# Bounded Arithmetic, Cryptography and Complexity

Samuel R. Buss*

**Abstract**

This survey discusses theories of bounded arithmetic, growth rates of definable functions, natural proofs, interpolation theorems, connections to cryptography, and the difficulty of obtaining independence results.

## 1 Introduction

The theories of bounded arithmetic, $S^i_1$ and $T^i_2$ for $i \geq 1$, are known to be closely related to the complexity classes of polynomial time hierarchy. These theories give a natural proof-theoretic framework for capturing the computational complexity of P, NP and other classes of the polynomial time hierarchy. Since no one has so far been able to resolve fundamental questions of computational complexity such as whether the polynomial time hierarchy is proper and whether P = NP, it is natural to look for analogues of these questions in the setting of bounded arithmetic. Two of the most natural such questions are, firstly, whether the hierarchy of theories of bounded arithmetic is proper and whether the theories $S^i_1$ and $T^i_2$ are all distinct, and secondly, whether open problems in complexity theory such as the P vs. NP question might be independent of theories of bounded arithmetic.† Resolving these questions about bounded arithmetic would not automatically answer open questions in complexity theory: for example, it may be that the theories of bounded arithmetic are all distinct whilst the polynomial time hierarchy collapses but does not $S_2$-provably collapse. However, there is hope that if we can resolve the questions about bounded arithmetic, that this would be a significant step towards resolving the P =?NP question. On the other hand, the questions about bounded arithmetic might be easier to resolve precisely because they do not necessarily imply results in computational complexity.

*Department of Mathematics, University of California, San Diego. Supported in part by NSF grant DMS-9503247.

†These two questions are not entirely independent, since it is known that the theories of bounded arithmetic collapse to some finite level if and only if $S_2$ is finitely axiomatized and if and only if $S_2$ can prove the polynomial time hierarchy collapses in a strong way [14, 6, 23].

Recently, there have been a number of results relating the strength of bounded arithmetic to computational issues arising from the study of cryptography and pseudorandom generators. Specifically, Razborov and Rudich [22] introduced a notion of "natural proofs" of $\mathrm{P} = \mathrm{NP}$, and Razborov [21] proved that, under some widely accepted cryptographic assumptions, $S^2_2$ cannot prove superpolynomial lower bounds on the size of circuits computing satisfiability (SAT). Razborov's proof methods were based on a Craig interpolation theorem; Krajíček and Pudlák [13] subsequently established another striking connection between interpolation theorems and cryptography by showing that if a certain form of interpolation, $\Delta^b_1$-interpolation, holds for $S^1_2$, then the RSA encryption function is insecure. They also showed that if a $\Pi^b_1$-extension $S^1_2 + \Phi$ of $S^1_2$ admits $\Delta^b_1$-interpolation, then the discrete logarithm function is insecure. Since these two cryptographic protocols are currently some of the best candidates for secure cryptography, this is fairly strong evidence against the possibility that $S^1_2$ enjoys the $\Delta^b_1$-interpolation property. This paper gives a similar result for another cryptographic protocol.

The outline of this paper is as follows. We presume the reader has knowledge of the P, NP and the polynomial time hierarchy. We also presume knowledge of the main theories of bounded arithmetic, but we restate some of the definitions and the main witnessing theorems needed for this paper (see [3, 9, 10] for background on bounded arithmetic). We discuss the impossibility of getting better bounds on the degrees of the polynomial runtimes of functions extracted from $S^1_2$-proofs. Turning to the central theme of the paper, we discuss a natural approach that one might hope could establish the independence of $\mathrm{P} = \mathrm{NP}$ from $S^1_2$, but give evidence based on the theory of pseudorandom number generators that this approach is unlikely to work. Then we discuss the notion of natural proofs and the theorems of Razborov and Rudich stating that if a certain cryptographic conjecture holds, then there are no natural proofs and no $S^2_2(\alpha)$-proofs that NP requires superpolynomial size circuits. Finally we define the $\Delta^b_1$-interpolation property, and prove that if $S^1_2 + \Phi$ has the $\Delta^b_1$-interpolation property then the Rabin encryption function $x \mapsto x^2 \bmod N$ is not secure. This result extends the above-mentioned results of Krajíček and Pudlák to a third cryptographic protocol, which is also generally believed to be cryptographically secure.

# 2 Provably total functions of bounded arithmetic

## 2.1 Bounded arithmetic and the polynomial time hierarchy

Recall that bounded arithmetic is a first-order theory of the natural numbers with non-logical symbols $0$, $S$, $+$, $\cdot$, $\lfloor \frac{1}{2}x \rfloor$, $|x|$, $\#$, and $\leq$ where $|x| = \lceil \log_2(x+1) \rceil$ and where $x \# y = 2^{|x| \cdot |y|}$. The language of first-order logic is enlarged to incorporate *bounded quantifiers* $(\forall x \leq t)$ and $(\exists x \leq t)$; when the term is of the form $t = |s|$, then the quantifier is *sharply bounded*. The hierarchy $\Sigma^b_i$ and $\Pi^b_1$ of bounded formulas is defined by counting alternations

of bounded quantifiers, ignoring sharply bounded quantifiers (much as the arithmetic hierarchy can be defined by counting alternations of unbounded quantifiers, ignoring bounded quantifiers). The primary two forms of induction axioms used for bounded arithmetic are the following schemes:

$$\Psi\text{-IND:} \qquad A(0) \wedge (\forall x)(A(x) \rightarrow A(x+1)) \rightarrow (\forall x)A(x)$$

$$\Psi\text{-PIND:} \qquad A(0) \wedge (\forall x)(A(\lfloor \tfrac{1}{2}x \rfloor) \rightarrow A(x)) \rightarrow (\forall x)A(x)$$

where, in both axioms schemes, $A$ may be any formula in $\Psi$, possibly containing additional free parameter variables.

The theories $S_2^i$, $i \geq 0$, are defined to be axiomatized with the $\Sigma_i^b$-PIND induction axioms plus a finite set of quantifier-free axioms which define the non-logical symbols. The theories $T_2^i$ are axiomatized with the same quantifier-free axioms, but with the $\Sigma_i^b$-IND axioms instead of the $\Sigma_i^b$-PIND axioms.

Proofs in bounded arithmetic are formalized in Gentzen's sequent calculus, extended to have rules of inference for bounded quantifiers and extended to have induction rules in place of the induction axioms displayed above. The details of the sequent calculus formulations of bounded arithmetic can be found in [3, 4].

There are several close connections between bounded arithmetic and the polynomial time hierarchy. Firstly, the $\Sigma_i^b$-formulas define exactly the predicates in the class $\Sigma_i^p$ of the polynomial time hierarchy. Thus, in the case $i = 1$, the $\Sigma_1^b$-formulas define precisely the NP-predicates. Secondly, there is a very close relationship between the definable functions of $S_2^i$ and the levels of the polynomial time hierarchy. A function $f : \mathbb{N} \rightarrow \mathbb{N}$ is said to be $\Sigma_i^b$*-definable* by a theory $R$ provided $R$ can prove $(\forall x)(\exists! y)A(x, y)$ for some $\Sigma_i^b$-formula $A$ such that $A(x, y)$ defines the predicate $y = f(x)$. Then we have:

**Theorem 1** *[3, 5] Let $i \geq 1$.*

(1) *The $\Sigma_i^b$-definable functions of $S_2^i$ are precisely the functions which are polynomial time computable relative to an oracle from $\Sigma_{i-1}^b$ (i.e., precisely the $\Box_i^p$-functions).*

(2) *($i > 1$) The $\Sigma_i^b$-definable functions of $T_2^{i-1}$ are also precisely the $\Box_i^p$-functions.*

Probably the most interesting case of the theorem is the $i = 1$ case, where it states that the $\Sigma_1^b$-definable functions of $S_2^1$ are the polynomial time functions.

A predicate is $\Delta_i^b$*-definable* by a theory if and only if it is provably equivalent both to a $\Sigma_i^b$-formula and to $\Pi_i^b$-formula. A corollary to the above theorem is that the $\Delta_i^b$-predicates of $S_2^i$ (and of $T_2^{i-1}$) are precisely the predicates which are polynomial time computable relative to an oracle from $\Sigma_{i-1}^p$ (these are the $\Delta_i^p$-predicates of the polynomial time hierarchy). In particular, the $\Delta_1^b$-definable predicates of $S_2^1$ are precisely the polynomial time computable predicates.

For the $\Sigma_1^b$-definable functions of $T_2^1$, we have the following theorem:

**Theorem 2** *[8] The $\Sigma^b_1$-definable functions of $T^1_2$ are precisely the functions which can be expressed as the composition of a polynomial time function and a polynomial local search function.*

The class of polynomial local search functions is defined in [15]; since it is not important for our applications in this paper we omit the definition here. C. Pollett [16] has extended this theorem to characterize the $\Sigma^b_i$-definable functions of $T^i_2$ for $i \geq 1$.

## 2.2 Bounding the degrees of the polynomial runtimes

The proof of Theorem 1 has two parts: first cut-elimination is used to transform an $S^i_2$-proof of $(\forall x)(\exists y)A(x,y)$, where $A$ is a $\Sigma^b_i$-formula, into a free-cut free proof of $(\forall x)(\exists y \leq t)A(x,y)$, and secondly a polynomial time algorithm is extracted from the free-cut free proof. The first part of the proof construction, the cut-elimination, can make the proof grow superexponentially larger; whereas, the second part, the algorithm extraction is relatively better behaved, yielding a polynomial time algorithm with the degree of the polynomial exponentially bounded by the number of lines in the free-cut free proof. In this section, we take up the question of whether either of these upper bounds, the superexponential growth rate or the exponential growth rate, are necessary or whether they are merely an artifact of the proof method. In both cases, we shall see that these upper bounds cannot be substantially improved.

Recall that the predicate $y = 2^x$ is polynomial time recognizable. Also, defining the superexponential function by $2^x_0 = x$ and $2^x_{i+1} = 2^{2^x_i}$; it is well-known that the 3-ary predicate $y = 2^x_i$ is polynomial-time recognizable. Therefore both predicates are $\Delta^b_1$-definable in $S^1_2$; in fact, $S^1_2$ can prove basic facts about these predicates. Since these predicates are $\Delta^b_1$-definable, we can use them freely in formulas without any essential increase in quantifier complexity. We let $2 \uparrow m$ denote the value $2^1_m$; the predicate $y = 2 \uparrow x$ is also $\Delta^b_1$-definable by $S^1_2$.

We let $\underline{m}$ denote the canonical term with value $m$: $\underline{0}$ is just $0$, $\underline{2m}$ is $2 \cdot \underline{m}$ and $\underline{2m+1}$ is $S(2 \cdot \underline{m})$ (where 2 means $S(S(0))$).

The next theorem shows that the superexponential growth rate of the cut-elimination theorem is unavoidable; in fact, not only can the runtime involve constants which are superexponentially large, but also it can lead to runtimes which are polynomials of superexponentially large degree.

**Theorem 3** *There are $S^1_2$-proofs of the statements*

$$(\forall x)(\exists y)(y = 2^{|x|^{2 \uparrow \underline{m}}})$$

*which have $O(m)$ steps and $O(m^2)$ symbols.*

For fixed $m$, the function $x \mapsto 2^{|x|^{2 \uparrow m}}$ is polynomial time computable but, because of its growth rate, its runtime polynomial must have degree at least $2 \uparrow m$.

**Proof** (Sketch). The proof of Theorem 3 uses the speedup of induction method of Solovay. Define the predicate $A(m)$ to be the formula

$$(\forall x)(\exists y)(y = 2^{|x|^m}).$$

The formula $A(m)$ defines a $+$*-closed cut* since it satisfies the following four conditions: (1) $A(0)$, (2) $(\forall m)(A(m) \to A(m+1))$, (3) $(\forall m_1, m_2)(A(m_1) \wedge m_2 < m_1 \to A(m_2))$ and (4) $(\forall m)(A(m) \to A(m+m))$. Now define $A_0(m)$ to be the formula

$$(\forall x)(A(x) \to A(m \cdot x)).$$

$A_0(x)$ defines a $\cdot$*-closed cut* since it satisfies conditions (1)-(4) plus the condition (5) $(\forall m)(A_0(m) \to A_0(m \cdot m))$. If we define $A_1(m)$ to be the formula

$$(\forall x)(A_0(x) \to A_0(m\#x)),$$

then $A_1(m)$ defines a $\#$*-closed cut* since it is a $\cdot$-closed cut and also satisfies (6) $(\forall m)(A_1(m) \to A_1(m\#m))$.

More generally, for $i \geq 1$, define $A_{2i}(m)$ to be the formula $(\exists x)(x = 2^m \wedge A_{2i-1}(x))$; and define $A_{2i+1}(m)$ to be the formula $(\forall x)(A_{2i}(x) \to A_{2i}(m\#x))$. Then the following facts are $S^1_2$-provable (for $i \geq 0$):

(a) $A_{2i}(x)$ defines a $\cdot$-closed cut and $A_{2i+1}$ defines a $\#$-closed cut.

(b) $A_{2i+2}(x) \to (\exists y)(y = 2^x \wedge A_{2i}(y))$.

(c) $A_{2i}(z) \to (\forall x)(\exists y)(y = 2^{|x|^{2^{\frac{z}{i}}}})$.

The fact that (a)-(c) have $S^1_2$ proofs is easily established by induction on $i$; in fact, the $S^i_2$-proofs require only $O(i)$ many steps. By using standard methods (see [17]), the formulas $A_i$ can be expressed using only $O(i)$ many symbols. Therefore, the statements (a)-(c) have $S^1_2$-proofs of length $O(i^2)$ symbols.

Since $A_m(1)$ implies $(\forall x)(\exists y)(y = 2^{|x|^{2\uparrow m}})$, the theorem is proved. □

The intuitive reason why cut-elimination gives rise to superexponentially large degrees is that when unbounded quantifiers appear in an $S^1_2$-proof, then the cut-elimination process can cause the proof size to grow superexponentially. However the next theorem shows that the degrees of the polynomial runtimes can be exponentially large even when the $S^1_2$-proof has low quantifier complexity.

**Theorem 4** *The formulas* $(\exists y)(y = 2^{|x|^{2^m}})$ *have sequent calculus proofs in which every antecedent formula is in* $\Sigma^b_1$ *and every succedent formula is in* $\exists\Sigma^b_1$*-form, which have* $O(m)$ *steps and* $O(m^2)$ *symbols.*

**Proof** Let $t_1(x)$ be the term $x\#x$ and define $t_{i+1}(x)$ to be the term $t_i(x)\#t_i(x)$. It is easy to verify that $t_i(x) \geq 2^{|x|^{2^i}}$ for all $x$. Fix $m > 0$. Let $A_m(x_{m-1})$ be the formula

$$(\exists x_m \leq x_{m-1}\#x_{m-1})(x_m = x_{m-1}\#x_{m-1}),$$

and for $i < m$, let $A_i(x_{i-1})$ be the formula

$$(\exists x_i \leq x_{i-1}\#x_{i-1})(x_i = x_{i-1}\#x_{i-1} \wedge A_{i+1}(x_i)).$$

Clearly, $A_m(x_{m-1})$ has a sequent calculus proof. Also, the formula $A_i(x_{i-1})$ can be derived from $A_{i+1}(x_i)$ with a constant number of inferences involving only bounded formulas. Therefore there is an $S^1_2$-proof of $A_1(x_0)$ of size $O(m)$ steps and $O(m^2)$ symbols.

It is easy to verify that there is a short $S^1_2$-proof of the sequent

$$x_1 = x_0\#x_0, x_2 = x_1\#x_1, \ldots, x_m = x_{m-1}\#x_{m-1} \longrightarrow |x_m| \geq |x_0|^{2^m}.$$

and, from this, of the sequent

$$x_1 = x_0\#x_0, x_2 = x_1\#x_1, \ldots, x_m = x_{m-1}\#x_{m-1} \longrightarrow (\exists y)(y = 2^{|x_0|^{2^m}}).$$

Putting this together with the proof of $A_1(x_0)$ leads to the desired $S^1_2$-proof of $(\exists y)(y = 2^{|x_0|^{2^m}})$. □

# 3 Independence of $\mathrm{P} \neq \mathrm{NP}$ from $S^1_2$?

This section discusses a possible method for proving the independence of $\mathrm{P} \neq \mathrm{NP}$ from $S^1_2$, and also discusses a possible obstacle to this method. First, we must consider what it might mean for $\mathrm{P} \neq \mathrm{NP}$ to be independent of $S^1_2$ and why we might expect this independence. A promising start arises from the result of [3] discussed above that any predicate which is $\Delta^b_1$ w.r.t. $S^1_2$ is actually polynomial time computable. Being $\Delta^b_1$ is the same as being provably in $\mathrm{NP} \cap \mathrm{coNP}$; so we have that $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ is consistent with $S^1_2$ in at least some weak sense.[‡]

To formalize $\mathrm{P} \neq \mathrm{NP}$ as a potential theorem of $S^1_2$, we use the fact that satisfiability is complete for NP. Let $TRU(x, y)$ be a $\Delta^b_1$-formula expressing the condition that $x$ codes a Boolean formula and that $y$ codes a satisfying assignment for the Boolean formula. Then $\mathrm{P} = \mathrm{NP}$ if and only if there is a polynomial time function $f$ such that

$$TRU(x, y) \rightarrow TRU(x, f(x))$$

holds for all $x$, $y$. One might wish to express $\mathrm{P} \neq \mathrm{NP}$ by a formula

$$(\forall f \in \mathrm{P})(\exists x)(\exists y)(TRU(x, y) \wedge \neg TRU(x, f(x)),$$

but of course, $S^1_2$ is a first-order theory and cannot quantify over functions. So instead, we'll quantify over Boolean circuits which compute functions. Namely, let $Circuit(C)$ express the property that $C$ is (i.e., codes) a Boolean circuit,

[‡]However the significance and import of this consistency is unclear: see Krajíček's construction [11] of models of fragments of bounded arithmetic where $\mathrm{P} = \mathrm{NP} \cap \mathrm{coNP}$ for an investigation of this.

and we write $Output(C, x)$ for the output of $C$ on input $x$. A circuit may have multiple outputs, which are interpreted as the bits of the binary representation of an integer. Therefore, $Output(C, x)$ is a integer. For $m \geq 2$ consider the formula $\Psi_m$:

$$(\forall u > m)(\forall C \leq 2^{|u|^m})[\text{“}C \text{ is a circuit with } |u| \text{ inputs”} \rightarrow$$
$$(\exists x \leq u)((\exists y \leq x)\, TRU(x, y) \wedge \neg TRU(x, Output(C, x)))].$$

When $m = 2$ for instance, $\Psi_2$ implies that $TRU$ cannot be solved in the above sense by a circuit of size $n^2$. Thus, if $\Psi_m$ is true for all $m$, then NP $\not\subseteq$ P/*poly*, which implies that P $\neq$ NP. Therefore, it is reasonable to define that P = NP is consistent with $S^1_2$ iff the formulas $\Psi_m$ are not $S^1_2$-provable.

It is commonly conjectured that NP $\not\subseteq$ P/*poly*, but this seems to be hard to prove. We further conjecture that the formulas $\Psi_m$ are not theorems of $S^1_2$ and that furthermore it may be possible to prove this fact using present-day techniques. Having said that, we feel obligated to point out a barrier to proving that the formulas $\Psi_m$ are not theorems of $S^1_2$. One likely way to prove that $S^1_2 \nvdash \Psi_m$ would be to prove that the function $C \mapsto (x, y)$ is not polynomial time computable; since, if $S^1_2 \vdash \Psi_m$, then Theorem 1 implies that $x$ and $y$ satisfying the property of $\Psi_m$ can be obtained as a polynomial time function of $C$. However, we shall prove later that, assuming that a strong pseudorandom number generator conjecture (the SPRNG conjecture) holds, there is a non-uniform probabilistic polynomial time computation which obtains $(x, y)$ from $C$. Therefore, under the SPRNG conjecture, the function $C \mapsto (x, y)$ has polynomial size circuits. This means that, assuming the SPRNG conjecture, any proof that $C \mapsto (x, y)$ is not in polynomial time would have to work even though the function has polynomial size circuits.

It would of course be perfectly fine to assume P $\neq$ NP or even that NP does not have polynomial size circuits, to prove that $S^1_2$ does not prove the formulas $\Psi_m$; because $S^1_2$ can prove only true formulas.

# 4 Independence via Natural Proofs

We next discuss the theorem of Razborov's giving a conditional independence result which states that $S^2_2$ cannot prove superpolynomial circuit size lower bounds for NP problems [21]. This result is a “conditional” result since it depends on the truth of a strong pseudorandom number conjecture, which we call the SPRNG conjecture. Since Razborov [20] has also argued that all present-day proof methods for establishing circuit lower bounds are formalizable in $U^1_1$, this provides at least some evidence that present-day proof methods are inadequate for proving P $\neq$ NP. Razborov's independence proof depends heavily on the notion of the *natural proofs* earlier introduced by Razborov-Rudich [22]. They gave a definition of “natural proofs” and proved that present methods for circuit lower bounds give natural proofs; furthermore, they showed that the SPRNG conjecture implies that there are no natural proofs of P $\neq$ NP.

## 4.1 Natural Proofs

We now give the definition of natural proofs due to Razborov and Rudich [22]. The intuitive motivation is that 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 do not have polynomial size circuits (for example, the Boolean function which recognizes satisfiable propositional formulas). More precisely, a proof is natural provided it is possible to use the proof method to find families $C_n$ of $n$-ary Boolean functions which do not have polynomial size circuits. The Boolean functions $f(x_1, \ldots, x_n)$ in $C_n$ are represented by their truth tables (which of course have size $N = 2^n$); therefore, an $n$-ary Boolean function is identified with a string of $2^n$ 0's and 1's and $C_n$ is a set of binary strings of length $2^n$.

**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 three conditions hold:

**Constructivity:** The predicate "$f \in C_n$?" is decidable by 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 a (quasi)polynomial time property $\{C_n\}_n$ which is natural against $P/poly$. Of course, it would be expected that $\{C_n\}_n$ would contain NP functions; however, this is not formally required by the above definition. Note that 'quasipolynomial time', is measured as a function of the size of the truth table of $f_n$.

## 4.2 The SPRNG conjecture

We now give the definition of the Strong Pseudorandom 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 than $P \neq \mathrm{NP}$, and thus far the conjecture has not actually been proven.)

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

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

The intuitive motivation for the above definition of hardness, is that a good pseudorandom number generator ought to behave similarly to a random number generator. Thus it should not be possible to have a feasibly computable probabilistic test which can distinguish pseudorandom numbers (such as values $\overline{y}$ output by $G_m$) from truly random numbers $\overline{y}$. If P = NP then there are no good pseudorandom number generators; however, it is generally conjectured that good pseudorandom number generators do exist:

**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 fixed $\epsilon > 0$.

One of the main results of Razborov and Rudich [22] is the following theorem:

**Theorem 5** *If the SPRNG conjecture is true, then there are no properties which are quasipolynomial time/poly natural against P/poly.*

The proof of this theorem can be found in [22] and we omit it here. Instead the next section discusses Razborov's use of this theorem to get independence results for bounded arithmetic.

## 4.3 An Independence Result for $S^2_2(\alpha)$

Let $\alpha$ represent a new unary predicate symbol. The theories $S^i_2(\alpha)$ and $T^i_2(\alpha)$ are defined similarly to $S^i_2$ and $T^i_2$, but allowing $\alpha$ to be a new predicate symbol which may appear in induction formulas. Namely, we define the classes $\Sigma^b_i(\alpha)$ and $\Pi^b_i(\alpha)$ just like the classes $\Sigma^b_i$ and $\Pi^b_i$; counting quantifiers as before, but now allowing the predicate $\alpha$ in the formulas. $S^i_2(\alpha)$ and $T^i_2(\alpha)$ are axiomatized with the same open formulas as $S^i_2$ plus the $\Sigma^b_i(\alpha)$-PIND and $\Sigma^b_i(\alpha)$-IND induction axioms (respectively). Note that although $\alpha$ may appear in induction formulas, there are no axioms defining $\alpha$ — from the computational point of view, $\alpha$ acts like an oracle, which could be *any* predicate.

There is no loss of generality in assuming that $\alpha$ is a unary predicate. Firstly, a $n$-ary predicate $\beta(\vec{x})$ can be simulated by a unary predicate by using Gödel numbering to encode $n$ integers as a single integer; then $\alpha(\langle\vec{x}\rangle)$ could give the value of $\beta(\vec{x})$. Secondly, a function $f$ of polynomial growth rate can be simulated by $\alpha$ by letting $\alpha(\langle x, i\rangle)$ equal the $i$-th bit of the binary representation of $f(x)$. In this way, $S^i_2(\alpha)$, respectively $T^i_2(\alpha)$, has the same power as $S^i_2(\beta)$ or $S^i_2(f)$, respectively $T^i_2(\beta)$ or $T^i_2(f)$.

The predicate $\alpha$ can encode a circuit by letting $\alpha$ encode the connection language for the circuit. I.e., if $i$ and $j$ are gate numbers, then $\alpha(\langle i, j\rangle)$ is true iff the output of gate $i$ is an input to gate $j$; and if $\tau$ is a gate type, such as $\wedge, \vee, \neg$, then $\alpha(\langle 0, \tau, i\rangle)$ is true if gate $i$ is of type $\tau$. The predicate $\alpha$ may encode additional information about the circuit as well; for instance, for $\vec{x}$ a list of Boolean inputs to the circuit, we may have $\alpha(\langle\vec{x}, i\rangle)$ giving the output value of gate $i$ for those inputs. The property that $\alpha$ encodes a valid circuit is

expressible as a $\forall\Delta_1^b$-predicate, and $S_1^i(\alpha)$ can do elementary reasoning about the encoding and behavior of circuits.

We adopt the convention that if a circuit has $t$ gates, then the gates must be numbered from 1 to $t$ and if gate $i$ is an input to gate $j$, then $i < j$.

We next wish to give a first-order formula expressing superpolynomial lower bounds for polynomial hierarchy predicates. Let $S(x)$ be any bounded formula (i.e., any polynomial hierarchy predicate). We are mostly interested in the case where $S$ is the satisfiability predicate $SAT(x) \Leftrightarrow (\exists y \leq x)\, TRU(x, y)$.

**Definition** We define $LB(n, t, S, \alpha)$ to be a formula which states that the Boolean circuit encoded by $\alpha$ either does not correctly compute the predicate $S$ or has size greater than $t$. The circuit bound $t$, may be specified by a term or may be specified implicitly in terms of its graph. (E.g., in the latter case, $t$ could be the function $n \mapsto 2^n$, which cannot be expressed as a term in bounded arithmetic.) Formally, we let $LB(n, t, S, \alpha)$ be the statement:

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

Note that the only free variable in $LB(n, t, S, \alpha)$ is the variable $n$. If one lets $n = |z|$, then the quantifier $(\forall x \in \{0,1\}^n)$ is bounded. The first half, which expresses the condition $\alpha$ codes a valid circuit of size $t(n)$ is a $\forall\Delta_1^b$-formula with the universal quantifier ranging over numbers of length $t(n)^{O(1)}$. This universal quantifier will be bounded if $t$ is dominated by a polynomial, but will be unbounded otherwise.

When $S$ is a $\Sigma_1^b$-formula (an $NP$-formula) such as $SAT$, then $LB(|z|, t, S, \alpha)$ is an $\exists\Pi_1^b$-formula. It follows that it is provable in $S_2^2(\alpha)$ if and only if it is provable in $T_2^1(\alpha)$, since the latter theory is conservative over the former [5]. But Razborov [21] established the following theorem.

**Theorem 6** *Assume the SPRNG conjecture. Let $t(n) = n^{\omega(1)}$ be of superpolynomial growth rate. Then*

$$S_2^2(\alpha) \nvdash LB(|z|, t, S, \alpha).$$

Razborov's original proof was based on the characterization of the $\Sigma_1^b$-definable functions of $T_2^1$ (and thus $S_2^2$) in terms of polynomial local search functions, plus a version of the interpolation theorem. A second and simpler method for proving this theorem was developed by him in [19], see also Krajíček [12], which was also based on a Craig interpolation theorem. A complete proof based on Razborov's second method can be found in [7]; we omit the rather lengthy proof here, however.

The obvious question at this point is what is the import of Theorem 6. Firstly, if $S_2^2(\alpha)$ could be replaced by $U_1^1(\alpha)$, this would clearly become a very strong result that would say something quite meaningful about the difficulty

of proving that NP $\not\subseteq$ P/*poly*. But with $S^2_2(\alpha)$, the theorem is a good deal weaker: in particular, since $t(n) = n^{\omega(1)}$, $S^2_2(\alpha)$ is unable to code the circuit with a first-order object, and therefore cannot do induction on the number of gates in the circuit.

As an intermediate goal, midway between Theorem 6 and replacing $S^2_2(\alpha)$ with $U^1_1(\alpha)$, one might try the following: let $m = t(n)$ so that $n = m^{o(1)}$, which we express as $n = n(m)$. Then it would desirable to prove that

$$LB[n(|z|), |z|, S, \alpha]$$

is independent of $S^1_2(\alpha)$. In this setting, with the circuit size equal to $|z|$, $S^1_2(\alpha)$ would be able to code the circuit as a first-order object and be able to do induction on the number of gates in the circuit.

### 4.4 An Application of the SPRNG Conjecture

In section 3 we mentioned that, under the assumption that the SPRNG conjecture holds, there is a non-uniform probabilistic polynomial time algorithm computing $C \mapsto (x, y)$ such that for all $C$ encoding a Boolean circuit, $x$ codes a Boolean formula which is satisfied by $y$, but $C$ on input $x$ fails to output a satisfying assignment for $x$. In turns out that this is easy to prove from the SPRNG conjecture.

We need to describe the algorithm which computes $C \mapsto (x, y)$. The input to the algorithm is a value $C$ which is presumed to encode a polynomial size circuit with $n$ inputs. Since the circuit is polynomial size, $|C| = n^{O(1)}$. The desired values $(x, y)$ will witness the fact that $C$ does not correctly solve the satisfiability predicate. Let $m = n^{1/k}$ (the constant $k$ will be specified in a moment) and let $G_m$ be a polynomial size circuit which computes a pseudorandom number generator — the circuit $G_m$ exists since we are assuming the SPRNG conjecture holds. Thus $G_m : \{0,1\}^m \to \{0,1\}^{2m}$. Letting $z \in \{0,1\}^{2m}$, the statement

$$InRange_{G_m}(z) \;\Leftrightarrow\; (\exists u \in \{0,1\}^m)(Output(G_m, u) = z)$$

is an NP-predicate, and therefore can be expressed as an instance $x = x(z, G_m, n)$ of *SAT*. That is to say, $x(z, G_m, n)$ is polynomial time computable and $SAT(x(z, G_m, n))$ is true iff $z$ is in the range of $G_m$. We choose the constant $k$ so that $|x(z, G_m, n)| \leq m^k$; in fact, w.l.o.g., $|x(z, G_m, n)| = m^k = n$ and therefore the value $x(z, G_m, n)$ can be input to the circuit $C$.

We claim that for $u$ chosen at random from $\{0,1\}^m$ and for $z = Output(G_m, u)$, the probability that $C$ fails to output a satisfying assignment to $SAT(x(z, G_m, n))$ is greater than $1/2$; in fact the probability must be close to $1 - 1/S = 1 - 1/2^{m^\epsilon}$ or greater. To prove this claim, we use the inequality from the definition of the hardness of the pseudorandom number generator. Namely, for $u \in \{0,1\}^m$, let $x(u) = x(Output(G_m, u), G_m, n)$ and let $Prob_u[TRU(x(u), C(x(u)))] = a/2^m$.

From this, we have $Prob_y[TRU(y, C(y))] \leq a/2^{2m}$. Therefore, since $a/2^m - a/2^{2m} < 1/S$, we have that

$$Prob_u[TRU(x(y), C(x(u)))] = a/2^m < \frac{1}{S}(1 + \frac{2^{m+1}}{2^{2m}}) < \frac{1}{2},$$

and 1 minus this probability is the probability that $C$ fails to find a satisfying assignment on inputs $x(u)$ generated from values in the range of $G_m$.

The algorithm for computing $C \mapsto (x, y)$ is now easily describable: one repeatedly picks $u \in \{0,1\}^m$ at random and computes $z = Output(G_m, u)$ and $x = x(z, G_m, n)$. One also computes $y = y(u)$ which is a satisfying assignment to the formula $\phi_x$ encoded by $x$. If $Output(C, x)$ is not a satisfying assignment for $\phi_x$, then the value $(x, y)$ is output. Otherwise the process is repeated for another random $u$. Repeating the process a polynomial number of times, the probability that the algorithm succeeds is exponentially close to 1.

By the well-known technique of Adleman [1], the above non-uniform probabilistic algorithm can be shown to have polynomial size circuits. Namely, since for a given $C$, a random chosen $u$ succeeds with high probability, there must be a small set of values for $u$, such that for all $C$ of a given size, at least one of the $u$ values succeeds. In fact, only $\approx \log_{2^{n^\epsilon}}(2^n) = n^{1-\epsilon}$ values for $u$ are needed. Given these values for $u$ and the circuit $G_m$, it is easy to compute $x$ and $y$ in polynomial time. Therefore, the function $C \mapsto (x, y)$ is computable by a polynomial size circuit.

# 5 Interpolation and Cryptography

This section discusses a striking connection between open problems in bounded arithmetic and the security of cryptographic protocols. We discuss the work of Krajíček and Pudlák [13], which arose from the use of interpolation and disjoint NP-pairs in [21, 19]. We state first the results of [13] relating interpolation theorems and the RSA function and the discrete logarithm function. We conclude with an extension of their theorems to the Rabin encryption function.

**Definition** Let $\vec{x}$, $\vec{y}$, $\vec{z}$ be disjoint vectors of variables. Let $A(\vec{x}, \vec{y})$ and $B(\vec{x}, \vec{z})$ be formulas such that $A \rightarrow B$ is valid. Then a (Craig) interpolant for $A$ and $B$ is a formula $C(\vec{x})$ involving only the variables common to $A$ and $B$ such that both $A \rightarrow C$ and $C \rightarrow B$ are valid.

We shall be interested in situations where $A \rightarrow B$ is provable in a theory of bounded arithmetic, with $A$ and $B$ both $\Delta_1^b$-formulas. We are particularly interested in bounding the computational complexity of the interpolant $C$ (regardless of whether the implications $A \rightarrow C$ and $C \rightarrow B$ are provable in the theory of bounded arithmetic).

By using pairing, a vector of variables can be reduced to a single variable, so let $A(x, y)$ and $B(x, z)$ be in $\Delta_1^b$ w.r.t. a theory $T$ of bounded arithmetic. Then if $T$ proves $A \rightarrow B$, it also proves

$$(\exists y \leq s(x))A(x, y) \longrightarrow (\forall z \leq t(x))B(x, z).$$

In other words, $T$ proves that the two NP-sets

$$\mathfrak{A} \;=\; \{x : (\exists y \leq s(x))A(x,y)\}$$

and

$$\mathfrak{B} \;=\; \{x : (\exists z \leq t(x))(\neg B(x,z))\}$$

are disjoint. (Note that $B$ is negated!) An interpolant $C(x)$ such that $A \rightarrow C$ and $C \rightarrow B$ are valid would define a set $\mathfrak{C} = \{x : C(x)\}$ which *separates* $\mathfrak{A}$ and $\mathfrak{B}$ in that $\mathfrak{C} \supseteq \mathfrak{A}$ and $\overline{\mathfrak{C}} \supseteq \mathfrak{B}$. Therefore, the problem of finding an interpolant for $\Delta_1^b$-predicates is the same as finding a set that separates two $\Sigma_1^b$-predicates.

**Definition** A theory $T$ *has* P*-interpolants for* $\Delta_1^b$*-formulas* if and only if whenever $A(x,y)$ and $B(x,z)$ are formulas which are $\Delta_1^b$-definable w.r.t. $T$ such that $T$ proves

$$(\exists y \leq s(x))A(x,y) \rightarrow (\forall z \leq t(x))(x,z),$$

then they have an interpolant $C(x)$ which is polynomial time recognizable.

The following theorem was first stated in [13] in terms of P/*poly* instead of P:

**Theorem 7** *If* $S_2^1$ *has* P*-interpolants for* $\Delta_1^b$*-formulas, then the RSA function is not secure against polynomial time computability.*

It is conjectured that the RSA function function is secure against any type of feasible computability; this conjecture obviously implies that $S_2^1$ does not have P-interpolants for $\Delta_1^b$-formulas.

Next we state a second similar result from [13] for the discrete logarithm encryption function. But first, we need to introduce the notion of Pratt certificates for primes and a $\Pi_1^b$-sentence $\Phi$ that relates Pratt certificates to primality. If $N$ is a prime, then a Pratt certificate for $N$ consists of the following:

a. The prime factorization of $N-1$,

b. A generator $g$ for the multiplicative group $Z_N^*$, and

c. Pratt certificates for each of the primes in the prime factorization of part a.

One can check in polynomial time whether a purported Pratt certificate is valid, by checking that the product of the numbers in the prime factorization equals $N-1$, checking that the generator $g$ satisfies $g^{N-1} = 1$ and $g^{(N-1)/q} \neq 1$ for each prime $q$ in the factorization of $N-1$, and then recursively checking that the Pratt certificates for the factors of $N-1$ are valid. We write *Pratt*$(w,N)$ for the $\Delta_1^b$-predicate that expresses the condition that $w$ is a valid Pratt certificate for $N$.

It is open whether $S^1_2$ can prove that numbers with Pratt certificates are prime in the usual sense, so following [13], we let $\Phi$ be the $\forall\Delta^b_1$-formula

$$(\forall w)(\forall N)[\mathit{Pratt}(w,N) \rightarrow \neg(\exists a < N)(\exists b < N)(a \cdot b = N]).$$

and now work with the theory $S^1_2 + \Phi$.

**Theorem 8** *[13] If $S^1_2 + \Phi$ has $\mathrm{P}$-interpolants for $\Delta^b_1$-formulas, then the discrete logarithm encryption function is insecure.*

We will not define the RSA and discrete logarithm encryption functions but will instead introduce a different and somewhat simpler encryption function, namely, the Rabin function. We let $(a,b)$ denote the greatest common divisor of two integers $a$ and $b$. Let $N = p \cdot q$ be a product of two distinct primes, and let $0 < x < N$ such that $(x,N) = 1$. Further, let $y = x^2 \bmod N$. The function $x \mapsto y$ is the Rabin function [18]. The inverse to the Rabin function can be defined to be the set of four values $0 < x_1 < x_2 < x_3 < x_4 < N$ such that $x_i^2 \equiv y \bmod N$. It is known that if there is a feasible algorithm to compute the inverse to the Rabin function (as a function of $y$ and $N$ with $(y,N) = 1$), then there is a feasible algorithm for factoring integers [18, 2].

**Theorem 9** *Let $A(x)$ be any polynomial time predicate and let $\phi_A(x_1,x_2,x_3,x_4,w_p,w_q,p,q,y,N)$ be the formula*

$$N = p \cdot q \wedge \mathit{Pratt}(w_p,p) \wedge \mathit{Pratt}(w_q,q) \wedge (y,N) = 1 \wedge \bigwedge_{i=1}^{4} x_i^2 \equiv y \bmod N$$

$$\wedge x_1 < x_2 < x_3 < x_4 \wedge A(x_1,x_2,x_3,x_4).$$

*and define $\phi_{\neg A}$ similarly with $\neg A$ in place of $A$. Then $S^1_2$ proves*

$$\phi_A(x_1,x_2,x_3,x_4,w_p,w_q,p,q,y,N) \tag{1}$$

$$\longrightarrow \neg\phi_{\neg A}(u_1,u_2,u_3,u_4,w_{p'},w_{q'},p',q',y,N).$$

*Therefore, if $S^1_2 + \Phi$ has $\mathrm{P}$-interpolants for $\Sigma^b_1$-formulas, the Rabin function is insecure.*

**Proof** After the discussion at the beginning of this section, the only thing left to show is that the implication (1) is $S^1_2$-provable. It suffices to show that $S^1_2$ proves

$$\begin{aligned} N = p \cdot q \wedge N = p' \cdot q' \wedge \mathit{Pratt}(w_p,p) \wedge \mathit{Pratt}(w_q,q) \\ \wedge \mathit{Pratt}(w_{p'},p') \wedge \mathit{Pratt}(w_{q'},q') \wedge (y,N) = 1 \\ \wedge \bigwedge_{i=1}^{4} x_i^2 = y \bmod N \wedge \bigwedge_{i=1}^{4} u_i^2 \equiv y \bmod N \\ \wedge x_1 < x_2 < x_3 < x_4 \wedge u_1 < u_2 < u_3 < u_4 \\ \longrightarrow \bigwedge_{i=1}^{4} x_i = u_i \end{aligned} \tag{2}$$

It was shown in [13] that $S_2^1 + \Phi$ can prove that if $p \cdot q = p' \cdot q'$ and $p, q, p', q'$ have Pratt certificates, then $\{p, q\} = \{p', q'\}$. So in finding the $S_2^1 + \Phi$-proof of (2), we may assume w.l.o.g. that $p = p'$ and $q = q'$.

**Claim** $S_2^1 + \Phi$ can prove

$$v_1 < p \wedge v_2 < p \wedge \mathit{Pratt}(w, p) \wedge v_1^2 \equiv v_2^2 \bmod p \longrightarrow v_1 = v_2 \vee v_1 = p - v_2.$$

To prove the claim, consider the product

$$(v_1 - v_2) \cdot (v_1 + v_2) \;=\; v_1^2 - v_2^2 \;\equiv\; 0 \bmod p.$$

From the claim on page 213 of [13], $S_2^1 + \Phi$ can prove that one of the terms $v_1 - v_2$ or $v_1 + v_2$ is divisible by $p$ since their product is, and our claim is proved.

To finish the $S_2^1 + \Phi$ proof of (2), we note that since the values $x_i^2$ and $y_i^2$ are all equal modulo $N$, then they are also equal modulo $p$. Therefore, by the claim, there are only two possible values allowed for $x_i \bmod p$ and for $u_i \bmod p$. Likewise, there are only two possible values for $x_i \bmod q$ and $u_i \bmod q$. And, we have the following:

**Claim** $S_2^1 + \Phi$ can prove

$$0 \leq a \leq b < N \wedge a \equiv b \bmod p \wedge a \equiv b \bmod q \longrightarrow a = b.$$

To prove the claim, we argue inside $S_2^1 + \Phi$ and consider the difference $b - a$: this is congruent to $0 \bmod p$ and hence is, provably in $S_2^1 + \Phi$ a multiple of $p$, say $b - a = \alpha \cdot p$. By the reasoning of the previous claim, $\alpha$ is congruent to $0 \bmod q$ and so is a multiple of $q$. Therefore $b - a$ is a multiple of $p \cdot q$ and so $b - a = 0$. □

It would be desirable to remove the presence of the extra axiom $\Phi$ from Theorems 8 and 9; so far we have been unable to accomplish this.

**Acknowledgement:** We thank R. Impagliazzo for pointing out the construction of section 4.4 and J. Krajíček, P. Pudlák and C. Pollett for comments on a preliminary version of this paper.

# References

[1] L. Adleman, *Two theorems on random polynomial time*, in Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science, 1978, pp. 75–83.

[2] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, *RSA/Rabin bits are* $1/2+1/\mathit{poly}(\log N)$ *secure*, in Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science, 1984, pp. 449–457.

[3] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986. Revision of 1985 Princeton University Ph.D. thesis.

[4] ———, *First-order proof theory of arithmetic*. Typeset manuscript, to appear in *Handbook of Proof Theory*, 199?

[5] ———, *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.

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

[7] ———, *Bounded Arithmetic and Propositional Proof Complexity*, Springer-Verlag, Berlin, 1997, pp. 67–121.

[8] 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.

[9] P. Hájek and P. Pudlák, *Metamathematics of First-order Arithmetic*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1993.

[10] J. Krajíček, *Bounded Arithmetic, Propositional Calculus and Complexity Theory*, Cambridge University Press, 1995.

[11] ———, *Extensions of models of $PV$*. Typeset manuscript, 1996.

[12] ———, *Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic*, Journal of Symbolic Logic, 62 (1997), pp. 457–486.

[13] J. Krajíček and P. Pudlák, *Some consequences of cryptographic conjectures for $S^1_2$ and $EF$*, in Logic and Computational Complexity, D. Leivant, ed., Lecture Notes in Computer Science #960, Berlin, 1995, Springer-Verlag, pp. 210–220.

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

[15] 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.

[16] C. Pollett, *Arithmetic theories with prenex normal form induction*. Typeset manuscript, 1996.

[17] P. Pudlák, *The lengths of proofs*, in Handbook of Proof Theory, S. R. Buss, ed., Elsevier North-Holland, 1998, pp. 547–637.

[18] M. O. Rabin, *Digitalized signatures as intractable as factorization*, Tech. Rep. MIT/LCS/TR-212, MIT, 1979.

[19] 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.

[20] ——, *Bounded arithmetic and lower bounds in Boolean complexity*, in Feasible Mathematics II, P. Clote and J. Remmel, eds., Boston, 1995, Birkhäuser, pp. 344–386.

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

[22] 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.

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