Paper:
Beckmann, A. & Buss, S. The NP Search Problems of Frege and Extended Frege Proofs. *ACM Transactions on Computational Logic*

The NP Search Problems of Frege and Extended Frege Proofs

ARNOLD BECKMANN, Swansea University
SAM BUSS, University of California, San Diego

We study consistency search problems for Frege and extended Frege proofs, namely the NP search problems of finding syntactic errors in Frege and extended Frege proofs of contradictions. The input is a polynomial time function, or an oracle, describing a proof of a contradiction; the output is the location of a syntactic error in the proof. The consistency search problems for Frege and extended Frege systems are shown to be many-one complete for the provably total NP search problems of the second order bounded arithmetic theories U^1_2 and V^1_2, respectively.

CCS Concepts: •**Theory of computation** → **Complexity classes; Proof complexity; Proof theory;** •**Computing methodologies** → *Theorem proving algorithms;*

Additional Key Words and Phrases: bounded arithmetic, Frege proofs, extended Frege proofs, NP search problems, propositional logic, total functions, proof complexity

ACM Reference Format:
Arnold Beckmann and Sam Buss. 2017. The NP Search Problems of Frege and Extended Frege Proofs *ACM Trans. Comput. Logic* V, N, Article A (January YYYY), 18 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

This paper studies the consistency search problems for Frege and extended Frege proofs, namely the total NP search problems of finding syntactic errors in Frege and extended Frege proofs of contradictions. A total NP search problem is a total, multi-valued, polynomial growth rate function with polynomial time recognizable graph. The class TFNP of total NP search problems has been extensively studied from the point of view of computational complexity (arising from the work of [Johnson et al. 1988; Papadimitriou 1990; 1994]), and more recently from the point of view of bounded arithmetic. For bounded arithmetic, TFNP problems correspond to Σ^{b}_1-definable functions; more precisely, the Σ^{b}_1-definable functions of a bounded arithmetic theory T are exactly the functions which can be obtained as projections, which are polynomial time computable, of total NP search problems of T. The second author's Ph.D. thesis [Buss 1986] studied the Σ^{b}_1-definable functions of S^1_2; this also characterized the total NP search functions of S^1_2. However, [Buss 1986] studied only the Σ^{b}_i-definable functions of S^i_2 and T^{i-1}_2 for $i > 1$ and only the $\Sigma^{1,\mathrm{b}}_1$-definable functions of the second-order theories U^1_2 and V^1_2. The Σ^{b}_1-definable functions of T^1_2 and S^2_2 were characterized in terms of the TFNP class Polynomial Local Search (PLS) by Buss and Krajíček [Buss and Krajíček 1994]. A number of recent papers [Krajíček et al. 2007; Pudlák and Thapen 2012; Skelley and Thapen 2011] have characterized the provably total NP search problems of the first-order theories T^{i-1}_2 and S^i_2 for $i > 2$. Even more recently, Kołodziejczyk-Nguyen-Thapen [Kołodziejczyk et al. 2011] and Beckmann-Buss [Beckmann and Buss 2014] have

Author's addresses: A. Beckmann, Department of Computer Science, Swansea University, Swansea SA2 8PP, UK; S. Buss, Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112, USA. S. Buss was supported in part by NSF grants CCF-121351 and DMS-1101228, and a Simons Foundation Fellowship 306202.

DOI: http://dx.doi.org/10.1145/0000000.0000000

given characterizations of the provably total NP search problems of the second-order theories U^1_2 and V^1_2 in terms of a variety of local improvement principles. Specifically, the results of [Kołodziejczyk et al. 2011; Beckmann and Buss 2014] together show that the following TFNP problems are many-one complete for V^1_2: the local improvement principles LI and LI_1 and the rectangular local improvement principles RLI and $\mathrm{RLI}_{\log n}$. For U^1_2, they show the following are many-one complete: the rectangular local improvement principle RLI_1 and the linear local improvement principles LLI and $\mathrm{LLI}_{\log n}$.

The present paper gives new characterizations of the TFNP problems which are provably total in U^1_2 and V^1_2. The new characterizations are based on the consistency search problems for Frege and extended Frege proofs; that is to say, based on the complexity of searching for syntactic errors in Frege or extended Frege proofs of contradictions. A closely related result for V^1_2 was obtained earlier by Krajíček [Krajíček 2016]; other related results for the theories T^i_2 were proved before that by Krajíček, Skelley, and Thapen [Krajíček et al. 2007] and Skelley and Thapen [Skelley and Thapen 2011].

A Frege proof is a“textbook style” proof system for propositional logic; typically a Frege proof system has modus ponens as the only rule of inference. Extended Frege proofs are allowed to use an additional “extension rule” that allows the introduction of new variables abbreviating more complex formulas. We briefly define (extended) Frege proofs here, but for more information see e.g. [Cook and Reckhow 1979; Buss 1998; 1999; Krajíček 1995]. For the purposes of the present paper, we use the propositional connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$. Frege proofs have a finite set of axiom schemes, for instance $A \rightarrow (B \rightarrow A)$ where A and B may be any formula; our Frege proofs have modus ponens as the only rule of inference: namely, from A and $A \rightarrow B$, infer B (again, for any formulas A and B). More generally, Frege systems can use any finite complete set of propositional connectives, and any finite implicationally sound and implicationally complete set of axiom schemes and inference schemes. All such Frege systems are p-equivalent (polynomially equivalent) [Cook and Reckhow 1979; Reckhow 1976]. These p-equivalences involve non-local constructions that do not immediately apply in our setting; nonetheless, our constructions below are general enough so that the results of the present paper apply to any Frege system.

We will use symbols $\top$ and $\bot$ as abbreviations for the true formula $(x_1 \vee (\neg x_1))$ and the false formula $(x_1 \wedge (\neg x_1))$, respectively. With no loss of generality, we presume that $\top$ is an axiom for our Frege system.

An extended Frege proof system is the Frege proof system augmented with the *extension rule*:

$$x \leftrightarrow \varphi,$$

where φ is a formula, and x is a new variable which does not appear earlier in the proof, in φ, or in the last formula of the proof. This effectively allows the variable x to abbreviate φ. The extension rule can be applied iteratively, and this conjecturally means that extended Frege proofs can be exponentially shorter than Frege proofs, where proof size is measured in terms of the number of symbols in a proof. (This is an open problem, however.)

We shall define TFNP problems $\mathcal{F}\textsc{Con}(\Omega, 0^n)$ and $e\mathcal{F}\textsc{Con}(\Omega, 0^n)$ which find syntactic errors in (purported) Frege and extended Frege proofs of contradictions. These are called the *consistency search problems* for Frege and extended Frege. The input 0^n serves only as a size parameter; in the following, “polynomial time/size” or “exponential time/size” will always be relative to n, and mean either $n^{O(1)}$ or $2^{n^{O(1)}}$, respectively. The second-order input Ω codes a string of symbols $\Omega(0), \ldots, \Omega(2^n{-}1)$ which are supposed to encode a valid Frege (resp., extended Frege) proof P_Ω of a contradiction. The search problem must output a place where P_Ω fails to be a valid proof of a contradiction. In other words, $\mathcal{F}\textsc{Con}(\Omega, 0^n)$ or $e\mathcal{F}\textsc{Con}(\Omega, 0^n)$ must output some set of positions in the proof P_Ω so that it can be verified in polynomial time that these positions in Ω reveal a syntactic mistake showing that P_Ω

is not a valid proof of a contradiction. Section 2.2 gives more details on how $\mathcal{F}\text{Con}$ and $e\mathcal{F}\text{Con}$ are defined and on how Ω encodes a proof P_Ω.

The size parameter 0^n means that Ω encodes a proof of exponential size, with 2^n many symbols. In this way, when a theory U^1_2 or V^1_2 gives a Σ^{b}_1-definition of $\mathcal{F}\text{Con}(\Omega, 0^n)$ or $e\mathcal{F}\text{Con}(\Omega, 0^n)$, it is proving the consistency of exponentially long Frege or extended Frege proofs (respectively). It is important to note that P_Ω may have exponentially many steps and may contain exponentially long formulas. The ability to have exponentially long formulas is not important for $e\mathcal{F}\text{Con}$, as the extension rule means that extended Frege proofs may be assumed to have only short formulas, even only formulas with constantly many symbols. For $\mathcal{F}\text{Con}$ however, the possibility that formulas can contain exponentially many symbols is crucial. Indeed, T^1_2 can Σ^{b}_1-define, and prove the totality of, the consistency search problem for Frege proofs in which formulas may contain only polynomially many symbols. This is because T^1_2 can define the truth of polynomial size propositional formulas, and prove by induction that every formula in such a Frege proof is true (say, under the assignment mapping all variables to *False*).

It is interesting to note that the extension rule may be iterated exponentially many times in an extended Frege proof P_Ω. This allows extension variables to represent exactly values which can be computed with exponential size Boolean circuits. Thus, an exponentially long extended Frege proof is, in effect, able to reason about exponential size circuits (cf. Jeřábek [Jeřábek 2004]). In fact, U^1_2 can Σ^{b}_1-define the consistency search problem for extended Frege proofs that are restricted to have only polynomial depth nesting of the extension rule.[1]

The next two theorems are the main results of the paper. For the definition of "many-one complete provably in S^1_2", see Sections 2.1 and 2.3.

THEOREM 1.1. *The Extended Frege Consistency search problem, $e\mathcal{F}\text{Con}$, is many-one complete (provably in S^1_2) for the provably total NP search problems of V^1_2.*

THEOREM 1.2. *The Frege Consistency search problem, $\mathcal{F}\text{Con}$, is many-one complete (provably in S^1_2) for the provably total NP search problems of U^1_2.*

These two theorems apply to the oracle, or "relativized", versions of $e\mathcal{F}\text{Con}$ and $\mathcal{F}\text{Con}$, and the second order, or "relativized", versions of S^1_2 as defined in Section 2.3. As an immediate corollary, they also apply to the usual, un-relativized, versions of S^1_2, and instances of $e\mathcal{F}\text{Con}$ and $\mathcal{F}\text{Con}$ w.r.t. polynomial time relations.

We mention in passing that the standard TFNP classes PPP, PPA, PPAD, PPADS, etc. are all many-one reducible to $\mathcal{F}\text{Con}$: As PSPACE functions can count sizes of sets, complete multi-functions in the classes such as PPP, PPA, PPAD, and PPADS are all provably total in the theory U^1_2. Hence by Theorem 1.2, they are many-one reducible to $\mathcal{F}\text{Con}$. For related results see Goldberg and Papadimitriou [Goldberg and Papadimitriou 2016].

There has already been extensive work relating bounded arithmetic theories to propositional proof complexity, including the Paris-Wilkie translation [Paris and Wilkie 1985], Cook's characterization of PV [Cook 1975], and many subsequent papers. Skelley and Thapen [Skelley and Thapen 2011] characterize the TFNP problems of the theories T^{i+2}_2 in terms of 1-reflection for depth i Frege systems (where depth zero is resolution). They do

[1] To prove this, one can use the fact that there is a polynomial space (PSPACE) algorithm which evaluates the truth of propositional formulas in which the extension rule is nested only to polynomial depth. One way to form this PSPACE algorithm is to non-deterministically perform a depth-first traversal of the Boolean formula, expanding extension rules as needed, traversing always smaller subformulas first. For traversing propositional formulas in order of smaller subformulas first, see [Buss 1987]; for formalizing nondeterministic PSPACE algorithms and Savitch's theorem in U^1_2, see [Beckmann and Buss 2014].

This construction is a uniform analogue of the fact that polynomial size extended Frege proofs in which the extension rule is only nested logarithmically can be quasi-polynomially simulated by Frege proofs.

not explicitly discuss consistency search problems, but it is not hard to recast their results in terms of consistency search problems for bounded depth Frege systems [Thapen, personal communication]. The most similar prior work to the present paper is that a version of Theorem 1.1 was already established by Krajíček [Krajíček 2016], using, for the proof, the notion of implicit proofs [Krajíček 2004]. One advantage of Theorem 1.1 is that it is stated directly in terms of propositional consistency, which we feel makes for a more direct and intuitive statement.

The proofs of Theorems 1.1 and 1.2 are similar. Perhaps the principal difference is that the latter theorem requires a refined method of encoding Frege proofs so that a syntactic error in an exponential size Frege proof may be described with only a polynomial amount of information. For this, see Section 2.2.

It is an open question whether the $e\mathcal{F}\textsc{Con}$ search problem is many-one reducible to the $\mathcal{F}\textsc{Con}$ search problem. If it is provably so in U^1_2, then there would be some surprising consequences. First, it would follow that V^1_2 is $\forall\Sigma^{\mathrm{b}}_1$-conservative over U^1_2, so U^1_2 and V^1_2 would have the same provably total NP search problems. Second, it would give a quasipolynomial simulation of extended Frege proofs by Frege proofs. (This second implication has not been proved in the literature, but one way to prove it is as follows. Conservativity over V^1_2 would mean that U^1_2 can prove the consistency of extended Frege proofs coded by second-order predicates. It would follow that propositional translations of this statement have quasipolynomial size Frege proofs [Krajíček 1995, Theorem 9.1.6]. By techniques of Cook [Cook 1975], this would further imply that Frege systems quasipolynomially simulate extended Frege systems.)

It is more likely, however, that the $e\mathcal{F}\textsc{Con}$ search problem is not many-one reducible to the $\mathcal{F}\textsc{Con}$ search problem, and consequently that V^1_2 is neither equal to, nor conservative over, U^1_2.

2. PRELIMINARIES

2.1. Total NP search problems

A TFNP problem, or total NP search problem [Papadimitriou 1994], is a total, polynomial growth rate, multivalued function defined by a polynomial time relation (relative to an oracle). Typical classes of TFNP problems such as PPA, PPAD, PPP, etc. are based on combinatorial properties such as the parity principle or the pigeonhole principle. For these classes, the input to the TFNP problem is a description of an exponentially large combinatorial object (e.g., a low-degree graph); the output is a witness to the combinatorial principle (e.g., a node of degree one, or a node of indegree two, etc.) The TFNP problem takes a polynomial size parameter as an input in addition to the exponentially large combinatorial object. In the initial definition of TFNP problems [Papadimitriou 1994], the exponentially large combinatorial object was defined in terms of a polynomial size circuit given as part of the input. Beame et al. [Beame et al. 1998] suggested instead using an oracle to encode the combinatorial object. The advantage of using an oracle is that it makes the definition of the TFNP classes more uniform, and especially that it allowed [Beame et al. 1998] to prove oracle separation results between classes such as PPA, PPAD, PPADS, and PPP. In this paper, we work with the relativized versions of TFNP problems where an oracle is used to specify the combinatorial object. Since we prove only reductions, not separations, this only makes our results more general.

The formalization of TFNP problems in bounded arithmetic uses a second-order predicate as the oracle encoding the combinatorial object. Formally, let T be a theory of bounded arithmetic, and let a TFNP problem be given by a polynomial time predicate A which may involve an oracle X, and a term t which bounds the size of the function values. The TFNP problem is *provably total* in a theory T provided T proves:

$$(\forall x)\,(\exists y{\le}t(x))\,A(x,y,X).$$

We write $f_A(x, X) = y$ when $y \leq t(x)$ and $A(x, y, X)$ holds. Observe that f_A is a multi-function, thus the y need not be unique. The second-order predicate X is part of the input, but algorithmically serves as an oracle.

Let A' and t' define a second TFNP problem $f_{A'}$. A *many-one reduction* of f_A to $f_{A'}$, denoted $f_A \preccurlyeq_m f_{A'}$, is a pair of polynomial time computable functions $g(x, X)$ and $h(x, y', X)$ and a polynomial time computable relation $z(u, x, X)$ such that

$$\begin{aligned}(\forall x)(\forall y')\,[y' {\leq} t'(g(x, X)) \wedge A'(g(x, X), y', Z)&\\ \rightarrow h(x, y', X) {\leq} t(x) \wedge A(x, h(x, y', X), X)],&\end{aligned} \tag{1}$$

where Z represents the predicate with value $Z(u)$ defined to equal $z(u, x, X)$. In other words, given inputs x and X to the TFNP problem f_A, letting Z be the predicate defined by z, and letting x' equal $g(x, X)$, we have that if y' is a solution to $f_{A'}(x', Z)$ then $h(x, y', X)$ is a solution to $f_A(x, X)$.

When S^1_2 proves (1), then we say the many-one reduction is provable in S^1_2. For this, we use the conservative extension of S^1_2 with the second-order predicate symbol X added to the language. As usual, the predicate X may be used in induction axioms, but second-order quantifiers are not permitted in induction axioms.

2.2. Encoding Frege and extended Frege proofs

We now discuss how to encode exponentially long Frege or extended Frege proofs using an oracle X. For simplicity, we now allow X to be a polynomial growth rate function oracle instead of a predicate: that is, $X(x, a)$ will be an integer instead of a true/false value. In bounded arithmetic theories, X is used as a new function symbol: for each first-order x, a, $X(x, a)$ is also a first-order value. As a polynomial growth rate function, $X(x, a)$ is by hypothesis always $\leq s(x, a)$ for some fixed first order term s. Treating X as a function instead of a predicate can be done without loss of generality, since if we wished to use only predicates, any atomic formula containing terms involving X could be replaced by a formula using the graph of X, and the graph of X could be expressed by a simple (sharply bounded) formula using the predicate giving the bit-graph of X — such a replacement does not increase the quantifier complexity of bounded formulas significantly.

When writing $X(x, a)$, x is intended to denote the size of an (extended) Frege proof, and a a position $\leq x$ within this proof. Thus x serves only as a size parameter, and for convenience, we will suppress mentioning x in the following, and write $X(a)$ instead of $X(x, a)$. The value $X(a)$ encodes the a-th symbol in a Frege or extended Frege proof, plus auxiliary information about the structure of the proof. The auxiliary information will enable syntactic errors to be identified with a polynomial amount of information, even though the proof may contain exponentially long formulas. We assume formulas are fully parenthesized, so that the valid formulas have the following forms: x for x a variable; $(\neg\varphi)$; or $(\varphi \circ \psi)$ where $\circ$ is $\wedge$, $\vee$, $\rightarrow$ or $\leftrightarrow$. Formulas in the proof coded by X are separated by commas; in fact every formula is preceded by a comma, including the first formula. We also allow "null" formulas with zero symbols. Null formulas are effectively viewed as the constant "True", and are represented by two adjacent commas in the proof coded by X. Inserting commas (null formulas) allows us to pad out proofs with blanks; this lets us construct uniform proofs with the property that there is a polynomial time function $f(i)$ that computes the i-th symbol of the proof.

To encode a Frege or extended Frege proof with X, write out the proof as a sequence of symbols $\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_N$. We let $X(0) = N$ encode the length of the proof. Every other value of $X(a)$ will be a Gödel number of a finite sequence. Any standard Gödel numbering scheme which is formalizable in S^1_2 may be used: the notation $\langle a_1, \ldots, a_k\rangle$ denotes the Gödel number of the finite sequence $a_1, \ldots, a_k$. For $a > 0$, $X(a)$ will have the form $\langle p, \ldots\rangle$, where p is an integer representing one of the finitely many axiom schemes or rules of inference, or representing one of the symbols "variable", or "left parenthesis", "right parenthesis", or one of the logical connectives. The latter values for p are written as p_x, $p_($, $p_)$, $p_\neg$, $p_\wedge$, $p_\vee$, $p_\to$, and $p_\leftrightarrow$. The former values for p are written $p_{(k)}$ denoting the k-th schematic axiom or rule of inference. For $a = 1, \ldots, N$, we set

(a) If σ_a is the comma preceding a formula φ, then $X(a) = \langle p_{(k)}, a_1, a_2\rangle$ where the k-th axiom or rule of inference is used to infer φ, and σ_{a_1} and σ_{a_2} are the commas preceding the (up to two) formulas φ_1 and φ_2 used to infer φ (e.g., by modus ponens).
(b) If σ_a is a parenthesis symbol "(" or ")", then $X(a) = \langle p, a'\rangle$ with p either "$p_($" or "$p_)$", and a' the index of the matching parenthesis. I.e., $\sigma_{a'}$ is the matching parenthesis.
(c) If σ_a is a variable x_i, then $X(a) = \langle p_x, i\rangle$; here p_x is the value for denoting variables.
(d) If σ_a is a logical symbol, $\neg$, $\wedge$, $\vee$, $\to$ or $\leftrightarrow$, then $X(a) = \langle p\rangle$ where p is corresponding value $p_\neg$, $p_\wedge$, $p_\vee$, $p_\to$ or $p_\leftrightarrow$.

The only unusual aspect of the just-described encodings of proofs is the use in (b) of a', which serves as a pointer to the matching parenthesis. This is included so that syntactic errors in the proof encoded by the function X can be witnessed by finitely many values of $X(i)$: that is, whenever X fails to correctly encode the computation, there is a constant size set of pairs $\langle i, X(i)\rangle$ which serve to witness an error in the encoded computation. To illustrate this, we list some representative ways in which X may fail to encode a valid proof.

(i) There may be two adjacent symbols $X(a)$ and $X(a{+}1)$ which contain symbols that cannot appear consecutively in a valid proof or formula. E.g., $X(a)$ and $X(a{+}1)$ might be two consecutive variables, or two adjacent propositional connectives, or a left parenthesis and a right parenthesis, or a propositional connective followed by a right parenthesis, etc. In all these cases (and others), the syntactic error may be specified by a and the values of $X(a)$ and $X(a{+}1)$.
(ii) A right (respectively, left) parenthesis may not have its pointer indicating a matching left (respectively, right) parenthesis. For some examples, suppose $X(a)$ is $\langle p_(, a'\rangle$. Then we may have $a' \le a{+}2$, or we may have $X(b)$ indicating the b-th symbol is a comma for $a < b \le a'$, or we may have $X(a')$ not encoding a right parenthesis that matches back to the left parenthesis at $X(a)$. Any of these conditions can be witnessed by giving values for a and possibly either b or a', and by giving the corresponding $X(\cdot)$ values. Other conditions can be witnessed similarly.
(iii) Two pairs of parentheses might not be properly nested. For instance, $X(a_1) = \langle p_(, a_1'\rangle$ and $X(a_2) = \langle p_(, a_2'\rangle$ can hold information about matching parentheses, but the positions a_1, a_1' and a_2, a_2' may not indicate positions compatible with properly nested parentheses. This failure can be witnessed by giving the values of a_1, a_2, $X(a_1)$ and $X(a_2)$.
(vi) Finally, a schematic axiom or inference rule may be misimplemented. For one example, suppose a syntactically incorrect instance of modus ponens has the form

$$\frac{A \qquad (A \to B)}{B'}$$

with B not equal to B'. This syntactic error can be witnessed with the following finite amount of information: (a) The positions a and b, the value $X(a)$ giving the comma before the hypothesis $(A \to B)$, and the value $X(b)$ for the comma preceding the conclusion B'. (b) The value of $X(a{+}2)$ giving the first symbol of A as either a propositional variable or an open parenthesis. In the former case, $X(c)$ is the first symbol of the subformula B for $c = a{+}4$; in the latter case, $X(a{+}2) = \langle p_(, a'\rangle$ and the first symbol $X(c)$ of B is at $c = a'{+}2$. (c) The value of $X(b{+}1)$, from which we obtain the length of B'. (d) If B and B' are well-formed distinct formulas, then for some s less than the length of B', the s-th symbols of B and B' are distinct. This is witnessed by specifying s and giving the values of $X(b{+}1{+}s)$ and $X(c{+}s)$.

All other cases of how $X(\cdot)$ can fail to encode a valid proof are similar; as in the above examples, they can all be witnessed by values of $X(a)$ for constantly many a's. Since $X(\cdots)$ has polynomial growth rate and each value $X(a)$ is encoded by polynomially many bits, there is a simple straightforward polynomial time algorithm to verify that such a set of witness values correctly identifies a syntactic error in the proof coded by X.

Definition 2.1. The TFNP search problem $\mathcal{F}\textsc{Con}$ (respectively, $e\mathcal{F}\textsc{Con}$) is a multi-function $f(x, X)$ which either (a) letting $N = X(0)$, returns the value 0 if $N \leq 8$ or $N > x$, or $X(N{-}8)$ is not the position of a comma followed by the formula $(x_1 \wedge (\neg x_1))$, or (b) returns a constant number of values $a_1, \ldots, a_k \leq x$ such that the pairs $(a_i, X(a_i))$ witness that X does not encode a valid Frege (respectively, extended Frege) proof.

2.3. Bounded arithmetic theories S^1_2, U^1_2, and V^1_2

We presume familiarity with the essentials of the theories S^1_2, U^1_2 and V^1_2 of bounded arithmetic of [Buss 1986]; but we give a quick review to establish notation. For more background see [Buss 1986; Krajíček 1995]; we also draw heavily from the conventions of [Beckmann and Buss 2014] for results about U^1_2 and V^1_2. Our theories all use the non-logical language $0, S, +, \cdot, |\cdot|, \lfloor\cdot/2\rfloor, \#, \leq, \mathrm{MSP}$. The inclusion of the most significant part function $\mathrm{MSP}(x,i) = \lfloor x/2^i \rfloor$ is a little non-standard but makes no essential difference to the power of the theories of bounded arithmetic. The advantage is that with MSP it is easier to formalize concepts such as sequence coding with sharply bounded formulas: for this see Jeřábek [Jeřábek 2006].

We will work exclusively with two-sorted theories which use both first- and second order variables; this includes the theories S^1_2 and T^1_2 which are traditionally first-order theories.[2] Second order variables range over *predicates*, that is, sets of first-order objects; they are denoted variously with capital Roman letters $X, Y, \ldots$. A *bounded quantifier* is a first order quantifier of the form $(\exists x{\leq}t)$ or $(\forall x{\leq}t)$. If the term t is of the form $|s|$, the quantifier is *sharply bounded*. Second order quantifiers have the form $(\forall X)$ or $(\exists X)$. The classes Σ^{b}_i and Π^{b}_i are defined by counting alternations of bounded quantifiers, ignoring sharply bounded quantifiers. Second order quantifiers are not allowed in Σ^{b}_i or Π^{b}_i formulas. A formula which contains only bounded (first order) quantifiers is called a *bounded formula*.

Formulas with second order quantifiers, but no unbounded first order quantifiers are classified with the classes $\Sigma^{1,\mathrm{b}}_i$ and $\Pi^{1,\mathrm{b}}_i$ by counting the alternations of second order quantifiers, ignoring any first order quantifiers. The class $\Sigma^{1,\mathrm{b}}_0$ is the set of bounded formulas.

The theories S^i_2 are axiomatized with a finite set BASIC of open axioms defining the non-logical symbols, plus polynomial induction Σ^{b}_i-PIND, or equivalently length induction Σ^{b}_i-LIND. The theories T^i_2 are axiomatized with the axioms of BASIC plus Σ^{b}_i-IND, namely the usual induction axioms. The theory T_2 is the union of the theories T^i_2 for $i \geq 0$. Like all our theories, S^i_2 and T^i_2 are formulated in a second-order language; for these theories, the induction formulas may contain second order variables but not second order quantifiers.

The axioms for both U^1_2 and V^1_2 include the axioms of T_2 and the $\Sigma^{1,\mathrm{b}}_0$-comprehension axioms,

$$(\forall \vec{x})(\forall \vec{X})(\exists Z)(\forall y{\leq}t(\vec{x}))[y \in Z \leftrightarrow \varphi(y, \vec{x}, \vec{X})] \tag{2}$$

for every bounded formula φ and every term t. This axiom states that any set (on a bounded domain) defined by a bounded formula φ with parameters is coded by some second order

[2]Sometimes the notations S^{1+}_2 and T^{1+}_2, or $\mathrm{S}^1_2(X)$ and $\mathrm{T}^1_2(X)$, are used to denote these conservative extensions of S^1_2 and T^1_2 to two-sorted (second order) theories; however, we prefer to use the simpler notations S^1_2 and T^1_2 (following [Beckmann and Buss 2014]). Similarly, we eschew notations such as $\Sigma^{\mathrm{b}+}_i$ or $\Sigma^{\mathrm{b}}_i(X)$, and use simply Σ^{b}_i.

object Z.[3] The theory U^1_2 has in addition the $\Sigma^{1,\mathrm{b}}_1$-PIND axioms. The theory V^1_2 has instead the $\Sigma^{1,\mathrm{b}}_1$-IND axioms. It is known that $\mathrm{V}^1_2 \vdash \mathrm{U}^1_2$, and that U^1_2 proves the $\Delta^{1,\mathrm{b}}_1$-IND axioms, namely induction for formulas which are U^1_2-provably equivalent to both a $\Sigma^{1,\mathrm{b}}_1$-formula and a $\Pi^{1,\mathrm{b}}_1$-formula, see [Buss 1986].

The theory S^1_2 has proof-theoretic strength which corresponds to polynomial time computability [Buss 1986], and this is true also relative to second order predicates used as oracles. Let a function f be polynomial time computable relative to an oracle X. We write $f^X(\vec{y})$ to denote the function f with first order inputs $\vec{y}$ and second order input X. As a polynomial time computable function, f is computed by an oracle Turing machine M_f with a polynomial runtime $p(\vec{n})$. The first order inputs $y_1, \ldots, y_k$ are given to M_f on its input tape as strings in $\{0,1\}^*$ encoding the integers y_i in binary. The second order input X is given to M_f as an oracle. For all $\vec{y}$ and X, the Turing machine $M^X_f(\vec{y})$ computes $f^X(\vec{y})$ within time $p(|\vec{y}|)$.

The theory S^1_2 can Σ^{b}_1-*define* any polynomial time function f. This means that S^1_2 can prove the existence of computations of $M^X_f(\vec{y})$ for all $\vec{y}$ and X, and can express the property that $z = f^X(\vec{y})$ with a Σ^{b}_1-formula $\varphi(\vec{y}, z, X)$. The converse holds as well and is the relativized version of the "main theorem" for S^1_2:

THEOREM 2.2. ([Buss 1986]) *Every Σ^{b}_1-definable function of S^1_2 is, provably in S^1_2, a polynomial time function. Indeed, if φ is a Σ^{b}_1-formula and if S^1_2 proves $(\forall X)(\forall \vec{y})(\exists z)\varphi(\vec{y}, z, X)$, then there is a polynomial time function $f^X(\vec{y})$ so that S^1_2 proves $(\forall X)(\forall \vec{y})\varphi(\vec{y}, f^X(\vec{y}), X)$.*

The corresponding witnessing theorems for U^1_2 and V^1_2 state that the $\Sigma^{1,\mathrm{b}}_1$-definable functions of U^1_2, respectively V^1_2, are precisely the polynomial growth rate functions which are computable in polynomial space, respectively in exponential time [Buss 1986]. These witnessing theorems also hold for oracle computability, but both polynomial space and exponential time oracle computations are constrained to make only polynomial size oracle queries. For the present paper, we need the stronger "new-style" form of the witnessing theorems for U^1_2 and V^1_2 as established by Beckmann and Buss [Beckmann and Buss 2014].

THEOREM 2.3. (Theorem 4.5.a of [Beckmann and Buss 2014]) *Suppose U^1_2 proves $(\exists y)\varphi(y, \vec{a}, \vec{A})$ for φ a $\Sigma^{1,\mathrm{b}}_0$-formula. Then there is a polynomial space oracle Turing machine M such that S^1_2 proves "If Y encodes a complete computation of $M^{\vec{A}}(\vec{a})$, then $\varphi(\mathrm{out}(Y), \vec{a}, \vec{A})$ is true", where $\mathrm{out}(Y)$ denotes the output of the computation coded by Y.*

THEOREM 2.4. (Theorem 4.7.a of [Beckmann and Buss 2014]) *Suppose V^1_2 proves $(\exists y)\varphi(y, \vec{a}, \vec{A})$ for φ a $\Sigma^{1,\mathrm{b}}_0$-formula. Then there is an exponential time oracle Turing machine M such that S^1_2 proves "If Y encodes a complete computation of $M^{\vec{A}}(\vec{a})$, then $\varphi(\mathrm{out}(Y), \vec{a}, \vec{A})$ is true".*

Section 2.4 will sketch a definition of what it means for the second order object Y to encode a complete computation of $M^{\vec{A}}(\vec{a})$. The only unexpected part of the definition is that the computation will be stretched out so that answers to oracle queries are obtained only at specific times. In fact, an oracle query "$A(c)$?" can be answered only at specific times which depend on c.

[3] The original definition [Buss 1986] of U^1_2 used an unbounded version of the comprehension axiom. We are using here the bounded version of the comprehension axiom, as has been preferred by most subsequent authors, e.g., [Beckmann and Buss 2014; Cook and Nguyen 2010; Krajíček 1995; 2011]. It makes no essential difference to the strengths of the theories.

Theorems 2.3 and 2.4 are weaker than the results proved by [Beckmann and Buss 2014] since they only assert that $\varphi(\mathrm{out}(Y), \vec{a}, \vec{A})$ is true and say nothing about canonical verifications. But, Theorems 2.3 and 2.4 suffice for our purposes, since we apply them only to Σ_0^{b}-formulas $\varphi(y, \vec{a}, \vec{A})$, and since it is a polynomial time operation to check whether $\varphi(\mathrm{out}(Y), \vec{a}, \vec{A})$ is true.

The next theorem establishes the easy direction of Theorems 1.1 and 1.2.

THEOREM 2.5.

(a) U_2^1 *proves that the consistency search problem* $\mathcal{F}\mathrm{CON}$ *satisfying the properties of Definition 2.1 is total.*
(b) V_2^1 *proves that the consistency search problem* $e\mathcal{F}\mathrm{CON}$ *satisfying the properties of Definition 2.1 is total.*

The proof is a straightforward application of the facts that U_2^1 and V_2^1 can define and reason about polynomial space and exponential time functions. Part (b), about V_2^1, is the analogue of the fact that S_2^1 can prove the consistency of extended Frege proofs encoded by first order numbers [Cook 1975]. The proof for V_2^1 is based on the fact that there is an exponential time algorithm which, given an exponential size circuit (coded by an oracle) with 0/1 inputs, determines the truth value of each gate in the circuit; this is analogous to the fact that the Boolean circuit value problem is in polynomial time. An extended Frege proof has input variables and extension variables. If the input variables are assigned fixed true/false values, then the extension rules define exponential size circuits giving the induced truth values of the extension variables. Therefore, there is an exponential time function which, given truth values for the input variables and given oracle access to X, computes the truth values of all formulas in the exponential size proof coded by X. V_2^1 can $\Sigma_1^{1,\mathrm{b}}$-define this function and prove its properties; with this, V_2^1 can use induction to show that every line in the proof evaluates to the value true if the input variables are set to, say, false.

Part (a), about U_2^1, uses instead the fact that there is a (poly)logarithmic space algorithm for evaluating Boolean formulas (in fact, there is even an NC^1 algorithm [Buss 1987; Buss et al. 1992; Buss 1993]). Analogously, there is a polynomial space algorithm which, given an exponential size Boolean sentence encoded by an oracle computes the value of the sentence. U_2^1 can $\Sigma_1^{1,\mathrm{b}}$-define this function and prove its properties; it can then use induction to show that every line in the proof evaluates to the value true if the input variables are set to false.

Part (a) is a weak analogue of the fact that the bounded arithmetic theories AID [Arai 2000] and VNC^1 [Cook and Morioka 2005; Cook and Nguyen 2010] can prove the reflection principle for Frege proofs. (The results for AID and VNC^1 are stronger, as they use an NC^1 algorithm for Boolean formula evaluation instead only a logarithmic space algorithm.)

2.4. Encoding polynomial space and exponential time computation with oracles

We now discuss at a high level the details of how an oracle Y encodes the computation of an exponential time oracle Turing machine, as needed for Theorems 2.3 and 2.4. For simplicity, suppose M has one first-order (ordinary) input a and one second-order input A. M is a single tape Turing machine. Initially, the tape contains the bits of a and is otherwise blank. When not in a query state, M uses its transition function to update the current tape symbol and possibly move the tape head one position left or right. For oracle queries, the polynomial length query w is by convention written on the tape starting one cell to the right of the tape head; after a query state, the query answer $A(w)$ is written under the tape head. A configuration of $M^A(a)$'s computation consists of the string of symbols on the work tape, with a symbol denoting the current state marking the tape head position. The tape symbols are drawn from a finite alphabet, and then encoded as fixed length bit strings;

configurations are padded with blanks so that they are fixed length bit strings. The values $Y(x)$ give the bits encoding an exponentially long description of the complete computation of $M^A(a)$. We assume the query value x encodes a triple $x = \langle t, s, c\rangle$; here t is the time, and $Y(\langle t, s, c\rangle)$ gives the s-th bit of the t-th configuration of $M^A(a)$.

The c is a counter which controls which value of A can be accessed by an oracle query. We design Y's encoding of the computation of M so that the correctness of any single bit $Y(x)$, where $x = \langle t, s, c\rangle$, depends on only polynomially many earlier bits $Y(x')$ and on a single value $A(w)$ where $w = w(c)$ is determined by c. The purpose of the counter c is to ensure that an oracle query (if made) is made only to a specific bit of A. In this way, we avoid having the correctness of $Y(x)$ depend on exponentially many bits of A. For each value of t, the counter c cycles through all of its possible values, so that $w(c)$ cycles through all possible queries to A. This is explained in more detail next.

Let the input a have length $n = |a|$. There is a polynomial $p(n)$ such that $M^A(a)$ runs for time $2^{p(n)}$ and only queries $A(w)$ for $|w| < p(n)$ — the polynomial bound on queries reflects the model of oracle exponential time computation from [Buss 1986]. W.l.o.g., the position values s are bounded by $2^{p(n)}$ as well. (In fact, when M is polynomial space, then s values are bounded by just $p(n)$.) We let c range over strings of length exactly $p(n)$; if $c = 0^i1w$, this means the value of $A(w)$ is accessible as an answer for an oracle query. For $c \neq 0^{p(n)}$, we write $c{-}1$ for the predecessor to c, as obtained by binary subtraction.

With these conventions, we can arrange that, in order for Y to correctly encode the computation of $M^A(a)$, the value $Y(\langle t, s, c\rangle)$ is determined by (for some constant C):

a. The value $A(w)$ if $c = 0^i1w$, and
b. The bits of the first-order input a (needed only when $t = 0$ and $c = 0^{p(n)}$), and
c. If $c = 0^{p(n)}$ and $t > 0$, the values $Y(\langle t{-}1, s', 1^{p(n)}\rangle)$ where $|s' - s| \leq C \cdot p(n)$, and
d. If $c \neq 0^{p(n)}$, the values $Y(\langle t, s', c{-}1\rangle)$ where $|s' - s| \leq C \cdot p(n)$. In this case, we think of the time t being held fixed while c is incremented through all possible values.

Thus, $Y(\langle t, s, c\rangle)$ is determined by the $O(p(n))$ earlier values as specified by a.-d.

The idea is that t is the ordinary Turing machine time, but the computation is slowed-down by letting c increment through all possible values at each step of the Turing machine. When case c. applies and $c = 0^{p(n)}$, the Turing machine is taking an ordinary step, e.g., using its transition function to update the state, the current tape cell symbol, and the tape head position. When a. and d. apply, and $c = 0^i1w$, the Turing machine may update its current tape cell with the value $A(w)$ provided it is in a query state with w the current query. For $c \neq 0^{p(n)}$, the time t is held fixed, and the Turing machine does not make an ordinary step. The constant C is chosen large enough so that the values $Y(\langle t, s', \cdots\rangle)$ for $|s' - s| \leq C \cdot p(n)$ specify the current state, and the tape contents of the current and adjacent tape squares, including the value of the current query string w (if any).

When Y does *not* correctly encode the computation of $M^A(a)$, then there must be a value $Y(\langle t, s, c\rangle)$, and $O(p(n))$ earlier values of $Y(\langle t', s', c'\rangle)$ as given by c. and d. above, and which witness that Y does not correctly encode the computation. We call such a set of values (for $\langle t, s, c\rangle$, $Y(\langle t, s, c\rangle)$ and the $O(p(n))$ many values $Y(\langle t', s', c'\rangle)$ of the oracle) a *local witness* that Y does not correctly encode the computation of $M^A(a)$. Since a local witness has polynomial size, all the information of the local witness can be coded with a first-order object σ. There is a straightforward polynomial time procedure $C^{A,Y}(a, \sigma)$ for verifying that σ is a local witness.

The extended Frege proofs constructed in Section 3 for the proof of Theorem 1.1 will introduce propositional variables $y_{t,s,c}$ intended to be the values of $Y(\langle t, s, c\rangle)$. In order for these to be correct values, they must satisfy conditions of the forms (3)-(5) below. By construction (in particular, by the use of the counter c controlling queries to A), these conditions will be polynomial size formulas. Let α_w be a propositional variable intended to be the truth value of $A(w)$, and $a_1, \ldots, a_n$ be n propositional variables intended to be the bits of the first-order input a.

The condition for a $c = 0^i 1w \in \{0,1\}^{p(n)}$ has the form

$$y_{t,s,c} \;\leftrightarrow\; \varphi_{t,s,c}(\alpha_w, y_{t,s-Cp(n),c-1}, \ldots, y_{t,s+Cp(n),c-1}) \tag{3}$$

(The y's with subscripts out of range are to be omitted.) For $c = 0^{p(n)}$ and $t > 0$, the condition has the form

$$y_{t,s,0^{p(n)}} \;\leftrightarrow\; \varphi_{t,s,0^{p(n)}}(y_{t-1,s-Cp(n),1^{p(n)}}, \ldots, y_{t-1,s+Cp(n),1^{p(n)}}) \tag{4}$$

For $t = 0$ and $c = 0^{p(n)}$, the form is

$$y_{0,s,0^{p(n)}} \;\leftrightarrow\; \varphi_{0,s,0^{p(n)}}(a_1, \ldots, a_n). \tag{5}$$

In all three types of conditions, the formula $\varphi_{t,s,c}$ is constructible in time polynomial in n. Indeed, these conditions can be taken to be in disjunctive normal form, as they merely express that a single step of the Turing machine is correctly carried out.

2.5. Encoding an oracle polynomial time computation

We also will need to encode the computation of a polynomial time function f which has oracle access to the two oracles A and Y, and takes the first-order a as an ordinary input. For this, we will use propositional variables $u_{t,s}$ that encode the bits of a straightforward encoding of the computation of the Turing machine $N_f^{A,Y}(a)$ which computes f. We no longer care about expressing the correctness of the values $u_{t,s}$ with propositional formulas; instead, we only want the correctness to be polynomial time checkable. Thus, we can simplify the construction from the previous subsection for encoding computations of $M^A(a)$ by dropping the subscript "c" and not using the slowed-down handling of oracle queries.

The convention is that $u_{t,s}$ is the s-th bit of the t-th configuration of $N_f^{A,Y}(a)$. Let $\vec{u}$ be the binary string concatenation of the $u_{t,s}$'s; since f is polynomial time, the string $\vec{u}$ is polynomial length. There are polynomial time algorithms (relative to A and Y, taking $\vec{u}$ and a as inputs) for parsing $\vec{u}$ and checking whether it correctly encodes the computation of f, including checking whether the answers to the oracle queries are correct. Furthermore, the theory S^1_2 can Σ^{b}_1-define these algorithms and prove their properties.

3. MAIN THEOREM FOR V^1_2

The V^1_2 proof is little less complicated, so we do it first.

PROOF PROOF OF THEOREM 1.1. Theorem 2.5 already states that $e\mathcal{F}$CON is provably total in V^1_2. We still need to show that any provably total NP search problem of V^1_2 is many-one reducible to $e\mathcal{F}$CON, provably in S^1_2.

Let $\psi(y, a, A)$ define a provably total NP search problem for V^1_2, so that V^1_2 proves $(\exists y{\le}\tau(a))\psi(y, a, A)$ for some term $\tau(a)$. We may assume w.l.o.g. that ψ is a sharply bounded formula, since any outermost existential quantifier in ψ may be incorporated into the bounded quantifier "$(\exists y{\le}\tau(a))$". Let $M^A(a)$ be as given by Theorem 2.4 so that S^1_2 proves:

$$\begin{aligned}
&(\exists\sigma)[\sigma \text{ encodes a local witness showing that}\\
&\qquad\qquad Y \text{ does not correctly encode a computation of } M^A(a)]\\
&\vee (\exists y{\le}\tau(a))(y = \mathrm{out}_M(Y) \wedge \psi(y, a, A)).
\end{aligned} \tag{6}$$

By Theorem 2.2, there is a polynomial time function $f^{A,Y}$ so that, provably in S^1_2, $f^{A,Y}(a)$ always produces either σ or y satisfying (6). When $f^{A,Y}(a)$ outputs a value σ, this means that σ codes a (polynomial size) set of values x so that the values $Y(x)$ for $x \in \sigma$ violate the correctness of Y as an encoding of the computation of $M^A(a)$. Furthermore, without loss of generality, $f^{A,Y}(a)$ has queried Y at all of the values x in the set σ (since if not, it could just query Y at these values before halting). Similarly, when $f^{A,Y}(a)$ outputs a value for y, we may assume w.l.o.g. that the computation has queried values $A(s)$ for sufficiently many values s so as to verify that $\psi(y,a,A)$ is true. Indeed, $\psi(y,a,A)$ is sharply bounded, so the quantified variables range over polynomially many values. Thus, for fixed y and a, the truth of $\psi(y,a,A)$ can be ascertained by examining a polynomial time computable set $S = S(y,a)$ of values so that $\psi(y,a,A)$ depends only on $A(s)$ for $s \in S$ — since ψ is sharply bounded, S only depends on the first order parameters y, a of ψ and not on A. All these properties of $f^{A,Y}(a)$ are provable in S^1_2.

The correctness condition for a value $Y(x)$, $x = \langle t,s,c\rangle$, is given by a polynomial size propositional formula $\varphi_{t,s,c}$ as shown in equations (3)-(5). Recall that, depending on the values of t,s,c, this correctness condition involves variables $y_{t',s',c'}$ and may also involve variables a_i representing the input bits of a and a variable α_w representing the value $A(w)$. As already mentioned, when $f^{A,Y}(a)$ outputs a local witness set σ for the value $x = \langle t,s,c\rangle$, $f^{A,Y}(a)$ is assumed to have queried all the values of $y_{t',s',c'}$'s needed to show that some $\varphi_{t,s,c}$ evaluates to false.

To prove Theorem 1.1, we define a polynomial time, many-one reduction from the TFNP problem defined by $\psi(y,a,A)$ to the $e\mathcal{F}$CON problem. For this, we must construct an extended Frege "proof" of a contradiction. This extended Frege proof is exponential size (that is size $2^{n^{O(1)}}$, where $n = |a|$), and is constructed as a function of the first-order input a and the second order predicate A. Then, we must show that any local witness for a syntactic error in the extended Frege proof gives a solution y to the problem $\psi(y,a,A)$. The extended Frege proof must be uniform; namely, its i-th symbol must be computable by a polynomial time function $h^A(i,a)$, and S^1_2 must be able to prove simple syntactic properties of the extended Frege proof as encoded by $h^A(i,a)$. The formulas in the extended Frege proof will have polynomial size and will be polynomial time constructible from parameters related to their location. Thus, to achieve uniformity, we only need to pad the proof so that the location of formulas also becomes polynomial time computable from parameters.

The first phase of the extended Frege proof introduces new variables $y_{t,s,c}$ with the extension rules as given by equations (3)-(5); these variables encode the computation of $M^A(a)$. The formulas (3)-(5) involve variables α_w and $a_1,\ldots,a_n$: in the extension rules, these variables are just replaced by the constant formulas $\top$ (True) or $\bot$ (False) depending on whether $A(w)$ is true or not, and on the bits of a. That is, there are no variables α_w or a_i in the constructed extended Frege proof; instead, they are replaced by constants based on the inputs a and A. This is permitted since the extended Frege proof being constructed depends on a and A. The extension axioms are polynomial size, and can be padded to occupy a fixed (polynomial) length by adding commas. Thus their locations in the proof can be given by a simple polynomial time function of t,s,c so that the resulting proof is suitably uniform. The variables $y_{t,s,c}$ completely define the computation of $M^A(a)$.

The second phase of the extended Frege proof consists of exponentially many "axioms" corresponding to the truth values of $\neg\psi(m,a,A)$ for $m = 0,\ldots,\tau(a)$. In other words, the second phase of the extended Frege proof lists a sequence of $\tau(a){+}1$ many formulas $\top$ or $\bot$ depending on whether the formulas $\neg\psi(m,a,A)$ are true or false (respectively). These formulas are all labeled as being axioms, correctly so in the case of formulas $\top$ and incorrectly so for the formulas $\bot$. Since $\psi(m,a,A)$ will be true for at least one value of m (by virtue of $(\exists y{\le}\tau(a))\psi(y,a,A)$ being provable), some of these "axioms" will be fallacious.

The formulas $\top$ and $\bot$ are both length 8, so their positions in the proof are a simple function of a. These places where the extended Frege proof falsely adds $\bot$ as an axiom are the only errors in the extended Frege proof. Hence, given an occurrence of $\bot$, one immediately obtains a value for m satisfying $\psi(m,a,A)$.

Observe that we cannot stop at this point with the construction of the extended Frege proof of $\bot$, as we lack a polynomial time function giving the location of an "axiom" $\bot$; this prevents us from inferring $\bot$ as the last line of the extended Frege proof.

The third, and most complicated, phase of the extended Frege proof thus works with the polynomial time computation of $f^{A,Y}(a)$. The intuition is to show that the computation of $f^{A,Y}(a)$ does not exist, via a brute force truth table proof: that is, to show for every sequence of truth values, that the sequence does not encode a computation of $f^{A,Y}(a)$.

Recall from Section 2.5 that the computation of $f^{A,Y}(a)$ is encoded by bits $u_{t,s}$, for $t, s < p(n)$, with $\vec{u}$ denoting their string concatenation. The values $u_{t,0}, \ldots, u_{t,p(n)-1}$ give the configuration of f at time t: this string of bits is denoted $\vec{u}_t$. Suppose the values of $u_{t,s}$ are fixed to particular true/false values. To check whether these correctly encode the computation of $f^{A,Y}(a)$, we need only to check whether each $u_{t,s}$ is correctly set according to the values of $u_{t-1,s'}$ and the bits a_i of the input, and (for oracle queries) the value of some $A(w)$ or $Y(w)$. Specifically, there is a polynomial time function $G^A(a, t, s, \vec{u}_{t-1})$ which outputs an atomic propositional formula $\gamma_{t,s}$ which specifies what the value of $u_{t,s}$ must be in a correct calculation — observe that the definition of $\gamma_{t,s}$ depends on the chosen values of $\vec{u}$. The formula $\gamma_{t,s}$ has at most one variable, namely a $y_{t',s',c}$: it can be one of the following:

1. When $t = 0$, $\gamma_{t,s}$ can be the constant a_i indicating that $u_{0,s}$ codes an input bit. Note that a_i is a constant $\top$ or $\bot$ since a is fixed.
2. When the computation of $f^{A,Y}(a)$ as encoded by $\vec{u}_{t-1}$ is making an oracle query $A(w)$ or $Y(\langle t', s', c\rangle)$, then $\gamma_{t,s}$ can be the constant α_w or the variable $y_{t',s',c}$, respectively. Note that α_w is a constant $\top$ or $\bot$ since A is fixed. Thus in this case, the formula $\gamma_{t,s}$ is $\top$ or $\bot$ or $y_{t',s',c}$.
3. If 1. and 2. do not apply, then $\gamma_{t,s}$ is a constant $\top$ or $\bot$ based on the earlier values $u_{t-1,s'}$.

The function $G^A(a, t, s, \vec{u}_{t-1})$ and its output $\gamma_{t,s}$ can be Σ^{b}_1-defined in S^1_2, and S^1_2 proves its simple properties. Still working with a fixed setting of true/false values for the $u_{t,s}$'s, let $U_{t,s}$ be the constant $\top$ or $\bot$ giving the truth value of $u_{t,s}$. (We can fix $U_{t,s}$ in this way since the value of $u_{t,s}$ has been fixed: $u_{t,s}$ is a truth value, and $U_{t,s}$ is a propositional formula.) We can thus form in polynomial time the conjunction

$$\bigwedge_{t,s} (U_{t,s} \leftrightarrow \gamma_{t,s}). \tag{7}$$

where the conjuction is taken over the $p(n)^2$ many values for t, s needed to code the polynomial time computation of $f^{A,Y}(a)$. The formula (7) expresses that $\vec{U}$ represents a correct encoding of the computation of $f^{A,Y}(a)$.

The third phase of the extended Frege proof disproves the formulas (7) for each of the $2^{p(n)^2}$ many choices of $\vec{u} \in \{0,1\}^{p(n)^2}$. There are three kinds of possible disproofs:

1. The conjunction (7) contains a conjunct $\bot \leftrightarrow \top$ or conversely $\top \leftrightarrow \bot$, or some variable $y_{t,s,c}$ appears twice, equivalent to both values $\bot$ and $\top$. The conjuction is easy to disprove in this case.
2. Condition 1. does not hold, and the values $u_{t,s}$ encode a computation of $f^{A,Y}(a)$ that outputs a (polynomial size) local witness set σ consisting of values for variables $y_{t,s,c}$ which violate one of the formulas (3)-(5). In this case the conjunction (7) gives an explicit true/false value to each $y_{t',s',c}$ in the local witness set, namely by including a conjunct $\top \leftrightarrow y_{t',s',c}$ or $\bot \leftrightarrow y_{t',s',c}$. These explicit values violate one of (3)-(5). On the other hand, the conditions (3)-(5) are extension axioms in the extended Frege proof, so a contradiction is obtained with a polynomial size Frege proof, thus disproving (7) in this case.

A formal proof of $\neg(7)$ can be obtained in the following way: We are in the situation that the witness set σ identifies a $y_{t',s',c}$ whose extension axiom is violated, based on the other variables in σ and their truth values fixed in (7). From the extension axiom for $y_{t',s',c}$ we can then easily derive $\neg(7)$, using the following more general fact: If $\chi(z_1,..,z_k)$ is a formula, and $v_1,\dots,v_k \in \{0,1\}$ a satisfying assignment, then there is a polynomial size proof (in χ and $\vec{v}$) of the formula $(\bigwedge_i (V_i \leftrightarrow z_i)) \to \chi(\vec{z})$ where V_i again is the truth formula $\bot$ or $\top$ corresponding to the truth value v_i being 0 or 1, respectively.

3. Otherwise, the values of $u_{t,s}$ encode a correct computation of $f^{A,Y}(a)$ that outputs a value $y \le \tau(a)$ such that $\psi(y,a,A)$ holds. In this case, the formula $\psi(y,a,A)$ is true, and the second phase added $\bot$ as a (fallacious) axiom for $m = y$, which we can use to obtain a contradiction, again disproving (7). We are able to use the formula $\bot$ at this point, because the value $m = y$ tells us where $\bot$ appears earlier as a formula in the Frege proof.

Given $\vec{u}$, we can decide in polynomial time which case 1.-3. applies. For each case we can construct in polynomial time a proof of $\neg(7)$ of fixed polynomial size using padding.

The fourth and final phase of the extended Frege proof combines the $2^{p(n)^2}$ many instances of formulas $\neg(7)$ to obtain a contradiction. There is one formula (7) for each $\vec{u} \in \{0,1\}^{p(n)^2}$. Taking the variables $\vec{u}$ ordered lexicographically first by t and then by s, reindex $\vec{u}$ as $u_0,\dots,u_{p(n)^2-1}$. Let $\vec{u}{\restriction}\ell$ be $u_0,\dots,u_{\ell-1}$. For fixed $\vec{u} \in \{0,1\}^{p(n)^2}$, let $(7)_{\vec{u}\restriction\ell}$ be the conjunction of the first ℓ components of (7), where $(7)_{\vec{u}\restriction 0}$ is defined as $\top$. It is trivial to give a Frege proof of $\neg(7)_{\vec{u}\restriction\ell}$ from assumptions $\neg(7)_{\vec{u}\restriction\ell,0}$ and $\neg(7)_{\vec{u}\restriction\ell,1}$. We can choose these to be of fixed polynomial size using padding. Thus we can successively provide proofs of $\neg(7)_{\vec{u}\restriction\ell}$ starting from the already proved formulas $\neg(7)$. For $\ell = 0$ this provides a proof of contradiction.

Examining the four phases of the construction of the extended Frege proof of a contradiction, the only place where the proof can contain an error is in phase two, where at least one false formula $\neg\psi(m,a,A)$ has led to an axiom of the form $\bot$. Furthermore, identifying one of these places in the Frege proof also identifies a value y such that $\psi(y,a.A)$ is true.

Therefore, there is a many-one reduction from the NP search problem given by $(\forall x)(\exists y{\le}\tau(x))\psi(y,x,A)$ to the consistency search problem for extended Frege proofs. This completes the proof of Theorem 1.1. $\square$

It is interesting to note that the constructed extended Frege proof, albeit exponentially large, contains only polynomial size formulas. But of course, the use of the extension rule is nested exponentially many times, hence it is implicitly defining values defined by exponential size Boolean circuits.

4. MAIN THEOREM FOR U^1_2

The strategy for the proof of Theorem 1.2 is similar to the proof of Theorem 1.1 above; however, phase one becomes more complicated. The variables $y_{t,s,c}$ are no longer introduced by extension; they are instead defined by a divide-and-conquer Nepomnjaščiĭ-Savitch style construction for defining polynomial space computations, which allows the variables $y_{t,s,c}$ to be defined with formulas that are (merely) exponentially long. This is essentially the usual Nepomnjaščiĭ-Savitch construction for defining space-restricted computation; in particular, it is a modified version of the formulation of Savitch's theorem in the theory of U^1_2 that is carried out in [Beckmann and Buss 2014].

Proof Proof of Theorem 1.2. We shall only sketch the differences between the proofs of Theorems 1.1 and 1.2. Let $\psi(y,a,A)$ define a provably total NP search problem for U^1_2, so that U^1_2 proves $(\exists y{\le}\tau(a))\psi(y,a,A)$ for some term $\tau(a)$. We may assume w.l.o.g. that ψ is a sharply bounded formula. Let the polynomial space $M^A(a)$ be as given by Theorem 2.3 so that S^1_2 proves:

$$\begin{aligned}(\exists\sigma)[&\sigma \text{ encodes a local witness showing that}\\ &\qquad Y \text{ does not correctly encode a computation of } M^A(a)]\\ &\vee (\exists y \le \tau(a))(y = \mathrm{out}_M(Y) \wedge \psi(y, a, A)).\end{aligned} \tag{8}$$

By Theorem 2.2, there is a polynomial time function f so that, provably in S^1_2, $f^{A,Y}(a)$ always produces either σ or y satisfying (8). This function f satisfies the same properties as were assumed for the proof of Theorem 1.1.

In order to prove Theorem 1.2, we shall construct a polynomial time uniform Frege "proof" of a contradiction, similar to the way the extended Frege "proof" was constructed for Theorem 1.1. The first phase of the extended Frege proof of Theorem 1.1 repeatedly used the extension rule to define the variables $y_{t,s,c}$ defining the computation of $M^A(a)$: this is no longer possible since we must construct a Frege proof. Nor is it possible to just unwind the extension rules (3)-(5), since that would yield double exponentially long formulas, and these would be too long to be used in an exponentially large Frege proof. Instead, we use the Nepomnjaščiĭ-Savitch construction to introduce formulas $\zeta_{t,s,c}$ which define the same values as $y_{t,s,c}$, but have only exponential size.

For $t = c = 0$, $\zeta_{0,s,0}$ will be defined using the formula $\varphi_{0,s,0}$ from (5). For other values of t and c, $\zeta_{t,s,c}$ will be defined with a divide-and-conquer construction. The values of t and c can be viewed as integers ranging from 0 to $2^{p(n)} - 1$, where $n = |a|$. Since M is polynomial space bounded, the value of s ranges w.l.o.g. from 0 to $p(n)-1$. To simplify the definition of $\zeta_{t,s,c}$ by divide-and-conquer, we unify the values t and c by letting $T = T(t, c) = 2^{p(n)}t + c$; and write $\zeta_{T,s}$ in place of $\zeta_{t,s,c}$. This T denotes the "slowed-down" computation time.

Consider $T = 0$. The formulas $\zeta_{0,s} = \zeta_{0,s,0}$ are defined to be either $\top$ or $\bot$, matching the truth value of the formulas $\varphi_{0,s,0}$ from (5) where the variables a_i are replaced with the constants $\top$ or $\bot$ depending on the value of a_i. (We are able to make this substitution of the constants for the a_i's because the Frege proof depends on a and A.) Since $\varphi_{0,s,0}$ with the a_i's replaced with $\top$ or $\bot$ is polynomial size and closed, its truth value can be computed in polynomial time.

For $T \neq 0$, we define $\zeta_{T,s}$ with a divide-and-conquer construction. Let $\vec{e}$ and $\vec{f}$ be vectors of $p(n)$ constants $\top$ and $\bot$. Thus, each e_i and f_i is the formula $\top$ or $\bot$. Let $0 \le T_2 < T_1 < 2^{2p(n)}$, and further suppose that, for some d, $2^d | T_2$ and $T_1 \le T_2 + 2^d$. The intent is that e_s (respectively, f_s) is equal to the value of $\zeta_{T_1,s}$ (respectively, $\zeta_{T_2,s}$) for $s = 0, \ldots, p(n)-1$. We shall define formulas $\mathrm{Next}_{T_1,T_2}(\vec{e}, \vec{f})$ expressing the property that if $\vec{f}$ defines the values of the formulas $\zeta_{T_2,s}$ then $\vec{e}$ defines the values of the formulas $\zeta_{T_1,s}$. In other words, Next_{T_1,T_2} defines the effect of transitioning from T_2 to T_1.

For fixed $T_1 = T_2+1$, the formula $\mathrm{Next}_{T_1,T_2}(\vec{e}, \vec{f})$ is either the constant $\top$ or the constant $\bot$ depending on whether $\vec{e}$ correctly represents the configuration of $M^A(a)$ that results at (slowed-down) time T_1 if $\vec{f}$ is the configuration at (slowed-down) time T_2. The formula $\mathrm{Next}_{T_1,T_2}(\vec{e}, \vec{f})$ can be constructed by evaluating the conditions (3) or (4) for all $p(n)$ many values of s. For any fixed s, the condition (3) or (4) has as its righthand side a polynomial size formula involving the values $\vec{f}$ and possibly the value $A(w)$ (as represented by α_w), and has as its lefthand side the value $\vec{e}$. Since we have oracle access to A, the value of α_w is known. Since $\vec{e}$ and $\vec{f}$ are also constants, each condition evaluates to true or false. Thus $\mathrm{Next}_{T_1,T_2}(\vec{e}, \vec{f})$ can be represented as $\top$ or $\bot$. The determination of the formula $\mathrm{Next}_{T_1,T_2}(\vec{e}, \vec{f})$ can be carried out in polynomial time as a function of $\vec{a}$, T_1, T_2, $\vec{e}$ and $\vec{f}$ relative to the oracle A.

Now consider fixed $T_1 > T_2+1$. Suppose also that $d \geq 1$ and $2^d | T_2$ and $T_2+2^{d-1} < T_1 \leq T_2+2^d$. Consider fixed sequences $\vec{e}$ and $\vec{f}$ of constants $\top$ and $\bot$. We define $\mathrm{Next}_{T_1,T_2}(\vec{e},\vec{f})$ as the following formula, where $g_0,\ldots,g_{p(n)-1}$ ranges over all possible vectors of $p(n)$ constants $\top$ and $\bot$ (so there are $2^{p(n)}$ many possible choices for the constants $g_0,\ldots,g_{p(n)-1}$):

$$\bigvee_{\vec{g}} \Big(\mathrm{Next}_{T,T_2}(\vec{g},\vec{f}) \wedge \mathrm{Next}_{T_1,T}(\vec{e},\vec{g})\Big)$$

where $T = T_2+2^{d-1}$, so $T_2 < T < T_1$. Finally, let $\zeta_{T,s} = \zeta_{t,s,c}$ be the formula

$$\bigvee_{\vec{e} \text{ s.t. } e_s = \top} \mathrm{Next}_{T,0}(\vec{e},\zeta_{0,0,0},\ldots,\zeta_{0,p(n)-1,0}).$$

The formulas $\zeta_{0,s,0}$ were by definition constants $\top$ or $\bot$. The formulas $\zeta_{T,s}$ are variable-free. Written as a tree, they have height $O(p(n))$ with nodes of fan-in $\leq 2^{p(n)}$, hence have size $2^{p(n)^{O(1)}}$. This exponential size means that $\zeta_{T,s}$ cannot be evaluated in polynomial time. For this reason, even though $\zeta_{T,s}$ evaluates to a constant value, the Frege proof must use an exponential size formula for $\zeta_{T,s}$ in order to have the Frege proof be polynomial time uniform.

The formula $\mathrm{Next}_{T_1,T_2}(\vec{e},\vec{f})$ can be viewed as a tree of polynomial height with $\bigvee$-gates of fanin $2^{p(n)}$ alternating with binary $\wedge$-gates. A path through the tree starting from the root can be coded as a polynomial length sequence of numbers $z_1,j_1,z_2,j_2,\ldots$ with $z_i < 2^{p(n)}$ describing a child of an $\bigvee$-node, and $j_i < 2$ denoting one of the two children of a binary $\wedge$. Each leaf of the tree is a formula of the form $\mathrm{Next}_{T+1,T}(\vec{e}\,',\vec{f}\,')$; its arguments $\vec{e}\,'$ and $\vec{f}\,'$ are also sequences of constants $\top$ and $\bot$ and they, as well as T, can be computed as a polynomial time function of the path and of T_1, T_2, $\vec{e}$ and $\vec{f}$. The leaf formulas $\mathrm{Next}_{T+1,T}(\vec{e}\,',\vec{f}\,')$ are either $\top$ or $\bot$ and thus all have the same length $|\top| = |\bot| = 8$. From this, it is easy to compute the size of the formula $\mathrm{Next}_{T_1,T_2}(\vec{e},\vec{f})$ as a polynomial time function of T_1 and T_2. This makes it possible to compute in polynomial time, the identity of the i-th symbol of $\mathrm{Next}_{T_1,T_2}(\vec{e},\vec{f})$ as a function of i, T_1, T_2, $\vec{e}$ and $\vec{f}$.

The first phase of the Frege proof consists of proofs of the statements corresponding to (3) and (4) for $T = T(t,s) > 0$:

$$\zeta_{T,s} \;\leftrightarrow\; \varphi_{t,s,c}(\alpha_w,\zeta_{T-1,s-Cp(n)},\ldots,\zeta_{T-1,s+Cp(n)}). \tag{9}$$

(As before, the ζ's with subscripts out of range are to be omitted.) The statements that correspond to (5) have simple proofs based on evaluating simple closed formulas, as the formula $\zeta_{0,s}$ is defined as the truth value of the formula on the right hand side of the equivalence in (5).

Before proving the formulas (9), the Frege proof needs to prove uniqueness statements

$$\neg\mathrm{Next}_{T_1,T_2}(\vec{e},\vec{f}) \vee \neg\mathrm{Next}_{T_1,T_2}(\vec{e}\,',\vec{f}). \tag{10}$$

for all appropriate T_1, T_2, and all sequences $\vec{e}$, $\vec{e}\,'$, $\vec{f}$ such that $\vec{e} \neq \vec{e}\,'$. The uniqueness statements are easily proved for $T_1 = T_2+1$. For $T_1 > T_2+1$, they are likewise straightforward to prove using the recursive definition of Next, for successively larger values of $T_1 - T_2$.

The formulas (9) are proved by proving more general statements

$$\mathrm{Next}_{T+1,T_2}(\vec{e}\,',\vec{f}) \wedge \mathrm{Next}_{T,T_2}(\vec{e},\vec{f}) \rightarrow \mathrm{Next}_{T+1,T}(\vec{e}\,',\vec{e}). \tag{11}$$

We denote these statements by $(11)_{T,T_2}$, and prove them for $T_2 < T < T_2+2^d$ where 2^d is the largest power of 2 dividing T_2 (or $d = p(n)$ if $T_2 = 0$). The formulas (9) will then follow easily from the formulas $(11)_{T,0}$.

For each formula $(11)_{T,T_2}$, T_2 is equal to $\alpha 2^d$ with α odd, or with $\alpha = 0$ and $d = p(n)$, and $T_2 < T < T_2+2^d$. These formulas are proved for successively larger values of $T - T_2$. Choose

d' so that $T_2 + 2^{d'} \le T < T{+}1 \le T_2 + 2^{d'+1}$. If $T = T_2 + 2^{d'}$ (and in particular if $d = 1$), then $(11)_{T,T_2}$ follows immediately from the definition of Next_{T+1,T_2} and the uniqueness statement (10) for Next_{T,T_2}. If $T > T_2 + 2^{d'}$, then $(11)_{T,T_2}$ follows immediately from the definitions of Next_{T,T_2} and Next_{T+1,T_2} and the $2^{p(n)}$ many instances of $(11)_{T,T_2+2^{d'}}$ with all $2^{p(n)}$ possible constants $\vec{e}$. Those formulas are available since the statements $(11)_{T,T_2}$ are proved in increasing order of $T - T_2$.

The Frege proof contains proofs of $(11)_{T_1,T_2}$ for each appropriate set of values for T_1, T_2, $\vec{e}$, $\vec{e}\,'$ and $\vec{f}$. These proofs are arranged in increasing order of $T_1 - T_2$, namely according to the lexicographic order of $T_1{-}T_2, T_2, \vec{e}, \vec{e}\,', \vec{f}$. The proofs of the formulas $(11)_{T_1,T_2}$ can be made to be fixed exponential size using padding (by adding commas). When T_1 and T_2 do not satisfy the condition $T_1 \le T_2 + 2^d$ with $2^d | T_2$, that portion of the proof is left blank by filling it in with commas. The proofs are straightforward to construct, and there is a polynomial time algorithm which, given T_1, T_2, $\vec{e}$, $\vec{e}\,'$ and $\vec{f}$ and given $i > 0$, determines the location of the corresponding proof and determines the i-th symbol in that subproof. Thus, this part of the Frege proof is polynomial time uniform.

The rest of the proof of Theorem 1.2 is similar to the proof of Theorem 1.1. The only change is that the formulas $\zeta_{T,s} = \zeta_{t,s,c}$ are used instead of the variables $y_{t,s,c}$. The formulas (9) play the same role as the formulas (3)-(5) which were used in the proof of Theorem 1.1. Since the formulas $\zeta_{t,s,c}$ are exponential size, additional care is required to keep the Frege proof uniformly describable by a polynomial time function with its syntactic properties provable in S^1_2. Nonetheless, the details of how the Frege proof is uniformly describable are straightforward, so we omit them. The overall size of the Frege proof is $2^{p(n)^{O(1)}} = 2^{n^{O(1)}}$.

The result is a uniformly polynomial time description of a Frege "proof" of a contradiction. The only false steps in the Frege "proof" are places where $\perp$ corresponding to a true $\psi(y, a, A)$ was introduced as an axiom. Thus, finding an error in the Frege "proof" gives a value y witnessing $\psi(y, a, A)$. This shows that the NP search problem for $\exists y{\le}\tau(a)\, \psi(y, a, A)$ is many-one reducible to the consistency search problem for Frege systems. □

ACKNOWLEDGMENTS

We thank Jan Krajíček, Neil Thapen, and the two referees for helpful substantive comments on earlier drafts of this paper.

REFERENCES

Toshiyasu Arai. 2000. A Bounded Arithmetic AID for Frege Systems. *Annals of Pure and Applied Logic* 103 (2000), 155–199.

Paul Beame, Stephen Cook, Jeff Edmonds, Russell Impagliazzo, and Toniann Pitassi. 1998. The Relative Complexity of NP Search Problems. *J. Comput. System Sci.* 57, 1 (1998), 3–19.

Arnold Beckmann and Samuel R. Buss. 2014. Improved Witnessing and Local Improvement Principles for Second-Order Bounded Arithmetic. *ACM Transactions on Computational Logic* 15, 1, Article 2 (2014), 35 pages.

Samuel R. Buss. 1986. *Bounded Arithmetic*. Bibliopolis, Naples, Italy. Revision of 1985 Princeton University Ph.D. thesis.

Samuel R. Buss. 1987. The Boolean Formula Value Problem is in ALOGTIME. In *Proceedings of the 19-th Annual ACM Symposium on Theory of Computing*. 123–131.

Samuel R. Buss. 1993. Algorithms for Boolean Formula Evaluation and for Tree Contraction. In *Arithmetic, Proof Theory and Computational Complexity*, P. Clote and J. Krajíček (Eds.). Oxford University Press, 96–115.

Samuel R. Buss. 1998. An Introduction to Proof Theory. In *Handbook of Proof Theory*, S. R. Buss (Ed.). North-Holland, 1–78.

Samuel R. Buss. 1999. Propositional Proof Complexity: An Introduction. In *Computational Logic*, U. Berger and H. Schwichtenberg (Eds.). Springer-Verlag, Berlin, 127–178.

Samuel R. Buss, Steven A. Cook, Arvind Gupta, and Vijaya Ramachandran. 1992. An Optimal Parallel Algorithm for Formula Evaluation. *SIAM J. Comput.* 21 (1992), 755–780.

Samuel R. Buss and Jan Krajíček. 1994. An Application of Boolean Complexity to Separation Problems in Bounded Arithmetic. *Proceedings of the London Mathematical Society* 69 (1994), 1–21.

Stephen A. Cook. 1975. Feasibly Constructive Proofs and the Propositional Calculus. In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, 83–97.

Stephen A. Cook and Tsuyoshi Morioka. 2005. Quantified Propositional Calculus and A Second-Order Theory for NC^1. *Archive for Mathematical Logic* 44 (2005), 711–749.

Stephen A. Cook and Phuong Nguyen. 2010. *Foundations of Proof Complexity: Bounded Arithmetic and Propositional Translations*. ASL and Cambridge University Press. 496 pages.

Stephen A. Cook and Robert A. Reckhow. 1979. The Relative Efficiency of Propositional Proof Systems. *Journal of Symbolic Logic* 44 (1979), 36–50.

Paul Goldberg and Christos Papadimitriou. 2016. Towards a Unified Complexity Theory of Total Functions. (2016). Draft 'working paper', available online.

Emil Jeřábek. 2004. Dual Weak Pigeonhole Principle, Boolean Complexity, and Derandomization. *Annals of Pure and Applied Logic* 124 (2004), 1–37.

Emil Jeřábek. 2006. The Strength of Sharply Bounded Induction. *Mathematical Logic Quarterly* 6 (2006), 613–624.

David S. Johnson, Christos H. Papadimitriou, and Mihalis Yannakakis. 1988. How Easy is Local Search? *J. Comput. System Sci.* 37 (1988), 79–100.

Leszek Aleksander Kołodziejczyk, Phuong Nguyen, and Neil Thapen. 2011. The Provably Total NP Search Problems of Weak Second-Order Bounded Arithmetic. *Annals of Pure and Applied Logic* 162, 2 (2011), 419–446.

Jan Krajíček. 1995. *Bounded Arithmetic, Propositional Calculus and Complexity Theory*. Cambridge University Press, Heidelberg.

Jan Krajíček. 2004. Implicit Proofs. *Journal of Symbol Logic* 69, 2 (2004), 387–397.

Jan Krajíček. 2011. *Forcing with Random Variables and Proof Complexity*. Cambraidge University Press.

Jan Krajíček. 2016. Consistency of Circuit Evaluation, Extended Resolution, and Total NP Search Problems. *Forum of Mathematics, Sigma* 4 (2016), e15 (13 pages).

Jan Krajíček, Alan Skelley, and Neil Thapen. 2007. NP Search Problems in Low Fragments of Bounded Arithmetic. *Journal of Symbolic Logic* 72, 2 (2007), 649–672.

Christos H. Papadimitriou. 1990. On Graph-Theoretic Lemmata and Complexity Classes (Extended Abstract). In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (Volume II)*. IEEE Computer Society, 794–801.

Christos H. Papadimitriou. 1994. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *J. Comput. System Sci.* 48, 3 (1994), 498–532.

Jeff B. Paris and Alex J. Wilkie. 1985. Counting Problems in Bounded Arithmetic. In *Methods in Mathematical Logic, Lecture Notes in Mathematics #1130*. Springer-Verlag, 317–340.

Pavel Pudlák and Neil Thapen. 2012. Alternating Minima and Maxima, Nash Equilibria and Bounded Arithmetic. *Annals of Pure and Applied Logic* 163 (2012), 604–614.

Robert A. Reckhow. 1976. *On the Lengths of Proofs in the Propositional Calculus*. Ph.D. Dissertation. Department of Computer Science, University of Toronto. Technical Report #87.

Alan Skelley and Neil Thapen. 2011. The Provably Total Search Problems of Bounded Arithmetic. *Proceedings of the London Mathematical Society* 103, 1 (2011), 106–138.