Bounded Arithmetic and Propositional Proof Complexity

Samuel R. Buss*
Departments of Mathematics and Computer Science
University of California, San Deigo
La Jolla, CA 92093-0112

Abstract

This is a survey of basic facts about bounded arithmetic and about the relationships between bounded arithmetic and propositional proof complexity. We introduce the theories S^i_2 and T^i_2 of bounded arithmetic and characterize their proof theoretic strength and their provably total functions in terms of the polynomial time hierarchy. We discuss other axiomatizations of bounded arithmetic, such as minimization axioms. It is shown that the bounded arithmetic hierarchy collapses if and only if bounded arithmetic proves that the polynomial hierarchy collapses.

We discuss Frege and extended Frege proof length, and the two translations from bounded arithmetic proofs into propositional proofs. We present some theorems on bounding the lengths of propositional interpolants in terms of cut-free proof length and in terms of the lengths of resolution refutations. We then define the Razborov-Rudich notion of natural proofs of $P \neq \text{NP}$ and discuss Razborov's theorem that certain fragments of bounded arithmetic cannot prove superpolynomial lower bounds on circuit size, assuming a strong cryptographic conjecture. Finally, a complete presentation of a proof of the theorem of Razborov is given.

1 Review of Computational Complexity

1.1 Feasibility

This article will be concerned with various "feasible" forms of computability and of provability. For something to be *feasibly* computable, it must be computable in practice in the real world, not merely effectively computable in the sense of being recursively computable.

*Supported in part by NSF grants DMS-9503247 and DMS-9205181.

The notion of *effective* computation is an intuitive (nonformal) concept: a problem is effectively decidable if there is a definite procedure which, in principle, can solve every instance of the problem. Church's Thesis (for which there is strong evidence) states that a problem is effectively computable if and only if it can be recursively computed, or equivalently, if and only if it can be decided by a Turing machine computation. However, effectively computable problems may not be feasibly computable since the computational resources such as time and space needed to solve an effectively computable problem may be enormous. By "feasible" we mean "computable in practice" or "computable in the real world"; or more to the point, a problem is feasibly computable if for any reasonably sized instance of the problem, we are able to solve that instance, albeit perhaps with a large (but not impossibly large) commitment of computing resources.

Like the notion of "effective", the notion of "feasible" is an intuitive concept; so it is natural to look for a feasible analog of Church's Thesis that will give a formal, mathematical model for feasible computation. A big advantage of having a mathematical model for feasible computability is that this allows a mathematical investigation of the power and limits of feasible computation. There is a widespread opinion that polynomial time computability is the correct mathematical model of feasible computation. This widespread belief is based primarily on two facts. First, the class of polynomial time computable functions and predicates is a robust and well-behaved class which has nice closure properties and is invariant for many natural models of computation. Second, and more importantly, it is an empirical observation about actual, real-world algorithms that it is nearly always the case that decision problems which are known to have polynomial time algorithms are also known to be feasibly decidable (see Cook [17] for a discussion of this).

1.1.1 Feasibility Versus Infeasibility — An Example

For an example of feasibility and infeasibility, we will consider the problem of integer multiplication. We'll give two algorithms, the first one is effective but not feasible, while the second one is effective and feasible.

For multiplication, we presume we are given two integers x and y in binary notation. We also presume that we already have feasible algorithms for addition which will be used as subroutines in our algorithms for multiplication. Let the length of x, denoted $|x|$, be the number of bits in the binary representation of x. For the purposes of computing runtimes, we presume that our subroutine for adding x and y uses $|x| + |y|$ steps (where a "step" involves only a constant number of bit operations).

The infeasible algorithm: This is what we might call the *first-grade algorithm* for multiplication, since in the U.S., students might learn this algorithm in the first grade. The algorithm is very simple: given two positive integers x and y, add x to itself repeatedly $y - 1$ times. The runtime of

this algorithm is $(y-1)\cdot(|x|+|y|)$ steps.

The feasible algorithm: This algorithm can be called the *fourth-grade algorithm* for multiplication (in California, this would be taught in the fourth grade). This is the usual grade school algorithm for multiplication; e.g., to multiply 6 and 5 in binary notation, one writes:

$$\begin{array}{rl} 110 & (=6) \\ 101 & (=5) \\ \hline 110 & \\ 000 & \\ 110 & \\ \hline 11110 & (=30) \end{array}$$

The runtime of this algorithm is approximately $|x|\cdot|y|$ steps.

To see why the first algorithm is infeasible and the second one is feasible, consider two 20 digit numbers x and y. To be as concrete as possible, we estimate the runtimes in seconds, assuming that each "step" takes one microsecond. Since $x \approx 10^{20}$, we have $|x| \approx 70$. Likewise, $|y| \approx 70$. The runtime of the infeasible algorithm is therefore

$$\approx\ 140 * 10^{20}\mu s\ \approx\ 1000 \times (\text{age of the universe}).$$

On the other hand, the runtime of the feasible algorithm is

$$\approx\ 70^2\mu\text{s}\ =\ 4900\mu\text{s}\ \approx\ \frac{1}{200}\text{s}$$

Clearly there is an impressive difference between the infeasible algorithm and the feasible algorithm! In this example, it is hopeless to carry out the infeasible algorithm to completion; whereas, the feasible algorithm is extremely quick on modern computers.

Definition Let n denote the sums of the lengths of the inputs to an algorithm. An algorithm is *polynomial time* if its runtime is bounded by n^c for some constant c. An algorithm is has *exponential runtime* if its runtime is bounded by 2^{n^c} for some constant c.

Clearly the infeasible algorithm has exponential runtime (but not polynomial runtime); whereas the feasible algorithm is polynomial time.

Remarks: From our theoretical viewpoint, we are primarily interested in the asymptotic behavior of the algorithms. We are also usually not interested in constant factors. Thus, the fact that modern day computers have builtin operations for multiplication of 32-bit or even 64-bit integers does not make a significant difference, since it only speeds up the above estimates of runtime by a factor of 32 or 64. Likewise, the fact that computers are currently doubling in speed every couple years does not affect our interest in feasible versus infeasible. (Actually, the doubling of speed is one of the primary

reasons for neglecting constant factors in runtime.)[†] The quadratic time, $O(n^2)$, algorithm for multiplication is not asymptotically optimal: $O(n \log n)$ runtime is possible using fast Fourier transform techniques.

1.2 P, NP, and the Polynomial-Time Hierarchy

We let $\mathbf{N}$ denote the set of non-negative integers.

Definition P is the set of polynomial time recognizable predicates on $\mathbf{N}$. Or equivalently, P is the set of polynomial time recognizable subsets of $\mathbf{N}$

FP is the set of polynomial time computable *functions*.

We let $|x| = \lceil \log_2(x+1) \rceil$ denote the length of the binary representation of x. We write $|\vec{x}| = |x_1|, |x_2|, \ldots, |x_k|$.

Remark: For us, functions and predicates are *arithmetic*: polynomial time means in terms of the length $|x|$ of the input x. Generally, computer scientists use the convention that functions and predicates operate on strings of symbols; but this is equivalent to operations on integers, if one identifies integers with their binary representation.

Cobham [15] defined FP as the closure of some base functions under composition and *limited iteration on notation*. As base functions, we can take 0, S (successor), $\lfloor \frac{1}{2} x \rfloor$, $2 \cdot x$,

$$x \leq y \quad = \quad \begin{cases} 1 & \text{if } x \leq y \\ 0 & \text{otherwise} \end{cases}$$

$$Choice(x, y, z) \quad = \quad \begin{cases} y & \text{if } x > 0 \\ z & \text{otherwise} \end{cases}$$

Definition Let q be a polynomial. f is defined from g and h by *limited iteration on notation* with space bound q iff

$$\begin{aligned} f(\vec{x}, 0) &= g(\vec{x}) \\ f(\vec{x}, y) &= h(\vec{x}, y, f(\vec{x}, \lfloor \tfrac{1}{2} y \rfloor)) \end{aligned}$$

provided $|f(\vec{x}, y)| \leq q(|\vec{x}|, |y|)$ for all $\vec{x}, y$.

Theorem 1 (Cobham [15]) *FP is equal to the set of functions which can be obtained from the above base functions by using composition and limited iteration on notation.*

[†]The rate of increase in computing performance presently shows little sign of abating; however, if should not be expected to continue forever, since presumably physical limits will eventually be met. It is interesting to note that if the rate of increase could be sustained indefinitely, then exponential time algorithms would actually be feasible, since one would merely wait a polynomial amount of time for computer power to increase sufficiently to run the algorithm quickly.

A nondeterministic Turing machine makes "choices" or "guesses": the machine accepts its input iff there is at least one sequence of choices that leads to an accepting state. On the other hand, a co-nondeterministic Turing machine makes "universal choices": it accepts iff every possible sequence of choices leads to an accepting state. NP is defined to be the set of predicates accepted by non-deterministic polynomial time Turing machines. The class coNP is defined to equal the set of predicates accepted by co-nondeterministic Turing machines, i.e., coNP is the set of complements of NP predicates. For example, the set of (codes of) satisfiable propositional formulas is in NP. And the set of (codes of) tautologies is in coNP.

We now give a second, equivalent, purely logical method of defining NP and coNP.

Definition If Ψ is a set of predicates, $PB\exists(\Psi)$ is the set of predicates A expressible as

$$\vec{x} \in A \Leftrightarrow (\exists y \leq 2^{p(|\vec{x}|)})B(\vec{x}, y)$$

for some polynomial p and some $B \in \Psi$.
$PB\forall(\Psi)$ is defined similarly with universal polynomially bounded quantification.

Definition NP $= PB\exists(P)$ and coNP $= PB\forall(P)$. In other words, NP is the set of predicates $A(\vec{x})$ expressible in the form

$$A(\vec{x}) \;\Leftrightarrow\; (\exists y \leq 2^{p(|\vec{x}|)})B(\vec{x}, y)$$

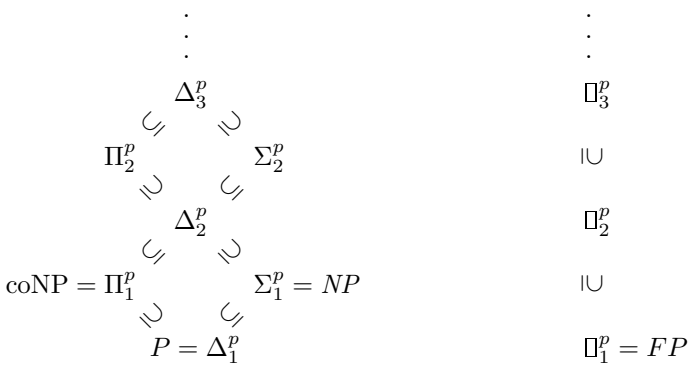
for some $B \in P$ and some polynomial p. The class coNP is similarly defined with a universal polynomially bounded quantifier.

Definition If Ψ is a set of predicates, P^{Ψ} (resp., FP^{Ψ}) is the set of predicates (resp., functions) polynomial time recognizable with oracles for a finite number of predicates in Ψ.

The notions of P, FP, NP and coNP are generalized by defining the polynomial time hierarchy:

Definition The *Polynomial Time "Hierarchy"*:

$$\begin{aligned}
\Pi_1^p &= FP \\
\Delta_1^p &= P \\
\Sigma_k^p &= PB\exists(\Delta_k^p) \\
\Pi_k^p &= PB\forall(\Delta_k^p) \\
\Delta_{k+1}^p &= P^{\Sigma_k^p} = P^{\Pi_k^p} \\
\Pi_{k+1}^p &= FP^{\Sigma_k^p} = FP^{\Pi_k^p}
\end{aligned}$$



It is an open question whether $P = \mathrm{NP}$, $\mathrm{NP} = \mathrm{coNP}$ and whether the polynomial hierarchy is proper. To sum up the current state of knowledge, all that is known is that if any two of the classes in the above hierarchy are equal, then the hierarchy collapses to those classes.

Remark: The above definitions have given purely logical characterizations of the classes in the polynomial time hierarchy in terms of *expressibility* in a formal language. In the next section we will give purely logical characterizations of these classes in terms of *derivability* in a formal theory. The most important such theory is S_2^1 in which the provably recursive functions are precisely the polynomial time computable functions. Thus we can characterize the proof-theoretic strength of S_2^1 as corresponding to polynomial time computability.

2 Bounded Arithmetic

A *constructive* proof system is one in which proofs of existence contain, or imply the existence of, algorithms for finding the object which is proved to exist. For a feasibly constructive system, the algorithm will be feasible, not merely effective. For instance, if $\forall x \exists y A(x, y)$ is provable then there should be a feasible algorithm to find y as a function of x. In the next section, we introduce feasible proof systems for number theory: more precisely, S_2^1 will be a feasible proof system, and other systems, S_2^i and T_2^i are systems that have proof-theoretic strength corresponding to higher levels of the polynomial time hierarchy.

2.1 The Language of Bounded Arithmetic

The theories of bounded arithmetic will be first-order theories for the natural numbers $\mathbf{N} = \{0, 1, 2, \ldots\}$. The first-order language for bounded arithmetic contains the predicates $=$ and $\leq$ and contains function symbols 0, S

(successor), $+$, $\cdot$, $\lfloor \frac{1}{2}x \rfloor$, $|x|$, $\#$ and relation symbol $\leq$, where

$$x\#y \;=\; 2^{|x|\cdot|y|}$$

It is easy to check that the $\#$ (pronounced "smash") function allows us to express $2^{q(|\vec{a}|)}$ for q any polynomial with positive integer coefficients.

Definition A *bounded quantifier* is a quantifier of the form $(Qx \leq t)$ with t a term not involving x. A *sharply bounded quantifier* is one of the form $(Qx \leq |t|)$. $(\forall x)$ and $(\exists x)$ are unbounded quantifiers. A *bounded* formula is one with no unbounded quantifiers.

A hierarchy of classes Σ^b_k, Π^b_k of bounded formulas is defined by counting alternations of bounded quantifiers, ignoring sharply bounded quantifiers. (Analogously to defining the arithmetic hierarchy by counting unbounded quantifiers, ignoring bounded quantifiers.)

Definition $\Sigma^b_0 = \Pi^b_0$ is the set of formulas with only sharply bounded quantifiers.

If $A \in \Sigma^b_k$ then $(\forall x \leq |t|)A$ and $(\exists x \leq t)A$ are in Σ^b_k and $(\forall x \leq t)A$ is in Π^b_{k+1}. Dually, if $A \in \Pi^b_k$ then $(\exists x \leq |t|)A$ and $(\forall x \leq t)A$ are in Π^b_k and $(\exists x \leq t)A$ is in Σ^b_{k+1}. For formulas not in prenex form, we say that a formula is in Σ^b_i (resp., Π^b_i) iff prenex operations can be used to put the formula in to the prenex Σ^b_1 (resp., Π^b_i) form defined above.

One of the primary justifications for the definition of Σ^b_i- and Π^b_i-formulas is the following theorem.

Theorem 2 *Fix $k \geq 1$. A predicate Q is in Σ^p_k iff there is a Σ^b_k formula which defines it.*

This theorem is essentially due to Stockmeyer [48] and Wrathall [51]; Kent and Hodgson [31] were the first to prove a full version of this.

Remarks: There are several reasons why the $\#$ function and sharply bounded quantifiers are natural choices for inclusion in the language of bounded arithmetic:

- The $\#$ function has the right growth rate for polynomial time computation.
- The above theorem defines the Σ, Π classes of the polynomial hierarchy syntactically (without use of computation),
- The presence of $\#$ in the language gives precisely the right growth rate so that the following *Quantifier Exchange Principle* holds:

$$\begin{aligned}(\forall x \leq |a|)(\exists y \leq b)A(x,y) \leftrightarrow \\ \leftrightarrow (\exists y \leq (2a+1)\#(4(2b+1)^2))(\forall x \leq |a|) \\ [A(x,\beta(x+1,y)) \wedge \beta(x+1,y) \leq b]\end{aligned}$$

- The original use of the # function was by E. Nelson [38]; his primary reason for its introduction was that the growth rate of # allows a smooth treatment of sequence coding and of the metamathematics of substitution. Wilkie and Paris [50] independently introduced the axiom Ω_1 ("$x^{\log x}$ is total") for similar reasons, since it gives similar growth rate.

The original use of polynomially bounded quantifiers was by Bennett [1]; they were first defined in form given above by [3, 4].

2.2 Induction Axioms for Bounded Arithmetic

The *IND* axioms are the usual induction axioms. The *PIND* and *LIND* axioms are "polynomial" and "length" induction axioms that are intended to be feasibly effective forms of induction.

Definition Let $i \geq 0$. The following are axiom schemes often used for theories of bounded arithmetic.

Σ^b_k**-IND**: $A(0) \wedge (\forall x)(A(x) \supset A(x+1)) \supset (\forall x)A(x)$ for $A \in \Sigma^b_k$.

Σ^b_k**-PIND**: $A(0) \wedge (\forall x)(A(\lfloor \frac{1}{2}x \rfloor) \supset A(x)) \supset (\forall x)A(x)$ for $A \in \Sigma^b_k$.

Σ^b_k**-LIND**: $A(0) \wedge (\forall x)(A(x) \supset A(x+1)) \supset (\forall x)A(|x|)$ for $A \in \Sigma^b_k$.

The axiom schemes Σ^b_k-LIND and Σ^b_k-PIND typically are equivalent and are (strictly?) weaker than Σ^b_k-IND. Since exponentiation is not provably total in Bounded Arithmetic, the $|x|$ function is not provably surjective; therefore, the LIND axioms do not appear to equal to the IND axioms in strength.

2.3 Theories of Bounded Arithmetic

Definition Let $i \geq 0$. T^i_2 is the first-order theory with language 0, S, $+$, $\cdot$, $\lfloor \frac{1}{2}x \rfloor$, $|x|$, $\#$ and $\leq$ and axioms:

(1) A finite set, BASIC, of (universal closures of) open axioms defining simple properties of the function and relation symbols. BASIC properly contains Robinson's Q since it has to be used with weaker induction axioms.

(2) The Σ^b_i-IND axioms.

T^{-1}_2 has no induction axioms. T_2 is the union of the T^i_2's.

T_2 is equivalent to $I\Delta_0 + \Omega_1$ (see Parikh [40] and Wilkie and Paris [50]) modulo differences in the nonlogical language.

Definition Let $i \geq 0$. S^i_2 is the first-order theory with language 0, S, $+$, $\cdot$, $\lfloor \frac{1}{2}x \rfloor$, $|x|$, $\#$ and $\leq$ and axioms:

(1) The BASIC axioms, and

(2) The Σ_i^b-PIND axioms.

$S_2^{-1} = T_2^{-1}$ has no induction axioms. S_2 is the union of the S_2^i's.

Remark: The theory S_2^1, which we will relate closely to polynomial computability, is defined by PIND on NP properties (in light of Theorem 2).

The following, somewhat surprising, relationship holds between the hierarchy of theories S_2^i and the hierarchy of theories T_2^i.

Theorem 3 *(Buss [3, 4]). Let $i \geq 1$. $T_2^i \vdash S_2^i$ and $S_2^i \vdash T_2^{i-1}$. So $S_2 \equiv T_2$.*

Open Question: Are the following inclusions proper?

$$S_2^1 \subseteq T_2^1 \subseteq S_2^2 \subseteq T_2^2 \subseteq \cdots$$

2.4 Provably Recursive and Σ_i^b-Definable Functions

We know come to one of the centrally important definitions for describing the proof-theoretic complexity of bounded arithmetic:

Definition Let $f: \mathbf{N}^k \mapsto \mathbf{N}$. The function f is *Σ_i^b-definable* by a theory R iff there is a formula $A(\vec{x}, y) \in \Sigma_i^b$ so that

(1) For all $\vec{n} \in \mathbf{N}^k$, $A(\vec{n}, f(\vec{n}))$ is true.

(2) $R \vdash (\forall \vec{x})(\exists y) A(\vec{x}, y)$

(3) $R \vdash (\forall \vec{x}, y, z)(A(\vec{x}, y) \wedge A(\vec{x}, z) \supset y = z)$

When a function is Σ_1^b-definable by a theory R, then we also say that the function is *provably recursive* in R. By "provably recursive" we are intending that there should be a Turing machine M which computes the function so that R can prove the Turing machine always halts within polynomial time. By a "bootstrapping procedure" it is possible to show that, for a given Turing machine M and a given polynomial time bound $p(n)$, the statement that M always halts within time $p(|x|)$ can be expressed in the form $(\forall x)(A_{M,p}(x))$ where $A_{M,p}$ is a Σ_1^b-formula. (Actually, this fact is immediate from Theorem 2, so no further bootstrapping is needed.) On the other hand, any Σ_1^b-definable function is certainly Turing computable; it can be computed by a (non-polynomial time) brute-force search if nothing else. Therefore, it makes sense to identify "provably recursive" and "Σ_1^b-definable".

The concept of Σ_i^b-definability applies to *functions* only. The analogue for predicates is the notion of Δ_i^b-definability:

Definition Let $Q \subseteq \mathbf{N}$. Q is Δ_i^b-definable by a theory R iff there is a Σ_i^b-formula A and Π_i^b-formula B that define Q so that A and B are provably equivalent in R. A formula is Δ_i^b with respect to R iff it is provably equivalent to a Σ_i^b- and to a Π_i^b-formula.

2.5 Bootstrapping Theorems

Theorem 4 *[3, 4] Every polynomial time function is Σ_1^b-definable by S_2^1 and every polynomial time predicate is Δ_1^b-definable by S_2^1.*

The above theorem shows that S_2^1, and the Σ_i^b-definable functions and Δ_1^b-definable predicates are sufficiently strong to introduce polynomial time properties. We will omit the proof of this theorem here: for details, the reader can refer to Buss [3, 4], some improvements to the bootstrapping can be found in Buss-Ignjatović [13] and a proof outline can be found in [7]. Alternative approaches to bootstrapping in different settings are given by Wilkie-Paris [50] and in Hájek-Pudlák [25].

A large part of the importance of Σ_1^b-definable functions and Δ_1^b-predicates comes from the following theorem:

Theorem 5 *[3, 4] Let $i \geq 1$. Any Σ_1^b-definable function or Δ_1^b-definable predicate of S_2^i may be introduced into the nonlogical language and used freely in induction axioms. The same holds for T_2^i in place of S_2^i.*

Proof (Sketch) Suppose f is Σ_1^b-defined by R, so that

$$R \vdash (\forall x)(\exists! y \leq r(\vec{x}))A_f(\vec{x}, y).$$

Then any atomic formula $\varphi(f(\vec{s}))$ is equivalent to both

$$(\exists y \leq r(\vec{s}))(A_f(\vec{s}, y) \wedge \varphi(y))$$

and

$$(\forall y \leq r(\vec{s}))(A_f(\vec{s}, y) \supset \varphi(y)).$$

Note that the first formula is in Σ_i^b and the second is in Π_i^b. Thus, for i odd, any Σ_i^b-formula involving f is equivalent to one not involving f by transforming atomic subformulas as above using the first equivalent formula for positively occurring subformulas and the second equivalent formula for negatively occurring subformulas. For i even, the roles of positive and negative occurrences are reversed. (We have omitted some details from this proof, since it is also necessary to remove f from terms in quantifier bounds; this is possible because of the known bound $r(\vec{x})$ on the value of $f(\vec{x})$.) $\square$

2.6 Main Theorems for S_2^i

The so-called "main theorems" for the theories S_2^i give an exact characterization of the Σ_i^b-definable functions of S_2^i in terms of the computational complexity.

Theorem 6 *(Buss [3, 4]) Let $i \geq 1$. Let A be a Σ_i^b-formula. Suppose $S_2^i \vdash (\forall \vec{x})(\exists y)A(\vec{x}, y)$. Then there is a Σ_i^b-formula B and a function $f \in \Box_i^p$ and a term t so that*

(1) $S_2^i \vdash (\forall \vec{x}, y)(B(\vec{x}, y) \supset A(\vec{x}, y))$.

(2) $S_2^i \vdash (\forall \vec{x})(\exists! y)B(\vec{x}, y)$.

(3) $S_2^i \vdash (\forall \vec{x})(\exists y \leq t)B(\vec{x}, y)$. *(see Parikh [40])*

(4) *For all $\vec{n}$, $\mathbf{N} \models B(\vec{n}, f(\vec{n}))$.*

Theorem 7 *If $f \in \Box_i^p$ then there is a formula $B \in \Sigma_i^b$ and a term t so that (2), (3) and (4) hold.*

Corollary 8 *($i \geq 1$) The Σ_i^b-definable functions of S_2^i are precisely the functions in $\Box_i^p$.*

The most interesting case of the above theorems and corollary is probably the $i = 1$ case. For this, we have:

Corollary 9 *The Σ_1^b-definable functions of S_2^1 are precisely the polynomial time functions.*

It is this corollary that allows us to state that the proof-theoretic strength of S_2^1 corresponds to polynomial time computability.

The above theorems characterize the Σ_i^b-definable functions of S_2^i. These can be restated to characterize the Δ_i^b-definable predicates of S_2^i.

Definition A predicate $Q(\vec{x})$ is *Δ_i^b-definable* in a theory T provided it is defined by a Σ_i^b-formula $A(\vec{x})$ and a Π_i^b-formula $B(\vec{x})$ such that T proves $(\forall \vec{x})(A(\vec{x}) \leftrightarrow B(\vec{x}))$.

Theorem 10 *($i \geq 1$). Suppose $A(\vec{x}) \in \Sigma_i^b$ and $B(\vec{x}) \in \Pi_i^b$ and $S_2^i \vdash A \leftrightarrow B$. Then there is a predicate $Q \in \Delta_i^p$ so that, for all $\vec{n}$,*

$$Q(\vec{n}) \quad \Leftrightarrow \quad \mathbf{N} \models A(\vec{n}) \quad \Leftrightarrow \quad \mathbf{N} \models B(\vec{n})$$

Conversely, if $Q \in \Delta_i^p$ then there are A and B so that the above holds.

In other words, the Δ_i^b-definable predicates of S_2^i are precisely the Δ_i^p-predicates.

The most interesting case of the last theorem is again probably the $i = 1$ case. For this, we have:

Corollary 11 *If A is a formula which is S_2^1-provably in $NP \cap \mathrm{coNP}$ then A defines a polynomial time predicate (provably in S_2^1). Being provably in $NP \cap \mathrm{coNP}$ means provably equivalent to a Σ_1^b- and to a Π_1^b-formula.*

We shall sketch a proof of the main theorem below; but first, we need to take a diversion into the sequent calculus. The sequent calculus and the cut elimination theorem will be one of the main tools in our proof of the main theorems.

3 The Sequent Calculus

3.1 Gentzen's Sequent Calculus

To prove the Main Theorem, we shall formalize S_2^i in Gentzen's sequent calculus.

We shall work in first-order logic with $\wedge$, $\vee$, $\neg$, $\supset$, $\forall$, $\exists$ the logical symbols of our first language. In addition, there is one further symbol, $\longrightarrow$, which is the *sequent connective*. The sequent connective does not occur in first-order formulas per se, but instead is used to mark the middle of a sequent:

Definition A *sequent* is an expression of the form

$$A_1, A_2, \ldots, A_n \longrightarrow B_1, B_2, \ldots, B_k$$

where the A_i's and B_i's are formulas. Its intended meaning is

$$(A_1 \wedge A_2 \wedge \cdots \wedge A_n) \supset (B_1 \vee B_2 \vee \cdots \vee B_k)$$

We shall use Greek letters $\Gamma, \Delta, \ldots$ to denote finite sequences of formulas separated by commas. A sequent is thus denoted $\Gamma \longrightarrow \Delta$. The cedents Γ and Δ are called the *antecedent* and *succedent* of the sequent (respectively).

Gentzen's sequent calculus is a proof system where lines in the proofs are sequents. The sequent calculus is generally denoted LK, from "Logische Kalkul".

Definition An *LK-proof* is a tree of sequents: the leaves or *initial sequents* must be of the form $A \longrightarrow A$; the root, or *endsequent*, is what is proved; and the valid inferences are:

$$\frac{\Gamma \longrightarrow \Delta, A}{\neg A, \Gamma \longrightarrow , \Delta} \qquad \frac{A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg A}$$

$$\frac{A, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A \qquad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B}$$

$$\frac{B, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A}{\Gamma \longrightarrow \Delta, A \vee B}$$

$$\frac{A, \Gamma \longrightarrow \Delta \qquad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \vee B}$$

$$\frac{\Gamma \longrightarrow \Delta, A \qquad B, \Gamma \longrightarrow \Delta}{A \supset B, \Gamma \longrightarrow \Delta} \qquad \frac{A, \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \supset B}$$

$$\frac{A(b), \Gamma \longrightarrow \Delta}{(\exists x)A(x), \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A(t)}{\Gamma \longrightarrow \Delta, (\exists x)A(x)}$$

$$\frac{A(t), \Gamma \longrightarrow \Delta}{(\forall x)A(x), \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A(b)}{\Gamma \longrightarrow \Delta, (\forall x)A(x)}$$

In the quantifier inferences, the free variable b is called the *eigenvariable* and must not appear in the lower sequent.

$$\frac{\Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, A}$$

$$\frac{\Gamma, A, B, \Pi \longrightarrow \Delta}{\Gamma, B, A, \Pi \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A, B, \Lambda}{\Gamma \longrightarrow \Delta, B, A, \Lambda}$$

$$\frac{A, A, \Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A, A}{\Gamma \longrightarrow \Delta, A}$$

Cut: $$\frac{\Gamma \longrightarrow \Delta, A \qquad A, \Pi \longrightarrow \Lambda}{\Gamma, \Pi \longrightarrow \Delta, \Lambda}$$

The system LK, as defined above, gives a sound and complete proof system for first-order logic. It is probably the most elegant way of formulating first-order logic; and a primary factor in its elegance is the following fundamental theorem:

Theorem 12 *(Gentzen [23])*

- *LK is complete.*
- *LK without the Cut inference is complete.*

In particular, if P is an LK-proof of $\Gamma \longrightarrow \Delta$ then there is a cut-free proof P^* of $\Gamma \longrightarrow \Delta$. There is an effective (but not feasible) procedure to obtain P^* from P.

Examination of the rules of inference for LK reveals that, with the exception of the cut rule, every inference has the *subformula property* that the formulas in the hypotheses of the inference are subformulas of formulas in the conclusion (the lower sequent) of the inference. More precisely, every formula in the upper sequents is obtained from a subformula of a formula in the lower sequent, possibly after substitution of a term for a (formerly) bound variable. In other words, every formula in an upper sequent is a subformula *in the wide sense* of some formula in the lower sequent.

Thus, if there are no cuts, the logical complexity of formulas in the proof will be at most the logical complexity of formulas in the endsequent $\Gamma \longrightarrow \Delta$. We therefore say that cut-free sequent calculus proofs enjoy the *subformula property*.

3.2 Sequent Calculus Formulations of Bounded Arithmetic

We now wish to formulate a sequent calculus version of the theories of bounded arithmetic in a way that preserves the essence of the this subformula property. We there for enlarge LK as follows:

(1) Allow equality axioms and BASIC axioms as initial sequents. An initial sequent will contain only atomic formulas.

(2) Add inferences for bounded quantifiers (the variable b occurs only as indicated):

$$\frac{b \leq s, A(b), \Gamma \longrightarrow \Delta}{(\exists x \leq s)A(x), \Gamma \longrightarrow \Delta} \qquad \frac{\Gamma \longrightarrow \Delta, A(t)}{t \leq s, \Gamma \longrightarrow \Delta, (\exists x \leq s)A(x)}$$

$$\frac{A(t), \Gamma \longrightarrow \Delta}{t \leq s, (\forall x \leq s)A(x), \Gamma \longrightarrow \Delta} \qquad \frac{b \leq s, \Gamma \longrightarrow \Delta, A(b)}{\Gamma \longrightarrow \Delta, (\forall x \leq s)A(x)}$$

(3) Allow induction inferences: (for $A \in \Sigma_i^b$)

$$\Sigma_i^b\text{-IND} \qquad \frac{A(b), \Gamma \longrightarrow \Delta, A(b+1)}{A(0), \Gamma \longrightarrow \Delta, A(t)}$$

$$\Sigma_i^b\text{-PIND} \qquad \frac{A(\lfloor \frac{1}{2} b \rfloor), \Gamma \longrightarrow \Delta, A(b)}{A(0), \Gamma \longrightarrow \Delta, A(t)}$$

Definition S_2^i and T_2^i are formulated as sequent calculus systems with BASIC axioms as initial sequents and with Σ_i^b-PIND and Σ_i^b-IND inference rules, respectively. With side formulas, the induction inferences are equivalent to the induction axioms.

3.3 Free-Cut Elimination

Definition A cut inference

$$\frac{\Gamma \longrightarrow \Delta, A \qquad A, \Pi \longrightarrow \Lambda}{\Gamma, \Pi \longrightarrow \Delta, \Lambda}$$

is *free* unless A is the direct descendant either of a formula in an initial sequent or of a principal formula of an induction inference.

The next theorem is due to Gentzen and Takeuti (for fragments of Peano arithmetic).

Theorem 13 FREE-CUT ELIMINATION THEOREM *If P is an S^i_2-proof (or T^i_2-proof) then there is a proof P^* in the same theory with the same endsequent which contains no free cuts.*

In a free-cut free proof, every formula will be a subformula (in the wide sense) of an induction formula, of a formula in an axiom or of a formula in the conclusion.

In S^i_2 and T^i_2, cut formulas may be restricted to be Σ^b_i-formulas. Therefore, if $\Gamma \longrightarrow \Delta$ is a sequent of Σ^b_i-formulas which is a consequence of S^i_2 or T^i_2, then $\Gamma \longrightarrow \Delta$ has an S^i_2-proof or T^i_2-proof (respectively) such that every formula appearing in the proof is in Σ^b_i.

3.4 Outline of Proof of the Main Theorem

There are two steps in the proof of Theorem 6

Step 1: By assumption, there is an S^i_2-proof P of

$$\longrightarrow (\exists y) A(\vec{c}, y).$$

Therefore, by free-cut elimination, there is an S^i_2 proof P^* of

$$\longrightarrow (\exists y \leq t) A(\vec{c}, y)$$

such that every formula in P^* is a Σ^b_i-formula.

Step 2: Given the proof P^* we will extract an algorithm to compute a function $f(\vec{c})$ such that $A(\vec{n}, f(\vec{n}))$ is true for all n. The function f will be in $\Box^p_i$ and will be Σ^b_i-defined by S^i_2. Furthermore, S^i_2 will prove $(\forall \vec{x}) A(\vec{x}, f(\vec{x}))$. Thus, P^* can be thought of as a program plus a proof that it is correct.

Step 1 is immediate from the free-cut elimination theorem. Before, we can carry out Step 2, we need to introduce the *Witness* predicate.

3.5 The Witness Predicate

Definition Fix $i \geq 1$. Let $B(\vec{a})$ be a Σ^b_i-formula with all free variables indicated. Then $Witness^{i,\vec{a}}_B(w, \vec{a})$ is a formula defined inductively by:

(1) If $B \in \Sigma^b_{i-1} \cup \Pi^b_{i-1}$ then $Witness^{i,\vec{a}}_B(w, \vec{a}) \Leftrightarrow B(\vec{a})$.

(2) If $B = C \vee D$ then

$$Witness^{i,\vec{a}}_B(w, \vec{a}) \Leftrightarrow Witness^{i,\vec{a}}_C(\beta(1, w), \vec{a}) \vee Witness^{i,\vec{a}}_D(\beta(2, w), \vec{a}).$$

(3) If $B = C \wedge D$ then

$$Witness^{i,\vec{a}}_B(w, \vec{a}) \Leftrightarrow Witness^{i,\vec{a}}_C(\beta(1, w), \vec{a}) \wedge Witness^{i,\vec{a}}_D(\beta(2, w), \vec{a}).$$

(4) If $B = (\exists x \leq t)C(\vec{a}, x)$ then

$$Witness_B^{i,\vec{a}}(w, \vec{a}) \Leftrightarrow \beta(1, w) \leq t \wedge Witness_{C(\vec{a},b)}^{i,\vec{a},b}(\beta(2, w), \vec{a}, \beta(1, w)).$$

(5) If $B = (\forall x \leq |t|)C(\vec{a}, x)$ then

$$Witness_B^{i,\vec{a}}(w, \vec{a}) \Leftrightarrow (\forall x \leq |t|)\, Witness_{C(\vec{a},b)}^{i,\vec{a},b}(\beta(x + 1, w), \vec{a}, x).$$

(6) If $B = \neg C$ use prenex operations to push the negation sign inside.

Lemma 14 *Fix $i \geq 1$. Let $B \in \Sigma_i^b$.*

(1) *For some term t_B, S_2^i proves*

$$B(\vec{a}) \leftrightarrow (\exists w \leq t_B)\, Witness_B^{i,\vec{a}}(w, \vec{a}).$$

(2) *$Witness_B^{i,\vec{a}} \in \Delta_i^p$ $(= P$ if $i = 1)$.*

(3) *$Witness_B^{i,\vec{a}}$ is Δ_i^b with respect to S_2^i.*

Proof: This is easily proved by induction on the complexity of B.

3.6 The Main Lemma

Lemma 15 *Suppose $S_2^i \vdash \Gamma \longrightarrow \Delta$ where Γ and Δ contain only Σ_i^b-formulas. Let $\vec{c}$ be the free variables in Γ and Δ. Then, there is a function f such that*

(a) *f is Σ_i^b-defined by S_2^i*

(b) *$S_2^i \vdash Witness_{\bigwedge \Gamma}^{i,\vec{c}}(w, \vec{c}) \supset Witness_{\bigvee \Delta}^{i,\vec{c}}(f(w, \vec{c}), \vec{c})$*

(c) *$f \in \Pi_i^p$ $(= FP$ if $i = 1)$*

Proof The proof is by induction on the number of inferences in a free-cut free S_2^i-proof of $\Gamma \longrightarrow \Delta$. As an example of one case of the proof of the main lemma, suppose that P is a free-cut free proof and ends with the inference

$$\frac{B(\lfloor \frac{1}{2}a \rfloor) \longrightarrow B(a)}{B(0) \longrightarrow B(t)}$$

By the induction hypothesis, there is a function g so that

(1) g is Σ_i^b-defined by S_2^i

(2) g is in Π_i^p $(= FP$ if $i = 1)$

(3) $S_2^i \vdash Witness_{B(\lfloor \frac{1}{2}a \rfloor)}^{i,a,\vec{c}}(w, a, \vec{c}) \supset Witness_{B(a)}^{i,a,\vec{c}}(g(w, a, \vec{c}), a, \vec{c})$.

(4) $S_2^i \vdash (\forall a, \vec{c})[g(w, a, \vec{c}) \leq t_B(a, \vec{c})]$

Now define f by limited iteration as

$$\begin{aligned} f(w, 0, \vec{c}) &= g(w, 0, \vec{c}) \\ f(w, a, \vec{c}) &= g(f(w, \lfloor \tfrac{1}{2} a \rfloor, \vec{c}), a, \vec{c}) \end{aligned}$$

so $f(w, a, \vec{c}) \leq t_B(a, \vec{c})$ and the following hold:

(1) $f \in \Box_i^p \quad (= FP \text{ if } i = 1)$
Pf: Since f is defined by limited iteration from g.

(2) S_2^i can Σ_i^b-define f and prove that f satisfies the above conditions

(3) $S_2^i \vdash \mathit{Witness}^{i,a,\vec{c}}_{B(0,\vec{c})}(w, a, \vec{c}) \supset \mathit{Witness}^{i,a,\vec{c}}_{B(a,\vec{c})}(f(w, a, \vec{c}), a, \vec{c})$.

Pf: Since $\mathit{Witness}^{i,b,\vec{c}}_{B(a,\vec{c})}$ is a Σ_i^b-formula, S_2^i can prove this by Σ_i^b-PIND directly from the induction hypothesis.

Q.E.D. Main Lemma 15

3.7 Conclusion of Proof of Main Theorem

We can now prove the Main Theorem 6 for S_2^i from the above Lemma 15.

Proof Suppose $S_2^i \vdash (\forall \vec{x})(\exists y) A(\vec{x}, y)$. By a theorem of Parikh, there is a term t so that S_2^i proves $\longrightarrow (\exists y \leq t) A(\vec{c}, y)$. By the Main Lemma,

$$S_2^i \vdash \mathit{Witness}^{i,\vec{c}}_{(\exists y \leq t) A}(g(\vec{c}), \vec{c})$$

for some Σ_i^b-defined function g. Define $B(\vec{c}, y)$ to be the formula

$$y = \beta(1, g(\vec{c})).$$

Since g is Σ_i^b-defined by S_2^i, B is a Σ_i^b-formula and by the properties of *Witness*,

$$S_2^i \vdash (\forall \vec{x}, y)(B(\vec{x}, y) \supset A(\vec{x}, y))$$

Finally, define $f(\vec{c}) = \beta(1, g(\vec{c}))$.
Q.E.D. Main Theorem 6

4 Other Systems of Bounded Arithmetic

4.1 Alternative Axioms for Bounded Arithmetic

Let Ψ be a set of formulas. The axioms below are schemes where $A \in \Psi$:

Ψ-MIN: (Minimization)

$$(\exists x)A(x) \supset (\exists x)[A(x) \wedge (\forall y < x)(\neg A(y))]$$

Ψ-LMIN: (Length minimization)

$$(\exists x)A(x) \supset A(0) \vee (\exists x)[A(x) \wedge (\forall y \leq \lfloor \tfrac{1}{2}x \rfloor)(\neg A(y))]$$

Ψ-replacement:

$$(\forall x \leq |t|)(\exists y \leq s)A(x,y) \leftrightarrow$$
$$\leftrightarrow (\exists w \leq SqBd(t,s))(\forall x \leq |t|)(A(x,\beta(Sx,w)) \wedge \beta(Sx,w) \leq s)$$

strong Ψ-replacement:

$$(\exists w \leq SqBd(t,s))(\forall x \leq |t|)$$
$$[(\exists y \leq s)A(x,y) \leftrightarrow A(x,\beta(Sx,w)) \wedge \beta(Sx,w) \leq s]$$

The figure below shows the known relationships between fragments of bounded arithmetic: (for $i \geq 1$, relative to S_2^1)

Σ_i^b-IND $\Leftrightarrow$ Π_i^b-IND $\Leftrightarrow$ Σ_i^b-MIN $\Leftrightarrow$ Δ_{i+1}^b-IND

$\Downarrow$

Σ_i^b-PIND $\Leftrightarrow$ Π_i^b-PIND $\Leftrightarrow$ Σ_i^b-LIND $\Leftrightarrow$ Π_i^b-LIND

$\Updownarrow$

Σ_i^b-LMIN $\Leftrightarrow$ strong Σ_i^b-replacement

$\Downarrow$ $\qquad\qquad$ $\Updownarrow$

Σ_{i-1}^b-IND $\qquad\qquad$ $(\Sigma_{i+1}^b \cap \Pi_{i+1}^b)$-PIND

Σ_{i+1}^b-MIN $\Leftrightarrow$ Π_i^b-MIN

Σ_{i+1}^b-replacement $\Rightarrow$ Σ_i^b-PIND $\Rightarrow$ Σ_i^b-replacement

$S_2^{i+1} \underset{\Sigma_{i+1}^b}{\succ} T_2^i$

$S_2^{i+1} \underset{\mathcal{B}(\Sigma_{i+1}^b)}{\succ} T_2^i + \Sigma_{i+1}^b$-replacement

Due to space limitations, we shall only sketch proofs of two of the facts pictured in the above figure.

Theorem 16 *[3, 4]* $S_2^1 + \Sigma_i^b\text{-}PIND \vdash \Delta_i^b$-IND. *Hence* $S_2^i \supset T_2^{i-1}$.

Proof Suppose A is Δ_i^b w.r.t. S_2^i. Assume $(\forall x)(A(x) \supset A(x+1))$ and argue inside S_2^i. Let $B(x,z)$ be the formula

$$(\forall w \leq x)(\forall y \leq z+1)(A(w \dot{-} y) \supset A(w)).$$

So B is equivalent to a Π_i^b-formula. Now by definition of B, $(\forall x,z)(B(x,\lfloor \frac{1}{2}z \rfloor) \supset B(x,z))$ and hence by Π_i^b-PIND on $B(x,z)$ w.r.t. z,

$$(\forall x)(B(x,0) \supset B(x,x)).$$

By the assumption, $(\forall x)B(x,0)$; hence $(\forall x)B(x,x)$, from whence

$$(\forall x)(A(0) \supset A(x)) \qquad \square$$

The second result is concerns the conservation results between T_2^i and S_2^{i+1}:

Theorem 17 *[8] Fix $i \geq 1$. S_2^{i+1} is conservative over T_2^i with respect to Σ_{i+1}^b-formulas, and hence with respect to $\forall\exists\Sigma_{i+1}^b$-sentences.*

This means that any $\forall\exists\Sigma_{i+1}^b$-formula which is S_2^{i+1}-provable is also T_2^i-provable.

Proof (Idea). Fix $i \geq 1$ and let Z be PV or T_2^{i-1} as appropriate. First show that every $\Box_i^p$-function is definable in Z in an appropriate sense. For $i = 1$, there is a function symbol for every polynomial time function; for $i > 1$, we show that every $\Box_i^p$-function can be "Q_i-defined" — this is stronger than "Σ_i^b-defined". Second, prove a stronger version of the Main Lemma above; in essence, we partially formalize the Main Lemma in Z and prove that the witnessing function f is defined appropriately in Z. Namely, we prove that if $S_2^i \vdash A$ with $A \in \Sigma_i^b$ then $Z \vdash A$.

We omit the rest of the proof details.

A nice consequence of the previous theorem and of the Main Theorem for S_2^i is:

Theorem 18 *[8] The Σ_{i+1}^b-definable theories of T_2^i are precisely the $\Box_{i+1}^p$-functions.*

4.2 Witnessing Theorem for T_2^1

The class *Polynomial Local Search*, PLS, was defined by Papadimitriou [39] to capture a common kind of search problem. Two representative examples of PLS problems are (1) linear programming and (2) simulated annealing. Of course, it is known that there are polynomial time algorithms for linear programming. On the other hand, there is no polynomial time algorithm which is known to find even a local optimum solution based on simulated annealing. So, it is open whether PLS is in P.

Definition A *Polynomial Local Search (PLS)*, problem is specified by two polynomial time functions N, c and a polynomial time predicate F which satisfy the following conditions:

(1) $c(s,x)$ is a *cost function*,

(2) $N(s,x)$ is a neighborhood function, such that for all s such that $F(s,x)$ holds, we have

$$c(N(s,x),x) \leq c(s,x) \qquad \text{and} \qquad F(N(s,x),x).$$

(3) $\{s : F(s,x)\}$ is the solution space for input x. $F(0,x)$ always holds; and if $F(s,x)$, then $|s| < p(|x|)$ for p some polynomial.

A solution to the PLS problem is a (multivalued) function f, s.t., for all x,

$$c(N(f(x),s),s) = c(f(x),x) \qquad \text{and} \qquad F(f(x),x).$$

Theorem 19 *[14] Suppose T_2^1 proves $(\forall x)(\exists y)A(x,y)$ where $A \in \Sigma_1^b$. Then there is a PLS function $f(x) = y$ and a polynomial time function π such that*

$$T_2^1 \vdash (\forall x)A(x, \pi \circ f(x)).$$

Furthermore, every PLS function (and every function $\pi \circ f$) is Σ_1^b-definable by T_2^1.

Corollary 20 *The same holds for S_2^2 by conservativity of S_2^2 over T_2^1.*

Proof-idea: A free-cut free T_2^1-proof can be transformed into a PLS problem. □

4.3 Herbrand's Theorem and the KPT Witnessing Theorem

The following theorem is a version of Herbrand's Theorem from Herbrand's dissertation [27, 30, 28]. A proof of (a strengthening of) this version of Herbrand's theorem can be found in [10].

Theorem 21 *Let T be a theory axiomatized by purely universal axioms. Let $A(x,y,z)$ be quantifier-free. Suppose T proves*

$$(\forall x)(\exists y)(\forall z)A(x,y,z).$$

Then there is an integer $k > 0$ and there are terms $t_1(x)$, $t_2(x,z_1)$, $t_3(x,z_1,z_2), \ldots, t_k(x,z_1,\ldots,z_{k-1})$ so that:

$$\begin{aligned} T \vdash\ (\forall x)[&(\forall z_1)[A(x,t_1(x),z_1) \vee \\ &(\forall z_2)[A(x,t_2(x,z_1),z_2) \vee \\ &(\forall z_3)[A(x,t_3(x,z_1,z_2),z_3) \vee \\ &\quad \vdots \\ &(\forall z_k)[A(x,t_k(x,z_1,\ldots,z_{k-1}),z_k)] \cdots]]]. \end{aligned}$$

The KPT Witnessing Theorem (Theorem 22 below) applies this theorem to the theories T_2^i. In order to do this, however, we must re-axiomatize T_2^i to have purely universal axioms. In order to give a set of purely universal axioms, we must enlarge the language of T_2^i by adding function symbols all Π_{i+1}^p-functions. Of course, we want to have only a conservative extension of T_2^i when we enlarge the language: By Theorem 18, T_2^i can already Σ_{i+1}^b-define all Π_{i+1}^b functions; therefore, we can add symbols for these functions to the language of T_2^i and add the defining equations for these new function symbols and still have a conservative extension of T_2^i.

Towards this end, we define:

Definition Let $i \geq 1$. The theory $PV_i = T_2^i(\Pi_{i+1}^p)$ is defined to be the theory T_2^i with language enlarged to include function symbols for all Π_{i+1}^p-functions. In addition, PV_i includes the axioms that define these Π_{i+1}^p-functions.

Using these new function symbols as Skolem functions, any Σ_i^b-formula of T_2^i is equivalent to a quantifier-free formula. The induction axioms

$$A(0) \wedge (\forall x)(A(x) \supset A(x+1)) \supset (\forall x)A(x)$$

of T_2^i can be replaced by

$$A(0) \wedge \neg A(c) \supset A(f_A(c)) \wedge \neg A(f_A(c)+1)$$

where f_A is the Π_{i+1}^p-function such that

$$f_A(c) \;=\; \begin{cases} \text{least } x < c \text{ such that } A(x) \text{ and } \neg A(x+1) \\ 0 \qquad \text{if no such } x < c \text{ exists} \end{cases}$$

It therefore be shown that PV_i can be axiomatized by purely universal formulas.

It is also possible to define PV_1 as being the natural first-order, purely universal theory which has function symbols from all polynomial time functions (this is the same as the theory PV of Cook [16], except extended to first-order logic).

Using PV_i in place of T_2^i, and applying Herbrand's theorem to PV_i, one obtains the following witnessing theorem.

Theorem 22 *(Krajíček-Pudlák-Takeuti [35]) Suppose $A \in \Sigma_{i+2}^b$ and T_2^i proves $(\forall x)(\exists y)(\forall z)A(x,y,z)$. Then there are $k > 0$ and functions $f_i(x, z_1, ..., z_{i-1})$ so that*

(1) *Each f_i is Σ_{i+1}^b-defined by T_2^i.*

(2) T_2^i *proves*

$$\begin{aligned}(\forall x)[(\forall z_1)[&A(x, f_1(x), z_1) \vee \\ &(\forall z_2)[A(x, f_2(x, z_1), z_2) \vee \\ &(\forall z_3)[A(x, f_3(x, z_1, z_2), z_3) \vee \\ &\quad \vdots \\ &(\forall z_k)[A(x, f_k(x, z_1, \ldots, z_{k-1}), z_k)] \cdots]]].\end{aligned}$$

4.4 Collapse of the Polynomial Hierarchy

As mentioned earlier, it is open whether the hierarchy of theories of bounded arithmetic is proper. Because of the close relationship between the fragments of bounded arithmetic and the levels of the polynomial time hierarchy, it is natural to try to find a connection between the questions of whether these two hierarchies are proper. This is answered by the following theorem, which shows that the hierarchy of theories of bounded arithmetic collapses if and only if bounded arithmetic is able to prove that the polynomial time hierarchy is proper.

Theorem 23 *(Krajíček-Pudlák-Takeuti [35], Buss [11], Zambella [52]) If $T_2^i = S_2^{i+1}$, then the polynomial time hierarchy collapses, provably in T_2^i. In fact, in this case, T_2^i proves that every Σ_{i+3}^p predicate is (a) equivalent to a Boolean combination of Σ_{i+2}^p-predicates and (b) is in $\Sigma_{i+1}^p/poly$.*

Proof (Idea) For simplicity, assume $i = 0$. Suppose $T_2^0(PV) = S_2^1$. Let $\overline{\varphi}$ represent a vector of Boolean formula $\overline{\varphi} = \langle \varphi_1, \ldots, \varphi_n \rangle$. Then $T_2^0(PV)$ proves

$$\begin{aligned}\forall \overline{\varphi} (\exists \ell \leq n)(\exists \langle w_1, \ldots, w_\ell \rangle) \\ [(\forall j \leq \ell)(w_j \text{ satisfies } \varphi_j) \wedge \text{“}\ell = n \text{ or } \varphi_{\ell+1} \text{ is unsatisfiable”}]\end{aligned}$$

The formula in $[\cdots]$ is in Π_1^b, so the KPT witnessing theorem can be applied to get $k > 0$ and polynomial time functions $f_1, \ldots, f_k$ so that $T_2^0(PV)$ proves (setting $n = k$) that given $\varphi_1, \ldots, \varphi_k$ satisfied by $w_1, \ldots w_k$, that one of $f_j(\overline{\varphi}, w_1, \ldots, w_{j-1})$ produces a witness to φ_j. [Note that f_j has all φ_i's as input.]

Let $PreAdvice(a, \langle \varphi_{\ell+1}, \ldots, \varphi_k \rangle)$ mean that for all $\varphi_1, \ldots, \varphi_\ell < a$ satisfied by $w_1, \ldots, w_\ell$, $f_j(\overline{\varphi}, w_1, \ldots, w_{j-1})$ satisfies φ_j for some $j \leq \ell$. Let $Advice(a, \langle \varphi_{\ell+1}, \ldots, \varphi_k \rangle)$ mean that $PreAdvice$ holds, and that ℓ is the minimum possible value for which there is such *PreAdvice*.

Claim $T_2^0(PV)$ proves, that if $\varphi_\ell < a$ and if $Advice(a, \langle \varphi_{\ell+1}, \ldots, \varphi_k \rangle)$, then φ_ℓ is satisfiable if and only if for all $\varphi_1, \ldots, \varphi_\ell$, satisfied by $w_1, \ldots, w_\ell$, there is $j \leq \ell$ such that $f_j(\overline{\varphi}, w_1, \ldots, w_{j-1})$ satisfies φ_j.

Proof of claim: If the latter condition is true, then the only way for $\langle \varphi_\ell, \ldots, \varphi_k \rangle$ to not be "preadvice", (which it isn't, by definition of "advice") is for φ_ℓ to be satisfied by $f_\ell(\overline{\varphi}, \vec{w})$ for some $\varphi_1, \ldots, \varphi_{\ell-1}$, $w_1, \ldots, w_{\ell-1}$. $\square$

Note that this means that the NP complete property of satisfiability is in coNP relative to the polynomial size advice, $\langle \varphi_\ell, \ldots, \varphi_k \rangle$.

The above shows that $T^0_2(PV)$ would prove NP $\subseteq$ coNP/*poly*. From this, Karp-Lipton style methods can show that $T^0_2(PV)$ proves the polynomial time hierarchy collapses. In fact it can be shown that $T^0_2(PV)$ proves that every polynomial time hierarchy predicate is equivalent to Boolean combination of Σ^p_2 predicates. The proof idea is that the property $PreAdvice$ is in coNP and therefore, property

$$PAlen(\ell) \equiv \exists \langle \varphi_{\ell+1}, \ldots, \varphi_k \rangle PreAdvice(a, \langle \vec{\varphi} \rangle)$$

is a Σ^p_2-property.

Similar methods work for $i \geq 1$. Q.E.D.

5 Lengths of Propositional Proofs

5.1 Frege and Extended Frege Systems

Definition *Propositional Formulas* are formed with logical connectives $\wedge$, $\vee$, $\neg$ and $\supset$, variables $p_1, p_2, \ldots$, and parentheses.

Cook's Theorem: *$P = NP$ if and only if there is a polynomial time algorithm for determining if a propositional formula is valid.*

Frege systems are the usual "textbook" proof systems for propositional logic:

Definition A *Frege* ($\mathcal{F}$) proof system is a usual proof system for propositional logic with a finite set of axiom schemes and with only the modus ponens rule. $\mathcal{F}$ is sound and complete.

Open Question: Does every tautology have a polynomial size $\mathcal{F}$-proof?

If the answer to this question is "Yes", then NP=coNP. This is since the set of tautologies is coNP complete and having a polynomial size $\mathcal{F}$-proof is an NP property.

The size of Frege proofs is measured in terms of the number of symbols occurring in the proof. Of course, the number of symbols may be large either because there are a lot of steps in the proof, or because the formulas which occur in the proof are large. In the latter case, there will generally be many repeated subformulas; this motivates the definition of "extended Frege proof systems" where repeated subformulas may be abbreviated by new symbols.

Definition The extended Frege ($e\mathcal{F}$) proof system is a Frege proof system plus the *extension rule*:

Extension Rule: whenever q is a variable which has not been used in the proof so far and does not appear in the final line of the proof or in φ then we may infer

$$q \leftrightarrow \varphi.$$

This allows us to use q as abbreviation for φ. By iterating uses of extension rule the extension rule can apparently make proofs logarithmically smaller by reducing the formula size.

Tseĭtin [49] first used the extension rule, for resolution proofs. See also Statman [47] and Cook-Reckhow [18, 19]. Statman proved that the number of symbols in an extended Frege proof can be polynomially bounded in terms on the number of steps in the proof.

Theorem 24 *Reckhow [45] The choice of axiom schemas or of logical language does not affect the lengths of $\mathcal{F}$- or $e\mathcal{F}$-proofs by more than a polynomial amount.*

5.2 Abstract Proof Systems

The following definition of propositional proof system is due to Cook [16]:

Definition A *propositional proof system* is a polynomial time function f with range equal to the set of all valid formulas.

An (extended) Frege proof system can be viewed as a propositional proof system by letting $f(w)$ equal the last line of w if w is a valid (e)$\mathcal{F}$-proof. Similarly, any theory (e.g. set theory) can be viewed as a propositional proof system. For instance, ZF serves a proof system by letting $f(w)$ be defined so that if w codes a ZF-proof that φ is a tautology, then $f(w)$ equals (the code for) φ.

Theorem 25 *(Cook [16])* NP = coNP *if and only if there is a proof system f for which tautologies have polynomial size proofs.*

To prove this theorem, note that if f is such a proof system, then then a formula is a tautology iff it has a proof of polynomial length. Therefore, the coNP complete property of being a tautology would be in NP, so NP would equal coNP. Conversely, an NP algorithm can be turned into a proof system, by letting its accepting computations serve as proofs.

Definition A proof system f in which all tautologies have polynomial size proof is called *super*.

It is open whether there exist super proof systems.

Definition Let S and T be proof systems (with the same propositional language). S *simulates* T iff there is a polynomial p so that for any T-proof of size n there is an S-proof of the same formula of size $\leq p(n)$. S *p-simulates* T iff the S-proof is obtainable as a polynomial time function of the T-proof.

Open Question: Does $\mathcal{F}$ simulate $e\mathcal{F}$?

This open question is related to the question of whether Boolean circuits have equivalent polynomial size formulas. By Ladner [36] and Buss [5] this is a non-uniform version of the open question of whether P is equal to alternating logarithmic time (ALOGTIME).

Open Question: Is there a *maximal* proof system which simulates all other propositional proof systems?

Krajíček and Pudlák [33] have shown that if NEXP (non-deterministic exponential time) is closed under complements then the answer is "Yes".

5.3 The Propositional Pigeonhole Principle

We now introduce tautologies that express the pigeonhole principle.

Definition The propositional pigeonhole principle PHP_n is the formula

$$\bigwedge_{0\leq i\leq n}\bigvee_{0\leq j<n} p_{i,j} \quad \supset \quad \bigvee_{0\leq i<m\leq n}\bigvee_{0\leq j<n} (p_{i,j}\wedge p_{m,j})$$

states that $n+1$ pigeons can't fit singly into n holes. $p_{i,j}$ means "pigeon i is in hole j".

Theorem 26 *(Cook-Reckhow [18, 19]) There are polynomial size $e\mathcal{F}$-proofs of PHP_n.*

Theorem 27 *([6]) There are polynomial size $\mathcal{F}$-proofs of PHP_n.*

Theorem 28 *(Haken [26]) The shortest resolution proofs of PHP_n are of exponential size.*

Cook and Reckhow had proposed PHP_n as an example for showing that $\mathcal{F}$ could not simulate $e\mathcal{F}$. However, Theorem 27 implies this is not the case. Presently there are no very good candidates of combinatorial principles that might separate $\mathcal{F}$ from $e\mathcal{F}$; the paper [2] describes some attempts to find such combinatorial principles.

5.4 PHP_n has Polysize $e\mathcal{F}$-Proofs

The pigeonhole principle provides a nice example of how extended Frege ($e\mathcal{F}$) proofs can grow exponentially larger when translated straightforwardly into Frege ($\mathcal{F}$) proofs. Although it is known (Theorem 27) that PHP has polynomial size Frege proofs, the proofs are rather different than the extended Frege proofs below.

We first give an intuitive, conceptual version of a proof of the pigeonhole principle and then describe how it is formalized as an extended Frege proof. Let $[n] = \{0, 1, 2, \ldots, n\}$.

Conceptual Version: (by contradiction)

Given $f\colon [n] \stackrel{1-1}{\longmapsto} [n-1]$

define $f_k\colon [k] \stackrel{1-1}{\longmapsto} [k-1]$

as $f_n(i) = f(i)$,

$$f_k(i) = \begin{cases} f_{k+1}(i) & \text{if } f_{k+1}(i) < k \\ f_{k+1}(k+1) & \text{otherwise.} \end{cases}$$

For $k = 1$, $f_1 : [1] \stackrel{1-1}{\longmapsto} [0]$ — contradiction.

The $e\mathcal{F}$-proof: Uses $q^k_{i,j}$ for "$f_k(i) = j$".

$$\begin{aligned} q^n_{i,j} &\leftrightarrow p_{i,j} \\ q^k_{i,j} &\leftrightarrow q^{k+1}_{i,j} \vee (q^{k+1}_{i,k} \wedge q^{k+1}_{k+1,j}) \end{aligned}$$

Then prove for $k = n, n-1, \ldots, 1$ that $q^k_{i,j}$'s code a one-to-one function from $[k]$ to $[k-1]$. For $k = 1$, we have "f_1 is total and one-to-one":

$$q^1_{0,0} \wedge q^1_{1,0} \wedge \neg(q^1_{0,0} \wedge q^1_{1,0})$$

which is impossible. □

It a useful exercise to see what happens when the above polynomial size extended Frege proofs are translated into Frege proofs by unwinding the definitions of the abbreviations $q^n_{i,j}$. Since $q^k_{i,j}$ abbreviates a formula involving three occurrences of variables $q^{k+1}_{i,j}$, the variable $q^1_{i,j}$ abbreviates a formula containing 3^{n-1} occurrences of symbols.

6 Translations from Bounded Arithmetic into Propositional Logic

There are two important translations of proofs in fragments of bounded arithmetic into proofs in propositional logic. The first, due to Cook [16], is a translation from PV-proofs (or more-or-less equivalently, from S^1_2-proofs) into polynomial size extended Frege proofs. The second, due to Paris and Wilkie [41], is a translation from proofs in the theory $I\Delta_0$ or $I\Delta_0 + \Omega - 1$ or $S_2 = T_2$ into constant depth, polynomial size Frege proofs.

6.1 S^1_2 and Polysize $e\mathcal{F}$ Proofs

PV is an equational theory of polynomial time functions; above, we discussed $PV_1 = T^0_2(\,\square^p_1)$ which is the conservative first-order extension of PV. PV was first introduced by Cook [16], who showed that there is an intimate translation between PV-proofs and polynomial size $e\mathcal{F}$-proofs. Namely, he showed that if $A(x)$ is a polynomial time equation provable in PV, then there is a family of tautologies $[\![A]\!]^n$ such that

(1) $[\![A]\!]^n$ is a polynomial size propositional formula,

(2) $[\![A]\!]^n$ says that $A(x)$ is true whenever $|x| \leq n$,

(3) $[\![A]\!]^n$ has polynomial size $e\mathcal{F}$-proofs.

(Some generalizations of these results have been proved by Dowd [21] for PSPACE and Krajíček and Pudlak [34] for various of the theories of bounded arithmetic.)

In these notes, we shall prove the version of Cook's theorem for S^1_2 and Π^b_2-formulas A. (Our proof below follows the version in [12]). This version is completely analogous to Cook's original theorem, in view of the $\forall\Sigma^b_1$-conservativity of S^1_2 over PV_1.

Definition Let $t(\vec{a})$ be a term. The *bounding polynomial* of t is a polynomial $q_t(\vec{n})$ such that

$$(\forall \vec{x})(|t(\vec{x})| \leq q_t(\max\{|\vec{x}|\})).$$

The inductive definition is:

$$\begin{aligned} q_0(n) &= 1 \\ q_a(n) &= n \quad \text{for } a \text{ a variable} \\ q_{S(t)}(n) &= q_t(n) + 1 \\ q_{s+t}(n) &= q_s(n) + q_t(n) \\ q_{s\cdot t}(n) &= q_s(n) + q_t(n) \\ q_{s\#t}(n) &= q_s(n) \cdot q_t(n) + 1 \\ q_{|t|}(n) &= q_{\lfloor \frac{1}{2}t \rfloor}(n) = q_t(n) \end{aligned}$$

Definition Let $A(\vec{a})$ be a bounded formula. The *bounding polynomial* of A is a polynomial $q_A(\vec{a})$ so that if $|a_i| \leq n$ for all a_i in $\vec{a}$, then $A(\vec{a})$ refers only to numbers of length $\leq q_A(n)$. The polynomial q_A is inductively defined by:

(1) $q_{s\leq t} = q_{s=t} = q_s + q_t$

(2) $q_{\neg A} = q_A$

(3) $q_{A\wedge B} = q_{A\vee B} = q_{A\supset B} = q_A + q_B$

(4) $q_{(Qx\leq t)A} = q_t(n) + q_A(n + q_t(n))$

Next we define $[\![t]\!]_m$ to be a vector of polynomial size formulas that define (compute) the term t for values of length $\leq m$. For this it is useful to think of formulas as being circuits of fanout 1.

Definition Let $[\![+]\!]_m$ be a polynomial size, fanout 1 circuit which accepts $2m$ binary inputs and outputs m binary signals; $[\![+]\!]_m$ computes the bitwise sum of two m-bit integers (and discards any overflow). Likewise define $[\![\cdot]\!]_m$, $[\![\#]\!]_m$, $[\![\lfloor \frac{1}{2}x \rfloor]\!]_m$, etc.

Definition Let $t(\vec{a})$ be a term and $m \geq q_t(n)$. $[\![t]\!]^n_m$ is a vector of m propositional formulas defining the lower m bits of the value of $t(\vec{a})$ when $|a_i| \leq n$.

For b a free variable in t, a propositional variable v^b_i represents the i-th bit of b's value.

(1) $[\![0]\!]^n_m$ is a sequence of m false formulas (for example $p \wedge \neg p$).

(2) For b a variable, $[\![b]\!]^n_m$ is a sequence of $m-n$ false formulas followed by $v^b_{n-1}, \ldots, v^b_0$.

(3) $[\![s+t]\!]^n_m$ is $[\![+]\!]_m([\![s]\!]^n_m, [\![t]\!]^n_m)$ (the formulas corresponding to the circuit for addition applied to the outputs of $[\![s]\!]^n_m$ and $[\![t]\!]^n_m$).

(4) And similarly for other cases.

Note that $[\![t]\!]^n_m$ is a polynomial size formula (in m and n).

For $A \in \Pi^b_2$, we define below a propositional formula $[\![A]\!]^n_m$ for $m \geq q_A(n)$. If B is a formula, we assign new 'existential' variables ϵ^B_i and new 'universal' variables μ^B_i to B $(i \geq 0)$. Different occurrences of B will generally get assigned different such variables.

Definition EQ_m is a circuit for equality:

$$EQ_m(\vec{p}, \vec{q}) \text{ is } \bigwedge_{k=0}^{m-1} (p_k \leftrightarrow q_k).$$

$LE_m(\vec{p}, \vec{q})$ is a circuit for $\leq$:

$$EQ_m(\vec{p}, \vec{q}) \vee \bigvee_{0 \leq i < m} \left(q_i \wedge \neg p_i \wedge \bigwedge_{i<j<m} (q_i \leftrightarrow p_i) \right).$$

Definition A is in negation-implication normal form (NINF) iff all negation signs are applied to atomic subformulas and there are no implications in A.

Definition Assume $A \in \Pi^b_2$ and A is in NINF and $m \geq q_A(n)$. Define $[\![A]\!]^n_m$ inductively by:

(1) $[\![s = t]\!]^n_m$ is $EQ_m([\![s]\!]^n_m, [\![t]\!]^n_m)$

(2) $[\![s \leq t]\!]^n_m$ is $LE_m([\![s]\!]^n_m, [\![t]\!]^n_m)$

(3) $[\![\neg A]\!]^n_m$ is $\neg[\![A]\!]^n_m$ for A atomic.

(4) $[\![A \wedge B]\!]^n_m$ is $[\![A]\!]^n_m \wedge [\![B]\!]^n_m$

(5) $[\![A \vee B]\!]^n_m$ is $[\![A]\!]^n_m \vee [\![B]\!]^n_m$

(6) $[\![(\exists x \leq t)A(x)]\!]^n_m$ is $[\![x \leq t \wedge A(x)]\!]^n_m(\{\varepsilon^A_i / v^x_i\}^{n-1}_{i=0})$

(7) $[\![(\forall x \leq t)A(x)]\!]^n_m$ is $[\![\neg x \leq t \vee A(x)]\!]^n_m(\{\mu^A_i / v^x_i\}^{n-1}_{i=0})$

(8) $[\![(\forall x \leq |t|)A(x)]\!]^n_m$ is $\bigwedge_{k=0}^{m-1} [\![\neg \underline{k} \leq |t| \vee A(\underline{k})]\!]^n_m$ Note that $|t| \leq m$ (by our assumption on m).

(9) $[\![(\exists x \leq |t|)A(x)]\!]^n_m$ is $\bigvee_{k=0}^{m-1} [\![\underline{k} \leq |t| \wedge A(\underline{k})]\!]^n_m$

Proposition 29 *The formula $[\![A]\!]^n_m$ is equivalent to A in that $A(\vec{a})$ is true ($|a_i| \leq n$) iff for all truth assignments to the universal variables in $[\![A]\!]^n_m$ there is an assignment to the existential variables which satisfies $[\![A]\!]^n_m$.*

We can extend the definition of $[\![A]\!]$ in the obvious way to formulas not in NINF, by using prenex operations to transform A into NINF form.

Definition An $e\mathcal{F}$-proof of $[\![A]\!]^n_m$ is defined like an ordinary $e\mathcal{F}$-proof except now we additionally allow the existential variables (but not the other variables) in $[\![A]\!]^n_m$ to be defined by the extension rule (each existential variable may be defined only once).

Theorem 30 *(essentially Cook [16]). If $A \in \Pi^b_2$ and $S^1_2 \vdash (\forall \vec{x})A(\vec{x})$ then there are polynomial size (in n) $e\mathcal{F}$-proofs of $[\![A]\!]^n_{q_A(n)}$. These $e\mathcal{F}$-proofs are obtainable in polynomial time.*

Proof If $\Gamma \longrightarrow \Delta$ is provable in S^1_2, we prove the theorem for

$$[\![\neg\Gamma \vee \Delta]\!].$$

By free-cut elimination it will suffice to do it for $\Gamma \subset \Sigma^b_1$ and $\Delta \subset \Pi^b_2$. We proceed by induction on the number of inferences in a free-cut free proof.

Case (1): A logical axiom $B \longrightarrow B$. Obviously

$$\left[\!\left[\neg B \vee B\right]\!\right] \;=\; \neg\left[\!\left[B\right]\!\right] \vee \left[\!\left[B\right]\!\right]$$

has a polynomial size $e\mathcal{F}$-proof.
Case (2): A BASIC axiom. For example,

$$\left[\!\left[(x+y)+z = x+(y+z)\right]\!\right]^n_{3n}$$

has straightforward polynomial size $\mathcal{F}$-proofs using techniques from [6] that allow formalization of addition and multiplication in Frege proofs.

Case (3) The proof ends with a contraction:

$$\frac{\Gamma \longrightarrow \Delta, B, B}{\Gamma \longrightarrow \Delta, B}$$

Recall that all three B's are assigned different existential and universal variables. The induction hypothesis says there are polynomial size $e\mathcal{F}$-proofs of

$$\left[\!\left[\neg\Gamma \vee \Delta \vee B \vee B\right]\!\right].$$

Modify these proofs by (1) identifying the universal variables for different B's; (2) at the end of the proof use extension to define

$$\epsilon''_j \;\leftrightarrow\; (\left[\!\left[B\right]\!\right](\vec{\epsilon}) \wedge \epsilon_j) \vee (\neg\left[\!\left[B\right]\!\right](\vec{\epsilon}) \wedge \epsilon'_j)$$

where $\vec{\epsilon}''$ are the existential variables for the lower B and the others are the existential variables for the upper B's; and (3) then extend to a proof of

$$\left[\!\left[\neg\Gamma\right]\!\right] \vee \left[\!\left[\Delta\right]\!\right] \vee \left[\!\left[B\right]\!\right](\vec{\epsilon}'').$$

Case (4) The proof ends with a Cut:

$$\frac{\Gamma \longrightarrow \Delta, B \qquad B, \Pi \longrightarrow \Lambda}{\Gamma, \Pi \longrightarrow \Delta, \Lambda}$$

By free cut elimination, $B \in \Sigma^b_1$; so B has existential variables $\vec{\epsilon}$ and $\neg B$ has universal variables $\vec{\mu}$. By induction hypothesis, there are polynomial size $e\mathcal{F}$-proofs of

$$\left[\!\left[\neg\Gamma\right]\!\right] \vee \left[\!\left[\Delta\right]\!\right] \vee \left[\!\left[B\right]\!\right](\vec{\epsilon})$$

and

$$\left[\!\left[\neg\Pi\right]\!\right] \vee \left[\!\left[\Lambda\right]\!\right] \vee \left[\!\left[\neg B\right]\!\right](\vec{\mu}).$$

The polynomial size $e\mathcal{F}$-proof of

$$\left[\!\left[\neg\Gamma \vee \neg\Pi \vee \Delta \vee \Lambda\right]\!\right]$$

consists of the first proof above followed by the second proof except with the $\vec{\mu}$'s changed to $\vec{\epsilon}$'s followed by a (simulated) cut.

Case (5) For Σ_1^b-PIND inferences, iterate the construction for Cut and contractions.

Case (6) If the proof ends with:

$$\frac{\Gamma \longrightarrow \Delta, A(t)}{t \leq s, \Gamma \longrightarrow \Delta, (\exists x \leq s)A(x)}$$

Let $\vec{\epsilon}$ be the existential variables for $(\exists x \leq s)A$. The desired $e\mathcal{F}$-proof contains:

(a) Extension: $\vec{\epsilon} \leftrightarrow [\![t]\!]$.

(b) The proof from the induction hypothesis of

$$[\![\neg\Gamma \vee \Delta \vee A(t)]\!].$$

(c) A further derivation of

$$[\![\neg t \leq s \vee \neg\Gamma \vee \Delta \vee (t \leq s \wedge A(t))]\!].$$

(d) A derivation of

$$[\![\neg t \leq s \vee \neg\Gamma \vee \Delta \vee (\exists x \leq s)A(x)]\!]$$

by changing some $[\![t]\!]$'s to ϵ's. □

6.2 Consequences of the S_2^1 and $e\mathcal{F}$ Correspondence

Theorems 31-33 are due to Cook [16], although he stated them for PV instead of for S_2^1.

Theorem 31 *Let $G \supseteq \mathcal{F}$ be a propositional proof system. If $S_2^1 \vdash Con(G)$ then $e\mathcal{F}$ p-simulates G.*

Theorem 32 *If $S_2^1 \vdash NP{=}coNP$ then $e\mathcal{F}$ is super.*

Theorem 33 *$e\mathcal{F}$ has polynomial size proofs of the propositional formulas $Con_{e\mathcal{F}}(n)$ which assert that there is no $e\mathcal{F}$-proof of $p \wedge \neg p$ of length $\leq n$.*

Theorem 34 *[9]. $\mathcal{F}$ has polynomial size proofs of the self-consistency formulas $Con_{\mathcal{F}}(n)$.*

Proof of Theorem 31 from Theorem 33: (Idea) Suppose there is a G proof P of a tautology φ. A polynomial size $e\mathcal{F}$ proof of φ is constructed as follows: Let $\vec{p}$ be the free variables in $\varphi(\vec{p})$. Reason inside $e\mathcal{F}$. First show that if $\neg\varphi$ then there is an $\mathcal{F}$-proof P_1 of $\neg\varphi(\underline{\vec{p}})$ where $\underline{\vec{p}}$ denotes a vector of $\top$'s and $\bot$'s: the truth values of $\vec{p}$. By substituting $\underline{\vec{p}}$ for $\vec{p}$ in P and combining this with P_1, we construct a G-proof P_2 of a contradiction. This proof has size polynomial in $|P|$ since P_1 has size polynomial in $|\varphi| \leq |P|$.

By Theorem 33 there is a polynomial size $e\mathcal{F}$-proof of $Con_G(|P_2|)$ so the assumption that $\neg\varphi$ is impossible; i.e., φ is true. □

Proof of Theorem 33: $S_2^1 \vdash Con(e\mathcal{F})$. □

Proof of Theorem 34: The $\mathcal{F}$-self-consistency proof is a "brute-force" proof that truth is preserved by axioms and modus ponens using the fact that the Boolean formula value problem is in ALOGTIME. □

Definition A *substitution Frege* $s\mathcal{F}$ proof system is a Frege proof system plus the substitution rule:

$$\frac{\psi(p)}{\psi(\varphi)}$$

for ψ, φ arbitrary formulas, all occurrences of p substituted for.

Theorem 35 *(Cook-Reckhow [19], Dowd [22], Krajíček and Pudlák [33]) $s\mathcal{F}$ and $e\mathcal{F}$ p-simulate each other.*

Proof (Idea) $s\mathcal{F}$ p-simulates $e\mathcal{F}$ is not hard to show directly. $e\mathcal{F}$ p-simulates $s\mathcal{F}$ since $S_2^1 \vdash Con(s\mathcal{F})$. □

6.3 Constant Depth Frege Proofs

Let propositional formulas use connectives $\wedge$ and $\vee$ with negations only on variables. The *depth* of a formula is the maximum number of blocks (alternations) of $\wedge$'s and $\vee$'s on any branch of the formula, viewed as a tree. The *depth* of a Frege proof is the maximum depth of formulas occurring in the proof.

Theorem 36 *Constant-depth Frege systems are complete (for constant depth tautologies).*

Proof By the cut-elimination theorem. □

6.4 Translation from $I\Delta_0/S_2$ into Constant Depth Frege

Paris and Wilkie [41] developed the following translation between provability in $I\Delta_0$ (or $I\Delta_0 + \Omega_1$) and the lengths of constant depth Frege proofs. The theory $I\Delta_0 + \Omega_1$ is essentially identical to the theory $S_2 = T_2$ — the only difference is the choice of first-order language: $I\Delta_0$ uses the first-order language containing the symbols $0, S, +, \cdot$ and $\leq$. Actually, we shall work with $I\Delta_0(\alpha, f)$ or $S_2(\alpha, f)$ where α and/or f are allowed to be new predicate or function symbols (resp.) which may be used in induction axioms.

Definition We translate closed (=variable-free) arithmetic formulas A into propositional formulas A^{PW}: this is defined inductively as follows.

(1) $(\alpha(t))^{PW}$ is the formula q_i, where i is the numeric value of the variable-free term t.

(2) $(f(t) = s)^{PW}$ is the formula $p_{i,j}$, where i and j are the numeric values of t and s. Without loss of generality, f occurs only in this context.

(3) For other atomic formulas, $P(\vec{t})^{PW}$ is defined to be either the constant $\top$ or the constant $\bot$.

(4) Boolean connectives are translated without any change. E.g., $(A \wedge B)^{PW}$ is $A^{PW} \wedge B^{PW}$.

(5) $[(\forall x \leq t)A(x)]^{PW}$ is $\bigwedge_{i=0}^{value(t)} [A(\underline{i})]^{PW}$.

(6) $[(\exists x \leq t)A(x)]^{PW}$ is $\bigvee_{i=0}^{value(t)} [A(\underline{i})]^{PW}$.

Theorem 37 *(Paris-Wilkie [41]) Suppose $I\Delta_0(\alpha, f)$ proves $(\forall x)A(x)$. Then the formulas $\{A(n)^{PW} : n \geq 0\}$ are tautologies and have polynomial size, constant-depth Frege proofs.*

Proof (Idea) Given an $I\Delta_0(\alpha, f)$ proof $P(x)$ of $A(x)$ and given $n \geq 0$, replace x everywhere with $\underline{n}$, to get a proof $P(n)$ of $A(\underline{n})$. W.l.o.g., $P(x)$ is free-cut free, so has only bounded formulas. Replace every formula B in $P(n)$ with its translation B^{PW}. Thus every sequent $\Gamma \longrightarrow \Delta$ in $P(n)$ becomes a propositional sequent $\Gamma^{PW} \longrightarrow \Delta^{PW}$.

(a) *Size of new formulas.* A simple size analysis gives that there is a constant c such that for every formula $A \in P(n)$, the formula A^{PW} as at most n^c many symbols. This is since every term $t(n)$ is bounded by n^c and there are finitely many formulas A in $P(n)$.

(b) *Size of propositional proofs* of $\Gamma^{PW} \longrightarrow \Delta^{PW}$ is likewise bounded by n^d for some constant d. To prove this, consider how the propositional proof is

obtained from the proof $P(n)$: the general idea is to work from the bottom of the proof upwards, always considering sequents in $P(n)$ with values assigned to all the free variables.

(b.i) A $\exists \leq$:right inference in $P(n)$:

$$\frac{\Gamma \longrightarrow \Delta, B(s)}{s \leq t, \Gamma \longrightarrow \Delta, (\exists x \leq t) B(x)}$$

If $s \leq t$, the propositional translation of this is:

$$\dfrac{\dfrac{\Gamma^{PW} \longrightarrow \Delta^{PW}, B(s)^{PW}}{\Gamma^{PW} \longrightarrow \Delta^{PW}, \bigvee_{i=0}^{t} B(i)^{PW}} \vee\text{:right's}}{\top, \Gamma^{PW} \longrightarrow \Delta^{PW}, \bigvee_{i=0}^{t} B(i)^{PW}}$$

(b.ii) A $\forall \leq$:right inference in $P(n)$:

$$\frac{a \leq t, \Gamma \longrightarrow \Delta, B(a)}{\Gamma \longrightarrow \Delta, (\forall x \leq t) B(x)}$$

has propositional translation:

$$\dfrac{\dfrac{\{\top, \Gamma^{PW} \longrightarrow \Delta^{PW}, B(i)^{PW}\}_{i=0}^{t}}{\Gamma^{PW} \longrightarrow \Delta^{PW}, \bigwedge_{i=0}^{t} B(i)^{PW}} \wedge\text{:right's}}{}$$

(b.iii) A induction inference in $P(n)$:

$$\frac{\Gamma, B(a) \longrightarrow B(a+1), \Delta}{\Gamma, B(0) \longrightarrow B(t), \Delta}$$

has propositional translation

$$\dfrac{\{\Gamma^{PW}, B(i)^{PW} \longrightarrow B(i+1)^{PW}, \Delta^{PW}\}_{i=0}^{t-1}}{\Gamma^{PW}, B(0)^{PW} \longrightarrow B(t)^{PW}, \Delta^{PW}}\ \text{Cuts}$$

Other inferences are handled similarly. Since the proof $P(n)$ has constant size, and since the values of terms are $\leq n^{\alpha}$, for some constant α, the size bound is proved. $\square$

When Ω_1 is present as an axiom, then the function $x \mapsto x^{\log x}$ is total. In this case, an argument similar to the above establishes the following theorem:

Theorem 38 *(Paris-Wilkie [41]) Suppose $I\Delta_0(\alpha, f) + \Omega_1$ proves $(\forall x)A(x)$. Then the formulas $\{A(n)^{PW} : n \geq 0\}$ are tautologies and have quasi-polynomial size, constant-depth Frege proofs.*

The above two theorems can be sharpened in the case of S_2^i- and T_2^i-proofs. First, at the cost of adding a finite set polynomial time functions such as the Gödel β function, we may assume that every formula in $\Sigma_i^b(\alpha, f)$ or $\Pi_i^b(\alpha, f)$ consists of exactly i bounded quantifiers, then a sharply bounded quantifier and then a Boolean combination of atomic formulas of the form $\alpha(t)$ or $f(t) = s$ or which do not use α or f. [Basically, because of the quantifier exchange property and by contracting like quantifiers.] With this convention, then if $A \in \Sigma_i^b$ or $A \in \Pi_i^b$ then the translation A^{PW} is a depth $i + 1$ propositional formula where the bottom depth has polylogarithmic fanin. This gives the following theorem:

Theorem 39 *Suppose $T_2^i \vdash \Gamma \longrightarrow \Delta$, sequent of $\Sigma_i^b \cup \Pi_i^b$ formulas. Then the sequents $\Gamma^{PW} \longrightarrow \Delta^{PW}$ have polynomial size propositional sequent calculus proofs of depth $i + 1$ in which every formula has polylogarithmic fanin at the bottom level.*

Furthermore, there is a constant c such that every sequent in the propositional proof has at most c formulas.

If every formula in the T_2^i-proof is in Π_i^b, then every formula in the propositional proofs starts with a (topmost) block of $\bigwedge$'s.

Proof Analogous to the above proof. We leave the details to the reader.

7 Interpolation Theorems for Propositional Logic

7.1 Craig's Theorem

The interpolation theorem is one of the fundamental theorems of mathematical logic; this was first proved in the setting of first-order logic by Craig [20]. For propositional logic, the interpolation is much simpler:

Theorem 40 *Let $A(\vec{p}, \vec{q})$ and $B(\vec{p}, \vec{r})$ be propositional formulas involving only the indicated variables. Suppose $A(\vec{p}, \vec{q}) \supset B(\vec{p}, \vec{r})$ is a tautology. Then there is a propositional formula $C(\vec{p})$ using only the common variables, so that $A \supset C$ and $C \supset B$ are tautologies.*

Proof Since $A(\vec{p}, \vec{q}) \models B(\vec{p}, \vec{r})$; if we have already assigned truth values to $\vec{p} = p_1, \ldots, p_k$, then it is not possible to extend this to a truth assignment on $\vec{p}, \vec{q}, \vec{r}$ such that both $A(\vec{p}, \vec{q})$ and $\neg B(\vec{p}, \vec{r})$ hold.

Let $\tau_1, \ldots, \tau_n$ be the truth assignments to $p_1, \ldots, p_k$ for which it is possible to make $A(\vec{p}, \vec{q})$ true by further assignment of truth values to $\vec{q}$.

Let $C(\vec{p})$ say that one of $\tau_1, \ldots, \tau_n$ holds for $\vec{p}$, i.e.,

$$C \;=\; \bigvee_{i=1}^{n} \left(p_1^{(i)} \wedge p_2^{(i)} \wedge \ldots \wedge p_k^{(i)} \right)$$

where

$$p_j^{(i)} \;=\; \begin{cases} p_j & \text{if } \tau_i(p_j) = \mathit{True} \\ \neg p_j & \text{otherwise} \end{cases}$$

Then clearly, $A(\vec{p}, \vec{q}) \models C(\vec{p})$. Also, by the comment at the beginning of the proof, $C(\vec{p}) \models B(\vec{p}, \vec{r})$. $\Box$

Note that $C(\vec{p})$ may be exponentially larger than $A(\vec{p}, \vec{q})$ and $B(\vec{p}, \vec{r})$.

Example: Let $p_1, \ldots, p_k$ code the binary representation of a k-bit integer P. Let $A(\vec{p}, \vec{q})$ be a formula which is satisfiable iff P is composite (e.g. q codes two integers > 1 with product P). Let $B(\vec{p}, \vec{r})$ be a formula which is satisfiable iff P is prime (i.e., $\vec{r}$ codes a Pratt-primality witness).

$$\begin{aligned} P \text{ is prime} \;&\Leftrightarrow\; \exists \vec{r} B(\vec{p}, \vec{r}) \\ &\Leftrightarrow\; \neg \exists \vec{q} A(\vec{p}, \vec{q}). \end{aligned}$$

and $A(\vec{p}, \vec{q}) \supset \neg B(\vec{p}, \vec{r})$ is a tautology.

An interpolant $C(\vec{p})$ must express "$\vec{p}$ codes a composite".

Generalizing the above example gives:

Theorem 41 *(Mundici [37] If there is a polynomial upper bound on the circuit size of interpolants in propositional logic, then*

$$NP/poly \cap \text{coNP}/poly = P/poly$$

Proof Let $\exists \vec{q} A(\vec{p}, \vec{q})$ express an $NP/poly$ property $R(\vec{p})$ and $\forall \vec{r} B(\vec{p}, \vec{r})$ express $R(\vec{p})$ in coNP/*poly* form. Then

$$\exists \vec{q} A(\vec{p}, \vec{q}) \models \forall \vec{r} B(\vec{p}, \vec{r}),$$

which is equivalent to

$$A(\vec{p}, \vec{q}) \supset B(\vec{p}, \vec{r})$$

being a tautology. Let $C(\vec{p})$ be a polynomial size interpolant s.t.,

$$A(\vec{p}, \vec{q}) \supset C(\vec{p}) \quad \text{and} \quad C(\vec{p}) \supset B(\vec{p}, \vec{r})$$

are tautologies. Thus

$$\exists \vec{q} A(\vec{p}, \vec{q}) \models C(\vec{p}) \models \forall \vec{r} B(\vec{p}, \vec{r}),$$

I.e., $R(\vec{p}) \Leftrightarrow C(\vec{p})$ and $R(\vec{p})$ has a polynomial size circuit, so $R(\vec{p})$ is in $P/poly$. $\Box$

7.2 Cut-Free Proofs and Interpolant Size

Definition Let PK be the propositional fragment of the Gentzen sequent calculus. The *size* of a PK-proof, $|P|$, is the number of steps in P. Normally, sequent calculus proofs are treelike; however, sometime we consider non-treelike proofs, and in this case, we write $|P|_{dag}$ to denote the size of P. $V(A)$ denotes the set of free variables in A. For C a formula, $|C|$ is the number of $\wedge$'s and $\vee$'s in C.

The next theorem states that tree-like cut-free proofs have interpolants of polynomial formula size, and general cut-free proofs have interpolants of polynomial circuit size.

Theorem 42 *Let P be a cut-free PK proof of $A \longrightarrow B$, where $V(A) \subseteq \{\vec{p}, \vec{q}\}$ and $V(B) \subseteq \{\vec{p}, \vec{q}\}$. Then there is an interpolant C such that*

(1) *$A \supset C$ and $C \supset B$ are valid,*

(2) *$V(C) \subseteq \{\vec{p}\}$,*

(3) *$|C| \leq |P|$ and $|C|_{dag} \leq |P|_{dag}$.*

Remark: The theorem also holds for proofs which have cuts only on formulas D such that $V(D) \subseteq \{\vec{p}, \vec{r}\}$ or $V(D) \subseteq \{\vec{p}, \vec{r}\}$.

Proof We use induction on the number of inferences in P to prove a slightly more general statement:

Claim: If P is a proof of $\Gamma_1, \Gamma_2 \longrightarrow \Delta_1, \Delta_2$ and if $V(\Gamma_1, \Delta_1) \subseteq \{\vec{p}, \vec{q}\}$ and $V(\Gamma_2, \Delta_2) \subseteq \{\vec{p}, \vec{r}\}$, then there is an interpolant C so that

(1) $\Gamma_1 \longrightarrow \Delta_1, C$ and $C, \Gamma_2 \longrightarrow \Delta_2$ are valid,

(2) $V(C) \subseteq \{\vec{p}\}$, and

(3) The polynomial size bounds hold too.

Base Case: Initial sequent. If the initial sequent is of the form $q_i \longrightarrow q_i$, take C to be $\bot$ since

$$q_i \longrightarrow q_i, \bot \qquad \text{and} \qquad \bot \longrightarrow$$

are valid. For an initial sequent of the form $r_i \longrightarrow r_i$, take C to be $\top$. For an initial sequent $p_i \longrightarrow p_i$, C will be either $\top$, $\bot$, p_i or $(\neg p_i)$ depending on how the p_i's are split into $\Gamma_1, \Gamma_2, \Delta_1, \Delta_2$.

Induction Step: There are a number of cases, depending on the type of the last inference in the proof.

(1) For last inference an $\vee$:right:

$$\frac{\Gamma \longrightarrow \Delta, A, B}{\Gamma \longrightarrow \Delta, A \vee B}$$

the interpolant for the upper sequent still works for the lower sequent, i.e., use C such that

$$\Gamma_1 \longrightarrow \Delta_1, A, B, C \quad \text{and} \quad C, \Gamma_2 \longrightarrow \Delta_2,$$

or

$$\Gamma_1 \longrightarrow \Delta_1, C \quad \text{and} \quad C, \Gamma_2 \longrightarrow \Delta_2, A, B,$$

depending on whether $A \vee B$ is in Δ_1 or Δ_2 (respectively).

(2) For last inference an $\wedge$:right:

$$\frac{\Gamma \longrightarrow \Delta, A \qquad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B}$$

(2.a) If $A \wedge B$ is in Δ_1, apply the induction hypothesis twice to have interpolants C_A and C_B so that

$$\Gamma_1 \longrightarrow \Delta_1^-, A, C_A \qquad C_A, \Gamma_2 \longrightarrow \Delta_2$$

$$\Gamma_1 \longrightarrow \Delta_1^-, B, C_B \qquad C_B, \Gamma_2 \longrightarrow \Delta_2$$

are valid. Now the derivations

$$\frac{\dfrac{\Gamma_1 \longrightarrow \Delta_1^-, A, C_A}{\Gamma_1 \longrightarrow \Delta_1^-, A, C_A \vee C_B} \qquad \dfrac{\Gamma_1 \longrightarrow \Delta_1^-, B, C_B}{\Gamma_1 \longrightarrow \Delta_1^-, B, C_A \vee C_B}}{\Gamma_1 \longrightarrow \Delta_1^-, A \wedge B, C_A \vee C_B}$$

and

$$\frac{C_A, \Gamma_2 \longrightarrow \Delta_2 \qquad C_B, \Gamma_2 \longrightarrow \Delta_2}{C_A \vee C_B, \Gamma_2 \longrightarrow \Delta_2}$$

show $(C_A \vee C_B)$ is an interpolant.

(2b) If $A \wedge B$ is in Δ_2 applying the induction hypothesis twice gives C_A and C_B so that

$$\Gamma_1 \longrightarrow \Delta_1, C_A \qquad C_A, \Gamma_2 \longrightarrow \Delta_2^-, A$$

$$\Gamma_1 \longrightarrow \Delta_1, C_B \qquad C_B, \Gamma_2 \longrightarrow \Delta_2^-, B$$

are valid. Now the two following derivations show $(C_A \wedge C_B)$ is an interpolant:

$$\frac{\dfrac{C_A, \Gamma_2 \longrightarrow \Delta_2^-, A}{C_A \wedge C_B, \Gamma_2 \longrightarrow \Delta_2^-, A} \qquad \dfrac{C_B, \Gamma_2 \longrightarrow \Delta_2^-, B}{C_A \wedge C_B, \Gamma_2 \longrightarrow \Delta_2^-, B}}{C_A \wedge C_B, \Gamma_2 \longrightarrow \Delta_2^-, A \wedge B}$$

$$\frac{\Gamma_1 \longrightarrow \Delta_1, C_A \qquad \Gamma_1 \longrightarrow \Delta_1, C_B}{\Gamma_1 \longrightarrow \Delta_1, C_A \wedge C_B}$$

The other cases are similar and the size bounds on C are immediate. $\square$

7.3 Interpolation Theorems for Resolution

We start with a quick review of the well-known system of resolution for propositional logic.

Definition A *literal* is a propositional variable p or a negated variable $\neg p$. $\overline{p}$ is $\neg p$, and $\overline{(\neg p)}$ is p. A *clause* is a set of literals; its intended meaning is the disjunction of its members. A *set of clauses* represents the conjunction of its members. Thus a set of clauses "is" a formula in conjunctive normal form. A *resolution inference* is an inference of the form:

$$\frac{C \cup \{p\} \qquad D \cup \{\overline{p}\}}{C \cup D}$$

For such resolution inferences, we assume w.l.o.g. that $p, \overline{p} \notin C$ and $p, \overline{p} \notin D$. A *resolution refutation* of a set Γ of clauses is a derivation of the empty clause $\emptyset$ from Γ by resolution inferences.

Theorem 43 *Resolution is refutation-complete (and sound).*

Since resolution is a refutation, it does not prove implications $A \supset B$. Therefore, to formulate the interpolation theorem for resolution, we work with sets of clauses $A \cup B$. An interpolant for the sets A and B is a formula C such that $A \supset C$ and $C \supset \neg B$. This gives the following form of the interpolation theorem:

Theorem 44 *Let* $\{A_1(\vec{p}, \vec{q}), \ldots, A_k(\vec{p}, \vec{q})\}$ *and* $\{B_1(\vec{p}, \vec{r}), \ldots, B_\ell(\vec{p}, \vec{r})\}$ *be sets of clauses, so that their union* Γ *is inconsistent. Then there is a formula* $C(\vec{p})$ *such that for any truth assignment* τ*,* $domain(\tau) \supseteq \{\vec{p}, \vec{q}, \vec{r}\}$*,*

(1) *If* $\tau(C(\vec{p})) = False$*, then* $\tau(A_i(\vec{p}, \vec{q})) = False$*,* for some i*.*

(2) *If* $\tau(C(\vec{p})) = True$*, then* $\tau(B_j(\vec{p}, \vec{q})) = False$*,* for some j*.*

Proof From Γ unsatisfiable, we have

$$A_1(\vec{p}, \vec{q}), \ldots, A_k(\vec{p}, \vec{q}) \longrightarrow \neg B_1(\vec{p}, \vec{r}), \ldots, \neg B_\ell(\vec{p}, \vec{r})$$

is valid. Thus there is an interpolant $C(\vec{p})$ such that

$$A_1(\vec{p}, \vec{q}), \ldots, A_k(\vec{p}, \vec{q}) \longrightarrow C(\vec{p})$$

and

$$C(\vec{p}) \longrightarrow \neg B_1(\vec{p}, \vec{r}), \ldots, \neg B_\ell(\vec{p}, \vec{r})$$

are valid. □

The next theorem gives bounds on the size or computational complexity of the interpolant, in terms of the number of inferences in the resolution refutation.

Theorem 45 *(Krajíček [32]) Let* $\{A_i(\vec{p},\vec{q})\}_i \cup \{B_j(\vec{p},\vec{r})\}_j$ *have a refutation* R *of* n *resolution inferences. Then an interpolant,* $C(\vec{p})$*, can be chosen with* $O(n)$ *symbols in dag representation.*

If R *is tree-like, then* $C(\vec{p})$ *is a formula with* $O(n)$ *symbols.*

Proof We view R as a dag or as a tree, each node corresponding to an inference and labeled with the clause inferred at that inference. For each clause E in R, define $C_E(\vec{p})$ as follows:

(1) For $E = A_i(\vec{p},\vec{q})$, a hypothesis, set $C_E = \bot$ (*False*).

(2) For $E = B_j(\vec{p},\vec{q})$, a hypothesis, set $C_E = \top$ (*True*).

(3) For an inference $\dfrac{F \cup \{q_i\} \qquad G \cup \{\overline{q}_i\}}{F \cup G}$

set $C_{F\cup G} = C_{F\cup\{q_i\}} \vee C_{G\cup\{\overline{q}_i\}}$.

(4) For an inference $\dfrac{F \cup \{r_i\} \qquad G \cup \{\overline{r}_i\}}{F \cup G}$

set $C_{F\cup G} = C_{F\cup\{r_i\}} \wedge C_{G\cup\{\overline{r}_i\}}$.

(5) For an inference $\dfrac{F \cup \{p_i\} \qquad G \cup \{\overline{p}_i\}}{F \cup G}$

set $C_{F\cup G} = (\overline{p}_i \wedge C_{F\cup\{p_i\}}) \vee (p_i \wedge C_{G\cup\{\overline{p}_i\}})$.

Lemma 46 *For all clauses* $F \in R$*,* $C_F(\vec{p})$ *satisfies the following condition that if* τ *is a truth assignment and* $\tau(F) =$ *False, then*

(a) *if* $\tau(C_F) =$ *False, then* $\tau(A_i(\vec{p},\vec{q})) =$ *False for some* i

(b) *if* $\tau(C_F) =$ *True, then* $\tau(B_j(\vec{p},\vec{r})) =$ *False for some* j

The proof of the lemma is by induction on the definition of C_F.
Q.E.D. Lemma and Theorem.

7.4 Resolution with Limited Extension

By "extension" is meant the introduction of variables that represent complex propositional formulas. We have already seen one system that uses extension, namely the extended Frege proof system. "Limited extension" means that extension variables may be introduced to abbreviate only formulas that appear in the formula being proved.

When A is a formula, we let σ_A be the extension variable for A; in particular, for p a variable, σ_p is just the same variable p, and for other formulas A, σ_A is a new variable.

Definition When A is a formula, $LE(A)$ is a set of clauses which define the meanings of the extensions variables for all subformulas of A; to wit:

$$LE(p) = \emptyset$$

$$LE(\neg A) = LE(A) \cup \left\{ \underbrace{\{\sigma_{\neg A}, \sigma_A\}}_{\neg\sigma_A \supset \sigma_{\neg A}}, \underbrace{\{\overline{\sigma_{\neg A}}, \overline{\sigma_A}\}}_{\sigma_{\neg A} \supset \neg\sigma_A} \right\}$$

$$LE(A \wedge B) = LE(A) \cup LE(B) \cup \left\{ \underbrace{\{\overline{\sigma_{A\wedge B}}, \sigma_A\}}_{\sigma_{A\wedge B} \supset \sigma_A}, \underbrace{\{\overline{\sigma_{A\wedge B}}, \sigma_B\}}_{\sigma_{A\wedge B} \supset \sigma_B}, \underbrace{\{\sigma_{A\wedge B}, \overline{\sigma_A}, \overline{\sigma_B}\}}_{\sigma_A \wedge \sigma_B \supset \sigma_{A\wedge B}} \right\}$$

$$LE(A \vee B) = LE(A) \cup LE(B) \cup \left\{ \underbrace{\{\overline{\sigma_A}, \sigma_{A\vee B}\}}_{\sigma_A \supset \sigma_{A\vee B}}, \underbrace{\{\overline{\sigma_B}, \sigma_{A\vee B}\}}_{\sigma_B \supset \sigma_{A\vee B}}, \underbrace{\{\sigma_A, \sigma_B, \overline{\sigma_{A\vee B}}\}}_{\sigma_{A\vee B} \supset \sigma_A \vee \sigma_B} \right\}$$

Definition Let $\mathcal{A}$ be a set of formulas. Then $LE(\mathcal{A})$ is $\cup_{A\in\mathcal{A}}\{LE(A)\}$. Similarly,

$$\begin{aligned} LE(\vec{p},\vec{q}) &= \cup\{LE(A) : V(A) \subseteq \{\vec{p},\vec{q}\}\}. \\ LE(\vec{p},\vec{r}) &= \cup\{LE(A) : V(A) \subseteq \{\vec{p},\vec{r}\}\}. \end{aligned}$$

Theorem 47 *Let Γ be the set of clauses*

$$\{A_i(\vec{p},\vec{q})\}_i \cup \{B_j(\vec{p},\vec{r})\}_j \cup LE(\vec{p},\vec{q}) \cup LE(\vec{p},\vec{r})$$

and suppose Γ has a refutation R of n resolution inferences. Then there is an interpolant $C(\vec{p})$ for the sets $\{A_i(\vec{p},\vec{q})\}_i$ and $\{B_j(\vec{p},\vec{r})\}_j$ of circuit size $O(n)$.

Proof Let $C(\vec{p})$ be the interpolant for $\{A_i(\vec{p},\vec{q})\}_i \cup LE(\vec{p},\vec{q})$ and $\{B_j(\vec{p},\vec{r})\}_j \cup LE(\vec{p},\vec{r})$ given by the earlier interpolation theorem.

Claim $C(\vec{p})$ is the desired interpolant.

Proof of claim: Any truth assignment τ with domain $\{\vec{p},\vec{q}\}$ can be uniquely extended to satisfy $LE(\vec{p},\vec{q})$. Suppose $\tau(C(\vec{p})) = \mathit{False}$. Extend τ so as to satisfy $LE(\vec{p},\vec{q})$. By choice of $C(\vec{p})$, τ makes a clause from $\{A_i(\vec{p},\vec{q})\}_i \cup LE(\vec{p},\vec{q})$ false, hence makes one of the A_i's false.

A similar argument shows that if $\tau(C(\vec{p})) = \mathit{True}$, then τ falsifies some $B_j(\vec{p},\vec{r})$.
Q.E.D. Claim and Theorem.

8 Natural Proofs, Interpolation and Bounded Arithmetic

8.1 Natural Proofs

The notion of natural proofs was introduced by Razborov and Rudich [44]. In order for a proof to be a natural proof of $P \neq \mathrm{NP}$, it must give a suitably constructive way of proving that certain Boolean functions are not in P (for example, the Boolean function *Sat* which recognized satisfiable propositional formulas). More precisely, a proof is natural provided it is possible to use the proof method to find a family C_n of Boolean functions which satisfies the property of the next definition. We represent a Boolean function $f_n(x_1, \ldots, x_n)$ by its truth table (which of course has size $N = 2^n$); therefore, an n-ary Boolean functions is identified with a string of 2^n 0's and 1's.

Definition $\mathcal{C} = \{C_n\}_n$ is *quasipolynomial-time natural against P/poly* if and only if each C_n is a set of (strings representing) truth tables of n-ary Boolean functions, and such that the following hold:

Constructivity: The predicate "$f_n \in C_n$?" has circuits of size $2^{n^{O(1)}}$, and

Largeness: $|C_n| \geq 2^{-cn} \cdot 2^{2^n}$ for some $c > 0$, and

Usefulness: If $f_n \in C_n$ for all n, then the family $\{f_n\}_n$ is not in *P/poly* (i.e., does not have polynomial size circuits).

The motivation for this definition of natural proofs is that "constructive" proofs that $NP \not\subset P/poly$ ought to give (quasi)polynomial time property which is natural against *P/poly*. Note that 'quasipolynomial time', is measured as a function of the size of the truth table of f_n.

8.2 Strong Pseudo-Random Number Generator Conjecture

We now introduce the Strong Pseudo-Random Number Generator (SPRNG) Conjecture: this conjecture implies the existence of pseudorandom number generators which generate pseudorandom numbers that pass any feasible test for randomness. This conjecture is closely related to the problem of the existence of one-way functions, and is generally conjectured to be true by researchers in cryptography. (But it is stronger that $P \neq \mathrm{NP}$, and so far it has not been proven.)

Definition Let $G_n : \{0,1\}^n \to \{0,1\}^{2n}$ be a pseudo-random number generator. The *hardness*, $H(G_n)$, of G_n is the least $S > 0$ such that, for

some circuit C of size S,

$$\left| \underset{\overline{x}\in\{0,1\}^n}{Prob}[C(G_n(\overline{x}))=1] - \underset{\overline{y}\in\{0,1\}^{2n}}{Prob}[C(\overline{y})=1] \right| \geq \frac{1}{S}$$

SPRNG Conjecture: There are pseudorandom number generators G_n, computed by polynomial size circuits, with hardness $H(G_n) \geq 2^{n^\epsilon}$, for some $\epsilon > 0$.

One of the main results of Razborov and Rudich [44] is the following theorem. We omit the proof from these notes.

Theorem 48 *If the SPRNG conjecture is true, then there are no properties which are quasipolynomial time/poly natural against $P/poly$.*

The remainder of this article presents a theorem of Razborov that states, in essence, that if S_2^2 can prove certain kinds of superpolynomial lower bounds on the sizes of circuits for polynomial hierarchy predicates (e.g., that SAT is not in NP), then there is a natural proof that $P \neq NP$ which, by the previous theorem, would imply the falsity of the SPRNG conjecture. Since the SPRNG conjecture is thought to be true, then we do not expect to be able to obtain such a lower bound proof in S_2^2.

Let α be a new predicate symbol: we will use α to encode information about a Boolean circuit. Let $LB(t, S, \alpha)$ be a formula which states that the Boolean circuit encoded by α either does not correctly compute the predicate S or has size greater than t. In other words, proving $LB(t, S, \alpha)$ proves that t is a lower bound on the size of any circuit computing S. (We give a more complete definition of $LB()$ in the next section.)

Razborov [43] proved the following theorem:

Theorem 49 *Let t denote any superpolynomial function and let S be any bounded formula. If $S_2^2(\alpha)$ proves $LB(t, S, \alpha)$, then the SPRNG conjecture is false.*

Razborov's first proof of Theorem 49 used the conservativity of $S_2^2(\alpha)$ over $T_2^1(\alpha)$ and used the resulting PLS algorithm to construct a kind of interpolant which serves as a natural proof. He later gave a second, more direct proof based on translating $T_2^1(\alpha)$-proofs into propositional proofs [42]. We give below a variation of this second proof based on propositional interpolation theorems, using a translation of $T_2^1(\alpha)$ proofs into limited extension resolution refutations, generalizing a construction of Krajíček [32].

The remainder of this article is devoted to this proof.

8.3 Split Bounded Arithmetic Theories

Let α and β be new unary predicate symbols. We define theories $S_2^i(\alpha, \beta)$ and $T_2^i(\alpha, \beta)$ of bounded arithmetic in the usual way by adjoining α and β as new predicate symbols. In addition, these theories admit induction on

$\Sigma_i^b(\alpha,\beta)$-formulas (which are Σ_i^b-formulas in which α and β may appear). We let $\Sigma_\infty^b(\alpha)$ denote all bounded formulas in the language of S_2 plus α. And we define the class of formulas $\mathcal{S}\Sigma_i^b$ to equal

$$\mathcal{S}\Sigma_i^b = \Sigma_i^b(\Sigma_\infty^b(\alpha), \Sigma_\infty^b(\beta))$$

where $\Sigma_1^b(X)$ indicates the closure of X under $\wedge$, $\vee$, sharply bounded quantification and existential bounded quantification, where $\Pi_1^b(X)$ is defined similarly and

$$\Sigma_{i+1}^b(X) = \Sigma_1^b(\Pi_i^b(X))$$

and $\Pi_{i+1}^b(X)$ is similarly defined.

The *split* versions of S_2^i and T_2^i are defined as follows:

Definition

$$\begin{aligned} \mathcal{S}S_2^i &= BASIC + \mathcal{S}\Sigma_i^b\text{-PIND} \\ \mathcal{S}T_2^i &= BASIC + \mathcal{S}\Sigma_i^b\text{-IND} \end{aligned}$$

We now describe the way in which circuits are coded in the theories $S_2^i(\alpha)$ and $T_2^i(\alpha)$, and the way in which superpolynomial lower bounds on circuit size are expressed in these theories. Let $N \geq 0$ and $n = |N| \approx \log N$ (where $n = |x|$). Also suppose $t(n) = n^{\omega(1)}$ (this will be the superpolynomial lower bound) — note that the fact that $t(n) = n^{\omega(1)}$ need not be provable in S_2. Finally, suppose that $S(N,x)$ is a Σ_∞^b-formula.

Definition Let $LB(t,S,\alpha)$ be the statement

$$\neg\Big[\alpha \text{ codes a circuit of size} \leq t(n) \text{ such that } (\forall x \in \{0,1\}^n)(\alpha(x) = 1 \leftrightarrow S(N,x))\Big]$$

where

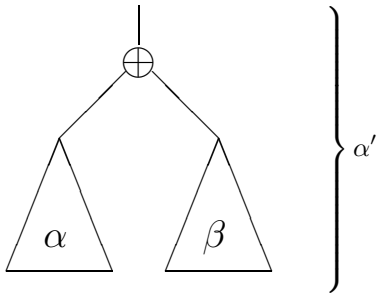
(1) The free variables of $LB(t,S,\alpha)$ are N and α.

(2) By "α encodes a circuit" we mean that α encodes gate types and gate connections in some straightforward manner, plus, α may encode the full truth table description of the functions computed by every gate in the circuit!

Theorem 50 *If* $S_2^2(\alpha) \vdash LB(t,S,\alpha)$, *then* $\mathcal{S}S_2^2 \vdash SLB(t,S,\alpha,\beta)$ *where* $SLB(t,S,\alpha,\beta)$ *is*

$$\neg\Big[\alpha \text{ codes a circuit of size} \leq t(n)/2 - 1 \text{ and } \beta \text{ codes a circuit of size} \leq t(n)/2 - 1 \text{ such that } \forall x \in \{0,1\}^n((\alpha \oplus \beta)(x) = 1 \leftrightarrow S(N,x))\Big]$$

Proof If $\neg SLB(t, S, \alpha, \beta)$, then the circuit α'



satisfies $\neg LB(t, S, \alpha)$. $\square$

In order to rephrase $SLB(t, S, \alpha, \beta)$, we let γ be a new predicate symbol. Theories such as $\mathcal{S}S(\alpha, \beta)^2_2$ and $\mathcal{S}T^1_2(\alpha, \beta)$ may use the symbol γ in induction formulas. If

$$\mathcal{S}S^2_2 \vdash SLB(t, S, \alpha, \beta),$$

then

$$\mathcal{S}S^2_2 \vdash \neg CC(t/2 - 1, \gamma, \alpha) \vee \neg CC(t/2 - 1, S \oplus \gamma, \beta)$$

where $CC(t, T(x), \alpha)$ states:

$$\Big[\alpha \text{ codes a circuit of size} \leq t(n) \text{ such that } \forall x \in \{0,1\}^n (\alpha(x) = 1 \leftrightarrow T(x))\Big]$$

or, in sequent form, $\mathcal{S}S^2_2$ proves

$$CC(t/2 - 1, \gamma, \alpha), CC(t/2 - 1, S \oplus \gamma, \beta) \longrightarrow$$

Since CC is a Σ^b_1 formula, this sequent is also provable in $\mathcal{S}T^1_2$ by $\forall\Sigma^b_2$-conservativity. (By the same proof that shows S^2_2 is conservative over T^1_2.)

Theorem 51 *(Razborov [43]) If $\mathcal{S}S^2_2 \vdash SLB(t, S, \alpha, \beta)$ for some $t = n^{\omega(1)}$ and $S \in \Sigma^b_\infty$, then the SPRNG conjecture is false.*

Corollary 52 *If the SPRNG conjecture holds, then S^2_2 does not prove superpolynomial lower bounds on circuit size for any bounded formula (i.e., for any polynomial time hierarchy predicate).*

The proof of Theorem 51 takes up the rest of this paper. We shall prove that if

$$\mathcal{S}T^1_2 \vdash CC(t, \gamma, \alpha), CC(t, S \oplus \gamma, \beta) \longrightarrow,$$

then there are quasipolynomial size circuits which are natural against $P/poly$.

The first step of the proof involves converting an $\mathcal{S}T^1_2$-proof and the sequent into a constant-depth propositional proof using the general method of Theorem 38. We then convert this to a resolution refutation with limited extension and finally apply the Interpolation Theorem 47. The interpolant obtained in this way will serve as a natural proof against $P/poly$.

8.4 Conversion to Constant Depth Frege Proofs

In this section, we discuss how to apply the Paris-Wilkie translation of Theorem 38 to $T^1_2(\alpha, \beta)$ proofs. The proof of Theorem 38 still mostly applies, and the only new ingredient required for the translation is that we must describe the translation of atomic formulas $\beta(t)$ and $\gamma(t)$. This is easily done by augmenting the definition of section 6.4 above to define $(\beta(t))^{PW}$ to equal r_i and $(\gamma(t))^{PW}$ to equal p_i, where i is the numeric value of the closed term t.

By expanding the language to include the sequence coding functions and using $\mathcal{S}\Pi^b_1$-IND and applying free cut-elimination, we may assume that every formula in the $\mathcal{S}T^1_2$ proof is of the form

$$(\forall y \leq r)(\exists z \leq |r'|)(\cdots)$$

where $(\cdots)$ is a Boolean combination of $\Sigma^b_\infty(\alpha)$ formulas and $\Pi^b_\infty(\beta)$ formulas and of formulas $\gamma(\cdots)$.

When translated into propositional logic by the Paris-Wilkie translation, this becomes

$$\bigwedge_{i=0}^{2^{n^{O(1)}}} \bigvee_{j=0}^{n^{O(1)}} E_{i,j}$$

where each formula $E_{i,j}$ either (1) is $\pm p_i$ or (2) involves only $\vec{p}, \vec{q}$ or (3) involves only $\vec{p}, \vec{r}$.

Suppose we have a $\mathcal{S}T^1_2$-proof of

$$CC(t, \gamma, \alpha), CC(t, S \oplus \gamma, \beta) \longrightarrow,$$

Fixing N and $f(x) = S(N, \alpha)$, the Paris-Wilkie translation gives a propositional sequent calculus proof of

$$\bigwedge_i A_i(\vec{p}, \vec{q}), \bigwedge_j B_j(\vec{p}, \vec{r}) \longrightarrow$$

where

(1) $\{A_i(\vec{p}, \vec{q})\}_i$ is a set of clauses stating that $\vec{q}$ codes a circuit of size t computing the function γ with graph given by $\vec{p}$.

(2) $\{B_j(\vec{p},\vec{q})\}_j$ is a set of clauses stating that $\vec{r}$ codes a circuit of size t computing the function $\gamma \oplus f$.

(3) f does not have a circuit of size $2t+1$

(4) Each formula in the proof is a conjunction of disjunctions of formulas involving just $\vec{p},\vec{q}$ or just $\vec{p},\vec{r}$.

(5) Each sequent has only c many formulas, where c is a constant independent of N.

(6) The proof has only $2^{n^{O(1)}}$ many symbols.

So far, we have obtained a polynomial size, tree-like sequent calculus proof in which every formula consists of conjunctions of clauses. The conjunctions may have polynomially many conjuncts; however, the clauses will contain at most $N^{O(1)}$ many literals. In addition, there is a constant upper bound, independent of N, on the number of formulas in each sequent of the proof. We shall now replace this with a propositional sequent calculus proof in which all formulas are merely clauses (i.e., disjunctions of atomic and negated atomic formulas). The new proof will still be polynomial size and tree-like; however, it will no longer have the property that there is a constant upper bound on the number of formulas in each sequent.

To form this new proof, we apply the following translation to the propositional proof obtained above:

(a) Given a sequent

$$\bigwedge_{i=1}^{p_1} E_{1,i}, \ldots, \bigwedge_{i=1}^{p_{c'}} E_{c',i} \longrightarrow \bigwedge_{i=1}^{q_1} F_{1,i}, \ldots, \bigwedge_{i=1}^{q_{c''}} F_{c'',i}$$

replace it with the $q_1 \cdot q_2 \cdot \cdots \cdot q_{c''}$ sequents

$$E_{1,1}, E_{1,2}, \ldots, E_{1,p_1}, E_{2,1}, \ldots, E_{c',p_{c'}} \longrightarrow F_{1,i_1}, F_{1,i_2}, \ldots, F_{c'',i_{c''}}$$

Since each $q_i = 2^{n^{O(1)}}$ and $c'' = O(1)$, this still only $2^{n^{O(1)})}$ many sequents.

(b) Build a new proof of *all* these sequents. The hardest case of making this a valid new proof, is the case of a cut on $\bigwedge_{i=1}^{p} F_i$. For this, an inference

$$\frac{\Gamma \longrightarrow \Delta, \bigwedge_{i=1}^{p} F_i \qquad \bigwedge_{i=1}^{p} F_i, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

is replaced by p cuts; i.e., by

$$\dfrac{\Gamma^* \longrightarrow \Delta^*, F_1 \quad \dfrac{F_1, F_2, \ldots, F_p, \Gamma^* \longrightarrow \Delta^*}{F_2, F_3, \ldots, F_p, \Gamma^* \longrightarrow \Delta^*}}{\Gamma^* \longrightarrow \Delta^*, F_2 \quad \dfrac{F_2, F_3, \ldots, F_p, \Gamma^* \longrightarrow \Delta^*}{F_3, \ldots, F_p, \Gamma^* \longrightarrow \Delta^*}}$$
$$\Gamma^* \longrightarrow \Delta^*, F_3 \quad \dfrac{F_3, \ldots, F_p, \Gamma^* \longrightarrow \Delta^*}{\vdots}$$
$$\Gamma^* \longrightarrow \Delta^*$$

At the end of the second step, we have a treelike sequent calculus proof of

$$A_1(\vec{p}, \vec{q}), \ldots, A_k(\vec{p}, \vec{q}), B_1(\vec{p}, \vec{r}), \ldots B_\ell(\vec{p}, \vec{r}) \longrightarrow$$

such that every formula in in the proof is a disjunction of formulas which either involve just $\vec{p}$ and $\vec{q}$ or involve just $\vec{p}$ and $\vec{r}$.

8.5 Translation to Resolution with Limited Extension

In the previous section, we have converted the $\mathcal{ST}_2^1(\alpha, \beta)$-proof into a polynomial size, tree-like, propositional sequent calculus proof containing only clauses as formulas. We next shall convert this proof into a resolution refutation with limited extension.

Each sequent in the proof obtained in the second step has the form

$$\bigvee_{i=1}^{p_1} E_{1,i}, \ldots, \bigvee_{i=1}^{p_u} E_{u,i}, \longrightarrow \bigvee_{i=1}^{q_1} F_{1,i}, \ldots, \bigvee_{i=1}^{q_v} F_{v,i} \qquad \text{(A)}$$

where each $E_{a,i}$, $F_{a,i}$ involves only $\{\vec{p}, \vec{q}\}$ or $\{\vec{p}, \vec{r}\}$.

Associate with sequent (A), the following set (B) of clauses:

$$\{\{E_{1,i}\}_{i=1}^{p_1}, \ldots, \{E_{u,i}\}_{i=1}^{p_u}, \{\neg F_{1,1}, \}, \{\neg F_{1,2}, \}, \ldots, \{\neg F_{v,q_v}, \}\} \qquad \text{(B)}$$

Now (B) is not really a proper set of clauses, since clauses are supposed to contain literals (not formulas). So instead of using (B), we introduce extension variables to form the following set (C) of clauses:

$$\left\{\{\sigma_{E_{1,i}}\}_{i=1}^{p_1}, \ldots, \{\sigma_{E_{u,i}}\}_{i=1}^{p_u}, \{\sigma_{\neg F_{1,1}}\}, \{\sigma_{\neg F_{1,2}}\}, \ldots, \{\sigma_{\neg F_{v,q_v}}\}\right\} \qquad \text{(C)}$$

If sequent (A) is $\Gamma \longrightarrow \Delta$, then the set (C) of clauses is denoted $(\Gamma \longrightarrow \Delta)^{LE}$. It is important for us that all the extension variables used in (C) are from $LE(\vec{p}, \vec{q})$ and $LE(\vec{p}, \vec{r})$.

Lemma 53 *If $\Gamma \longrightarrow \Delta$ is derived in m lines of the sequent calculus proof constructed in Step (2) above, then*

$$(\Gamma \longrightarrow \Delta)^{LE} \cup LE(\vec{p}, \vec{q}) \cup LE(\vec{p}, \vec{r})$$

has a resolution refutation (not necessarily tree-like) of $O(m^2)$ resolution inferences.

Proof By induction on m. — splits into cases depending on the last inference of the proof:

Case (1) $\Gamma \longrightarrow \Delta$ is $A \longrightarrow A$. If $A = \bigvee A_i$, then

$$\{\{\sigma_{A_1}, \ldots, \sigma_{A_u}\}, \{\sigma_{\neg A_1}\}, \ldots, \{\sigma_{\neg A_u}\}\} \cup LE(A)$$

has a resolution refutation of $O(u)$ inferences.

Case (2) Suppose $A = \bigvee_i A_i$ involves only $\vec{p}, \vec{q}$. Then $\{\sigma_A\}$ and $\{\sigma_{A_1}, \ldots, \sigma_{A_u}\}$ can be derived from each other (in the presence of $LE(A)$). Therefore it is not important how we express formulas as disjunctions when there is a choice.

Case (3) $\wedge$:left and $\vee$:right inferences involve only formulas that use just $\vec{p}, \vec{q}$ or just $\vec{p}, \vec{r}$; these are therefore straightforward (the $\wedge$:right is a little harder than the $\vee$:left case).

Case (4) An $\vee$:left inference can be:

$$\frac{\bigvee_i E_i, \Gamma \longrightarrow \Delta \qquad \bigvee_j F_j, \Gamma \longrightarrow \Delta}{\vee\{E_i, F_j\}_{i,j}, \Gamma \longrightarrow \Delta}$$

The induction hypotheses give refutations R_1 and R_2:

$$\left.\begin{array}{l}(\Gamma \longrightarrow \Delta)^{LE} \\ \{\sigma_{E_i}\}_i \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r})\end{array}\right\} \stackrel{R_1}{\Longrightarrow} \emptyset$$

and

$$\left.\begin{array}{l}(\Gamma \longrightarrow \Delta)^{LE} \\ \{\sigma_{F_j}\}_j \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r})\end{array}\right\} \stackrel{R_2}{\Longrightarrow} \emptyset$$

These can be combined as:

$$\left.\begin{array}{l}(\Gamma \longrightarrow \Delta)^{LE} \\ \{\sigma_{E_i}\}_i \cup \{\sigma_{F_j}\}_j \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r})\end{array}\right\} \stackrel{R_1'}{\Longrightarrow} \left.\begin{array}{l}\{\sigma_{F_j}\}_j \\ (\Gamma \longrightarrow \Delta)^{LE} \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r})\end{array}\right\} \stackrel{R_2}{\Longrightarrow} \emptyset$$

where R_1' is like R_1 but uses $\{\sigma_{E_i}\}_i \cup \{\sigma_{F_j}\}_j$ in place of $\{\sigma_{E_i}\}_i$.

(*Remark:* Note the refutation is not tree-like since $\{\sigma_{F_j}\}_j$ may be used multiple times in R_2.)

Case 5 Suppose the last inference is a cut:

$$\frac{\Gamma \longrightarrow \Delta, \bigvee_i A_i \qquad \bigvee_i A_i, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

The induction hypotheses give refutations R_1 and R_2:

$$\left.\begin{array}{l} (\Gamma \longrightarrow \Delta)^{LE} \\ \{\sigma_{\neg A_1}\}, \ldots, \{\sigma_{\neg A_u}\} \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r}) \end{array}\right\} \stackrel{R_1}{\Longrightarrow} \emptyset$$

and

$$\left.\begin{array}{l} (\Gamma \longrightarrow \Delta)^{LE} \\ \{\sigma_{A_1}, \ldots, \sigma_{A_u}\} \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r}) \end{array}\right\} \stackrel{R_2}{\Longrightarrow} \emptyset$$

Combine these as below, with R_1' equal to R_1 minus any uses of $\{\sigma_{\neg A_i}\}$'s:

$$\left.\begin{array}{l} (\Gamma \longrightarrow \Delta)^{LE} \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r}) \end{array}\right\} \stackrel{R_1'}{\Longrightarrow} \left.\begin{array}{l} \{\sigma_{A_1}, \ldots, \sigma_{A_u}\} \\ (\Gamma \longrightarrow \Delta)^{LE} \\ LE(\vec{p}, \vec{q}) \\ LE(\vec{p}, \vec{r}) \end{array}\right\} \stackrel{R_2}{\Longrightarrow} \emptyset$$

Q.E.D. Lemma.

8.6 Conclusion of the Proof

From the previous Lemma 53 and from the Interpolation Theorem 47 for resolution with limited extension, there must be a circuit $C(\vec{p})$ of size $2^{n^{O(1)}}$ such that

(1) if $C(\vec{p}) = 0$, then $\{A_i(\vec{p}, \vec{q})\}_i$ is unsatisfiable, and

(2) if $C(\vec{p}) = 1$, then $\{B_j(\vec{p}, \vec{q})\}_j$ is unsatisfiable.

Note the size of $C(\vec{p})$ is $2^{n^{O(1)}}$ which is quasipolynomial in $N = 2^n$. In case (1), when $C(\vec{p}) = 0$, the function $\gamma(x)$ does not have a circuit of size $t = n^{\omega(1)}$. In case (2), when $C(\vec{p}) = 1$, the function $(\gamma \oplus f)(x)$ does not have a circuit of size $t = n^{\omega(1)}$. (Recall $f(x)$ does not have a circuit of size $2t + 1$.)

Definition Let

$$C^*(\vec{p}) \stackrel{df}{=} (\neg C(\vec{p})) \vee C(\vec{p} \oplus f),$$

where $\vec{p} \oplus f$ is $p_0 \oplus f(0), \ldots p_{N-1} \oplus f(\underline{N-1})$. (Each $f(i)$ is 0 or 1, of course.)

Claim Under the above assumptions, $C^*(\vec{p})$ is a quasipolynomial time property against $P/poly$.

Proof There are three things to show:

(1) *Constructivity:* C^* has circuits of size $2^{n^{O(1)}}$ since C does.

(2) *Largeness:* For all γ, either $C^*(\gamma)$ or $C^*(\gamma \oplus f)$ holds (since either $\neg C(\gamma)$ holds, or $C((\gamma \oplus f) \oplus f)$ holds). Therefore, $C^*(\gamma)$ holds for at least half of the γ's.

(3) *Usefulness:* then γ does not have a polynomial size circuit.

(3.a) If $\neg C(\vec{p})$, i.e., $C(\vec{p}) = 0$, then $\gamma = \vec{p}$ does not have a circuit of size t, by choice of C.

(3.b) If $C(\vec{p} \oplus f)$, i.e., $C(\vec{p} \oplus f) = 1$, then $(\vec{p} \oplus f) \oplus f = \vec{p}$ $(= \gamma)$ likewise does not have a circuit of size t.

Q.E.D. Theorem 51

References

[1] J. Bennett, *On Spectra*, PhD thesis, Princeton University, 1962.

[2] M. L. Bonet, S. R. Buss, and T. Pitassi, *Are there hard examples for Frege systems?*, in Feasible Mathematics II, P. Clote and J. Remmel, eds., Boston, 1995, Birkhäuser, pp. 30–56.

[3] S. R. Buss, *Bounded Arithmetic*, PhD thesis, Princeton University, 1985.

[4] ——, *Bounded Arithmetic*, Bibliopolis, 1986. Revision of 1985 Princeton University Ph.D. thesis.

[5] ——, *The Boolean formula value problem is in ALOGTIME*, in Proceedings of the 19-th Annual ACM Symposium on Theory of Computing, May 1987, pp. 123–131.

[6] ——, *Polynomial size proofs of the propositional pigeonhole principle*, Journal of Symbolic Logic, 52 (1987), pp. 916–927.

[7] ——, *An introduction to proof theory.* Typeset manuscript, to appear in S.R. Buss (ed.) *Handbook of Proof Theory*, North-Holland, 199?

[8] ——, *Axiomatizations and conservation results for fragments of bounded arithmetic*, in Logic and Computation, proceedings of a Workshop held Carnegie-Mellon University, 1987, vol. 106 of Contemporary Mathematics, American Mathematical Society, 1990, pp. 57–84.

[9] ——, *Propositional consistency proofs*, Annals of Pure and Applied Logic, 52 (1991), pp. 3–29.

[10] ——, *On Herbrand's theorem*, in Logic and Computational Complexity, Lecture Notes in Computer Science #960, D. Leivant, ed., Berlin, 1995, Springer Verlag, pp. 195–209.

[11] ——, *Relating the bounded arithmetic and polynomial-time hierarchies*, Annals of Pure and Applied Logic, 75 (1995), pp. 67–77.

[12] S. R. Buss and et al., *Weak formal systems and connections to computational complexity*. Student-written Lecture Notes for a Topics Course at U.C. Berkeley, January–May 1988.

[13] S. R. Buss and A. Ignjatović, *Unprovability of consistency statements in fragments of bounded arithmetic*, Annals of Pure and Applied Logic, 74 (1995), pp. 221–244.

[14] S. R. Buss and J. Krajíček, *An application of Boolean complexity to separation problems in bounded arithmetic*, Proc. London Math. Society, 69 (1994), pp. 1–21.

[15] A. Cobham, *The intrinsic computational difficulty of functions*, in Logic, Methodology and Philosophy of Science, Proceedings of the Second International Congress, held in Jerusalem, 1964, Y. Bar-Hillel, ed., Amsterdam, 1965, North-Holland.

[16] S. A. Cook, *Feasibly constructive proofs and the propositional calculus*, in Proceedings of the Seventh Annual ACM Symposium on Theory of Computing, 1975, pp. 83–97.

[17] ——, *Computational complexity of higher type functions*, in Proceedings of the International Congress of Mathematicians, 1990, pp. 55–69.

[18] S. A. Cook and R. A. Reckhow, *On the lengths of proofs in the propositional calculus, preliminary version*, in Proceedings of the Sixth Annual ACM Symposium on the Theory of Computing, 1974, pp. 135–148.

[19] ——, *The relative efficiency of propositional proof systems*, Journal of Symbolic Logic, 44 (1979), pp. 36–50.

[20] W. Craig, *Linear reasoning. A new form of the Herbrand-Gentzen theorem*, Journal of Symbolic Logic, 22 (1957), pp. 250–268.

[21] M. Dowd, *Propositional representation of arithmetic proofs*, in Proceedings of the 10th ACM Symposium on Theory of Computing, 1978, pp. 246–252.

[22] ——, *Model-theoretic aspects of $P \neq NP$.* Typewritten manuscript, 1985.

[23] G. Gentzen, *Die Widerspruchsfreiheit der reinen Zahlentheorie*, Mathematische Annalen, 112 (1936), pp. 493–565. English translation in [24], pp. 132-213.

[24] ——, *Collected Papers of Gerhard Gentzen*, North-Holland, 1969. Editted by M. E. Szabo.

[25] P. Hájek and P. Pudlák, *Metamathematics of First-order Arithmetic*, Springer-Verlag, Berlin, 1993.

[26] A. Haken, *The intractability of resolution*, Theoretical Computer Science, 39 (1985), pp. 297–308.

[27] J. Herbrand, *Recherches sur la théorie de la démonstration*, PhD thesis, University of Paris, 1930.

[28] ——, *Investigations in proof theory: The properties of true propositions*, in From Frege to Gödel: A Source Book in Mathematical Logic, 1978-1931, J. van Heijenoort, ed., Harvard University Press, Cambridge, Massachusetts, 1967, pp. 525–581. Translation of chapter 5 of [27], with commentary and notes, by J. van Heijenoort and B. Dreben.

[29] ——, *Écrits logique*, Presses Universitaires de France, Paris, 1968. Ed. by J. van Heijenoort.

[30] ——, *Logical Writings*, D. Reidel, Dordrecht-Holland, 1971. Ed. by W. Goldfarb, Translation of [29].

[31] C. F. Kent and B. R. Hodgson, *An arithmetic characterization of NP*, Theoretical Comput. Sci., 21 (1982), pp. 255–267.

[32] J. Krajíček, *Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic.* To appear in *Journal of Symbolic Logic.*

[33] J. Krajíček and P. Pudlák, *Propositional proof systems, the consistency of first-order theories and the complexity of computations*, Journal of Symbolic Logic, 54 (1989), pp. 1063–1079.

[34] ——, *Quantified propositional calculi and fragments of bounded arithmetic*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 36 (1990), pp. 29–46.

[35] J. Krajíček, P. Pudlák, and G. Takeuti, *Bounded arithmetic and the polynomial hierarchy*, Annals of Pure and Applied Logic, 52 (1991), pp. 143–153.

[36] R. E. Ladner, *The circuit value problem is log space complete for P*, SIGACT News, 7 (1975), pp. 18–20.

[37] D. Mundici, *Tautologies with a unique Craig interpolant*, Annals of Pure and Applied Logic, 27 (1984), pp. 265–273.

[38] E. Nelson, *Predicative Arithmetic*, Princeton University Press, 1986.

[39] C. H. Papadimitriou, *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, 1990, pp. 794–801.

[40] R. J. Parikh, *Existence and feasibility in arithmetic*, Journal of Symbolic Logic, 36 (1971), pp. 494–508.

[41] J. B. Paris and A. J. Wilkie, Δ_0 *sets and induction*, in Open Days in Model Theory and Set Theory, W. Guzicki, W. Marek, A. Pelc, and C. Rauszer, eds., 1981, pp. 237–248.

[42] A. A. Razborov, *On provably disjoint NP-pairs*, Tech. Rep. RS-94-36, Basic Research in Computer Science Center, Aarhus, Denmark, November 1994. http://www.brics.dk/index.html.

[43] ——, *Unprovability of lower bounds on the circuit size in certain fragments of bounded arithmetic*, Izvestiya of the RAN, 59 (1995), pp. 201–224.

[44] A. A. Razborov and S. Rudich, *Natural proofs*, in Proceedings of the 26-th Annual ACM Symposium on Theory of Computing, 1994, pp. 204–213.

[45] R. A. Reckhow, *On the Lengths of Proofs in the Propositional Calculus*, PhD thesis, Department of Computer Science, University of Toronto, 1976. Technical Report #87.

[46] J. Siekmann and G. Wrightson, *Automation of Reasoning*, vol. 1&2, Springer-Verlag, Berlin, 1983.

[47] R. Statman, *Complexity of derivations from quantifier-free Horn formulae, mechanical introduction of explicit definitions, and refinement of completeness theorems*, in Logic Colloquium '76, Amsterdam, 1977, North-Holland, pp. 505–517.

[48] L. J. Stockmeyer, *The polynomial-time hierarchy*, Theoretical Comput. Sci., 3 (1976), pp. 1–22.

[49] G. S. Tsejtin, *On the complexity of derivation in propositional logic*, Studies in Constructive Mathematics and Mathematical Logic, 2 (1968), pp. 115–125. Reprinted in: [46, vol 2].

[50] A. J. Wilkie and J. B. Paris, *On the scheme of induction for bounded arithmetic formulas*, Annals of Pure and Applied Logic, 35 (1987), pp. 261–302.

[51] C. Wrathall, *Complete sets and the polynomial-time hierarchy*, Theoretical Comput. Sci., 3 (1976), pp. 23–33.

[52] D. Zambella, *Notes on polynomially bounded arithmetic*, Journal of Symbolic Logic, 61 (1996), pp. 942–966.