Proofs, Codes, and Polynomial-Time Reducibilities

Ravi Kumar*
IBM Almaden

D. Sivakumar†
University of Houston

Abstract

We show how to construct proof systems for NP languages where a deterministic polynomial-time verifier can check membership, given any $N^{(2/3)+\epsilon}$ bits of an N-bit witness of membership. We also provide a slightly super-polynomial time proof system where the verifier can check membership, given only $N^{(1/2)+\epsilon}$ bits of an N-bit witness. In addition, we construct proof systems where a deterministic polynomial-time verifier can check membership, given an N-bit string that agrees with a legitimate witness on just $(N/2) + N^{(4/5)+\epsilon}$ bits.

Our results and framework have applications for two related areas of research in complexity theory: proof systems for NP, and the relative power of Cook reductions and Karp-Levin type reductions. Our proof techniques are based on algebraic coding theory and small sample space constructions.

The complexity class NP has served as the prime mover for the theory of computational complexity for nearly three decades. NP crisply captures the notion of efficient verifiability of proofs of membership, a single abstract property that is common to a wide variety of decision problems of tremendous significance. The interest in the P vs. NP question stems directly from our ambition to answer the question: can we efficiently solve those problems for which we know how to verify a solution efficiently?

Considering the fact that virtually all the important questions about NP still remain open, it is not surprising that the viewpoint of NP as the class of languages with efficient *proof systems* has been explored in much greater detail from various angles. Various refinements, generalizations, and restrictions of proof systems for NP languages have been pursued, and a rich body of results is known. Such studies have not only led to a significantly better understanding of the structure of NP, but have also had unexpected pay-offs in other areas as well. Some of the more significant pursuits along these lines include the work on interactive proof systems [GMR89, Bab85, BM88, LFKN92], multiple-prover interactive proof systems [GMW91, CCL94, BFL91], probabilistically checkable proof systems [BFLS91, AS92b, ALM+92], proof systems with verifiers of restricted complexity [Con93, AAI+97], as well as a considerable body of work on various structural aspects of proof systems [BKT94, JT95, BT96, FFNR96] (see [Sel96, JT97] for recent surveys).

0.1 Partially publishable proof systems

In this paper, we study proof systems for NP from the following viewpoint. Let L be a language in NP, and let $R_L(x, y)$ be a polynomial-time decidable binary predicate such that for all x, $x \in L$ iff

*IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA. email: ravi@almaden.ibm.com

†Research supported in part by NSF CAREER award CCR-9734164. Part of this work was done during a visit to IBM Almaden. Department of Computer Science, University of Houston, Houston, TX 77204, USA. email: siva@cs.uh.edu

there is a "witness string" w (of length polynomial in $|x|$) s.t. $R_L(x, w)$ holds. Notice that to test if $x \in L$ is satisfiable, the "verifier" R_L needs all the bits of the witness w. The main question we address is:

> For every $L \in$ NP, is there a polynomial-time decidable binary predicate $R'_L(x, \omega)$ such that $x \in L \iff (\exists \omega)\ R'_L(x, \omega)$, and *furthermore*, whether $R'_L(x, \omega)$ holds can be decided efficiently, given *any* subset of the bits (of sufficiently large size) of the witness ω? In addition, are there efficient transformations between witnesses w for the predicate R_L and witnesses ω for the predicate R'_L?

The parameter of interest here is the size of the subset of bits that the verifier is given.

One may think of such predicates as providing "partially publishable proof systems" for NP languages. It can be easily seen that the above question is equivalent to the following more naturally-stated variant in terms of circuit satisfiability. Given a Boolean circuit C with n input variables, produce a Boolean circuit C' in $N = n^{O(1)}$ variables such that:

(1) There is some input w s.t. $C(w) = 1$ iff there is some input ω s.t. $C'(\omega) = 1$.

(2) Given a partial assignment to the input variables of C' with the promise that it can be extended to an assignment ω s.t. $C'(\omega) = 1$, an assignment w to the input variables of C s.t. $C(w) = 1$ can be constructed efficiently.

We note that essentially the same question was raised recently by Gál, Halevi, Lipton, and Petrank [GHLP97], although with somewhat different motivations; in fact, this paper was directly motivated by the paper [GHLP97], and examines some of the same issues in the complexity-theoretic setting of proof systems. Our concrete results are stronger than those of [GHLP97].

Our partially publishable proof systems are also motivated by the recent success story of probabilistically checkable proof systems (PCP). The concern there is to design witness predicates $R''_L(x, \alpha)$ s.t. $x \in L \iff (\exists \alpha)\ R''_L(x, \alpha)$, and *furthermore*, whether $R''_L(x, \alpha)$ is TRUE can be decided by probing a randomly chosen set of $O(1)$ bits of α. The main difference between our proof systems and PCPs is the following: in PCPs, the verifier randomly chooses a set of bits of the witness, whereas in our setting, the verifier is *given* a set of bits of the witness that could be chosen *adversarially*. Consequently, we can't expect the verifier to function correctly when it is given sub-polynomial number of bits (unless NP is contained in sub-exponential time).

0.2 Proof systems and polynomial-time reducibilities

One of the long-standing open questions in complexity theory is whether Cook reducibility is strictly more powerful than Karp-Levin reducibility. This question can be refined further into a variety of more specific versions, some of which we state now:

(1) Is there a set $C \in$ NP that is complete for NP under Cook reductions but not under Karp-Levin reductions?

(2) Are there sets $A, B \in$ NP such that A is Cook-reducible to B but A is not reducible to B by a Karp-Levin reduction?

These questions are intimately tied to other major open questions in complexity theory. For example, an affirmative answer to either question above would imply that P $\neq$ NP. As a consequence,

very little is known unconditionally about this kind of questions. What is perhaps more disappointing is that not much is known, *even* under various complexity theory hypotheses. For example, since distinguishing Cook reducibility from the other reducibilities would separate NP from P, it might be natural to prove that these reducibilities are different under the assumption that $\mathrm{P} \neq \mathrm{NP}$; however, even this more modest goal is yet to be achieved.

A major application of Cook reducibility in complexity theory is in the reduction of NP search problems to NP decision problems. Every proof system for an NP language defines in the natural way a corresponding search problem, namely the problem of mapping instances of the language to their witnesses of membership under the proof system. A remarkable property of NP-complete languages is the fact that they are all self-reducible, and hence their search versions are reducible by Cook reductions to their decision versions. (Various refinements can be studied in this connection; see the papers [BF92, BG94, HNOS96, FFNR96] for several fine results in this direction.)

Being an (apparently) inherently sequential process with multiple queries, the task of reducing search to decision for NP languages offers a natural ground in which to compare the power of Cook reductions with that of the weaker reducibilities. However, we must note that the solution to a search problem is a (polynomially) long string, not just one bit, and hence we have to be careful in defining what Karp-Levin reducibility means in this context. It is clear that we need the decision oracle to supply us with $n^{\Omega(1)}$ bits (unless NP is contained in sub-exponential time). Therefore, one natural formulation of the Cook vs. weaker reductions question in this context is to ask if search reduces to decision for NP problems via *non-adaptive* Cook reductions, that is, Cook reductions in which all questions to the decision oracle are submitted simultaneously. The seminal result of Valiant and Vazirani [VV86] demonstrates, among other things, a probabilistic non-adaptive Cook reduction from the search versions for standard NP-complete languages to their decision versions.

We offer the notion of a partially publishable proof system that we study in this paper as a meaningful, interesting, and natural candidate for comparison with Cook reducibility in the context of reducing search to decision.

We begin by observing that the standard reduction from search to decision for NP languages would work just fine even if the oracle had the choice of which bit of the witness to give each time it is invoked. More concretely, suppose that we're given an n-variable Boolean formula φ, and we wish to construct a satisfying assignment for φ using some NP (decision) oracle. We may submit φ to it, and suppose it assigns the bit b_i to the variable x_i; we may set $x_i = b_i$ in φ to obtain a new formula φ' with which we may repeat the process until we obtain all the bits of a satisfying assignment for φ.

On the other hand, suppose that we're given an n-variable Boolean formula φ, and we wish to construct a satisfying assignment for φ using some NP (decision) oracle that we have *one-time* access to (as in a Karp-Levin reduction). This is precisely the setting of a partially publishable proof system, where the oracle gives us $n^{\Omega(1)}$ bits of *its* choice, from among all the (related) decision queries that we ask. We believe that this is a faithful adaptation of the notion of a Karp-Levin reduction for computing NP search functions. (We also note that this is incomparable to a non-adaptive Cook reduction from search to decision since in a partially publishable proof system, we demand the oracle to provide all the bits that it provides to be consistent with a single witness string.)

0.3 Main results

We construct partially publishable proof systems that can work with a very small (asymptotically vanishing) fraction of the witness bits. Let $L \in \text{NP}$ and let $R_L(x, y)$ be a polynomial-time decidable witness predicate that has the property $(\forall x)\ [(x \in L) \iff (\exists y)\ [|y| = |x|^{O(1)} \wedge R_L(x, y)]]$. Then we say that a partially publishable proof system for L is *optimal* if for every $\delta > 0$, there is a polynomial-time decidable witness predicate $R_L^\delta(x, z)$, where the size of the witnesses $|z| = N = |x|^{O(1)}$, and where the verifier can make its decisions when given any N^δ of the N bits of the witness z. (Such a proof system, if it exists, cannot be improved unless NP is contained in sub-exponential time.)

Our proof systems are not optimal, but come close. We present three proof systems; the first one is a polynomial-time proof system that works for $\delta > 3/4$. Then we present a proof system construction (essentially the derandomized version of a probabilistic argument) that works for $\delta > 1/2$, but where the proof length and the running time of the verifier are slightly super-polynomial. Inspired by this proof system, we present a third proof system that works in polynomial time, and works for $\delta > 2/3$. (The results of [GHLP97] that partly motivated our work can be used to give probabilistically defined proof systems for $\delta > 1/2$. However the proof systems thus obtained are probabilistically defined, i.e., it is not just that the verifier in the proof system is probabilistic, but what we obtain is really a distribution over proof systems. One may obtain genuine proof systems from their work, but that would make the verifier *non-uniform*, that is, the verifier has some auxiliary information of polynomial size. It is neither known nor clear how to derandomize their probabilistic argument.)

Since in all our constructions the verifier transforms a partially supplied witness z for the predicate R_L^δ into a (full) witness y for the predicate R_L, we reap the following result as a by-product: the analogue of Karp-Levin reductions defined above from search to decision is nearly as powerful as the standard Cook reductions from search to decision. This follows from the fact that every bit of the witness z in the predicate $R_L^\delta(x, z)$ is an NP decision question.

Our main proof techniques involve building various bit-level *erasure codes*, where one can recover the message word from a very small (asymptotically vanishing) fraction of the bits of the code word. An especially crucial component in all of our constructions is the recent surprisingly powerful polynomial reconstruction algorithm due to Madhu Sudan [Sud96], building on earlier work by Ar *et al.* [ALRS92]. This elegant algorithm has already found many complexity applications [AS97, Siv98, CPS98]. Our bit-level erasure code constructions based on this algorithm may also find other applications in complexity theory.

0.4 Extensions

We generalize the notion of a partially publishable proof system to that of a *noisy* proof system for NP languages. Here the verifier is given a string ω_1 of length N that agrees with a legitimate witness ω for $x \in L$ on $\geq (N/2) + N^{\frac{1}{2}+\epsilon}$ positions. The positions where ω_1 agrees with ω may be chosen adversarially. Note that if we produce a random string ω_r of length N, then we can expect that with good probability, ω_r will agree with ω on $\geq (N/2) + O(\sqrt{N})$ positions. Thus the question here is, if we're given a string that agrees with a good witness on just slightly (but asymptotically) more than places than a random string would, can we then decide membership in L?

It can be shown that variants of our constructions in Section 1, along with a bound due to S. Johnson [MS77], can be used to solve this problem for $\epsilon > 3/10$. Extending this to arbitrary $\epsilon > 0$ appears to be a challenging and interesting open question both from the viewpoint of complexity theory and from the viewpoint of designing error-correcting codes with special properties.

1 Erasure-resilient proof systems

Let $L \in \text{NP}$, and let $R_L(x, y)$ be a standard witness predicate s.t. for some constant c and for all x, $x \in L \iff (\exists y)\ [|y| = |x|^c \ \wedge\ R_L(x, y)]$. The various witness predicates R we construct for the partially publishable proof systems all have the following high-level structure: for some constant d (usually d will be much larger than c) and for all x,

$$x \in L \iff (\exists z)\ [|z| = |x|^d \ \wedge\ (\exists y)\ [|y| = |x|^c \ \wedge\ z = E(y) \ \wedge\ R_L(x, y)]],$$

where E is a suitable erasure code, that is, a code where y may be constructed given sufficiently many bits of z.

In general, erasure codes are not as well-understood as error-correcting codes; there has been some renewed interest in these in recent years [AEL95, LMS+97]. However, the recent constructions have all been aimed towards constructing efficiently encodable and decodable erasure codes, and not so much on the fraction of erasures that can be tolerated. More importantly, they all have the more stringent requirement that the message word be uniquely identified from parts of the code word. However, as we shall see, for our problem, the more general *list decoding* is sufficient (where the decoder is allowed to produce a list of message words that is guaranteed to contain the original message word); this flexibility allows us to tolerate significantly more erasures, a property crucial to obtaining near-optimal proof systems in our framework.

1.1 The top-level construction

In the subsequent sections, we present three erasure code constructions that we use for the proof system constructions. All the codes we construct are compositions of the well-known Reed-Solomon code (or polynomial codes) with appropriately designed "inner codes." We will rely heavily on the powerful decoder for Reed-Solomon codes, developed by Madhu Sudan [Sud96], who builds on earlier work by Ar *et al.* [ALRS92].

Theorem 1 ([Sud96]) *Let* F *be a finite field. Given* L *distinct pairs* $(u_i, v_i) \in \text{F} \times \text{F}$ *for* $i = 1, \ldots, L$*, in time polynomial in* L*,* d*, and* $|\text{F}|$*, one can produce a list of at most* $\sqrt{L/d}$ *polynomials over* F *that contains every polynomial* p *of degree at most* d *that satisfies* $|\{i \mid v_i = p(u_i)\}| \geq \sqrt{2Ld}$*.*

Depending on the characteristics of the inner code that we use, we obtain proof systems of varying levels of optimality. In all our discussions, we will constantly refer to the (arbitrary) NP language L introduced above, and some canonical witness predicate R_L for L.

1.2 Hadamard codes

In this section, we construct a witness predicate $R_1(x, y)$ by choosing the Hadamard code as the inner code to be composed with the outer Reed-Solomon code. Concretely, we will solve the following problem: and obtain given an n-bit string y (think of as the witness for $x \in L$), encode it as an N-bit string z where $N = n^{O(1)}$ such that given any $N^{\frac{3}{4}+\epsilon}$ bits of z, y can be reconstructed in polynomial time.

With hindsight, we will choose $q \geq (4n)^{\frac{1}{3\epsilon}}$ to be an appropriate power of two; by choosing q to be of specific nice forms (*eg.*, $q = 2^{2 \cdot 3^\ell}$), we can in fact make the finite field construction and computations extremely simple. We will also assume that $q^\epsilon \geq 2$. Construct the field F_q of q elements.

Interpret the n-bit message $y = y_{n-1}y_{n-2}\ldots y_1y_0$ as the coefficients of the polynomial $p_y(u) = \sum_i y_i u^i$ over the finite field F_q. For convenience, we will list the elements of F_q as $u_1, u_2, \ldots, u_q$. Evaluate p_y on all elements of the finite field F_q, to produce the following first-level encoding: $(p_y(u_1), p_y(u_2), \ldots, p_y(u_q))$. This is the first-level Reed-Solomon encoding. Note that each value $p_y(u_i)$ is an element of F_q, hence has a $(\log_2 q)$-bit representation, treating F_q as a vector space over GF(2). In fact, F_q is an inner product space, endowed with the inner product $\langle v, w\rangle$ defined by $\langle v, w\rangle = \sum_j v^{(j)}w^{(j)}$, where $v^{(j)}$ denotes the j-th bit of v and $w^{(j)}$ denotes the j-th bit of w, and the arithmetic is done over GF(2).

The second-level encoding is done with a Hadamard code, as follows: For each i, let $v_i = p_y(u_i)$, and treat the bits of v_i as representing the linear function over F_q that maps an element w to $\langle v_i, w\rangle$. In the second level of encoding, each v_i is now replaced by evaluating the inner products $\langle v_i, w\rangle$ for all $w \in \mathrm{F}_q$. Thus the total length of the code is exactly $M = q^2$ bits.

Suppose we are given at least $M^{\frac{3}{4}+\epsilon}$ bits of a code word. We will think of the q^2 bits of the codeword as q *blocks* of q bits each; the i-th block corresponds to the evaluation of the polynomial p_y on the element $u_i \in \mathrm{F}_q$ and then encoding $p_y(u_i)$ by the second-level (Hadamard) code of all GF(2) inner products $\langle p_y(u_i), w\rangle$ for all $w \in \mathrm{F}_q$.

Given that we are given at least $M^{\frac{3}{4}+\epsilon} = q^{\frac{3}{2}+2\epsilon}$ bits of a valid codeword, the average number of bits exposed within a block is $\geq q^{\frac{1}{2}+2\epsilon}$. Equivalently, $\xi \leq q - q^{\frac{1}{2}+2\epsilon}$ is the average number of missing bits in a block. We will call a block *good* if $> q^{\frac{1}{2}+\epsilon}$ bits in the block are exposed, and we will call it *bad* otherwise. By Markov's inequality, the number s of bad blocks satisfies

$$\begin{aligned}
s &\leq q \cdot \frac{\mathrm{Exp}[\xi]}{q - q^{\frac{1}{2}+\epsilon}} \\
&= q \cdot \frac{q - q^{\frac{3}{2}+2\epsilon}}{q - q^{\frac{1}{2}+\epsilon}} \\
&= q \cdot \frac{(q - q^{\frac{3}{2}+\epsilon}) - (q^{\frac{3}{2}+2\epsilon} - q^{\frac{3}{2}+\epsilon})}{q - q^{\frac{1}{2}+\epsilon}} \\
&= q - \frac{q^{\frac{3}{2}+2\epsilon} - q^{\frac{3}{2}+\epsilon}}{q - q^{\frac{1}{2}+\epsilon}} \\
&\leq q - \frac{q^{\frac{3}{2}+2\epsilon}/2}{q} \\
&= q - (q^{\frac{1}{2}+2\epsilon}/2).
\end{aligned}$$

(The last "$\leq$" in this chain uses the assumption $q^\epsilon \geq 2$.)

Thus at least $q^{\frac{1}{2}+2\epsilon}/2$ blocks are good. Recall that in a good block, at least $q^{\frac{1}{2}+\epsilon}$ bits are exposed. Suppose the i-th block is good. The bits in the i-th block are the inner products $\langle p_y(u_i), w\rangle$ for all $w \in \mathrm{F}_q$. Since the i-th block is good, we have at least $q^{\frac{1}{2}+\epsilon}$ out of the q inner products of $p_y(u_i)$. These inner products thus give us at least $q^{\frac{1}{2}+\epsilon}$ linear equations (over GF(2)) on the $\log_2 q$ bits of $p_y(u_i)$. Since there are at least $q^{\frac{1}{2}+\epsilon}$ linear equations, the rank of the system of equations must be at least $(\frac{1}{2}+\epsilon)\log_2 q$. This gives us at most $2^{\log_2 q - (\frac{1}{2}+\epsilon)\log_2 q} = q^{\frac{1}{2}-\epsilon}$ possibilities for the value of $p_y(u_i)$.

In summary, we have at least $q^{\frac{1}{2}+2\epsilon}/2$ good blocks, and if the i-th block is good, we have at most $q^{\frac{1}{2}-\epsilon}$ possibilities for the value of $p_y(u_i)$.

The decoding algorithm is now simple: Discard all blocks that are not good, and restrict attention to exactly $q^{\frac{1}{2}+2\epsilon}/2$ good blocks. Produce a list of pairs (u_i, v_i), where i-th block is good, and v_i is one the of the remaining possibilities for $p_y(u_i)$. This gives us a total of $L = (q^{\frac{1}{2}+2\epsilon}/2)q^{\frac{1}{2}-\epsilon} = q^{1+\epsilon}/2$ distinct pairs, of which at least $q^{\frac{1}{2}+2\epsilon}/2$ pairs are of the form $(u_i, p_y(u_i))$. Recall that p_y is a polynomial of degree less than n; also note that $q^{\frac{1}{2}+2\epsilon}/2 \geq \sqrt{(2nq^{1+\epsilon}/2)}$ is implied by the choice $q \geq (4n)^{\frac{1}{3\epsilon}}$. Thus the algorithm of [Sud96] will produce a list of polynomials that is guaranteed to contain p_y.

This completes the description of the erasure code, which we will call E_1. Returning to the original problem, we define the witness predicate R_1 for L as

$$R_1(x, z) \equiv [|z| = |x|^d \ \wedge \ (\exists y) \ [|y| = |x|^c \ \wedge \ z = E_1(y) \ \wedge \ R_L(x, y)]].$$

Suppose that $x \in L$, and let y of length $n = |x|^c$ be some witness s.t. $R_L(x, y)$ holds. Let N be the length of $z = E_1(y)$, and suppose we're given $N^{\frac{3}{4}+\epsilon}$ bits of z; by applying the decoding procedure above, we will have a list of at most polynomially many strings that is guaranteed to contain y.

The punch line is that since R_L is a polynomial-time decidable predicate, we can check, for each y produced by the decoder, if $R_L(x, y)$ holds. Clearly, such a y will be present if and only if $x \in L$.

We have proved the following result:

Theorem 2 *For every language $L \in$ NP, every witness predicate R_L for L and every $\epsilon > 0$, there is a (partially publishable) witness predicate R_1 s.t. given any $N^{\frac{3}{4}+\epsilon}$ of an N-bit witness z that satisfies $R_1(x, z)$, one can construct in polynomial time a witness y that satisfies $R_L(x, y)$.*

□

1.3 A quasi-polynomial time proof system

In this section, we use a probabilistic argument to show the existence of better inner codes (than the Hadamard code), and use such codes to define proof systems. However, to be a legitimate proof system, we need a uniform way to define these codes, not merely argue that they exist. For this purpose, we show how our argument can be derandomized easily to obtain appropriate inner codes that have the desired properties. By doing this, we construct a slightly super-polynomial time proof system whose verifier makes the correct decision when given any $N^{\frac{1}{2}+\epsilon}$ bits of an N-bit witness.

The plan of action is similar to the construction of previous erasure code, with Reed-Solomon code over a field F_q used as the outer code. For the inner code, we will first provide a probabilistic construction. Let α be such that

$$\frac{(1/2) - \epsilon}{(1/2) + \epsilon} < \alpha < \frac{(1/2) + \epsilon}{(1/2) - \epsilon},$$

and let $b > \alpha$. It is easy to see that such choices for α and b exist for any $\epsilon > 0$. Furthermore, pick $q > n^{(1/\alpha)+\gamma}$ to be a power of two for some tiny $\gamma > 0$. The reason for these choices of α will become clear shortly.

To encode an n-bit witness string y, we will evaluate the polynomial $p_y(u) = \sum_j y_j u^j$ on all $u \in F_q$. Next we will encode each value obtained this way by an inner code block of length q^α (to be described). Thus the total code length N is $q^{1+\alpha}$. Suppose that we're given at least $N^{\frac{1}{2}+\epsilon}$

of the N bits. This quantity is $q^{(1+\alpha)((1/2)+\epsilon)}$ bits. The average number of bits given per inner code block is $q^{(1+\alpha)((1/2)+\epsilon)-1}$, and the average number of missing bits per inner code block is $\xi \leq q^\alpha - q^{(1+\alpha)((1/2)+\epsilon)-1}$. Call a block *bad* if it has more than $q^\alpha - (1/2)q^{(1+\alpha)((1/2)+\epsilon)-1}$ missing bits, and call it *good* otherwise. By Markov's inequality, the number s of bad blocks obeys

$$\begin{aligned} s &\leq q \cdot \frac{\mathrm{Exp}[\xi]}{q^\alpha - (1/2)q^{(1+\alpha)((1/2)+\epsilon)-1}} \\ &\leq q \cdot \frac{q^\alpha - q^{(1+\alpha)((1/2)+\epsilon)-1}}{q^\alpha - (1/2)q^{(1+\alpha)((1/2)+\epsilon)-1}} \\ &\leq q - \frac{1}{2}q^{(1+\alpha)((1/2)+\epsilon)-\alpha}. \end{aligned}$$

Thus the number of good blocks G is at least $(1/2)q^{\epsilon+\epsilon\alpha+(1/2)-(\alpha/2)}$. In each good block, we have at least $(1/2)q^{\epsilon+\epsilon\alpha-(1/2)+(\alpha/2)}$ bits. By the choice of α, the exponent here is > 0; pick a constant less than this exponent, say β; thus we have in each good block at least q^β bits. For sufficiently large n (and hence q), $q^\beta \gg b \lg q$ (see definition of b above).

Now we spell out the property we need in the inner code. Suppose that we have a code where for any given assignment to $b \lg q$ out of q^α positions, there are no more than $a \lg q$ code words (for some constant $a > 1$) that agree with the given assignment on the given positions. Then we will have for at least G blocks corresponding to $u_i \in \mathrm{F}_q$, at most $a \lg q$ values for the value of $p_y(u_i)$.

Let us suppose for the moment that such a code exists. Now the decoding algorithm is simple: simply discard blocks that are not good; we know that we will have G good blocks, where $G \geq (1/2)q^{\epsilon+\epsilon\alpha+(1/2)-(\alpha/2)}$. Furthermore, if block i is good, we have no more than $c \lg q$ candidates for the value of $p_y(u_i)$; we produce all the pairs $(u_i, p_y(u_i))$ with these candidate values for $p_y(u_i)$. The choices of α and q once again imply that $G \geq \sqrt{2G(a \lg q)n}$, and hence the algorithm of [Sud96] will produce all polynomials that agree with at least G of the pairs given.

This completes the description of the erasure code (modulo the description of the inner code); we will call this erasure code call E_2. Returning to the original problem, we define the witness predicate R_2 for L as

$$R_2(x, z) \equiv [|z| = |x|^d \ \wedge \ (\exists y) \ [|y| = |x|^c \ \wedge \ z = E_2(y) \ \wedge \ R_L(x, y)]].$$

Suppose that $x \in L$, and let y of length $n = |x|^c$ be some witness s.t. $R_L(x, y)$ holds. Let N be the length of $z = E_2(y)$, and suppose we're given $N^{\frac{1}{2}+\epsilon}$ bits of z; by applying the decoding procedure above, we will have a list of at most polynomially many strings that is guaranteed to contain y. Once again, since R_L is a polynomial-time decidable predicate, we can check, for each y produced by the decoder, if $R_L(x, y)$ holds. Clearly, such a y will be present if and only if $x \in L$.

It remains to show how such an inner code can be constructed. Suppose that we pick a $q^\alpha \times q$ matrix M of 0's and 1's randomly (and independently) by picking each entry of M. Then the probability that M fails to be a code with the desired property is clearly upper bounded by

$$\binom{q^\alpha}{a \lg q}\binom{q}{b \lg q} \ 2^{b \lg q} \ 2^{-(ab(\lg q)^2)} \quad \leq \quad 2^{a\alpha(\lg q)^2 + b(\lg q)^2 + b \lg q - ab(\lg q)^2} \quad < \quad 1,$$

since $b > \alpha$ and $a > 1$. Here the first factor comes from summing over all ways of choosing of $a \lg q$ rows of M, the second factor comes from summing over all ways of choosing $b \lg q$ columns of M, the third factor comes from summing over all $b \lg q$ bit patterns t, of the probability that a certain

$a \lg q \times b \lg q$ sub-matrix has all its rows equal to the pattern t, which is the fourth factor, namely $2^{-ab(\lg q)^2}$.

The key observation in derandomizing this argument is to note that in the evaluation of the probability that a certain $a \lg q \times b \lg q$ sub-matrix has all its rows equal to some pattern t, the only events of the probabilistic process that we need to be independent are precisely the generation of the entries of this sub-matrix. Thus, the same probability bound applies even if we select the $q^{1+\alpha}$ bits of the matrix M in an ℓ-wise independent manner, where $\ell = ab(\lg q)^2$, that is, every set of ℓ of the $q^{1+\alpha}$ bits of M are made independently of each other. Such a choice can be made efficiently with as few as $(\lg q)(1+\alpha)(ab(\lg q)^2/2) = O((\lg q)^3)$ random bits [AS92a, Chapter 15.2]. Thus an algorithm that cycles through all such sequences of random bits runs in time slightly super-polynomial in q, and hence in n.

In the proof system, therefore, the verifier first cycles through all the sequences of $O((\lg q)^3)$ random bits and uses the first suitable matrix for the inner code; the witness predicate is formally defined by using the first such matrix as the inner code in E_2. We have proved the following result:

Theorem 3 *For every language $L \in$ NP, every witness predicate R_L for L and every $\epsilon > 0$, there is a (partially publishable) witness predicate R_2 s.t. given any $N^{\frac{1}{2}+\epsilon}$ of an N-bit witness z that satisfies $R_2(x,z)$, one can construct in polynomial time a witness y that satisfies $R_L(x,y)$.*

□

1.4 A construction using small sample spaces

Partly inspired the result of the last section, in this section, we use an "ℓ-wise δ-biased sample space" construction of [NN93, AGHP92] as the inner code, and construct a proof whose verifier works correctly, given any $N^{\frac{2}{3}+\epsilon}$ bits of an N-bit witness.

Definition 1 ([NN93, AGHP92]) *A $Q \times k$ matrix M is said to be an* ℓ-wise δ-biased sample space of k variables *if for any subset S of ℓ columns and any ℓ-bit vector t, the number of rows in which the pattern t occurs in the columns in S is within the interval $[(Q/2^\ell) - \delta Q, (Q/2^\ell) + \delta Q]$.*

The goal in these sample space constructions is to make Q as small as possible. Combining an idea of Naor and Naor [NN93] with a construction in [AGHP92], the authors of [AGHP92] build ℓ-wise δ-biased sample spaces of k variables with $\lg Q = (2 + o(1))(\lg \ell + (\ell/2) + \lg \lg k + \lg \delta^{-1})$. It turns out that in their construction, $\lg Q$ is an integer, a fact that is convenient to work with.

We will use such matrices as inner codes in erasure code constructions in the following way. To encode an n-bit string y, as before, we will view it as the polynomial $p_y(u) = \sum_j y_j u^j$, and evaluate it on all the q elements of a finite field F_q (the exact value of q will be specified shortly). Then as the inner encoding we do the following: Suppose $v_i = p_y(u_i)$; the value v_i is a $(\lg q)$-bit value, which we will treat as the index of a row in a $Q \times k$ matrix M of the kind described above, and encode v_i by the corresponding row of M. Our choices for the various parameters are: $Q = O(q)$, $k = q^\alpha$ where α satisfies the conditions

$$\frac{(1/3) - \epsilon}{(2/3) + \epsilon} < \alpha < \frac{(1/6) + \epsilon + \gamma}{(1/3) - \epsilon},$$

where $0 < \gamma < \epsilon$ is a very small constant, $\ell = C \log q$ for some constant $C > 1$, and $\delta = 1/\sqrt{q}$. It can be easily verified that for any $\epsilon > 0$, α can be chosen to satisfy the above condition; it might

help the reader to note that $\alpha \approx (1/2)$. Note that the choice of α implies that

$$\tau \doteq \epsilon + \epsilon\alpha + (2/3) - \alpha - 1/2 > 0,$$

and we take $q > n^{1/\tau}$.

The crucial fact that we need is that if, for some $\beta > 0$, we know at least $q^\beta \gg \ell$ bits of some k-bit vector that is purported to be a row of the matrix M, then there could be no more than $Q/2^\ell + \delta Q \leq O(\sqrt{q})$ rows of M that agree with the given bits in the given positions. (The reason for choosing a Q different from q is just to handle the pesky $o(1)$ term in the construction parameters of [AGHP92]; for easier reading, we may pretend that $Q = q$.)

Thus the total code length N is (ignoring lower order terms from the $o(1)$ term) $O(q^{1+\alpha})$. Suppose that we're given at least $N^{\frac{2}{3}+\epsilon}$ of the N bits. We will view the code as having q blocks of q^α bits each. The average number of bits given per block is at least $q^{(1+\alpha)((2/3)+\epsilon)-1}$, and the average number of missing bits per block is $\xi \leq q^\alpha - q^{(1+\alpha)((2/3)+\epsilon)-1}$. Call a block *bad* if it has more than $q^\alpha - (1/2)q^{(1+\alpha)((2/3)+\epsilon)-1}$ missing bits, and call it *good* otherwise. By Markov's inequality, the number s of bad blocks is at most

$$\begin{aligned} s &\leq q \cdot \frac{\mathrm{Exp}[\xi]}{q^\alpha - (1/2)q^{(1+\alpha)((2/3)+\epsilon)-1}} \\ &\leq q \cdot \frac{q^\alpha - q^{(1+\alpha)((2/3)+\epsilon)-1}}{q^\alpha - (1/2)q^{(1+\alpha)((2/3)+\epsilon)-1}} \\ &\leq q - \frac{1}{2}q^{(1+\alpha)((2/3)+\epsilon)-\alpha}. \end{aligned}$$

Thus the number of good blocks is at least $(1/2)q^{\epsilon+\epsilon\alpha+(2/3)-(\alpha/3)}$. In each good block, we have at least $(1/2)q^{\epsilon+\epsilon\alpha-(1/3)+(2\alpha/3)}$ bits. By the choice of α, the exponent here is > 0; pick a constant less than this exponent, say β; thus we have in each good block at least q^β bits. Therefore, there are at most $O(\sqrt{q})$ code words in the inner code that could agree with the given bits (see remarks above).

Now the decoding algorithm is simple: simply discard blocks that are not good; we know that we will still have at least G good blocks, where $G \geq (1/2)q^{\epsilon+\epsilon\alpha+(2/3)-(\alpha/3)}$. Furthermore, if block i is good, we have no more than $O(\sqrt{q})$ candidates for the value of $p_y(u_i)$; we produce all the pairs $(u_i, p_y(u_i))$ with these candidate values for $p_y(u_i)$. The choices of α and q once again imply that $G \geq O(\sqrt{G\sqrt{q}n})$, and hence the algorithm of [Sud96] will produce all polynomials that agree with at least G of the pairs given.

We will call this erasure code E_3. We are now ready to define a new witness predicate for L, as follows:

$$R_3(x,z) \equiv [|z| = |x|^d \ \wedge \ (\exists y)\ [|y| = |x|^c \ \wedge \ z = E_3(y) \ \wedge \ R_L(x,y)]].$$

If $x \in L$ and we are given at least $N^{\frac{2}{3}+\epsilon}$ bits of an N-bit witness z, some witness y s.t. $R_L(x,y)$ holds will be found, and it can be decided whether $x \in L$. Thus we have proved:

Theorem 4 *For every language $L \in$ NP, every witness predicate R_L for L and every $\epsilon > 0$, there is a (partially publishable) witness predicate R_3 s.t. given any $N^{\frac{2}{3}+\epsilon}$ of an N-bit witness z that satisfies $R_3(x,z)$, one can construct in polynomial time a witness y that satisfies $R_L(x,y)$.*

□

Acknowledgments

We are grateful to Erez Petrank for making [GHLP97] available on his home-page. Thanks to Anna Gál and Richard Lipton for helpful conversations during STOC '98 and Complexity '98.

References

[AAI+97] M. Agrawal, E. Allender, R. Impagliazzo, S. Rudich, and T. Pitassi. Reducing the complexity of reductions. In *Proc. 29th Annual ACM Symposium on the Theory of Computing*, pages 730–738, 1997.

[AEL95] N. Alon, J. Edmonds, and M. Luby. Linear time erasure codes with nearly optimal recovery. In *Proc. 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 512–519, 1995.

[AGHP92] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–303, 1992. A preliminary version appeared in FOCS 1991.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[ALRS92] S. Ar, R. Lipton, R. Rubinfeld, and M. Sudan. Reconstructing algebraic functions from erroneous data. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 503–512, 1992.

[AS92a] N. Alon and J. Spencer. *The Probabilistic Method*. Wiley, 1992. With an appendix by P. Erdős.

[AS92b] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[AS97] S. Arora and M. Sudan. Improved low-degree testing and its applications. In *Proc. 29th Annual ACM Symposium on the Theory of Computing*, 1997.

[Bab85] L. Babai. Trading group theory for randomness. In *Proc. 17th Annual ACM Symposium on the Theory of Computing*, pages 421–429, 1985.

[BF92] R. Beigel and J. Feigenbaum. On being incoherent without being very hard. *Computational Complexity*, 2:1–17, 1992.

[BFL91] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd Annual ACM Symposium on the Theory of Computing*, pages 21–31, 1991.

[BG94] M. Bellare and S. Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, 1994.

[BKT94] H. Buhrman, J. Kadin, and T. Thierauf. On functions computable with nonadaptive queries to NP. In *Proc. 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 43–53, 1994.

[BM88] L. Babai and S. Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *Journal of Comp. Sys. Sci.*, 36:254–276, 1988.

[BT96] H. Buhrman and T. Thierauf. The complexity of generating and checking proofs of membership. In *Proc. 13th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1046 of *Lecture Notes in Computer Science*, pages 75–86. Springer-Verlag, 1996.

[CCL94] J. Cai, A. Condon, and R. Lipton. PSPACE is provable by two provers in one round. *Journal of Comp. Sys. Sci.*, 48(1):183–193, 1994.

[Con93] A. Condon. The complexity of space-bounded interactive proof systems. In S. Homer, U. Schöning, and K. Ambos-Spies, editors, *Complexity Theory: Current Research*, pages 147–190. Cambridge University Press, 1993.

[CPS98] J. Cai, A. Pavan, and D. Sivakumar. On the hardness of permanent, 1998. Manuscript.

[FFNR96] S. Fenner, L. Fortnow, A. Naik, and J. Rogers. On inverting onto functions. In *Proc. 11th Annual IEEE Conference on Computational Complexity*, pages 213–222, 1996.

[GHLP97] A. Gál, S. Halevi, R. Lipton, and E. Petrank. Computing from partial solutions, 1997. Manuscript.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18:186–208, 1989.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity, or All languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38:691–729, 1991.

[HNOS96] E. Hemaspaandra, A. Naik, M. Ogihara, and A. Selman. P-selective sets, and reducing search to decision vs. self-reducibility. *Journal of Comp. Sys. Sci.*, 53(2):194–209, 1996.

[JT95] B. Jenner and J. Toŕan. Computing functions with parallel queries to NP. *Theoretical Computer Science*, 141:175–193, 1995.

[JT97] B. Jenner and J. Toŕan. The complexity of obtaining solutions for problems in NP. In L. Hemaspaandra and A. Selman, editors, *Complexity Theory Retrospective II*, pages 155–178. Springer-Verlag, 1997.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39:859–868, 1992.

[LMS+97] M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proc. 29th Annual ACM Symposium on the Theory of Computing*, 1997.

[MS77] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal on Computing*, 22:838–856, 1993. A preliminary version appeared in STOC 1990.

[Sel96] A. Selman. Much ado about functions. In *Proc. 11th Annual IEEE Conference on Computational Complexity*, pages 198–212, 1996.

[Siv98] D. Sivakumar. On membership comparable sets. In *Proc. 13th Annual IEEE Conference on Computational Complexity*, pages 2–7, 1998.

[Sud96] M. Sudan. Maximum likelihood decoding of Reed-Solomon codes. In *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 164–172, 1996.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.