Efficient Reductions from NP to Parity using Error-Correcting Codes (preliminary version)

Kenneth W. Regan*
State University of New York at Buffalo

July 8, 1993

Abstract

This paper proves that every language in NP is recognized by an RP[⊕P] machine whose time complexity is quasilinear, apart from the time to verify witnesses. The results significantly improve the number of random bits, success probability, and running time of Valiant and Vazirani's original construction [VV86], and beat both the $2n$ random bits and time/success tradeoff in subsequent methods based on universal hashing. Questions of further improvements are connected to open problems in the theory of error-correcting codes.

1. Introduction

Valiant and Vazirani [VV86] proved that every language $L \in$ NP is accepted by a polynomial-time bounded probabilistic oracle Turing machine M_o which, on any input x, computes a Boolean formula ϕ_x as an oracle query, and accepts if the answer is '1'. The oracle of M_o has the "promise property" that whenever ϕ_x has a unique satisfying assignment, it always answers '1', and whenever ϕ_x has no satisfying assignments, it answers '0'. An oracle which returns the parity of the number of satisfying assignments has this property, and it follows that NP $\subseteq$ RP[⊕P]. Toda [Tod89, Tod91] used this to show that the polynomial hierarchy (PH) is contained in BP[⊕P]. Toda and Ogiwara [TO91, TO92] extended this to obtain PH $\subseteq$ BP[$C_=$P] and related results, while Tarui [Tar91, Tar93] obtained similar results with zero error probability. Allender [All89] used the Valiant-Vazirani construction and techniques from the second part of Toda's paper [Tod91] in a circuit setting to prove that AC^0 circuits can be simulated by small depth-3 threshold circuits, and Allender and Hertrampf [AH90] gave a uniform version of this construction. Some other

*The author was supported in part by NSF Research Initiation Award CCR-9011248
Author's current address: Computer Science Department, 226 Bell Hall, UB North Campus, Buffalo, NY 14260-2000. Email: *regan@cs.buffalo.edu*, tel.: (716) 645–3189, fax: (716) 645–3464.

results apprear to require constructions which work under all oracles that have the promise property (see [BT91, BRS91]), rather than just the parity oracle. Because of all these applications, there is considerable interest in improving the complexity of the construction, in terms of the running time $t(n)$ of M_o, the number $r(n)$ of random bits M uses, and the success probability $c(n)$ that M_o accepts when $x \in L$. (Here M_o will always reject when $x \notin L$.)

We state our bounds in a more-detailed setting: Define a $(t(n), \nu(n), r(n))$-*bounded probabilistic NTM* to be a machine M which on any input x, $|x| = n$, first flips $r(n)$-many coins to form a 0-1 string w. Then M makes $\nu(n)$-many nondeterministic moves, which can be recorded as a string $y \in \{0,1\}^{\nu(n)}$. Finally, M evaluates a predicate $Q(x, w, y)$ in time $t(n) = t_Q(n + r(n) + \nu(n))$. Let a language L be given. Then M *accepts* L *with success probability* $c(n)$ if for all x ($n = |x|$),

$$x \in L \Longrightarrow \Pr_{w \in \{0,1\}^{r(n)}}[|\{ y \in \{0,1\}^{\nu(n)} : Q(x, w, y) \}| \text{ is odd}] \geq c(n), \tag{1}$$

$$x \notin L \Longrightarrow \Pr_{w \in \{0,1\}^{r(n)}}[|\{ y \in \{0,1\}^{\nu(n)} : Q(x, w, y) \}| \text{ is odd}] = 0. \tag{2}$$

(Here and below, '$\in$' refers to selection under uniform distribution, unless otherwise specified.) If there is a fixed constant $c > 0$ such that $c(n) > c$ for all n, then M is an RP[$\oplus$P]-*machine*. The main theorem of this paper is a construction which, under the simplying assumption that $L \in \mathrm{NTIME}[n]$, achieves

- $t(n) = O(n \log^+ n)$,
- $c(n) = 1/2 - \epsilon$, and
- $r(n) = 2n - \lceil n \log\log n / \log n \rceil + O(n/\log n)$,

where the constant in the 'O' is $1 + \log_2(1/\epsilon)$, and $\log^+ n$ is an abbreviation for $\log n \log\log n \log\log\log n$. All logarithms are to base 2 unless otherwise specified. Sometimes we say $\log_2$ for emphasis, such as in the bound $2^{\lceil \log_2 n \rceil} \leq 2n$.

M can be turned into the above kind of oracle machine M_o as follows: M_o on input x flips $r(n)$ coins to get w. Then the fixed deterministic recognizer for Q is converted via the efficient construction of Schnorr [Sch76] (see also Buss and Goldsmith [BG93]) into a formula $\phi_{x,w}$ of size $O(t(n) \log t(n))$ such that the number of satisfying assignments of $\phi_{x,w}$ equals the number of y such that $Q(x, w, y)$ holds. The time to construct $\phi_{x,w}$ is also $O(t(n) \log t(n))$. Thus M_o runs in time $t'(n) = O(t(n) \log t(n))$. In this paper we largely ignore the difference between $t'(n)$ and $t(n)$, and work with RP[$\oplus$P] machines.

1.1. Previous results

Valiant and Vazirani stated their result as a randomized reduction from *SAT* (or some other language $L \in$ NP) to an oracle for Unique *SAT* which meets the above promise condition. Given $L \in$ NP, let $R(x, y)$ be a *witness predicate* for L; i.e., such that for all x, $x \in L \iff (\exists y \in \{0,1\}^{p(n)})\, R(x, y)$, where p is a polynomial. For the case $L = SAT$, [VV86] took $p(n) = n$. Here we follow Toda [Tod89] and write p as short for $p(n)$. The machine M first flips $p^2 + \lceil \log(p+1) \rceil$ coins to form p-many vectors $w^1, \ldots, w^p \in \{0,1\}^p$ and an integer k, $0 \leq k \leq p$. Then M makes p-many nondeterministic moves to form y, and accepts iff

$$R(x,y) \ \wedge \ (\forall i : 1 \leq i \leq k)\, y \cdot w^k = 0, \tag{3}$$

where $\cdot$ denotes inner product of vectors over GF(2). Then M accepts L with success probability $1/4p(n)$. The success probability can be improved to $1/4$ at the cost of increasing $\nu(n)$ to $p^2 + 2p + 3$ as follows: Let $N^{w,k}$ be the NTM which results from M after k and $w = w^1, \ldots, w^p$ are fixed, and for any NTM N, let N_{+1} be obtained by a standard construction which adds one extra accepting path $\vec{c}$, using one more nondeterministic move. The new machine M' flips just p^2 coins to form w. Let N^w stand for the NTM which simulates the machines $N_{+1}^{w,0}, \ldots, N_{+1}^{w,p}$ in series, accepting iff all of them accept. Since a product of numbers is even iff at least one of the numbers is even, it follows from the lemma of [VV86] that $\Pr_w$[the number of accepting computations of N^w is even] $\geq 1/4$. Now modify N^w so that the path $\vec{c}^{\,p+1}$ is not an accepting path. Then the machine M' which forms w and simulates this N^w is an RP[$\oplus$P] machine which accepts L with success probability at least $1/4$. This gives $r(n) = p^2$, which in the case $p(n) = n$ equals quadratically many random bits. M' also makes quadratically many nondeterministic moves; we say it works on the "product witness space." The guarantee by M' is only for parity, whereas M "isolates a unique element" in the parlance of [CRS93].

A more-efficient variant of this construction uses families H of hash functions $h : A \to B$ with the collision-avoidance property that for some $c > 0$ and all distinct $y_1, y_2 \in A$, $\Pr_{h \in H}[h(y_1) = h(y_2)] \leq c/|B|$. (This is called "$c$-universal" in [DGMP92], and when $c = 1$ and the probability is exactly $1/|B|$, this is called "universal" [CLR90] or "universal$_1$" [CW79].) Let $S \subseteq A$ be nonempty, and put $s := |S|$, $b := |B|$. A simple argument using the inclusion-exclusion principle shows that when $\lceil b/3 \rceil \leq s \leq \lfloor 2b/3 \rfloor$, $1 \leq c \leq 39/35$, and $b \geq 16$,

$$\Pr_{h \in H, z \in B}[\text{there is exactly one } y \text{ such that } y \in S \text{ and } h(y) = z] \geq 1/8. \tag{4}$$

In the case $A = \{0,1\}^p$, $B = \{0,1\}^k$, we want uniform families $H^{pk} = \{ h_1^{pk}, \ldots h_{\eta(p,k)}^{pk} \}$ of hash functions such that the mapping $h_i^{pk}(y)$ is easy to compute given k, p, $y \in \{0,1\}^p$, and i, $1 \leq i \leq \eta(p,k)$. The machine M first flips $\lceil \log(p+1) \rceil$ coins to select $k \in 1, \ldots, p+1$, and then flips $\lceil \log \eta(p, p+1) \rceil + (p+1)$ coins to select $h \in H^{pk}$, and $z \in \{0,1\}^k$. Then M guesses $y \in \{0,1\}^p$, and accepts if $R(x,y) \ \wedge \ h(y) = z$. Since the ranges of s for which equation 4 is valid cover all of $[1 \ldots 2^p]$ for $1 \leq k \leq p+1$, it follows that whenever $x \in L$, M isolates a unique witness y with success probability at least $1/8(p+1)$. The same product construction as before gives an M' with success probability at least $1/8$, for parity (no longer uniqueness) on the product witness space. Note that this analysis does not require the selections of $h \in H^{pk}$ to be independent for different k. The function $\eta(p,k)$ need not increase with k—indeed, for $k = p$, H^{kp} could consist solely of the identity. There are families of hash functions such that M' needs only $2(p+1)$ coin flips (see [IZ89, DGMP92]), giving $r(n) = 2n + 2$ in the case $p(n) = n$.

S. Gupta [Gup93] constructs an M' which uses quadratically-many random bits and achieves success probability at least $3/16$ for parity. Unlike the above, this construction stays in the original witness space, thus making only p-many nondeterministic moves. Chari, Rohatgi, and Srinivasan [CRS93] report that when a bound Z on s is known in advance, then $O(\log n + \log Z)$ random bits suffice for an M which isolates a unique element, with success probability $1/8p$. (In Section 5 we note that this can be achieved via the above hashing method, and analyze the particular constant factor in the O notation.) Thus for $p(n) = n$, our method for parity improves previous constructions by a factor of n in at least two of the categories $r(n), t(n), c(n), \nu(n)$.

2. The Coding Idea

The RP[$\oplus$P] machine M in this paper has the following special form: Given the witness predicate $R(x, y)$ for L, M flips coins to form j, guesses a string y, and then the deterministic phase computes a predicate of the form.

$$Q(x, j, y) = R(x, y) \ \wedge \ G(y, j) \tag{5}$$

Here G will be the generator matrix of a large *error-correcting code*. Note that G is independent of x, and M has $\nu(n) = p(n)$.

An (n, d)-*code* over an alphabet Σ is a collection $C \subseteq \Sigma^n$ such that for all distinct strings $x, y \in C$ (called *codewords*), $d_{\mathrm{H}}(x, y) \geq d$. Here d_{H} stands for Hamming distance. The code is said to have *density* $\delta = d/n$, and its *rate* is $R = \log_{|\Sigma|}(|C|)/n$. If $|\Sigma|$ is a prime power q, then one can build onto Σ the structure of a field $F = (\Sigma, +, \cdot, 0, 1)$, and regard Σ^n as a vector space V of dimension n over F. The field is unique up to isomorphism, and is also denoted by GF(q). If an (n, d) code C is a subspace of V of dimension k, then C is called an $[n, k, d]$-*code*, and its rate is given by $R = k/n$. Generally, C is then called a *linear code*, and its minimum distance d equals the minimum *weight* (i.e., number of non-zero entries over F) of a non-zero codeword. Any basis for C can be formed into a $k \times n$ *generator matrix* G for the code, with entries from F. Rows of G are denoted by $G(i, \cdot)$, $1 \leq i \leq k$.

The basic idea is the following: Suppose we are given a generator matrix G for an $[\hat{n}, 2^p, \delta\hat{n}]$ code over GF(2), i.e. over $\{0, 1\}$. Let the probabilistic NTM M work as follows:

1. Flip $\lceil \log_2 \hat{n} \rceil$-many coins to choose a column j, $1 \leq j \leq \hat{n}$.
2. Guess a row i, $1 \leq i \leq 2^p$, identified with a possible witness string $y_i \in \{0, 1\}^p$.
3. Accept iff $R(x, y) \ \wedge \ G(i, j) = 1$.

Suppose $S := \{ y \in \{0, 1\}^p : R(x, y) \}$ is nonempty. To S there corresponds the unique non-zero codeword $w_S := \sum_{y \in S} G(y, \cdot)$, where the sum is over GF(2). Let N^j be the NTM which results from M when j is fixed. Then the number of accepting computations is odd iff the jth entry of w_S is a '1'. Since the weight of any non-zero codeword is at least $\delta\hat{n}$, it follows that step 1. of M isolates an odd subset of S with success probability at least δ.

Also note that expressed in terms of the rate, the number of random bits M uses is $\log_2(\hat{k}/R) = p + \log_2(1/R)$. In the case $p(n) = n$ and constant rate R, this gives $r(n) = n + O(1)$, which is intuitively the best one can hope for via the coding construction, absent the possibility of amplifying via pseudo-random generators in complexity-bounded settings. The desire to make the above construction efficiently uniform in n motivates the following two definitions, the first of which is standard in coding theory.

Definition 2.1. Let I be an infinite set of natural numbers, and let the field F be fixed. A sequence $[C_l]_{l\in I}$ of $[l, k_l, d_l]$-codes over F is *asymptotically good* if there are constants $\delta_0, R_0 > 0$ such that for all but finitely many $l \in I$, $d_l/l \geq \delta_0$ and $k_l/l \geq R_0$.

(The standard definition uses the slightly weaker condition that the limit-inferior of each sequence not fall below the constant.) It is intended that I not be too sparse. The second is similar to definitions in [KTV84, TV91, ABN$^+$92, She93]. Let $\mathcal{C}$ be some complexity class of functions.

Definition 2.2. A sequence $[C_{l(n)}]_{n=1}^{\infty}$ of codes represented by generator matrices $G_{l(n)}$ is $\mathcal{C}$*-uniform* if the function mapping $(n, i, j) \mapsto G_{l(n)}(i, j)$ is computable in the complexity class $\mathcal{C}$. Here we intend that n is written in unary, i and j in binary ($1 \leq i \leq k_{l(n)}$, $1 \leq j \leq l(n)$), that $l(n)$ is an increasing 1-1 function, and that the complexity is expressed in terms of n, not $|(n, i, j)|$.

If the lengths $l(n)$ are polynomial in n, then so are the dimensions $k_{l(n)}$, and the generator matrices $G_{l(n)}$ have polynomial size. Then with $\mathcal{C} = \mathrm{P}$ the above is the same as the cited definitions, which require that the encoding function $v \mapsto Gv$ be computable in polynomial time. We informally call such sequences *short codes*. In our case we will have $l(n) = \hat{n} = 2^{p(n)}/R$, which is exponential in n. Then we call the codes *long*. For polynomial-time uniformity, $l(n)$ cannot grow faster than 2^{poly}, else j will be too big. For long codes, time which is polynomial in n is polylog in $\hat{n}$. In the 1970's it was considered a major problem in coding theory to find a sequence of asymptotically good codes with $l(n) = n^{O(1)}$ which is polynomial-time uniform. This was attained by Tsfasman, Vladut, Zink [TVZ82] (see also the paper with Katsman, [KTV84]), with an algorithm running in time approximately $O(n^{30})$. This has been improved to reasonable runtimes by Justesen et.al. [JLJ$^+$89, JLJH92], who also give efficient decoding algorithms, and more strongly by Shen [She93] (see also [TV91, ABN$^+$92]). In our case, speaking in terms of $\hat{n}$, we have the problem of whether asymptotically good sequences of codes exist which are NC-uniform. The sequence of codes in the next section is *not* asymptotically good, but when $p(n) = n$ it is close to being linear-time uniform. It is also NC^1-uniform in terms of $\hat{n}$, when computing $G(i, j)$ is the same as constructing all of G.

Notes The basic idea of using error-correcting codes to reduce the number of random bits in certain applications, and to isolate unique elements, was observed by Naor and Naor [NN90] (see also [ABN$^+$92]). It was also known to Tarui [Tar92] in a non-uniform context. Large error-correcting codes of constant density are integral to the recent work on transparent (holographic) proofs [BFLS91, AS92, ALM$^+$92, Sud92, Bab93, BGLR93]. Indeed, the code in the next section is a known simplification of the code originally used by Babai et.al. [BFLS91]; see Sudan [Sud92] and the Acknowledgments below. Ours is the first formulation we know of the coding idea in the context of improving the bounds for reducing NP to parity, where large codes are involved and questions of uniformity and efficiency are important.

3. Examples of Codes and Concatenation

The *Hadamard* $\mathcal{H}_k$ over $\{0,1\}$ of length $n = 2^k$ has n codewords. The codewords can be arranged into an $n \times n$ array with rows and columns indexed by strings $u, v \in \{0,1\}^k$, and entries $u \cdot v$, where $\cdot$ is inner product over GF(2). $\mathcal{H}_k$ has distance $d_k = 2^{k-1}$, so $\delta_k = 1/2$ is constant, but its dimension is only k, and the rate $R_k = k/2^k$ tends rapidly to zero. (One can double the size of $\mathcal{H}_k$ and add 1 to the dimension by including the bitwise negations of the codewords.) Thus the $\mathcal{H}_k$ are not good codes, but individual entries of codewords are easy to compute.

The *full q-ary generalized Reed-Muller code* $\mathcal{R}_q(r,m)$ of *order* r, where $r < m(q-1)$, has length $N = q^m$ over the field $F = \mathrm{GF}(q)$. Each polynomial $f(x_1, \ldots x_m)$, in m variables over F of degree at most r, defines the codeword with entries $f(a_1, \ldots, a_m)$, where $\vec{a} = (a_1, \ldots, a_m)$ ranges over all sequences of arguments in F. We follow the description in [TV91] (p46) of a reduced code $\mathcal{R}'_q(r,m)$ for the case $r \leq q-1$. The generator matrix has one row for each monomial $x_1^{i_1} x_2^{i_2} \cdots x_m^{i_m}$ such that $i_1 + i_2 + \ldots + i_m \leq r$. Since $r \leq q-1$ these monomials are all distinct, and they are all linearly independent, so the dimension is $K = \binom{m+r}{r}$. By the so-called Schwartz inequality [Sch80] (cf. [BFLS91, Sud92]), for every two distinct polynomials f and g over F of degree at most r, and for every $I \subseteq F$,

$$|\{\, \vec{a} \in I^m : f(\vec{a}\,) = g(\vec{a}\,) \,\}| \leq r|I|^{m-1}. \tag{6}$$

With $I = F$, it follows that every two distinct codewords have distance over F at least $|F|^m - r|F|^{m-1}$, and some achieve this, so the density Δ equals $1 - r/|F|$. The rate is $\binom{m+r}{r}/q^m$.

We are interested in the case $F = \mathrm{GF}(2^k)$. Elements of $\mathrm{GF}(2^k)$ can be encoded as binary strings of length k. The code $\mathcal{R}'_q(r,m)$ could be simply *flattened* into a binary code of length $\hat{n} = Nk$, but since two distinct elements of $\mathrm{GF}(2^k)$ may differ in only one place as binary strings, the density $\hat{\delta}$ of the resulting code might be no better than Δ/k, which tends to 0 as k increases. The trick is to encode the binary representations of the field elements using a linear code of dimension k and constant density δ. Doing this guarantees $\hat{\delta} \geq \Delta \cdot \delta$. When the Hadamard code $\mathcal{H}_k$ is used for this encoding the result is a linear code $\hat{C}$ over $\{0,1\}$ with parameters $\hat{n} = N2^k$, $\hat{k} = K$, and $\hat{\delta} = 1/2 - \epsilon$, where $\epsilon = r/2^{k+1}$. The rate is $K/N2^k = \binom{m+r}{m}/2^{km+k} \leq \binom{m+2^k}{m}/2^{km+k} =$

$$\frac{2^k+m}{2^k m} \cdot \frac{2^k+(m-1)}{2^k(m-1)} \cdots \frac{2^k+1}{2^k} \cdot \frac{1}{2^k}, \tag{7}$$

which vanishes at least as fast as $1/2^k$ as k increases. Hence these codes fall short of being asymptotically good.

This general idea behind the construction of $\hat{C}$ is called *concatenation of codes* [For66] (see also [MS77]); here $\mathcal{R}'_q(r,m)$ is the *outer code* and $\mathcal{H}_k$ is the *inner code*. The calculations also work if we take a proper subset (of size 2^p or 2^n) of the rows of the generator matrix as a basis for the outer code.

4. Construction and Main Theorem

The last detail needed to build the code $\hat{C}$ is to find a representation for $\mathrm{GF}(2^k)$ and its product operation. The standard method is to find a one-variable polynomial $\alpha(x)$ of degree k which is *irreducible* over GF(2). Shoup [Sho88] gives a deterministic algorithm for finding such a polynomial α in time $O(k^4 \log^{O(1)} k)$. Since k will be about $2 \log n$, this time is negligible. Then an element b of F written bitwise as $b_0 b_1 \dots b_{k-1}$ corresponds to the polynomial $b_0 + b_1 x + \cdots + b_{k-1} x^{k-1}$. Products of such polynomials are reduced modulo α.

Now we can give the whole program for an RP[⊕P] machine M which accepts a given language $L \in \mathrm{NP}$ with witness predicate $R(x, y)$. For simplicity we assume that witnesses y have length n; the general case can be had by substituting $p(n)$ for n everywhere in the program.

1. Input x and ϵ; $n := |x|$
2. $b := \lceil \log_2 n \rceil$ /*block length for exponents*/
3. $d := 2^b - 1$ /*maximum degree in each variable*/
4. $m := \lceil n/b \rceil$ /*number of variables*/
5. $k := \lceil \log_2 d + \log_2 m + \log_2(1/\epsilon) - 1 \rceil$
6. Calculate an irreducible polynomial α of degree k over GF(2)
7. Flip $mk + k$ coins to form $j = \langle a_1, \dots, a_m, v \rangle$, where $v \in \{0,1\}^k$.
8. Guess $y \in \{0,1\}^n$
9. Taking b bits of y at a time, form integers $i_1, i_2, \dots i_{m-1}, i_m \in \{0, \dots, d\}$. It is OK for i_m to be truncated.
10. Compute $u := a_1^{i_1} \cdot a_2^{i_2} \cdots a_m^{i_m}$
11. Compute $G(y, j) := u \cdot v$
12. Accept iff $R(x, y) \ \wedge \ G(y, j) = 1$.

Theorem 4.1. *For any fixed $\epsilon < 1/2$, M is an* RP[⊕P] *machine which accepts L with success probability at least $1/2 - \epsilon$, using at most $r(n) = 2n - n \log\log n / \log n + (1 + \log(1/\epsilon)) n / \log n + O(\log n)$ random bits, running in time $O(n \log^+ n) + t_R(n)$, and working on the original witness space with $\nu(n) = n$.*

Proof. The length of the code is $N 2^k = 2^{km+k}$, so the number of random bits used is

$$k(m+1) \leq (\log_2 d + \log_2 m + \log_2(1/\epsilon))(m+1)$$

$$\leq (\log_2 2n + \log n - \log\log n + \log_2(1/\epsilon))n/(\log n) + O(\log n)$$

$$= 2n - n\log\log n/\log n + (1 + \log_2(1/\epsilon))n/\log n + O(\log n).$$

Steps 2.–6. and 11. take time polynomial in $\log n$, while steps 7.–9. take linear time, and step 12. takes time $t_R(n)$ (which we may ignore). This leaves step 10. To multiply two polynomials of degree $k-1$ over GF(2) and reduce them modulo α in the field GF(2^k) takes time $t_1 = O(k\log k\log\log k)$ on standard Turing machine models (see [AHU74] and [Rab80]). The time to compute a^i in GF(2^k) where $i \leq n$ is $t_2 = O(\log n \cdot 2k\log k\log\log k)$ via repeated squaring. Hence the time to evaluate the monomial is at most $O(mt_2 + mt_1) = O(m\log(n)k\log k\log\log k) = O(n\log^+ n)$, since $m = O(n/\log n)$ and $k = O(\log n)$. □

Now given definitions of the classes P, NP, PH, RP, BPP, PP, and ⊕P in terms of machines M, define QL, NQL, QLH, RQL, BQL, PQL, and ⊕QL to be the classes which result when M is required to run in quasilinear time; i.e., time $qlin = n(\log n)^{O(1)}$. (Remark: the probabilities in RQL and BQL can be amplified at least up to $2^{-polylog n}$.)

Corollary 4.2. NQL $\subseteq$ RQL[⊕QL].

We have not yet checked thoroughly whether the amplification is good enough to extend this to QLH $\subseteq$ BQL[⊕QL], or whether the quasilinear analogue PQL[⊕QL] $\subseteq$ $\mathrm{QL}^{\mathrm{PQL}}$ of Toda's second theorem [Tod91] holds. On the face of it, the latter encounters the problem that the amplifying polynomials used are higher than quadratic.

5. Possible Improvements and Applications

As observed in Section 2, if the rates of the uniform codes can be held above a constant R_0, as well as the densities held above δ_0, then a constant success probability is achieved using only $n + O(1)$ random bits. There are two main points of slack in the code of the last section. The lesser one is the use of "bad" Hadamard codes for the inner code; these are responsible for the $1/2^k$ term in equation 7. Since $k = O(\log n)$, the deterministic polynomial-time uniform, asymptotically good binary codes of Shen [She93] can be used in place of $\mathcal{H}_k$ without affecting the order of the running time. However, doing so only subtracts $k = O(\log n)$ random bits from $r(n)$. The greater one may be an inherent problem with the approach of scaling down the size of the base field F (to improve the runtime of the field arithmetic) by increasing the number m of variables. In order for the generator matrix G to have 2^n rows, we must have (roughly) $m\log(r/m) \geq n$. Even supposing that the inner code had length 1, we would still have $\log\hat{n} = \log N = \log(q^m) \geq m\log r$, since $r \leq q-1$. (Using the more-complicated q-ary Reed-Muller code that arises when $r \geq q$, or alternatively using some $I \subset F$ in Schwartz's Lemma, seems not to help in this case either.) Setting $\log\hat{n} = n + f(n)$ gives $f(n) \geq m\log m$. The only way to make $f(n) = O(1)$ is to hold m constant, but then the size of the field must be increased exponentially to compensate. Likewise, questions such as whether Shen's or some other known family of good codes can be used instead of $\mathcal{R}_q(r,m)$ altogether come down to problems of complexity in relatively large fields. It is desirable to construct large fields $\mathrm{GF}(2^k)$ using an element of degree k which is *primitive*, but this is still only known to be possible in deterministic polynomial time if the prime factorization of $2^k - 1$ is known. (See Shoup [Sho90].)

It is natural to ask how easy it is to amplify the success probability of the RP[$\oplus$P] machine M beyond $1/2$, without using too many more random bits. One idea is to have M probe the entries of u in more than one column, but this may involve another kind of product construction that compounds the runtime. We inquire whether deterministic amplification techniques involving expander graphs (cf. [ABN+92]) can be efficiently used in this setting.

Finally, we return to the remarks about hashing in Section 1.1. By an observation attributed to Bruck in [NN90], a c-universal family H of hash functions $h : \{0,1\}^n \rightarrow \{0,1\}^k$ is the same as a (not necessarily linear) code with alphabet $\{0,1\}^k$ of length $\eta = |H|$ and density $\delta = 1 - c/2^k$ over $\{0,1\}^k$. Having $c > 1$ makes the code density avoid the *Plotkin bound* (see [MS77]), and allows for *good* families of codes whose lengths are a small constant times $(n-1)2^k/c$. However, the bound $c \leq 39/35$ places the codes into the *low-rate* neighborhood studied by Shen and others. If a bound Z on $|S|$ is known in advance and good codes are used, this makes the number of random bits required for the hashing equal to $\log n + \log Z + O(1)$. This improves the unspecified constants in the 'O' notation used by Chari, Rohatgi, and Srinivasan [CRS93] to 1. If Z is not too large, the codes required are relatively short, and it appears that the above-cited families will yield good bounds on time as well as randomness complexity. There is, however, still the problem discussed in section 1.1 that either the success probability is non-constant or the number of nondeterministic moves is more than linear.

In further research we intend to explore the improvements to the theorems cited in Section 1 which result from the new constructions and complexity bounds.

Acknowledgments This paper grew out of a term project with D. Sivakumar, Ashish Naik, and K. Sundar (Buffalo) to seek improvements to the Valiant-Vazirani construction. The author is especially grateful to Dr. Madhu Sudan of Berkeley for explanations given at the FCRC of the results in his thesis and for help in tuning the parameters of the code used here. The author also acknowledges helpful conversations with Drs. Jun Tarui, Sanjay Gupta, Laszlo Babai, and Stephen Rudich.

References

[ABN+92] N. Alon, J. Bruck, J. Naor, M. Naor, and R. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–512, March 1992.

[AH90] E. Allender and U. Hertrampf. On the power of uniform families of constant-depth circuits. In *The Proceedings of the 15th International Symposium on Mathematical Foundations of Computer Science*, volume 452 of *Lecture Notes in Computer Science*, pages 158–164. Springer-Verlag, 1990.

[AHU74] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Mass., 1974.

[All89] E. Allender. A note on the power of threshold circuits. In *The Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 580–584, 1989.

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

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *The Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[Bab93] L. Babai. Transparent (holographic) proofs. In *The Proceedings of the 1993 Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lecture Notes in Computer Science*, pages 525–534. Springer-Verlag, 1993.

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

[BG93] J. Buss and J. Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22:nnn–nnn, 1993.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *The Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 294–304, 1993.

[BRS91] R. Beigel, N. Reingold, and D. Spielman. The perceptron strikes back. In *The Proceedings of the 6th Annual IEEE Conference on Structure in Complexity Theory*, pages 286–291, 1991.

[BT91] R. Beigel and J. Tarui. On ACC. In *The Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 783–792, 1991.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[CRS93] S. Chari, P. Rohatgi, and A. Srinivasan. Randomness-optimal unique element isolation, with applications to perfect matching and related problems. In *The Proceedings of the 25th Annual ACM Symposium on the Theory of Computing*, pages 458–467, 1993.

[CW79] J. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–154, 1979.

[DGMP92] M. Dietzfelbinger, J. Gil, Y. Matias, and N. Pippenger. Polynomial hash functions are reliable. In *The Proceedings of the 1992 International Conference on Automata, Languages, and Programming*, volume 623 of *Lecture Notes in Computer Science*, pages 235–246. Springer-Verlag, 1992.

[For66] G. Forney. *Concatenated Codes*. MIT Press, 1966.

[Gup93] S. Gupta. On isolating an odd number of elements, 1993.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *The Proceedings of the 1989 IEEE Symposium on Foundations of Computer Science*, pages 248–253, 1989.

[JLJ+89] J. Justesen, K. Larsen, H.E. Jensen, A. Havemose, and T. Hoholdt. Construction and decoding of a class of algebraic geometry codes. *IEEE Transactions on Information Theory*, 35(4):811–820, July 1989.

[JLJH92] J. Justesen, K. Larsen, H.E. Jensen, and T. Hoholdt. Fast decoding of codes from algebraic plane curves. *IEEE Transactions on Information Theory*, 38(1):111–119, January 1992.

[KTV84] G. Katsman, M. Tsfasman, and S. Vladut. Modular curves and codes with a polynomial construction. *IEEE Transactions on Information Theory*, IT-30:353–355, 1984.

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

[NN90] J. Naor and M. Naor. Small-bias probability spaces. In *The Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

[Rab80] M. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, pages 273–280, 1980.

[Sch76] C. Schnorr. The network complexity and the Turing machine complexity of finite functions. *Acta Informatica*, 7:95–107, 1976.

[Sch80] J.T. Schwartz. Fast probabilistic algorithms for polynomial identities. *Journal of the ACM*, 27:701–717, 1980.

[She93] B.-Z. Shen. A Justesen construction of binary concatenated codes than asymptotically meet the Zyablov bound for low rate. *IEEE Transactions on Information Theory*, 39(1):239–242, January 1993.

[Sho88] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. In *The Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 283–290, 1988.

[Sho90] V. Shoup. Searching for primitive roots in finite fields. In *The Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 546–554, 1990.

[Sud92] M. Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems*. PhD thesis, University of California, Berkeley, 1992.

[Tar91] J. Tarui. Randomized polynomials, threshold circuits, and the polynomial hierarchy. In *The Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science*, volume 480 of *Lecture Notes in Computer Science*, pages 238–250. Springer-Verlag, 1991.

[Tar92] J. Tarui. On singleton selectors and probabilistic polynomials. Manuscript, 1992.

[Tar93] J. Tarui. Probabilistic polynomials, AC^0 functions, and the polynomial-time hierarchy. *Theoretical Computer Science*, 113:167–183, 1993.

[TO91] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. In *The Proceedings of the 6th Annual IEEE Conference on Structure in Complexity Theory*, page to appear, 1991.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21:316–328, 1992.

[Tod89] S. Toda. On the computational power of PP and $\oplus$P. In *The Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20:865–877, 1991.

[TV91] M. Tsfasman and S. Vladut. *Algebraic-Geometric Codes*, volume 58 of *Mathematics and Its Applications (Soviet Series)*. Kluwer Academic, Dordrecht, 1991.

[TVZ82] M. Tsfasman, S. Vladut, and T. Zink. Modular curves, Shimura curves and Goppa codes, better than Varshamov-Gilbert bound. *Mathematische Nachrichten*, 109:21–28, 1982.

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