynomial-time function symbol encoding the polynomial-time verifier for the ITERATION problem. The formula $\mathrm{Reduction}_{\textsc{Iter}}(n, m, s, C, D, P, N)$ is defined to be the conjunction of the formulas:

- $\mathrm{isCkt}(n, C)$,
- $\mathrm{isCkt}(n + s, D)$,
- and the formula

$$\begin{aligned}\forall X < n, Y < s : \textsc{Iter}(s, \mathrm{Eval}(n, s, C, X), Y) \to \\ \mathrm{UNSAT}(n, m, P, N, X, \mathrm{Eval}(n + s, m, D, X \circ Y))\end{aligned}$$

  where $\circ$ denotes string concatenation.

We note for the reader that C is a circuit which itself *outputs* the description of a circuit, since the input to ITER is itself a circuit.

Statement of Main Lemmas and Proof of Theorem 4.2. Finally, using the notation introduced in the previous sections, we can now state the main lemmas which formalize the transformations between implicit Resolution proofs and non-uniform reductions inside of $\mathsf{V}^1(\mathsf{VPV})$.

Lemma 4.6. *There are polynomial-time algorithms A and B such that* $\mathsf{V}^1(\mathsf{VPV})$ *proves*

$$\mathrm{ImpProof}_{\mathrm{Res}}(n, m, s, C, P, N) \to \mathrm{Reduction}_{\mathrm{ITER}}(n, m, s, A(\vec{x}, \vec{X}), B(\vec{x}, \vec{X}), P, N),$$

where $\vec{x} = (n, m, s)$*, and* $\vec{X} = (C, P, N)$*.*

Lemma 4.7. *There is a polynomial-time algorithm R and a number term t such that* $\mathsf{V}^1(\mathsf{VPV})$ *proves*

$$\mathrm{Reduction}_{\mathrm{ITER}}(n, m, s, C, D, P, N) \to \mathrm{ImpProof}_{\mathrm{Res}}(n, m, t(\vec{y}, \vec{Y}), R(\vec{y}, \vec{Y}), P, N),$$

where $\vec{y} = (n, m, s)$ *and* $\vec{Y} = (C, D, P, N)$*.*

The proofs of these two lemmas will take up the rest of the section. However, assuming Lemma 4.6 and Lemma 4.7, we can now prove Theorem 4.2.

Proof of Theorem 4.2. This proof is an exercise in the use of propositional translations, so we sketch it and leave a fully formal proof to the reader.

First, we show that $[EF, \mathrm{Resolution}]$ p-simulates $\langle EF, \textsc{Iter}\rangle$. Suppose that (C, D, Π) is an $\langle EF, \textsc{Iter}\rangle$ refutation of a CNF formula $F(x_1, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$, and so Π is an Extended Frege proof of the propositional formula $\mathrm{Reduction}_{\mathrm{ITER}}(C, D, F)$. Let $(P^F, N^F) \in \{0,1\}^{m \times n}$ be strings encoding the CNF F. By propositionally translating the statement in Lemma 4.7, we have for any parameters $n', m', s' \in \mathbb{N}$, there are polynomial-size Extended Frege proofs of the implication

$$||\mathrm{Reduction}_{\mathrm{ITER}}(\vec{y}, \vec{Y})||_{n',m',s'} \to ||\mathrm{ImpProof}_{\mathrm{Res}}(n, m, t(\vec{y}, \vec{Y}), R(\vec{y}, \vec{Y}), P, N)||_{n',m',s'},$$

where we have used the notation $\vec{y}$ and $\vec{Y}$ from Lemma 4.7. From $\mathrm{Reduction}_{\mathrm{ITER}}(C, D, F)$, Extended Frege can prove the formula

$$||\mathrm{Reduction}_{\mathrm{ITER}}(n, m, s, C, D, P^F, N^F)||_{n,m,s}$$

obtained by propositionally translating the formula from Definition 4.5. Hence Extended Frege can prove

$$||\mathrm{ImpProof}_{\mathrm{Res}}(n, m, t(\vec{y}, \vec{Y}), R(\vec{y}, \vec{Y}), P^F, N^F)||_{n',m',s'}$$

and so we have proved that the circuit encoded by R encodes an implicit Resolution refutation of F.

The converse direction is similar. Suppose that (Π, C) is an $[EF, \mathrm{Resolution}]$ refutation of F, so Π is an Extended Frege proof of the propositional translation

$$||\mathrm{ImpProof}_{\mathrm{Res}}(n, m, s, C, P^F, N^F)||_{n,m,s}$$

for appropriately chosen $n, m, s \in \mathbb{N}$. By applying a similar argument as before, we propositionally translate Lemma 4.6 to attain an Extended Frege proof of the implication in that lemma, and hence we can derive in Extended Frege the formula

$$||\mathrm{Reduction}_{\mathrm{ITER}}(n, m, s, A(\vec{x}, \vec{y}), B(\vec{x}, \vec{Y}), P^F, N^F)||_{n,m,s}$$

in polynomial size. From this formula, and using the fact that P^F and N^F encode F, we can derive the propositional version of $\mathrm{Reduction}(A, B, F)$ in size polynomial in $|C|$ and $|\Pi|$ which used in the definition of $\langle EF, \textsc{Iter}\rangle$.

□

4.1.2 Proofs of Main Lemmas

In the remainder of this section we prove Lemma 4.6 and Lemma 4.7, and we begin by proving Lemma 4.6. Our proof is inspired by the equivalence between black-box total search problems efficiently reducible to $\textsc{Iter}^{dt}$ and low-width Resolution refutations, as observed by [Kam19], but analogous results were observed in the bounded arithmetic literature, see e.g. [Kra01b]. Indeed, our proof can, in some sense, be viewed as a *formalization* in $\mathsf{V}^1(\mathsf{VPV})$ of this theorem, although in fact moving to the white-box setting will simplify some steps and complicate others.

Proof of Lemma 4.6. We argue in $\mathsf{V}^1(\mathsf{VPV})$. Assume $\mathrm{ImpProof}_{\mathrm{Res}}(n, m, s, C, P, N)$, and let $\vec{x} = (n, m, s)$ and $\vec{X} = (C, P, N)$. To remember the types of the various objects, we recall that (P, N) are strings encoding a CNF formula $F(x_1, x_2, \ldots, x_n) = C_1 \wedge \cdots \wedge C_m$, and C is a circuit implicitly encoding a Resolution refutation of F. As input, C takes a string $Y \in \{0,1\}^s$ and outputs a corresponding line of the proof. The algorithms A and B that we define will *output* circuits S and G encoding an instance of $\textsc{Iter}$, such that any solution in the $\textsc{Iter}$ instance can be used to recover a false clause of F.

We now describe the circuits S and G which will be output by the algorithms A and B, respectively. The circuit S, on input X, Y with $X < n$ and $Y < s$, executes Algorithm 1.

To summarize the algorithm in Algorithm 1, we evaluate the circuit C at the index Y to get a line of the proof $C(Y)$. If $C(Y)$ is satisfied, or disabled, or an axiom of F, then we output Y as the successor. Otherwise, $C(Y)$ must be falsified, active, and therefore derived by either weakening or Resolution. We hence evaluate the lines used to derive C, and let the successor of Y be the falsified line that was used to derive $C(Y)$. This algorithm is computable in polynomial time, and the circuit computing S can be computed from n, s, and C in polynomial time as well. Given $\vec{x}, \vec{X}$, the algorithm A outputs the encoding of the circuit computing the function $X \mapsto S(X, Y)$.

The circuit G is even simpler. Given $X < n, Y < s$, we evaluate S at most twice to see if Y is a solution of the $\textsc{Iter}$ instance encoded by S. If Y is a solution, then by construction, $S(X, Y) = Y'$ is the index of a falsified clause of F on input X, and hence G will output a string $Z < m$ indexing the clause falsified by X on this input. Similarly, the circuit G can be computed from $\vec{x}$ and $\vec{X}$ in polynomial time, and so the algorithm B constructs G and outputs the encoding of the circuit computing the function $(X, Y) \mapsto G(X, Y)$.

Finally, we argue that $\mathsf{V}^1(\mathsf{VPV})$ can prove

$$\mathrm{Reduction}_{\mathsf{Iter}}(n, m, s, A(\vec{x}, \vec{X}), B(\vec{x}, \vec{X}), P, N).$$

By definition, A outputs the code of circuit S and B outputs the code of circuit G, and hence $\mathsf{V}^1(\mathsf{VPV})$ can prove $\mathrm{isCkt}(n, A(\vec{x}, \vec{X}))$ and $\mathrm{isCkt}(n+s, B(\vec{x}, \vec{X}))$. Now, let $X < n$

Algorithm 1: Successor Function $S(n, s, X, Y)$

Input: Strings X, Y with $X < n$ and $Y < s$
Output: A string $Y' < s$

Evaluate $C(Y)$, receiving a line $\mathcal{L}$ of the proof;
Let $\mathcal{L} = (P^{\mathcal{L}}, N^{\mathcal{L}}, T^{\mathcal{L}}, L^{\mathcal{L}}, R^{\mathcal{L}})$ be the components of the line;
if $\text{SATClause}(n, P^{\mathcal{L}}, N^{\mathcal{L}}, X)$ *or* $T^{\mathcal{L}} = 10$ *or* $T^{\mathcal{L}} = 11$ **then**
    return Y
end
if $T^{\mathcal{L}} = 00$ **then**
    /* $C(Y)$ is a weakening, so output the line that was weakened. */
    return $L^{\mathcal{L}}$
else if $T^{\mathcal{L}} = 01$ **then**
    /* $C(Y)$ is a resolution, so find the false clause */
    Evaluate $C(L^Y) = \mathcal{L}'$;
    Let $\mathcal{L}' = (P^{\mathcal{L}'}, N^{\mathcal{L}'}, T^{\mathcal{L}'}, L^{\mathcal{L}'}, R^{\mathcal{L}'})$ be the components of the line;
    if $\neg\text{SATClause}(n, P^{\mathcal{L}'}, N^{\mathcal{L}'})$ **then**
        return $L^{\mathcal{L}}$
    else
        return $R^{\mathcal{L}}$

and $Y < s$, and suppose that $\text{ITER}(s, \text{Eval}(n, s, A(\vec{x}, \vec{X}), Y)$ holds. Examining the code of $A(\vec{x}, \vec{X}) = S$ and $B(\vec{x}, \vec{X}) = G$, we have that G will output a string Z falsifying a clause of F, as can be proved using a case argument on the definition of G and S. Hence $\text{UNSAT}(n, m, P, N, X, \text{Eval}(n+s, m, B(\vec{x}, \vec{X}), X \circ Y))$ will hold, and this completes the proof. □

We now prove Lemma 4.7.

Proof of Lemma 4.7. We again reason in $\mathsf{V}^1(\mathsf{VPV})$. Assume that

$$\text{Reduction}_{\text{ITER}}(n, m, s, C, D, P, N)$$

holds, and so C and D are circuits encoding a reduction from the search problem for the CNF F encoded by (P, N) to ITER. This means given an input assignment $X < n$, $C(X)$ will output the description of a circuit S that is an instance of ITERATION, and given $X < n$, $Y < s$, $D(X, Y)$ will output a solution Z of Search_F. For simplicity, we write $S(X, Y)$ to be the algorithm which first evaluates $C(X)$, obtaining a circuit S, and then evaluates $S(Y)$.

Let $\vec{y} = (n, m, s)$ be the number parameters and $\vec{Y} = (C, D, P, N)$ be the string parameters. We compute an implicit Resolution refutation using the algorithm H, whose code appears in Algorithm 2, although we will give a high-level overview of the structure of the proof now, and a depiction is given in Figure 1.

Each line of the implicit Resolution proof will be indexed by strings X, Y, Y' with $|X| \leq n+1$ and $|Y|, |Y'| < s$. The structure of the refutation will be in two separate blocks.

Algorithm 2: Implicit Refutation $H(X, Y, Y')$

Input: Strings $X \leq n+1, Y < s, Y' < s$.
Output: Strings encoding a line $\mathcal{L} = (P^{\mathcal{L}}, N^{\mathcal{L}}, T^{\mathcal{L}}, L^{\mathcal{L}}, R^{\mathcal{L}})$

if $X = 1^{n+1}, Y = 1^s, Y' = 1^s$ **then**
  // This is a special final "empty" clause to satisfy the definition of $\mathrm{ImpProof}_{\mathrm{Res}}$.
  return $\mathcal{L} = (\varepsilon, \varepsilon, 00, (\varepsilon, \varepsilon, \varepsilon))$
end
if $|X| = n + 1$ **then**
  // We are in the "first block", and so we use the successor function S to route clauses.
  Let $X' = X[0 : n-1]$ be the first n bits of X;
  if $S(X', Y) \neq Y'$ **then**
    // Disable the line.
    return *any line with* $T^{\mathcal{L}} = 11$
  end
  // Now $S(X', Y) = Y'$
  if $|Y'| = |Y| = 0$ *or* $Y' <_{\text{lex}} Y$ *or* $S(X', Y') = Y'$ **then**
    // Y is a solution, so $\mathrm{UNSAT}(n, m, P, N, X, D(X', Y))$ is satisfied and $D(X', Y)$ encodes a clause of F falsified under X'.
    Evaluate $D(X', Y) = Z$;
    Let $i := \min\{j : Z(j) = 1\}$;
    return $\mathcal{L} = (P^{\overline{X'}}, N^{\overline{X'}}, 10, \mathrm{Dyad}(i), 0^s)$
  else if $Y' >_{\text{lex}} Y$ **then**
    // Y is not a solution, so we just make it a weakening.
    Evaluate $S(X', Y') = Y''$;
    return $\mathcal{L} = (P^{\overline{X'}}, N^{\overline{X'}}, 00, (X, Y''), 0^s)$
else
  // $|X| < n + 1$, now we are in the second block, and must embed a complete tree-like Resolution refutation.
  if $|Y| > 0$ *or* $|Y'| > 0$ **then**
    // Disable the line.
    return *any line* $\mathcal{L}$ *with* $T^{\mathcal{L}} = 11$
  end
  if $|X| = n$ **then**
    // In this case we are a weakening of an earlier line
    Evaluate $S(X, Y') = Y''$;
    return $\mathcal{L} = (P^{\overline{X}}, N^{\overline{X}}, 00, (X \circ 1, Y''), 0^s)$
  else
    // Now $|X| < n$, so we are performing the complete tree-like Resolution refutation of size 2^n.
    return $\mathcal{L} = (P^{\overline{X}}, N^{\overline{X}}, 01, (X \circ 0, 0^s, 0^s), (X \circ 1, 0^s, 0^s)))$

In the first block, $|X| = n+1$, and in the second block $|X| \leq n$. When $|X| = n+1$, let X' be the first n bits of X, and note this can be obtained by simply setting the nth bit of X to 0. A line will be active in this case if $S(X', Y) = Y'$ and either $Y' >_{\text{lex}} Y$ or $Y' = 0$. If a line is active, then X' will define the clause encoded at the line by the strings $(P^{\overline{X}'}, N^{\overline{X}'})$, which is the unique clause of width n that is falsified by X'. (We note that for any such X', we can define $P^{\overline{X}'}, N^{\overline{X}'}$ using Σ_0^B-comprehension.) The clauses used to deduce this will be determined entirely by the successor function S from the ITERATION instance. If Y is a solution in S on the input X', meaning either $S(X', Y) = Y'$ and $S(X', Y') = Y'$, then we know that X' falsifies some clause of F, and the particular clause can be recovered using the circuit $D(X', Y)$. In this case, we label the line as a weakening of a clause of F and we are done. On the other hand, if Y is not a solution, then $S(X', Y) = Y'$ and $S(X', Y') \neq Y'$. Now the line indexed by $(X', Y', S(X', Y'))$ must also be active and labelled by the clause $(P^{\overline{X}'}, N^{\overline{X}'})$, and so we declare that the line (X', Y, Y') to be a (trivial) weakening of $(X', Y', S(X', Y'))$.

In the second block, $|X| \leq n$. Now a line is active only if $|Y| = |Y'| = 0$. Here we simply plant the complete tree-like Resolution refutation of size 2^n. The leaves of the refutation will be labelled by strings $|X| = n$, and we similarly declare these to be weakenings of corresponding clauses in the first block, and pointers to the internal clauses can be calculated quite easily.

Now, we let $R(\vec{y}, \vec{Y})$ denote the algorithm which outputs the circuit encoding H on these input lengths, and we let $t(\vec{y}, \vec{Y}) = n + 1 + 2s$. The algorithm H is polynomial-time computable and so R is a polynomial-time computable algorithm. Let us now argue that we can prove $\text{ImpProof}_{\text{Res}}(n, m, t(\vec{y}, \vec{Y}), R(\vec{y}, \vec{Y}), P, N)$ in $\mathsf{V}^1(\mathsf{VPV})$.

From the definition of $\text{ImpProof}_{\text{Res}}$, and from the fact that $t(\vec{y}, \vec{Y}) = t = n + 2s + 1$, we need to prove the following formulas:

- $\text{isCkt}(t, R(\vec{y}, \vec{Y}))$
- $\text{Active}(n, t, R(\vec{y}, \vec{Y}), 1^t) \wedge \text{Empty}(n, t, R(\vec{y}, \vec{Y}), 1^t)$
- The formula

$$\begin{aligned}\forall Y < t : \text{Active}(n, t, R(\vec{y}, \vec{Y}), Y) \rightarrow \\ \text{isWeaken}(n, t, R(\vec{y}, \vec{Y}), Y) \vee \text{isResolve}(n, t, R(\vec{y}, \vec{Y}), Y) \vee \\ \exists i < m : \text{isAxiomW}(n, m, t, i, P, N, R(\vec{y}, \vec{Y}), Y).\end{aligned}$$

The fact that R outputs a circuit is easily provable in $\mathsf{V}^1(\mathsf{VPV})$. For the remaining statements we must analyze the code of H.

To prove

$$\text{Active}(n, t, R(\vec{y}, \vec{Y}), 1^t) \wedge \text{Empty}(n, t, R(\vec{y}, \vec{Y}), 1^t),$$

we observe this follows from the first "If" statement of the description of Algorithm 2 – the line is active and empty by definition.

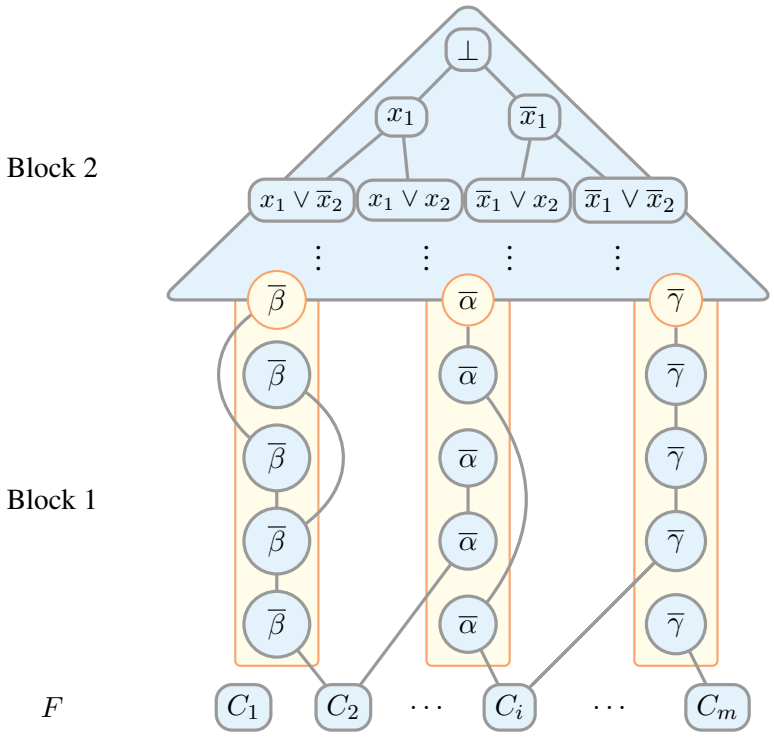


Figure 1: The structure of the implicit Resolution proof of $F = C_1 \wedge \ldots \wedge C_m$. The second block is a complete tree-like resolution refutation. Each leaf of this complete proof is labeled with a clause $\overline{\alpha}$ which is the negation of (the conjunct representing) a total assignment $\alpha \in \{0,1\}^n$; this constitutes the first block. At that leaf, the reduction from Search_F to Iter (marked in yellow) is run on the total assignment α and each node is labeled with $\overline{\alpha}$. The correctness of the Iter instance guarantees that we arrive at a clause falsified by α.

Similarly we can prove the formula

$$\forall Y < t : \mathrm{Active}(n,t,R(\vec{y},\vec{Y}),Y) \rightarrow \mathrm{isWeaken}(n,t,R(\vec{y},\vec{Y}),Y) \vee \mathrm{isResolve}(n,t,R(\vec{y},\vec{Y}),Y) \vee \exists i < m : \mathrm{isAxiomW}(n,m,t,i,P,N,R(\vec{y},\vec{Y}),Y).$$

using the code of Algorithm 2 and from the assumption that $\mathrm{Reduction}_{\mathrm{ITER}}$ holds. If a line is active, then by inspection of the algorithm, the step is either a weakening (with tag 00), a Resolution (with tag 01), or a weakening of an axiom (with tag 10). In the first two cases, the definition of Algorithm 2 shows that the output lines are correct formed, and hence isWeaken and isResolve will hold for these lines. In the third case, the line indexed is a solution, and this can only happen if Lines 13-17 execute in Algorithm 2. In this case, we must be sure that the output of $D(X',Y) = Z$ actually indeces into a false clause of the formula under the assignment X'. Since we have assumed that $\mathrm{Reduction}_{\mathrm{ITER}}$ holds, we

have (cf. Definition 4.5) that if C encodes a solution on the input X', Y, then the string output by $D(X', Y)$ is indeed an false clause of F. Hence we can conclude that this is a correct axiom weakening of a clause of F. □

4.2 G_1 is p-equivalent to $\langle EF, \mathsf{Iter}\rangle$

In this section we prove Theorem 4.3. Our proof will rely on the well known TV^1/G_1-witnessing theorem of [BK94,Ske05,BB09], stating that the $\forall\Sigma^B_1$-consequences of TV^1 are witnessed by PLS functions. This was originally proved in the uniform setting by Buss and Krajíček [BK94] for the first order T^1_2, and later reproved by Cook and Nguyen for TV^1 [CN10]. A follow-up work of Beckmann and Buss [BB09] generalized these witnessing theorems for TV^i, $i \geq 1$.

A priori, there is little similarity between G_1 and $\langle EF, \mathsf{Iter}\rangle$. However some intuition for the proof can be built by juxtaposing this result with the TFNP^{dt} characterizations of Tree-like Resolution with FP^{dt}, and bounded-width Resolution with Iter^{dt}. The reason that bounded-width resolution is not similarly characterized by efficient decision-tree protocols is due to the DAG-like nature of the proofs. This same issue precisely identifies the jump from EF to G_1.

Showing $\langle EF, \mathsf{Iter}\rangle \leq_p G_1$ will require reducing the G_1 witnessing problem to Iter, provably in Extended Frege. To do this, we formalize a G_1-witnessing theorem in Extended Frege by propositionally translating an efficient TV^1-witnessing theorem of [CN10]. To show $G_1 \leq_p \langle EF, \mathsf{Iter}\rangle$, we prove that TV^1 proves the reflection principle for $\langle EF, \mathsf{Iter}\rangle$, which will amount to proving the reflection principle of EF, as well as the totality of Iter.

We begin with some definitions.

Definition 4.8. The class $\Sigma^q_0 = \Pi^q_0$ consists of all (quantifer-free) propositional formulas. The class of Σ^q_1 formulas are of the form $\exists x_1, \ldots, x_n A(x_1, \ldots, x_n, y)$ where $A(x, y) \in \Sigma^q_0$, and $y = y_1, \ldots, y_m$.

The system G is a proof system for quantified Boolean formulas, that natural extends the sequent calculus proof system for Σ^q_0 formulas. See Appendix A for a formal description of G. The subsystem G_1 is G but with the restriction that all formulas occurring in all sequents of the proof are Σ^q_1 formulas.

Definition 4.9 (Witnessing for G_1)**.** Let $F = \exists y\, A(x, y)$, where A is a Σ^q_0 formula, and let Π be a G_1 proof of F. On input F, Π, and assignment $\alpha \in \{0,1\}^n$ to the x-variables, the G_1 *witnessing search problem* asks to find an assignment to the y-variables such that $A(x, y)$ holds. If Π is not a valid G_1-proof of F, then output $\perp$.

For simplicity, we will often write $\mathsf{Wit}(F, \Pi)$. We use the following theorem of Cook and Nguyen, which proves TV^1-witnessing in V^1.

Theorem 4.10 (Uniform TV^1 Witnessing, [CN10])**.** *Suppose that* $\mathsf{TV}^1 \vdash \psi$ *for* $\psi \equiv \forall x, X\, \exists y, Y\, \varphi(x, X, y, Y)$*, with* $\varphi \in \Sigma^B_0$*. Then there is an* Iter *instance defined by its*

VPV*-predicate* graph*,* $F(x, X, y, Y)$*, and* VPV*-functions* r_1, r_2*, such that*

$$\mathsf{V}^1(\mathsf{VPV}) \vdash [\forall x, X\, \exists y, Y\, F(x, X, y, Y)] \supset \varphi(x, X, r_1(x, X, y, Y), r_2(x, X, y, Y))^5$$

We additionally need one more claim for converting propositional unsatisfiable formulas into quantified tautologies.

Claim 4.11. Let $F = \bigwedge_{i=1}^{m} C_i$ be an unsatisfiable CNF, and suppose $G_1 \vdash \neg F$. Then

$$G_1 \vdash \exists i \left([\![i = 1]\!] \vee \cdots \vee [\![i = m]\!] \right) \wedge \left(([\![i = 1]\!] \rightarrow \neg C_1) \wedge \cdots \wedge ([\![i = m]\!] \rightarrow \neg C_m) \right).$$

This is the quantified version of the UNSAT_F formula of Definition Definition 3.1, and it is near identical to the first-order formula UNSAT_F used to define $\mathsf{Reduction}_{\mathrm{ITER}}$. We will denote this as UNSAT_F^q.

Lemma 4.12. $\mathsf{Wit}(F, \Pi, x)$ *is provably total in* TV^1.

Proof. We need the following well-known facts.

1. For fixed (F, Π) where Π is a valid G_1-proof of Σ_1^q-formula F, $\mathsf{V}^1 \vdash \mathsf{Proof}_{G_1}(F, \Pi)$,
2. $\mathsf{TV}^1 \vdash \Sigma_1^q - \mathsf{Refl}_{G_1}(F, \Pi, x)$

See [Kra19, CN10] for proofs of these facts. As well, see Appendix A.2 for a brief discussion of Σ_1^q-Refl_P.

By definition of $\mathsf{Wit}(F, \Pi, x)$, if Π is not a valid proof of F then we output $\perp$. For F, Π satisfying $\mathsf{Proof}_{G_1}(F, \Pi)$, we additionally have $\mathsf{TV}^1 \vdash \mathsf{Proof}_{G_1}(F, \Pi)$. Let $F \equiv \exists y \varphi(x, y)$, for $\varphi \in \Sigma_0^q$. As TV^1 proves G_1 reflection, if $G_1 \vdash F$, then $\mathsf{TV}^1 \vdash \exists Y\, \varphi(X, Y)$.

To complete the proof, we only need to give a $\mathsf{V}^1(\mathsf{VPV})$ definition of $\mathsf{Wit}(F, \Pi)$. We need the following standard VPV predicates and functions[6]:

- $\mathsf{Fmla}(F)$ — Expresses that F is a valid quantified propositional formula,
- $\mathsf{Sub}(F, X, W)$ — Outputs F with the terms X substituted into free variables, and W substituted into existentially quantified variables, removing the quantifiers

We may define $\mathsf{Wit}(F, \Pi, X) \equiv \mathsf{Fmla}(F) \supset \exists Y\, \mathsf{SAT}(\mathsf{Sub}(F, X, Y))$. Given this definition, it is clear that since $\mathsf{TV}^1 \vdash \exists Y\, \varphi(X, Y)$, then $\mathsf{TV}^1 \vdash \forall X \mathsf{Wit}(F, \Pi, X)$. □

Lemma 4.13. Search_F *reduces to* $\mathsf{Wit}_{F,\Pi}$ *in* V^1*. That is, for* Π *a* G_1 *refutation of CNF* F*,* $\mathsf{V}_1 \vdash \mathsf{Reduction}_{\mathsf{Wit}_{F,\Pi}}$.

Proof. The two polytime functions $A(F, X)$, $B(F, X, i)$ in the reductions are the obvious ones: $A(F, X)$ maps F to UNSAT_F^q, with X substituted in for the variables, and $B(F, X, i)$ maps index $i \in [m]$ to clause C_m of F. V^1 proves these functions form a valid reduction to $\mathsf{Wit}(F, \Pi)$ by proving $\forall i \leq m\, \mathsf{SAT}(\mathsf{Sub}(\mathrm{Unsat}_F^q, X, i)) \supset \neg \mathsf{Clause}(F, i)$. This follows by definition of UNSAT_F^q. □

[5] In fact, they only need V^0! This is shown by proving Iter with AC^0 circuits is still complete for PLS.

[6] With more work, these are further defined in VTC^0, [CN10]

We are now ready to prove the main theorem of this subsection, the equivalence between G_1 and $\langle EF, \mathsf{Iter}\rangle$.

Proof of Theorem 4.3. We begin by proving $G_1 \geq_p \langle EF, \textsc{Iter}\rangle$. Suppose there is a G_1 refutation of an unsatisfiable CNF formula F. Then by Claim 4.11, $G_1 \vdash \text{UNSAT}_F^q$. V^1 proves $\text{Reduction}_{\textsc{Iter}}(n, m, C, D, P, N)$ for circuits C, D defined by combining the reduction of Search_F to $\mathsf{Wit}(F, \Pi)$ (Claim 4.11) with the provable totality of $\mathsf{Wit}(F, \Pi)$ in TV^1 (Lemma 4.12) and TV^1 witnessing (Theorem 4.10). Applying the standard propositional translation for V^1, we get that there is an EF proof τ of $\|\mathsf{Reduction}_{\textsc{Iter}}\|$, with size $|\tau| = O(|\Pi|^k)$, for some fixed k dependent only on the proof of Theorem 4.10.

Next, we prove the converse direction, that $G_1 \leq_p \langle EF, \textsc{Iter}\rangle$. It is well known that if the theory TV^1 proves the reflection principle of a propositional proof system P, then $P \geq_p G_1$ [Kra95]. Hence, we only need to show that TV^1 proves the reflection principle of $\langle EF, \textsc{Iter}\rangle$. We give the explicit formula with a *refutation* predicate Ref indicating a valid refutation, and a *proof* predicate Proof indicating a valid proof. We need both to sensibly define reflection for $\langle EF, \textsc{Iter}\rangle$.

$$\mathsf{Refl}_{\langle EF,\textsc{Iter}\rangle} := \|\mathsf{Ref}_{\langle EF,\textsc{Iter}\rangle}(F, C, D, \Pi) \supset \neg\mathsf{SAT}(F, X)\|,$$

where $\mathsf{Ref}_{\langle EF,\textsc{Iter}\rangle}(F, C, D, \Pi) := \mathsf{Proof}_{EF}(\text{Reduction}_{\textsc{Iter}}(n, m, C, D, P, N), \Pi)$ and X is an assignment to the variables of F.

Assume $\mathsf{Ref}_{\langle EF,\textsc{Iter}\rangle}(F, C, D, \Pi)$ is true. Then, since $\mathsf{V}^1/\mathsf{TV}^1 \vdash \text{Refl}_{EF}$, we have

$$\mathsf{TV}^1 \vdash \mathsf{SAT}(\|\text{Reduction}_{\textsc{Iter}}(n, m, C, D, P, N), X)\|).$$

As a reminder, the formula $\text{Reduction}_{\textsc{Iter}}(n, m, s, C, D, P, N)$ is defined to be the conjunction of the formulas:

- $\text{isCkt}(n, C)$,
- $\text{isCkt}(n + s, D)$,
- and the formula

$$\forall X < n, Y < s : \textsc{Iter}(s, \text{Eval}(n, s, C, X), Y) \rightarrow \\ \text{UNSAT}(n, m, P, N, X, \text{Eval}(n + s, m, D, X \circ Y)).$$

Since TV^1 proves the totality of Iter, we can cut on the formula $\forall X < n, Y < s : \textsc{Iter}(s, \text{Eval}(n, s, C, X), Y)$ to derive UNSAT_F, which is provably equivalent to $\neg\mathsf{SAT}(F, X)$ in V^1. □

A Comparison with G_1^* and FP-Witnessing. Wang [Wan13] showed that $[EF, \text{TreeRes}]$ is polynomially equivalent to Extended Frege. Our proof simplifies and generalizes their proof strategy, which we summarize below.

To show that $[EF, \text{TreeRes}] \geq_p EF$, the same strategy as ours is used, by setting the succinct tree-like Resolution proof to be the trivial exponential size proof, and using

the original Extended Frege proof to show that the circuit is valid. For the reverse of $[EF, \text{TreeRes}] \leq_p EF$, Wang roughly does the following:

1. Let C succinctly represent proof Π, and Π_C be an EF proof of the validity of C. Use extension variables on the evaluation of C to define a path of polynomial length through the succinctly representing tree-like resolution proof Π, from root to axiom,
2. As EF proves the reflection principle of tree-like resolution, there must be a root-to-leaf path from $\bot$ to a falsified axiom of unsat CNF F.
3. Combine with Π_C to get a complete refutation of F.

This proof implicitly runs through G_1^*/V^1-witnessing without any mention, as the algorithm used to witness a root-to-leaf path finding a falsified clause is going to be the same witnessing algorithm suggested in the V^1-witnessing of [CN10]. Our proof of $[EF, Res] = G_1$ identifies that *efficiently provable witnessing* is the lynchpin of these equivalences.

4.3 Generalizing to G_i

By combining known witnessing theorems in the literature, as well as the connection between quantified proof systems G_i and the theories T_2^i [KP90], we can extend our main result to stronger systems. Krajíček, Skelley, and Thapen [KST07], as well as Skelley-Thapen [ST07], characterize the $\forall\Sigma_1^B$-consequences of TV^i by the *game induction principles*, GI_i, based on i-turn games.[7] In parallel work of Beckmann and Buss [BB09], they characterize the $\forall\Sigma_k^B$-consequences of TV^i, $0 \leq k \leq i$ by proving a general witnessing theorem. Moreover, they show their witnessing theorem is efficiently provable, like Theorem 4.10, in V^1. By combining these two works, we can generalize our main theorem to the following.

Theorem 4.14. $G_i \equiv_p \langle EF, \mathsf{GI}_i \rangle$.

The proof would follow the same template as Section 4.2, and uses that game induction is provably reducible to the witnessing problems of Beckmann-Buss.

5 A Generic Correspondence

In Section 4 we showed an equivalence between G_1 and $\langle EF, \text{Res} \rangle$. In this section we state necessary and sufficient conditions for a proof system to be equivalent to EF-provable reductions to a TFNP problem. Our results in this section can be seen as generalizing the equivalence between black-box TFNP classes and proof systems from [BFI23] to the white-box setting, under Extended Frege-provable reductions. At a high level, they showed that any TFNP^{dt} class (satisfying certain mild conditions) can be viewed as the total NP search problems that are decision tree reductions to the *reflection principle* (encoded as a

[7] Note that [KST07] do this for the first order theories T_2^i, which are equivalent to TV^i by the RSUV isomorphism [CN10].

propositional formula) for some proof system. For example, the problems reducible to Iter^{dt} are the problems reducible to the reflection principle for narrow Resolution. Conversely, they showed that if a proof system proves its own reflection principle and can simulate decision tree reductions then it is characterized by a TFNP^{dt} class.

In this section, we show that if a proof system P can prove its own reflection principle (cf. Definition 2.4) and is at least as strong as Extended Frege, then EF-provable reductions to the *Wrong Proof problem* for P are equivalent to P proofs.

Definition 5.1. $\mathrm{WrongProof}_P := \{\mathrm{WrongProof}_{P,n,m,s}\}_{n,m,s}$ is the false clause search problem associated with the unsatisfiable formula $\neg\mathrm{Refl}_P$, which asserts that a formula has both a P-proof and a falsifying assignment. Formally,

$$\begin{aligned}\mathrm{WrongProof}_{P,n,m,s}(x,z) :&= \mathrm{UNSAT}_{\neg\mathrm{Refl}_{P,n,m,s}}(x,z)\\ &= \mathrm{UNSAT}_{\mathrm{Proof}_{P,n,m,s}}(x,z_1) \vee \mathrm{UNSAT}_{\neg\mathrm{SAT}_{n,m}}(x,z_2), \quad (2)\end{aligned}$$

Recalling that we use a one-hot encoding in UNSAT, z_1 are the bits that represent the clauses of Proof, while z_2 are those belonging to SAT. As well, $x = (\alpha, F, \Pi)$ is a tuple encoding an assignment, a formula (claimed to be a tautology), and a proof Π. Recall from Definition 3.1 that this is a problem in TFNP.

Informally, $\mathrm{WrongProof}_P$ is the following search problem: Given a CNF formula F, an assignment x to the variables of F, and a proposed P-refutation Π of F, either output a clause of F that is falsified under x or an error in Π. We note that the WrongProof problem was originally introduced by Goldberg and Papadimitriou [GP18] in a slightly different setting — our usage is closer to that of [LLR24]. The main theorem of this section says that if a sufficiently strong proof system P can prove its own reflection principle, then it is characterized by EF-provable reductions to $\mathrm{WrongProof}_P$.

Theorem 5.2. *Let $P \geq_p EF$ be any proof system which has polynomial-size proofs of Refl_P. Then P is polynomially equivalent to $\langle EF, \mathrm{WrongProof}_P\rangle$*

Said differently, there is a size-s P-refutation of F iff there is an EF-provable many-one reduction from Search_F to $\mathrm{WrongProof}_P$ where both the reduction and the Extended Frege proof have size polynomial in s. It may be helpful to keep in mind the proofs of Proposition 3.6 and Proposition 3.8, as our proof of this theorem follows a similar trajectory.

We will use the following simple properties of Extended Frege.

Claim 5.3. For any CNF formula F on n variables, the following hold:

1. Extended Frege has $\mathsf{poly}(|F|)$-size proofs of $\neg F(x) \to \mathrm{UNSAT}_F(x,y)$, for new variables y, and $\mathrm{UNSAT}_F(x,y) \to \neg F(x)$.
2. For any circuit $C : \{0,1\}^t \to \{0,1\}^n$, if Extended Frege can prove F then it also has $\mathsf{poly}(|C|)$-size proofs of $\mathrm{Eval}(C, x', x) \wedge F(x)$.

Proof. We sketch the proof. Let $F = C_1 \wedge \ldots \wedge C_m$. For the first item, Extended Frege begins by deriving $F \vee \neg F$. Then, it introduces new variables y such that $y_i \iff \neg C_i$.

From $\neg F = \neg C_1 \vee \ldots \vee \neg C_m$ and the definition of the z_i's it then derives $\mathrm{UNSAT}_F(x, z)$ to obtain $F \to \mathrm{UNSAT}_F(x, z)$. The proof of the converse is similar.

The second item is standard. Beginning with $F(x)$, and treating x as the output gates of C, Extended Frege introduces new variables $x'_1, \ldots, x'_m, g_1, \ldots, g_{|S|}$, where g_i will represent the value of the i-th gate of C, such that the input variables are x', and clauses enforcing that the value of g_i is correctly derived from the values of its children in C. □

We prove Theorem 5.2 over the new two lemmas, the forward direction is captured by Lemma 5.4 taking $P = Q$, while the backwards direction is given in Lemma 5.5.

Lemma 5.4. *Let P, Q be proof systems such that $EF \leq_p P \leq_p Q$ and Q proves the reflection principle for P. If $\langle EF, \mathrm{WrongProof}_P \rangle$ has a size s refutation of F, then Q has a size $\mathsf{poly}(s)$ refutation of F.*

Proof. Let (C, D, Π) be an $\langle EF, \mathrm{WrongProof}_P \rangle$ refutation of an unsatisfiable CNF formula F. In particular, (C, D) is a mapping reduction and Π is an Extended Frege proof of $\mathrm{Reduction}_{\mathrm{WrongProof}_P}$. We will derive the premise of $\mathrm{Reduction}_{\mathrm{WrongProof}_P}$,

$$\mathrm{isCkt}(C) \wedge \mathrm{isCkt}(D) \wedge \mathrm{Eval}(C, x, x') \wedge \mathrm{Eval}(D, (x, y), z) \wedge \mathrm{WrongProof}_P(x', y), \tag{3}$$

and then cut it with the conclusion of Π to prove $\neg F$.

First, we use that P has small proofs of Refl_P in order to derive $\mathrm{UNSAT}_{\neg \mathrm{Refl}_P}(x', y)$, which is equivalent to $\mathrm{WrongProof}_P(x', y)$. More formally, by Claim 5.3, Extended Frege can prove

$$\neg \mathrm{Refl}_P(x') \implies \mathrm{UNSAT}_{\neg \mathrm{Refl}_P}(x', y),$$

which follows by first writing down the statement $\neg \mathrm{Refl}_P \vee \mathrm{Refl}_P$ and then, from $\neg \mathrm{Refl}_P$, introducing new variables y and deriving $\mathrm{UNSAT}_{\neg \mathrm{Refl}_P}(x', y)$. Cutting this formula with $\mathrm{Refl}_P(x')$, we obtain $\mathrm{WrongProof}_P(x', y)$.

Finally, we derive the statements about the circuits C and D. Since these are fixed circuits, $\mathrm{isCkt}(C)$ and $\mathrm{isCkt}(D)$ are constant true formulas. As well, by Claim 5.3, Extended Frege can derive $\mathrm{Eval}(C, x, x') \wedge \mathrm{Eval}(D, (x, y), z)$, where C and D are fixed. This completes the derivation of the premise (3). Using the supplied proof Π we can derive $\mathrm{Reduction}_{\mathrm{WrongProof}_P}$. Cutting this with (3) gives $\mathrm{UNSAT}_F(x, z)$, from which we can deduce $\neg F$ using Claim 5.3. □

Lemma 5.5. *Let $P \geq_p EF$ be any proof system such that P has polynomial-size proofs of Refl_P. If there is a size-s $\langle EF, \mathrm{WrongProof}_P \rangle$ of F then there is size $\mathsf{poly}(s)$ proof of F.*

Proof of Lemma 5.5. For the backwards direction, let Π' be a P-refutation of F, and let Π be a similar-sized proof that $\neg F$ is a tautology (for example by cutting Π with $F \vee \neg F$. We will need Π as we have phrased Refl for proofs of tautologies.

We construct a mapping reduction (C, D) from Search_F to $\mathrm{WrongProof}_P$. Recall that $\mathrm{WrongProof}_P$ has three inputs: a formula, an n-bit truth assignment claiming to falsify the

formula, and a P-proof claiming that the formula is a tautology. On input $x \in \{0,1\}^n$, the circuit C maps x to the truth assignment variables, and hard-codes Π for the proof variables and $\neg F$ for the formula variables. By assumption, Π is a valid P-proof of $\neg F$ and so $\mathrm{Proof}(C(x)) = \mathrm{Proof}(\Pi, \neg F)$ is the constant true formula. As well, denoting by $\mathrm{SAT}_{\neg F}$ the formula SAT with its formula-input fixed to $\neg F$, observe that

$$\mathrm{UNSAT}_{\neg \mathrm{SAT}}(A(x), z) = \mathrm{UNSAT}_{\neg \mathrm{SAT}_{\neg F}}(x, z) = \mathrm{UNSAT}_F(x, z),$$

by definition. Hence,

$$\begin{aligned}\mathrm{WrongProof}_P(C(x), z) &= \mathrm{UNSAT}_{\mathrm{Proof}}(C(x), z_1) \vee \mathrm{UNSAT}_{\neg \mathrm{SAT}}(C(x), z_2) \\ &= \bot \vee \mathrm{UNSAT}_{\neg \mathrm{SAT}_{\neg F}}(x, z_2) \\ &= \mathrm{UNSAT}_F(x, z_2).\end{aligned}$$

Finally, define the circuit D to map $(z_1, z_2) \mapsto z_2$, recalling that we use a one-hot encoding for z.

It remains to construct an Extended Frege proof of $\mathrm{Reduction}_{\mathrm{WrongProof}_P(C,D,F)}$ using the assumed proof Π. Beginning from the premise of this statement,

$$\mathrm{Eval}(C, x, x') \wedge \mathrm{Eval}(D, (x, y), z) \wedge \mathrm{WrongProof}_P(x', y),$$

where we have used that, since A and D are fixed, $\mathrm{isCkt}(C)$ and $\mathrm{isCkt}(D)$ are constant true formulas. Extended Frege must argue two things:

1. That C hard-wires $(\neg F, \Pi)$ and so $\mathrm{Proof}(\Pi, \neg F)$ is the constant true formula, and that D maps indices of falsified clauses of $\neg\mathrm{SAT}(x, \neg F)$ to indices of falsified clauses of F.
2. That Π is an P-proof of F.

Examining the gates of the circuit C, Extended Frege can deduce that the output string of $\mathrm{Eval}(C, x, x')$ has the form $x' = (x, \neg F, \Pi)$. Hence, it can also derive that

$$\mathrm{WrongProof}_P(x', y) = \mathrm{UNSAT}_{\mathrm{Proof}}(\neg F, \Pi, y_1) \vee \mathrm{UNSAT}_{\neg \mathrm{SAT}}(x, \neg F, y_2),$$

where $\neg F, C, \Pi$ are fixed, $y = y_1 \circ y_2$, and we have expanded the definition of $\mathrm{WrongProof}$. Use Π to derive $\mathrm{Proof}(\neg F, \Pi)$ and cut it on $\mathrm{UNSAT}_{\neg \mathrm{Proof}}(F, \Pi, y_1)$ in order to derive

$$\mathrm{UNSAT}_{\mathrm{SAT}}(x, \neg F, y_2).$$

Finally, examining the gates of D, Extended Frege can prove that the output string of $\mathrm{Eval}(D, (x, y), z)$ satisfies $z = y_2$, where $y = y_1 \circ y_2$. Therefore, it can derive

$$\mathrm{UNSAT}_{\neg \mathrm{SAT}}(x, \neg F, z) = \mathrm{UNSAT}_F(x, z),$$

where equality holds syntactically by definition since F is fixed. □

References

[AKPS24] Noel Arteche, Erfan Khaniki, Ján Pich, and Rahul Santhanam. From proof complexity to circuit complexity via interactive protocols. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 12:1–12:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of mathematics*, pages 781–793, 2004.

[BB09] Arnold Beckmann and Samuel R. Buss. Polynomial local search in the polynomial hierarchy and witnessing in fragments of bounded arithmetic. *J. Math. Log.*, 9(1), 2009.

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

[BFI23] Sam Buss, Noah Fleming, and Russell Impagliazzo. TFNP characterizations of proof systems and monotone circuits. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 30:1–30:40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[BIK+94] Paul Beame, Russell Impagliazzo, Jan Krajíček, Toniann Pitassi, and Pavel Pudlák. Lower bounds on Hilbert's Nullstellensatz and propositional proofs. In *Proceedings of the 35th Symposium on Foundations of Computer Science (FOCS)*, pages 794–806, 1994.

[BK94] Sam Buss and Jan Krajíček. An application of boolean complexity to separation problems in bounded arithmetic. In *Proceedings of the London Mathematical Society*, volume 69, pages 1–21, 1994.

[BKKK20] Sam Buss, Valentine Kabanets, Antonina Kolokolova, and Michal Koucký. Expander construction in vnc^1. *Ann. Pure Appl. Log.*, 171(7):102796, 2020.

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

[CN10] Stephen Cook and Phuong Nguyen. *Logical Foundations of Proof Complexity*. Cambridge University Press, 2010.

[DR23] Ben Davis and Robert Robere. Colourful TFNP and propositional proofs. In Amnon Ta-Shma, editor, *38th Computational Complexity Conference, CCC 2023, July 17-20, 2023, Warwick, UK*, volume 264 of *LIPIcs*, pages 36:1–36:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[FGIM26] Noah Fleming, Anna Gál, Deniz Imrek, and Christophe Marciot. Separations above TFNP from sherali-adams lower bounds. *CoRR*, abs/2602.16810, 2026.

[FGJ+26] Noah Fleming, Stefan Grosser, Siddhartha Jain, Jiawei Li, Hanlin Ren, Morgan Shirley, and Weiqiang Yuan. Total search problems in ZPP. In Shubhangi Saraf, editor, *17th Innovations in Theoretical Computer Science Conference, ITCS 2026, Bocconi University, Milan, Italy, January 27-30, 2026*, LIPIcs, pages 60:1–60:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2026.

[FGPR24] Noah Fleming, Stefan Grosser, Toniann Pitassi, and Robert Robere. Black-box PPP is not turing-closed. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1405–1414. ACM, 2024.

[FIM25] Noah Fleming, Deniz Imrek, and Christophe Marciot. Provably total functions in the polynomial hierarchy. In Srikanth Srinivasan, editor, *40th Computational Complexity Conference, CCC 2025, Toronto, Canada, August 5-8, 2025*, LIPIcs, pages 28:1–28:40. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[GHJ+22a] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Further collapses in TFNP. In *Proceedings of the 37th Computational Complexity Conference (CCC)*, pages 33:1–33:15, 2022.

[GHJ+22b] Mika Göös, Alexandros Hollender, Siddhartha Jain, Gilbert Maystre, William Pires, Robert Robere, and Ran Tao. Separations in proof complexity and TFNP. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1150–1161. IEEE, 2022.

[GKRS18] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 124, pages 38:1–38:19, 2018.

[GKRS19] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:19. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019.

[GMRS25] Mika Göös, Gilbert Maystre, Kilian Risse, and Dmitry Sokolov. Supercritical tradeoffs for monotone circuits. In Michal Koucký and Nikhil Bansal, editors,

Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025, pages 1359–1370. ACM, 2025.

[GP18] Paul W. Goldberg and Christos H. Papadimitriou. Towards a unified complexity theory of total functions. *J. Comput. Syst. Sci.*, 94:167–192, 2018.

[HKT24] Pavel Hubácek, Erfan Khaniki, and Neil Thapen. TFNP intersections through the lens of feasible disjunction. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, Berkeley, CA, USA, January 30 - February 2, 2024*, LIPIcs, pages 63:1–63:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Jer04] Emil Jerábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Ann. Pure Appl. Log.*, 129(1-3):1–37, 2004.

[Jer07] Emil Jerábek. Approximate counting in bounded arithmetic. *J. Symb. Log.*, 72(3):959–993, 2007.

[JJ26] Raheleh Jalali and Ondřej Ježil. Feasibility of primality in bounded arithmetic. In *Forum of Mathematics, Sigma*, volume 14, page e49. Cambridge University Press, 2026.

[Kam19] Pritish Kamath. *Some hardness escalation results in computational complexity theory*. PhD thesis, Massachusetts Institute of Technology, 2019.

[Kha24] Erfan Khaniki. Jump operators, interactive proofs and proof complexity generators. *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 573–593, 2024.

[KNY19] Ilan Komargodski, Moni Naor, and Eylon Yogev. White-box vs. black-box complexity of search problems: Ramsey and graph property testing. *J. ACM*, 66(5):34:1–34:28, 2019.

[KP90] Jan Krajíček and Pavel Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36(1):29–46, 1990.

[Kra95] Jan Krajicek. *Bounded arithmetic, propositional logic and complexity theory*, volume 60. Cambridge University Press, 1995.

[Kra01a] Jan Krajíček. Implicit proofs and propositional proof complexity. In S. Buss, P. Hajek, and P. Pudlák, editors, *Logic Colloquium '98*, volume 13 of *Lecture Notes in Logic*, pages 145–165. Association for Symbolic Logic, 2001.

[Kra01b] Jan Krajíček. On the weak pigeonhole principle. *Fundamenta Mathematicae*, 170:123–140, 2001.

[Kra04] Jan Krajíček. Implicit proofs. *Journal of Symbolic Logic*, 69(2):387–397, 2004.

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

[KST07] Jan Krajíček, Alan Skelley, and Neil Thapen. Np search problems in low fragments of bounded arithmetic. *The Journal of Symbolic Logic*, 72(2):649–672, 2007.

[LLR24] Jiawei Li, Yuhao Li, and Hanlin Ren. Metamathematics of resolution lower bounds: A tfnp perspective. *arXiv preprint arXiv:2411.15515*, 2024.

[LPR24] Yuhao Li, William Pires, and Robert Robere. Intersection classes in TFNP and proof complexity. In Venkatesan Guruswami, editor, *15th Innovations in Theoretical Computer Science Conference, ITCS 2024, Berkeley, CA, USA, January 30 - February 2, 2024*, LIPIcs, pages 74:1–74:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[MP91] Nimrod Megiddo and Christos Papadimitriou. On total functions, existence theorems and computational complexity. *Theoretical Computer Science*, 81(2):317–324, 1991.

[Pap94] Christos Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

[PT26] Pavel Pudlák and Neil Thapen. Quantified propositional calculi and narrow implicit proofs, 2026.

[Pud00] Pavel Pudlák. Proofs as games. *Am. Math. Mon.*, 107(6):541–550, 2000.

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

[Raz93] A. A. Razborov. An equivalence between second order bounded domain arithmetic and first order arithmetic. *In Clote and Krajicek (editors) Arithmetic, Proof Theory, and Computational Complexity*, 1993.

[Ske05] Alan Skelley. Some results concerning G1 and polynomial local search. Manuscript, 2005.

[ST07] Alan Skelley and Neil Thapen. The provably total search problems of bounded arithmetic. *submitted (preprint 2008)*, 2007.

[Wan13] Zi Chao Wang. Implicit resolution. *Log. Methods Comput. Sci.*, 9(4), 2013.

A Appendix: Proof Systems and Bounded Arithmetic

A.1 Propositional and QBF Proof Systems

We formally describe proof systems Frege, Extended Frege, and G_i. Refer to [Kra19] for results on the strength of these systems.

Frege and Extended Frege.

Definition A.1 (Gentzen's PK: Propositional Calculus)**.** Let all formulas be propositional and under the DeMorgan basis. A *PK proof* uses the following rules.

$$\text{(Ax)}\ \frac{}{A \to A}\ \text{Ax} \qquad \text{(Cut)}\ \frac{\Gamma \to A \qquad \Delta, A \to B}{\Gamma, \Delta \to B}\ \text{Cut}$$

$$\text{(WL)}\ \frac{\Gamma \to A}{\Gamma, B \to A}\ \text{WL} \qquad \text{(WR)}\ \frac{\Gamma \to A}{\Gamma \to A, B}\ \text{WR}$$

$$\text{(CL)}\ \frac{\Gamma, A, A \to B}{\Gamma, A \to B}\ \text{CL} \qquad \text{(CR)}\ \frac{\Gamma \to A, A}{\Gamma \to A}\ \text{CR}$$

$$\text{(EL)}\ \frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ \text{EL} \qquad \text{(ER)}\ \frac{\Gamma \to A, B, \Delta}{\Gamma \to B, A, \Delta}\ \text{ER}$$

$$\text{(neg L)}\ \frac{\Gamma \to A, \Delta}{\Gamma, \neg A \to \Delta}\ \text{neg L} \qquad \text{(neg R)}\ \frac{\Gamma, A \to \Delta}{\Gamma \to \neg A, \Delta}\ \text{neg R}$$

$$\text{(and L)}\ \frac{\Gamma, A, B \to \Delta}{\Gamma, A \wedge B \to \Delta}\ \text{and L} \qquad \text{(and R)}\ \frac{\Gamma \to A, \Delta \qquad \Gamma \to B, \Delta}{\Gamma \to A \wedge B, \Delta}\ \text{and R}$$

$$\text{(or L)}\ \frac{\Gamma, A \to \Delta \qquad \Gamma, B \to \Delta}{\Gamma, A \vee B \to \Delta}\ \text{or L} \qquad \text{(or R)}\ \frac{\Gamma \to A, B, \Delta}{\Gamma \to A \vee B, \Delta}\ \text{or R}$$

Extended Frege, or ePK, allows the addition of *extension variables* to PK.

Definition A.2. Let F be a PK-derivation from initial formulas $F_1, \ldots, F_k$ over variables p_i. Let e be a variable not among the p_i variables of F. An *extension rule* is the inference, $\to e \leftrightarrow A$, for A a formula over variables p_i. We say that e is an *extension variable*.

Extended Frege, or ePK, is the propositional proof system of PK, with the addition of extension rules.

Quantified Propositional Calculus and G_i systems. Adding quantified formulas and rules for quantifiers defines the more powerful G proof system and G_i subsystems.

The proof system G generalizes Frege and Extended Frege systems to general quantified Boolean formulas (QBF).

Definition A.3 (G and G_i Subsystems)**.** We adapt Frege to allow quantified formulas introduced by the following rules:

$$(\forall\text{-left})\ \frac{A(B),\ \Gamma \to \Delta}{\forall x A(x), \Gamma \to \Delta} \qquad (\forall\text{-right})\ \frac{\Gamma \to \Delta, A(p)}{\Gamma \to \Delta,\ \forall x\, A(x)}$$

$$(\exists\text{-left})\ \frac{A(p),\ \Gamma \to \Delta}{\exists x A(x),\ \Gamma \to \Delta} \qquad (\exists\text{-right})\ \frac{\Gamma \to \Delta, A(p)}{\Gamma \to \Delta,\ \exists x\, A(x)}$$

For the $\forall$-right and $\exists$-left rules, the variable p is called an *eigenvariable* and cannot appear in the bottom sequent. The G proof system is simply PK with these additional rules. For $i \geq 0$, G_i is a restriction of G to only allow cuts on $(\Sigma_i^q \cup \Pi_i^q)$-formulas. (That is, the cut rule for G_i can only be applied to formulas with at most i alternations of $\forall$ and $\exists$ quantifiers.)

While the G systems can prove validity of general QBF formulas, it is also interesting to understand their strength with respect to Boolean formulas. Since G (and even G_1) has all of EF rules plus additional rules, G can trivially p-simulate EF (with respect to Boolean formulas). And it is known that the tree-like version of G_1, called G_1^* is p-equivalent to EF. On the other hand, it is unknown whether or not G_1 is stronger than EF (with respect to Boolean formulas), and it is generally conjectured to be more powerful wrt p-simulations.

A.2 Bounded Arithmetic

We will assume certain preliminaries from bounded arithmetic, although we review some of the main points now. We work in *second-order* bounded arithmetic, where we are reasoning about objects with two types: *natural numbers* (denoted with lower-case variables x, y, z) and *sets of numbers* (denoted with upper-case variables X, Y, Z). We use the standard language

$$\mathcal{L}_A^2 := (0, 1, +, \cdot, |\cdot|; =_1, =_2, \leq, \in)$$

where $0, 1, +, \cdot$ are number terms representing $0, 1$ and usual addition and multiplication, and

$$|S| = \begin{cases} 0 & S = \emptyset \\ 1 + \max\limits_{i \in S} i & S \neq \emptyset \end{cases}$$

is a function which takes a set term and outputs a number which is one larger than the largest element of the set. The relations $=_1$ and $=_2$ are equality symbols for numbers and strings, respectively, $\leq$ is the normal ordering on numbers, and $x \in X$ is intended to mean usual set inclusion. We write $X(x) := x \in X$.

It is convenient to think of a set of numbers as a binary string. We follow Cook-Nguyen (CITE) and associate the set S with the string $w(S) = S(n)S(n-1)\cdots S(0)$, where n is the largest element of the set S. Note that $|S|$ is the length of the string associated with S. Moving forward we will refer to set terms as string terms, and think almost exclusively in terms of strings.

We sometimes treat strings as multi-dimensional string arrays, which can be done by standard methods using pairing relations. In this way, if X is an $n \times m$ string array, we use indexing notation $X(i,j)$ to denote the corresponding element at the index (i,j). We use $X^{[i]}$ to denote the length-m string encoded in the ith row of X. We will also sometimes need to take array slices. For instance, if $|X| = n$ and $i < j \leq n$, then we write $X(i:j)$ to mean the substring that starts at index i and ends at index j, including the first endpoint of the interval but excluding the last endpoint. If $j \leq i$ then $X(i:j)$ is just the empty string. We can also combine these notations, writing e.g. $X(i, j:k)$. All of these relations are definable using Σ^B_0 string comprehension and hence can be used freely.

V^1 and TV^1. We follow [CN10] in the presentation of V^1, a two-sorted theory for polynomial time which is equivalent to Buss' first order theory, S^1_2.

At the base of V^1 are the 2-BASIC axioms, which define the basic behaviors of the function symbols and predicates of $\mathcal{L}^2_A$. See [CN10] for a full list.

Definition A.4 (Σ^B_i-Comprehension)**.** For a formula $\varphi(x)$, the φ-COMP axiom is the following sentence,

$$\exists |X| \leq y \, \forall i < y. \, X(i) = 1 \longleftrightarrow \varphi(i).$$

If Γ is a set of formulas then Γ-COMP $:= \{\phi\text{-COMP} : \phi \in \Gamma\}$. We will be particularly interested in the collections Σ^B_i-COMP for each i.

The theory V^1 is then defined as,

$$V^1 := \text{2-BASIC} + \Sigma^B_1\text{-COMP}.$$

Theorem A.5 (FP Definability, [CN10])**.** *A function f is in FP if and only if f is Σ^B_1-definable in V^1.*

We list other well-known schemas which V^1 is able to prove.

Lemma A.6 ([CN10])**.** *(i) $V^1 \vdash \Sigma^B_1$-IND, (ii) $V^1 \vdash \Sigma^B_1$-MIN, and (iii) $V^1 \vdash \Sigma^B_1$-MAX, referring to the standard induction, minimization, and maximization principles respectively.*

The theory $V^1(VPV)$ is obtained from V^1 by extending the language $\mathcal{L}^2_A$ with $\mathcal{L}_{\mathsf{FP}}$, which has a function symbol for every polynomial-time function. The axioms of $V^1(VPV)$ are the union of the axioms of V^1 with the axioms of VPV, which contain defining Cobham axioms for each of the symbols in $\mathcal{L}_{\mathsf{FP}}$. Cook and Nguyen prove that $V^1(VPV)$ is a conservative extension of V^1, and it can prove the corresponding comprehension and induction axioms for $\Sigma^B_1(\mathcal{L}_{\mathsf{FP}})$ formulas.

The theory TV^1 is a stronger theory than V^1, axiomatized by the *string induction* schema. Interpreting a string X as a binary number, we may Σ^B_0 define a string successor function $S(X)$ that adds 1 to X via binary addition. See [CN10] for the definition.

Definition A.7. String induction is the two-sorted version of full induction. Let φ be a formula.

$$\varphi\text{-}\mathsf{SIND} \triangleq \varphi(\varepsilon) \wedge (\forall X \, \varphi(X) \rightarrow \varphi(S(X))) \rightarrow \forall X \, \varphi(X).$$

With string induction, we may define TV^1:

$$\mathsf{TV}^1 \equiv \text{2-BASIC} + \Sigma_1^B\text{-SIND}.$$

Theorem A.8 ([CN10]). $\mathsf{TV}^1 \supset \mathsf{V}^1$.

As we will see below, TV^1 corresponds to "PLS" reasoning in regards to the provability of $\forall\Sigma_1^B$-sentences.

A.3 Witnessing Theorems

We recall several witnessing theorems.

Theorem A.9 (See [CN10]). *Let $\phi(\vec{x}, \vec{y}, \vec{X}, \vec{Y})$ be a Σ_0^B formula. If*

$$V^1 \vdash \forall\vec{x}, \vec{X}\exists\vec{y}, \vec{Y}\phi(\vec{x}, \vec{y}, \vec{X}, \vec{Y}),$$

then there are polynomial-time functions $f_1, \ldots, f_k$, $F_1, \ldots, F_m$ such that

$$V^1(\vec{f}, \vec{F}) \vdash \forall\vec{x}, \vec{X}\phi(\vec{x}, \vec{f}(\vec{x}, \vec{X}), \vec{X}, \vec{F}(\vec{x}, \vec{X})).$$

We also have the following related which holds for $V^1(VPV)$ [CN10].

Theorem A.10. *Let $\phi(\vec{x}, \vec{y}, \vec{X}, \vec{Y})$ be a $\Sigma_0^B(VPV)$ formula. If*

$$V^1(VPV) \vdash \forall\vec{x}, \vec{X}\exists\vec{y}, \vec{Y}\phi(\vec{x}, \vec{y}, \vec{X}, \vec{Y}),$$

then there are polynomial-time functions $f_1, \ldots, f_k$, $F_1, \ldots, F_m$ such that

$$V^1(VPV) \vdash \forall\vec{x}, \vec{X}\phi(\vec{x}, \vec{f}(\vec{x}, \vec{X}), \vec{X}, \vec{F}(\vec{x}, \vec{X})).$$

For TV^1, we get PLS-witnessing.

Theorem A.11 ([CN10]). *Let $\phi(\vec{x}, \vec{y}, \vec{X}, \vec{Y})$ be a $\Sigma_0^B(VPV)$ formula. If*

$$\mathsf{TV}^1 \vdash \forall\vec{x}, \vec{X}\exists\vec{y}, \vec{Y}\phi(\vec{x}, \vec{y}, \vec{X}, \vec{Y}),$$

then there is a polynomial-time function F, and an Iter *instance with graph $G_\phi(x, X, Z)$ such that*

$$\mathsf{V}^1(\mathsf{VPV}) \vdash G_\phi(\vec{x}, \vec{X}) \longrightarrow \phi(\vec{x}, \vec{X}, F(\vec{x}, \vec{X}, Z)).$$

As TV^1 proves the totality of Iter, we can derive a statement more like Theorem A.9. Notice, however, that one only needs V^1 to "prove" the witnessing theorem. We will need this finegrained aspect of Theorem A.11 in Section 4.

A.4 Propositional Translations

The following translations will allow us to relate second-order bounded arithemtic theories and propositional proofs. We follow the treatment in Cook-Nguyen [CN10].

Let $\phi(\vec{x}, \vec{X})$ be a Σ_0^B-formula with free number variables $\vec{x}$ and string variables $\vec{X}$. We define a polynomial-size, bounded-depth family of propositional formulas

$$||\phi(\vec{x}, \vec{X})|| = \{||\phi(\vec{x}, \vec{X})||_{\vec{n},\vec{m}} : m_i, n_i \in \mathbb{N}\},$$

where each $||\phi(\vec{x}, \vec{X})||_{\vec{m},\vec{n}}$ is a propositional formula defined such that it is valid if and only if for every $\vec{x}$ and $\vec{X}$, if $|X_k| = n_k$ for each k, then $\phi(\vec{m}, \vec{X})$ is true in the standard model.

Let us now formally define propositional translations. We use 0 and 1 for the boolean constants False and True, and write $\underline{n}$ to mean the numeral n. If t is a closed term then $t^{\mathbb{N}}$ is the value that t takes in the standard model, and if ψ is a closed sentence, we write $\psi^{\mathbb{N}}$ to mean the truth value that ψ takes under the standard model. We begin by considering the case with a single string variable $\phi(X)$, and note that if ϕ had additional number variables $\phi(\vec{x}, X)$, then we define

$$||\phi(\vec{x}, X)||_{\vec{n},m} := ||\phi(\underline{\vec{n}}, X)||_m,$$

and hence we can safely reduce to this case.

The propositional formula $||\phi(X)||_m$ is defined as follows by induction. Introduce m propositional variables $p[X]_0, p[X]_1, \ldots, p[X]_{m-1}$ representing the values of $X(i)$ for $i < m$.

Atomic Formulas. For atomic formulas, we have the following cases:

- If $\phi(X) := X = X$ then $||\phi(X)||_m = 1$.
- If $\phi(X) := t(|X|) = u(|X|)$ then $||\phi(X)||_m := (t(\underline{m}) = u(\underline{m}))^{\mathbb{N}}$
- If $\phi(X) := X(t(|X|))$, then define

$$||\phi(X)||_m := \begin{cases} 0 & m = 0 \\ p[X]_{t(m)^{\mathbb{N}}} & t(m)^{\mathbb{N}} < m - 1 \\ 1 & t(m)^{\mathbb{N}} = m - 1 \\ 0 & t(m)^{\mathbb{N}} > m - 1 \end{cases}$$

Inductive Formulas. For the induction step, we define:

- If $\phi(X) = \psi(X) \circ \nu(X)$ then $||\phi(X)||_m = ||\psi(X)||_m \circ ||\nu(X)||_m$, for $\circ \in \{\wedge, \vee\}$.
- If $\phi(X) = \neg\psi(X)$ then $||\phi(X)||_m = \neg||\psi(X)||_m$.
- If $\phi(X) = \exists x \leq t(|X|)\psi(x, X)$ then $||\phi(X)||_m = \bigvee_{i=0}^{t(\underline{m})^{\mathbb{N}}} ||\psi(\underline{i}, X)||_m$
- If $\phi(X) = \forall x \leq t(|X|)\psi(x, X)$ then $||\phi(X)||_m = \bigwedge_{i=0}^{t(\underline{m})^{\mathbb{N}}} ||\psi(\underline{i}, X)||_m$.

The case of strictly bounded number quantifiers $\exists/\forall x < t$ is handled similarly. If we have a strictly bounded quantifier like $\exists x < 0$ or $\forall x < 0$, then the corresponding $\vee$ or $\wedge$ is empty, and hence we output 0 and 1, respectively.

If we have multiple string variables $\vec{X}$, then the above construction for $||\phi(\vec{X})||_{\vec{m}}$ proceeds identically by introducing propositional variables $p[X_i]_j$ for each X_i in the list $\vec{X}$ and each $j \leq m_i$. Now we have one extra atomic formula case to handle, namely $\phi(\vec{X}) := X_i = X_j$ for $i \neq j$. In this case, we can reduce to the single-variable case by defining

$$||\phi(\vec{X})||_{\vec{m}} := |||X_i| = |X_j| \wedge \forall x < |X_i| : X_i(x) \leftrightarrow X_j(x)||_{\vec{m}}.$$

Not only can first-order formulas be propositionally translated, but V^1 proofs (of Π^B_1 formulas) can also be translated into polynomial-size Extended Frege proofs:

Theorem A.12 ([CN10]). *Let φ be Π^B_1 formula. If $\mathsf{V}^1 \vdash \varphi$, then $EF \vdash ||\varphi||_n$. Furthermore, the EF proofs are of polynomial size.*

This extends to the theory TV^1 as well.

Theorem A.13 ([CN10]). *Let φ be a Π^B_1 formula. If $\mathsf{TV}^1 \vdash \varphi$, then $G_1 \vdash ||\varphi||_n$. Furthermore, the G_1 proofs are of polynomial size.*

A.5 Formalizing Propositional Logic in V^1

We give an informal and high-level survey of the formalization of the syntax and semantics of propositional logic in bounded arithmetic. For a complete treatment, see [CN10]. The goal of this section will be to show a back-and-forth relationship between propositional proof systems and several standard theories of bounded arithmetic.

Syntax of Propositional Logic. In V^1 (and in fact, in much weaker theories), valid formula and G_i-proof predicates Fmla, Proof_{G_i} are Δ^B_1-definable. All that is required to design such a predicate are TC^0-functions needed to verify proper syntax, as well as process parse trees of formulas.

Definition A.14 (Proofs, Satisfiability, and Reflection Principles)**.** For a proof system P (a polynomial-time verifier) and parameters n, m, s, a reflection principle for P is a CNF formula

$$\text{Refl}_P := \text{Proof}_P(F, \Pi, s) \supset \text{SAT}(F, X),$$

where SAT is the VPV-predicate stating that F is a formula satisfied by the truth assignment X. As well, Proof_P is a VPV-predicate which is true iff Π is a P-proof of F. Such a formula can be obtained, for example, by taking the Cook-Reckhow translation of the verifying Turing Machine for P-proofs. Reflection principles establish the *soundness* of a proof system.

The satisfiability predicate and reflection principles may be generalized to quantified boolean formulas in the natural way, by changing the SAT predicate to be $\text{SAT}_{\Sigma^q_1} \equiv \exists |W| <$

$|F|$ SAT(F, X, W), where values for W are substituted into existentially quantified variables of F. We refer to this reflection as Σ_1^q-reflection.

Below are important theorems on reflection for theories V^1 and TV^1.

Theorem A.15 (Chapter X, [CN10])**.** V^1 *proves the reflection principle for Extended Frege, and the reflection principle (for* Σ_1^q*-formulas) for tree-like* G_1*.* TV^1 *proves the reflection principle of* G_1*.*

Theorem A.16 (Informal, [CN10])**.** *Suppose that a theory* T*, which contains* V^1*, proves the reflection principle for propositional proof system* $\mathcal{P}$*. Then for a formula* F*,* $P \vdash F \rightarrow \mathsf{T} \vdash SAT(F, X)$*.*

Corollary A.17. *If* $EF \vdash F$*, then* $\mathsf{V}^1 \vdash SAT(F, X)$*. Similarly for* TV^1*, if* $G_1 \vdash F$*, then* $\mathsf{TV}^1 \vdash SAT(F, X)$*.*

Theorem A.16 establishes what is called a "back-and-forth" relationship between bounded arithmetic theories and propositional proof systems. On the one hand, if V^1 proves a $\forall\Sigma_1^B$-sentence $\forall X \exists Y < t.\, \varphi(X, Y)$, then by known propositional translations (Theorem A.12) , $EF \vdash \|\forall X \exists Y < t.\, \varphi(X, Y)\|_n$. Moreover, this translation is *provable* in V^1.

One the other hand, by Theorem A.16, if Extended Frege proves formula F, then V^1 proves F is a tautology. These two directions establish that we may go back-and-forth between the first-order setting and propositional proof systems for any pair (T, P) with the above relationships. In particular, we will need this between TV^1 and G_1, as well as $\mathsf{V}^1(\mathsf{VPV})$ and Extended Frege. For a complete picture of this relationship, see [Chapter X, [CN10]].