Quasilinear Time Complexity Theory

Ashish V. Naik
SUNY Buffalo
avnaik@cs.buffalo.edu

Kenneth W. Regan
SUNY Buffalo
regan@cs.buffalo.edu

D. Sivakumar
SUNY Buffalo
sivak-d@cs.buffalo.edu

August 20, 1993

Abstract

This paper furthers the study of quasi-linear time complexity initiated by Schnorr [Sch76] and Gurevich and Shelah [GS89]. We show that the fundamental properties of the polynomial-time hierarchy carry over to the quasilinear-time hierarchy. Whereas all previously known versions of the Valiant-Vazirani reduction from NP to parity run in quadratic time, we give a new construction using error-correcting codes that runs in quasilinear time. We show, however, that the important equivalence between search problems and decision problems in polynomial time is unlikely to carry over: if search reduces to decision for *SAT* in quasi-linear time, then all of NP is contained in quasi-polynomial time. Other connections to work by Stearns and Hunt [SH86, SH90, HS90] on "power indices" of NP languages are made.

Topics. Computational complexity.

1. Introduction

The notion of "feasible" computation has most often been identified with the concept of polynomial time. However, an algorithm which runs in time n^{100} or even time n^2 may not really be feasible on moderately large instances. Quasi-linear time, namely time $qlin := n \cdot (\log n)^{O(1)}$, largely avoids this objection to the size of the exponent of n. Let DQL and NQL stand for time *qlin* on deterministic and nondeterministic Turing machines. Schnorr [Sch76, Sch78] showed that *SAT* is complete for NQL under DQL many-one reductions ($\leq_{\mathrm{m}}^{\mathrm{ql}}$). Together with Stearns and Hunt [SH86, SH90], it was shown that many known NP-complete problems also belong to NQL and are complete for NQL under $\leq_{\mathrm{m}}^{\mathrm{ql}}$, so that the question NQL $\stackrel{?}{=}$ DQL takes on much the same shape as NP $\stackrel{?}{=}$ P. Related classes within P are studied by Buss and Goldsmith [BG93].

One theoretical difficulty with the concept of quasilinear time is that it appears not to share the degree of independence on particular machine models that makes polynomial time such a *robust* concept. Gurevich and Shelah [GS89] showed that a wide variety of models related to the RAM under log-cost criterion [CR73] accept the same class of languages in quasilinear time—we call this class DNLT. They also showed that nondeterministic *qlin* time for these machines, namely NNLT, equals NQL. However, currently it appears that DNLT is larger than DQL, and that for all $d > 1$, Turing machines with d-dimensional tapes accept more languages in time *qlin* than do TMs with $(d-1)$-dimensional tapes (cf. [WW86]). Our constructions all work for DQL as well as DNLT.

Our main motivation is to ask: How much of the known theory of complexity classes based on polynomial time carries over to the case of quasilinear time? Section 2 observes that the basic results for the polynomial hierarchy and PSPACE hold also for the quasilinear hierarchy (QLH) and QLSPACE.

Section 3 shows that the randomized reduction from NP to parity given by Valiant and Vazirani [VV86] and used by Toda [Tod91], previously proved by constructions which run in quadratic time (see [VV86, Tod89, CRS93, Gup93]), can be made to run in time *qlin*. Our construction also markedly improves the number of random bits needed and the success probability, and uses error-correcting codes in an interesting manner first noted in [NN90].

Section 4 studies what may be the major difference between polynomial and quasilinear time: the equivalence between functions and sets seems no longer to hold. It has long been known that *any* function can be computed in polynomial time using some set as an oracle. In contrast, we show that there exists a function f which cannot be computed in quasilinear time using any set as an oracle whatsoever. Next, we study the property of *reducing search to decision* for NP sets. While it is well-known that search reduces to decision for *SAT* in quadratic time (cf. [Sel88, JY90]), we show that search does *not* reduce to decision for *SAT* in quasilinear time, unless all of NP is contained in DTIME$[2^{\text{polylog}\, n}]$. We extend our techniques to show that the quadratic bound on reducing search to decision for *SAT* is optimal, unless *SAT* can be recognized in subexponential time. Finally, we establish an interesting connection between the time taken for reducing search to decision and the *power index* [SH90] of a language.

2. Notation and Basic Results

Let $\Sigma := \{0, 1\}$. Given strings $y_1, \ldots y_m \in \Sigma^*$, each y_i of length n_i, let $y = \langle y_1, \ldots y_m \rangle$ stand for the binary string of length $2r + 2m$ obtained by translating 0 to 00, 1 to 11, and 'comma' to 01, with an extra 01 at the end. For any language B we often write $B(x, y)$ in place of '$\langle x, y \rangle \in B$' and consider B as a predicate. For convenience we call q a *quasilinear function* if there are constants $k, c, d \geq 0$ such that for all n, $q(n) = cn(\log^k n) + d$. Where n is understood we write q as short for $q(n)$, and also write $(\exists^q y)$ for $(\exists y \in \{0, 1\}^{q(n)})$, $(\forall^q y)$ for $(\forall y \in \{0, 1\}^{q(n)})$. Standard padding lets us ignore the distinction between $|y| = q$ and $|y| \leq q$ for our present purposes. The notation $(\#^q y : B(x, y))$ means "the number of strings $y \in \{0, 1\}^{q(|x|)}$ such that $B(x, y)$ holds."

Definition 2.1. If $A \in$ NP, $B \in$ P, and p is a polynomial such that for all x, $x \in A \iff (\exists^p y)\, B(x, y)$, then we call B a *witness predicate* for A, with the length bound p understood. We use the same terms in the context of NQL and DQL.

We note the following provision about oracle Turing machines M made standard in [WW86] (see also [LL76, Wra77, Wra78]): Whenever M enters its query state $q_?$ with the query string z on its query tape, z is *erased* when the oracle gives its answer. If the oracle is a function g, we suppose that $g(z)$ replaces z on the query tape in the next step. If A and B are languages such that $L(M^B) = A$ and M^B runs in quasilinear time, then we write $A \leq_{\mathrm{T}}^{\mathrm{ql}} B$. As usual we may also write $A \in \mathrm{DQL}^B$ or $A \in \mathrm{DQL}(B)$, and if M is nondeterministic, $A \in \mathrm{NQL}^B$ or $A \in \mathrm{NQL}(B)$. Henceforth our notations and definitions of complexity classes are standard, with 'P' replaced by 'QL', except that we use square brackets for "class operators":

Definition 2.2. For any languages A and B,

(a) $A \in \mathrm{NQL}[B]$ if there is a quasilinear function q such that for all $x \in \Sigma^*$, $x \in A \iff (\exists^q y)\, B(x,y)$.

(b) $A \in \mathrm{UQL}[B]$ if there is a q such that for all $x \in \Sigma^*$, $x \in A \Longrightarrow (\#^q y : B(x,y)) = 1$, and $x \notin A \Longrightarrow (\#^q y : B(x,y)) = 0$.

(c) $A \in \oplus\mathrm{QL}[B]$ if there is a q such that for all $x \in \Sigma^*$, $x \in A \iff (\#^q y : B(x,y))$ is odd.

(d) $A \in \mathrm{BQL}[B]$ if there is a quasilinear function q such that for all $x \in \Sigma^*$, $x \in A \Longrightarrow (\#^q y : B(x,y))/2^q > 2/3$, and $x \notin A \Longrightarrow (\#^q y : B(x,y))/2^q < 1/3$.

(e) $A \in \mathrm{RQL}[B]$ if there are q and $\epsilon > 0$ such that for all $x \in \Sigma^*$, $x \in A \Longrightarrow (\#^q y : B(x,y))/2^q > 2/3$, and $x \notin A \Longrightarrow (\#^q y : B(x,y)) = 0$.

For any class $\mathcal{C}$ of languages, $\mathrm{NQL}[\mathcal{C}]$ equals $\cup_{B\in\mathcal{C}}\mathrm{NQL}[B]$, and similarly for the other operators. With $\mathcal{C} = \mathrm{DQL}$ these classes are simply written NQL, UQL, $\oplus$QL, BQL, and RQL. It is easy to check that "machine definitions" of these classes are equivalent to the above "quantifier definitions"; e.g. UQL is the class of languages accepted by unambiguous NTMs which run in quasilinear time. By standard "amplification by repeated trials," for any function $r = O(\log^k n)$, the classes BQL and RQL remain the same if '1/3' is replaced by $2^{-r(n)}$ and '2/3' by $1 - 2^{-r(n)}$; and similarly for $\mathrm{BQL}[\mathcal{C}]$ and $\mathrm{RQL}[\mathcal{C}]$ provided $\mathcal{C}$ is closed under "polylogarithmic majority truth table reductions." This is also enough to give $\mathrm{BQL}[\mathrm{BQL}[\mathcal{C}]] = \mathrm{BQL}[\mathcal{C}]$.

Definition 2.3. The *quasilinear time hierarchy* is defined by: $\Sigma_0^{ql} = \Pi_0^{ql} = \Delta_0^{ql} = \mathrm{DQL}$, and for $k \geq 1$,

$$\Sigma_k^{ql} = \mathrm{NQL}[\Pi_{k-1}^{ql}], \quad \Pi_k^{ql} = \text{co-}\Sigma_k^{ql}, \quad \Delta_k^{ql} = \mathrm{DQL}^{\Sigma_{k-1}^{ql}}.$$

Also $\mathrm{QLH} := \cup_{k=0}^{\infty}\Sigma_k^{ql}$, and $\mathrm{QLSPACE} := \mathrm{DSPACE}[qlin]$. By the results of [GS89], all these classes from NQL upward are the same for Turing machines and log-cost RAMs.

Next we observe the following concavity property of quasilinear functions:

Lemma 2.1. *(a) Let $q(n) = cn\log^k n$, let $n_1, \ldots n_m$ be nonnegative real numbers, and let $\sum_{i=1}^m n_i \leq r$. Then $\sum_{i=1}^m q(n_i) \leq q(r)$.*

(b) If $q(n) = cn\log^k n + d$, each $n_i \geq 1$, and $r = r(n)$ is another quasilinear function, then $\sum_{i=1}^m q(n_i)$ is bounded by a quasilinear function.

Proof. (a) True for $m = 1$. By the induction hypothesis for $m-1$, $\sum_{i=1}^{m} q(n_i) \leq q(r - n_m) + q(n_m)$. The second derivative of $q(r-x) + q(x)$ with respect to x is positive for $0 < x < r$, so the maxima on $[0, r]$ are with $n_m = 0$ or $n_m = r$, giving the upper bound $q(r)$.

(b) By (a), $\sum_{i=1}^{m} q(n_i) \leq q(r(n)) + dm$. Since each $n_i \geq 1$, $m \leq r(n)$, and so the additive term dm is quasilinear. If $r(n) = c'n\log^{k'} n + d'$, then substituting gives a quasilinear bound of the form $c''n\log^{k+k'} n + d''$, for some constants c'' and d''. $\square$

Corollary 2.2. *The relation $\leq_{\mathrm{T}}^{\mathrm{ql}}$ is transitive.*

Proof. Let $A = L(M_0^B)$ and $B = L(M^C)$, where M runs in time $q(n)$ and M_0 in time $r(n)$. Define M_1 on any input x to simulate $M_0(x)$ but use M to answer the queries $y_1, \ldots, y_m$ made by M_0. For each query y_i let $n_i := \max\{\, |y_i|, 1\,\}$. Then $\sum_i n_i$ is bounded by $r(n)$, $q(n_i)$ bounds the runtime of M on input y_i, and Lemma 2.1(b) bounds the total runtime of M_1. $\square$

With this in hand it is straightforward to show that the most fundamental properties of the polynomial hierarchy (from [Sto77, Wra77]) carry over to QLH.

Theorem 2.3.

(a) (Equivalence of oracles and quantifiers): For all $k \geq 1$, $\Sigma_k^{ql} = \mathrm{NQL}^{\Sigma_{k-1}^{ql}}$.

(b) (Downward separation): For all $k \geq 0$, if $\Sigma_k^{ql} = \Pi_k^{ql}$ then $\mathrm{QLH} = \Sigma_k^{ql}$.

(c) (Turing closure): For all $k \geq 0$, $\Sigma_k^{ql} \cap \Pi_k^{ql}$ is closed downward under $\leq_{\mathrm{T}}^{\mathrm{ql}}$. In particular, DQL *and* NQL $\cap$ *co-*NQL *are closed under $\leq_{\mathrm{T}}^{\mathrm{ql}}$.*

(d) For each $k \geq 1$, the language B_k of quantified Boolean formulas in prenex form with at most k alternating quantifier blocks beginning with '$\exists$' is complete for Σ_k^{ql} under DQL *many-one reductions.*

(e) QLH $\subseteq$ QLSPACE.

The case $k = 1$ of (d) is Schnorr's seminal result, and the higher cases follow quickly from this and (a). It is worth sketching Schnorr's construction (see also [BG93]): Take a time-$t(n)$ DTM M which decides a witness predicate $B(x, y)$ for the given language $A \in \mathrm{NQL}$. Convert M into $O(t(n)\log t(n))$-sized circuits C_n of fan-in 2 in variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_q$ such that for all x, $x \in A \iff (\exists y_1, \ldots, y_q)\, C_n(x_1, \ldots, x_n, y_1, \ldots, y_q) = 1$. Then assign a dummy variable to each of the $O(n \log n)$ wires in C_n and write a 3-CNF formula which expresses that each output wire has the correct value given its input wires. This reduces A to SAT and is computable in time $O(n \log n)$.

Let QBF stand for $\cup_k B_k$. While QBF is in alternating *qlin* space, it is not known to be in deterministic *qlin* space. Moreover, the standard reduction from a language $A \in$ PSPACE to QBF in [HU79] has a quadratic blowup in size (if A is in linear space). These apparent differences from PSPACE are connected to the issue of whether

Savitch's simulation of nondeterministic space $s(n) = \Omega(\log n)$ by deterministic space $O(s(n)^2)$ *must* have quadratic blowup. By the same token, the familiar "one-line proof" $\mathrm{NP}^{QBF} \subseteq \mathrm{NPSPACE} = \mathrm{PSPACE} = \mathrm{P}^{QBF}$ is not valid for QL. However, the result (a) below is still true:

Proposition 2.4. *(a)* $\mathrm{NQL}^{QBF} = \mathrm{DQL}^{QBF}$.

(b) There is an oracle B such that NQL^{B} *is not contained in* $\mathrm{DTIME}[2^{o(n)}]$.

The proof of (a) uses Schnorr's construction and Lemma 2.1, and in fact gives $\mathrm{NQL}^{QBF} = \mathrm{DQL}[QBF]$. Statement (b) holds for the standard oracle B separating NP^B from P^B in [HU79].

The result of [PZ83] that $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ also carries over because of the quasilinear bound on the total length of all queries in an oracle computation: $\oplus\mathrm{QL}^{\oplus\mathrm{QL}} = \oplus\mathrm{QL}$. However, it is unclear whether the theorem $\mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$ [Ko82] carries over, because the amplification of success probability to $1-2^{-\mathrm{polylog}}$ obtainable for BQL seems insufficient. For similar reasons we do not know whether Toda's lemma $\oplus\mathrm{P}[\mathrm{BP}[\mathcal{C}]] \subseteq \mathrm{BP}[\oplus\mathrm{P}[\mathcal{C}]]$ (for $\mathcal{C}$ closed under polynomial-time majority truth-table reductions), which was instrumental in proving $\mathrm{PH} \subseteq \mathrm{BP}[\oplus\mathrm{P}]$ [Tod91], carries over in the form $\oplus\mathrm{QL}[\mathrm{BQL}[\mathcal{C}]] \subseteq \mathrm{BQL}[\oplus\mathrm{QL}[\mathcal{C}]]$. However we are able to show, in the next section, that the other instrumental lemma, namely $\mathrm{NP} \subseteq \mathrm{BP}[\oplus\mathrm{P}]$ [VV86], *does* carry over by a new construction, where all previous known constructions were quadratic or worse.

3. Quasilinear-Time Reduction to Parity

Let $A \in \mathrm{NP}$ with witness predicate $B(x,y)$ and length bound $q = q(n)$, and for any x let $S_x := \{ y \in \{0,1\}^q : B(x,y) \}$ be the corresponding witness set, so that $x \in A \iff S_x \neq \emptyset$. Valiant and Vazirani [VV86] constructed a probabilistic NTM N which on any input x of length n first flips q^2-many coins to form q-many vectors $w_1, \ldots, w_q$ each of length q. N also flips coins to form a number j, $0 \leq j \leq q$. Then N guesses $y \in \{0,1\}^q$ and accepts iff $B(x,y)$ and for each i, $1 \leq i \leq j$, $y \cdot w_i = 0$, where $\cdot$ is inner product of vectors over GF(2). Let $N_{w,j}$ stand for the NTM N with $w = w_1, \ldots, w_q$ and j fixed. Clearly whenever $x \notin A$, for all w and i, the number $\#acc(N_{w,j}, x)$ of accepting computations of $N_{w,j}$ on input x is zero. The basic lemma of [VV86] states that whenever $x \in A$, $\Pr_w[(\exists j)\#acc(N_{w,j}, x) = 1] \geq 1/4$. In particular, $\Pr_{w,j}[\#acc(N_{w,j}, x)$ is odd$] \geq 1/4(q+1)$. A "product construction" yields an N' which flips coins to form just w, guesses strings $y_0, \ldots, y_q$, and achieves for all x,

$$x \in A \Longrightarrow \Pr_w[\#acc(N'_w, x) \text{ is odd}] \geq 1/4,$$

$$x \notin A \Longrightarrow \Pr_w[\#acc(N'_w, x) \text{ is odd}] = 0.$$

In symbols, this implies that $\mathrm{NP} \subseteq \mathrm{RP}[\oplus\mathrm{P}]$ (cf. [Tod91]).

However, in the case $A = SAT$ addressed by [VV86], with $q(n) = n$, N' runs in quadratic time—in fact, N' flips quadratically many coins and makes quadratically many nondeterministic moves. It was well known that by using small families $\mathcal{H} = \{ H_k \}$ of *universal*$_2$ hash functions [CW79, BCGL89] $h_k : \{0,1\}^q \to \{0,1\}^k$ $(1 \leq k \leq q+1)$ cuts the number $r(n)$ of random bits used to $2q(n)$. A related construction of [CRS93] achieves the same effect, still with quadratic runtime when $q(n) = n$. Gupta [Gup93] gives a randomized reduction to parity which achieves constant success probability 3/16 with only $\nu(n) = q(n)$ nondeterministic moves, but still using q^2-many random bits and quadratic time. The first construction of Naor and Naor [NN90, NN93] boils down to the following idea in this setting: N flips $2q+2$ coins to determine functions $h_k \in H_k$ for all k, and then flips $q+1$ more coins to form $u \in \{0,1\}^{q+1}$. Then N nondeterministically guesses $y \in \{0,1\}^q$ and k, $1 \leq k \leq q+1$, and accepts iff $B(x,y) \wedge h_k(y) = 0 \wedge u_k = 1$. This uses $3q+3$ random bits, achieves success probability at least 1/8, and runs in the time to compute h_k, which is $O(q \log q \operatorname{loglog} q)$. Our construction achieves better constants than this, and is better by an order of magnitude in (randomness or number of nondeterministic moves) and running time than all of the previous ones.

The idea of using error-correcting codes is mentioned by Naor and Naor [NN93] and ascribed to Bruck, referring the reader to [ABN$^+$92] for details. However, the construction in [ABN$^+$92] uses a code of Justesen (see [Jus72, MS77] whose implementation in our setting seems to require exponentiation of field elements of length polynomial in n, which is not known to be computable in (randomized) quasilinear time (cf. [AMV88, Sti90]). Our point is that by scaling down the size of the field used for basic arithmetic, and using multi-variable polynomials, one can achieve quasi-linear runtime. Our code is similar to those used in recent improvements of "holographic proof systems" [BFLS91, Sud92], and is only inferior to that of [ABN$^+$92] in the number of random bits.

Let ? be an alphabet of size 2^l. We can give ? the structure of the field $F = \mathrm{GF}(2^l)$; then $?^n$ becomes an n-dimensional vector space over F. An $[N,K,D]$ *code* over F is a set $C \subseteq ?^n$ which forms a vector subspace of dimension K (so $\|C\| = 2^k$), such that for all distinct $x, y \in C$, $d_H(x,y) \geq D$, where d_H is Hamming distance. Since C is closed under addition (i.e., a *linear* code), the *minimum distance* D equals the minimum *weight* (i.e., number of non-zero entries over F) of a non-zero codeword. The *rate* of the code is $R = K/N$, and the *density* is given by $\delta = D/N$. Any basis for C forms a $K \times N$ *generator matrix* for the code. If $F = \mathrm{GF}(2)$ we speak of a *binary* code.

The basic idea is to take a $2^q \times 2^{r(n)}$ generator matrix G for a binary code C of constant density $\delta = 1/2 - \epsilon$, and have the probabilistic NTM N work as follows:

1. Flip $r(n)$ coins to choose a column j.

2. Guess a row i, $1 \leq i \leq 2^q$, identified with a possible witness string $y_i \in \{0,1\}^q$.

3. Accept iff $B(x, y_i) \wedge G(i,j) = 1$.

Suppose $S = S_x$ is nonempty. Then 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). Then $\#acc(N_j, x)$ is odd iff the jth entry of w_S is a '1'. Since the proportion of non-0 entries of w_S is at least δ, $\Pr_j[\#acc(N_j,x) \text{ is odd}] > \delta$; that is, N reduces A to parity with success probability at least δ. And if S is empty, N has no accepting computations at all. Thus to show $\mathrm{NQL} \subseteq \mathrm{RQL}[\oplus\mathrm{QL}]$, we need to construct C so that $G(i,j)$ is computable in quasilinear time. Our code C is the *concatenation* of two simpler codes:

• The *Hadamard code* $\mathcal{H}_k$ 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.

• 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 $F = \mathrm{GF}(q)$.[1] 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. When $r \leq q-1$ a generator matrix for this code is easy to describe: it 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 $I \subseteq F$,

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

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 $1 - r/|F|$.

If with $q = 2^k$ we simply regarded $\mathcal{R}_q(r,m)$ as a binary code of length kN, we would only be able to assert that the density is at least $1/2k$, because two distinct elements $a_1, a_2 \in \mathrm{GF}(2^k)$ might differ in only one out of k places as binary strings. But if we apply the Hadamard code to a_1 and a_2, the two resulting strings, though of length 2^k, differ in at least $1/2$ their places, yielding our code C of length $2^k N$ and density $1/2 - \epsilon$, where $\epsilon = r/2|F|$. This is done by step 11. of our construction of N:

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

[1] Here q and r are different from their use above; we take $q = 2^k$ and $r = 2^b - 1$ below.

11. Compute $G(y, j) := u \cdot v$
12. Accept iff $B(x, y) \wedge G(y, j) = 1$.

Let t_B be the time to compute the witness predicate $B(x, y)$, and let $\log^+ n$ abbreviate $\log n \operatorname{loglog} n \log \operatorname{loglog} n$.

Theorem 3.1. *For any fixed $\epsilon < 1/2$, the probabilistic NTM N accepts A with success probability $1/2 - \epsilon$, making q nondeterministic moves and running in time $O(q \log^+ q)$ (apart from the time to recognize B), and uses a number of random bits bounded by*

$$r = 2q - q \operatorname{loglog} q / \log q + (1 + \log(1/\epsilon)) q / \log q + O(\log q).$$

Proof Sketch. Step 10 dominates the running time. To multiply two polynomials of degree $k - 1$ over GF(2) and reduce them modulo α in the field GF(2^k) takes time $t_1 = O(k \log k \operatorname{loglog} k)$ on standard Turing machine models (see [AHU74] and [Rab80]). The time to compute a^i in GF(2^k) where $i \leq n$ is $t_2 = O(\log n \cdot 2k \log k \operatorname{loglog} k)$ via repeated squaring. Hence the time to evaluate the monomial is at most $O(mt_2 + mt_1) = O(m \log(n) k \log k \operatorname{loglog} k) = O(n \log^+ n)$, since $m = O(n / \log n)$ and $k = O(\log n)$. $\square$

Corollary 3.2. NQL $\subseteq$ RQL[$\oplus$QL]. $\square$

The first open problem is whether two or more alternations can be done in quasilinear time; that is, whether $\mathrm{NQL}^{\mathrm{NQL}} \subseteq \mathrm{BQL}[\oplus\mathrm{QL}]$. The obstacle is the apparent need to amplify the success probabilities of the second level to $1 - 2^{-q}$, for which straightforward "amplification by repeated trials" takes time q^2. The second is whether the code can be improved and still give quasi-linear runtime. Our codes have rate $R = K/N = 2^q / 2^{(2q - \cdots)}$, which tends to 0 as q increases. Families of codes are known for which R (as well as δ) stays bounded below by a constant; such (families of) codes are called *good*. Good codes require only $q + O(1)$ random bits in the above construction. The codes in [Jus72, TV91, ABN+92, JLJH92, She93] are good. However, we do not know of any good codes which give quasi-linear runtime in the above construction.

4. Search Versus Decision in Quasilinear Time

The classical method of computing partial, multivalued functions using sets as oracles is the *prefix-set* method (cf. [Sel88]). To illustrate, let f be an arbitrary length-preserving, partial function from Σ^* to Σ^*. Define:

$$L_f = \{x\#w \mid w \text{ is a prefix of some value of } f(x)\}.$$

Clearly f is computable in quadratic time using L_f as an oracle. First we observe that for "random" functions f, quadratic time is best possible.

Theorem 4.1. *There exist length-preserving functions $f : \Sigma^* \to \Sigma^*$ with the property that there does not exist an oracle set B relative to which f is computable in less than $n^2 - n$ steps.*

Proof. Let B and an OTM M such that $M^B(x) = f(x)$ on all strings $x \in \{0,1\}^n$ be given, and suppose M^B runs in time $g(n)$. Then the following is a description of f on $\{0,1\}^n$:

- The finite control of M, plus finite descriptions of the function $g(n)$ and "this discussion" (see [LV90]). This has total length some constant C.
- A look-up table for all the strings of length $< n$ which belong to B—this is specifiable by a binary string of length $\sum_{i=0}^{n-1} 2^i = 2^n - 1 < 2^n$.
- For each $x \in \{0,1\}^n$, the answers given by B to those queries z made by M on input x such that $|z| \geq n$. There are at most $g(n)/n$ such queries. All of this is specifiable by a binary string of length $2^n g(n)/n$.

Now let K_f be the *Kolmogorov complexity* of f, relative to some fixed universal Turing machine. Then $C + 2^n + 2^n g(n)/n < K_f$, so $g(n) > nK_f/2^n - n - nC/2^n$. Since functions $f : \{0,1\}^n \to \{0,1\}^n$ are in 1-1 correspondence with binary strings of length $n2^n$, and a simple counting argument shows that some such strings have Kolmogorov complexity at least $n2^n$, there exist f with $K_f \geq n2^n$. Then $g(n) > n^2 - n$. $\square$

(Remarks: The $n^2 - n$ is close to tight—an upper bound of $g(n) \leq n^2 + 2n \log n$ is achievable by a modification of L_f. By diagonalization one can also construct such functions f which are computable in exponential time.)

Hence the equivalence between functions and sets does not carry over to quasilinear time complexity in general. Theorem 4.1 can be read as saying that Kolmogorov-random functions have so much information that large query strings are needed to encode it. We are interested in whether natural functions in NP, such as witness functions for NP-complete problems, pack information as tightly.

Let L be a language in NP and let B be some polynomial-time witness predicate for L. Define the partial multivalued function f_B by:

$$f_B(x) \mapsto y, \text{if } |y| = q(|x|) \text{ and } B(x,y).$$

Then f_B is called a *search function* for L. The following is a straightforward extension of the standard notion of search reducing to decision in polynomial time [BD76, BBFG91, NOS93, HNOS93] to other time bounds $t(n)$.

Definition 4.1. Let $L \in$ NP and a time bound $t(n)$ be given. Then we say that *search reduces to decision for L in time $t(n)$* if there exists a witness predicate B for L and a $t(n)$ time-bounded deterministic oracle TM M such that for all inputs x, if $x \in L$ then $M^L(x)$ outputs some y such that $f_B(x) \mapsto y$, and if $x \notin L$ then $M^L(x) = 0$.

Let polylog n abbreviate $(\log n)^{O(1)}$ as before. Then DTIME$[2^{\text{polylog}\, n}]$ is often referred to as *quasi-polynomial* time (cf. [Bar92]).

Theorem 4.2. *Let $L \in$ NP. If search reduces to decision for L in quasilinear time, then $L \in$ DTIME$[2^{\text{polylog}\, n}]$.*

Proof. Let M be the oracle TM from Definition 4.1, and let c and k be constants such that M runs in time $cn\log^k n$. We may suppose that M itself verifies that its output y satisfies $B(x,y)$. Let $f(n) := n/\log n$. Let n_0 be a fixed constant whose value we determine later; on inputs x of length $< n_0$, whether $x \in L$ is looked up in a table.

Now we create a non-oracle TM M' which operates as follows on any input x of length $n \geq n_0$: M' simulates M. Whenever M makes a query z and $|z| < n_0$, M' answers from the table. If $|z| > f(n)$, we call z a "large query." Here M' branches, simulating both a "yes" and a "no" answer to z. Finally, if $n_0 \leq |z| \leq f(n)$, then M' calls itself recursively on input z to answer the query. The above is a recursive description of what M' does. The actual machine M' simulates both the recursion and the branching on large queries using a stack, and halts and accepts iff at some point M outputs a string y such that $B(x,y)$ holds. Clearly M' accepts L.

Let $t_{M'}(n)$ stand for the worst-case running time of M' on inputs of length n. We show that for all n, $t_{M'}(n) \leq 2^{c\log^{k+2} n}$. Since table-lookup takes only linear time, this holds for $n < n_0$. Now consider the binary tree T whose nodes are large queries made by M, and whose edges represent computation paths by M' between large queries. Then T has depth at most $c\log^{k+1} n$ and at most $2^{c\log^{k+1} n}$ branches. The number of small queries on each branch is at most $c\log^k n$, and each such query has length at most $n/\log n$. Hence the time taken by M' to traverse all branches, namely $t_{M'}(n)$, meets the bound:

$$t_{M'}(n) \leq 2^{c\log^{k+1} n} \cdot cn\log^k n \cdot t_{M'}(n/\log n). \tag{2}$$

By induction hypothesis, $t_{M'}(n/\log n) \leq 2^{c(\log n - \mathrm{loglog}\, n)^{k+2}}$. Elementary calculation shows that there is a fixed n_0 such that for all $n \geq n_0$,

$$2^{c\log^{k+1} n} \cdot cn\log^k n \cdot 2^{c(\log n - \mathrm{loglog}\, n)^{k+2}} \leq 2^{c\log^{k+2} n}.$$

Thus $t_{M'}(n) \leq 2^{c\log^{k+2} n}$. □

The result also holds if L belongs to nondeterministic quasi-polynomial time. Our next corollary follows immediately from the NP-completeness of SAT.

Corollary 4.3. *If search reduces to decision for SAT is quasilinear time, then*

$$\mathrm{NP} \subseteq \mathrm{DTIME}[2^{\mathrm{polylog}\, n}].$$

Now we observe that the proof technique of Theorem 4.2 can be used to give some evidence that the quadratic bound on the search to decision reduction for SAT is optimal.

Corollary 4.4. *If there exists an $\epsilon > 0$ such that search reduces to decision for SAT in $\mathrm{DTIME}[n^{1+\epsilon}]$, then $SAT \in \mathrm{DTIME}[2^{n^\epsilon}]$, and also $\mathrm{NP} \subseteq \mathrm{DTIME}[2^{n^\epsilon}]$.*

Stearns and Hunt [SH90] define a language $L \in \mathrm{NP}$ to have *power index* ϵ if ϵ is the infimum of all δ such that $L \in \mathrm{DTIME}[2^{n^\delta}]$. They classify familiar NP-complete problems according to known bounds on their power indices, and conjecture that SAT has power index 1. In this setting, Corollary 4.4 can be restated as :

Corollary 4.5. *If there exists an $\epsilon > 0$ such that search reduces to decision for SAT in* DTIME$[n^{1+\epsilon}]$*, then SAT has power index at most ϵ.*

This establishes a relation between reducing search to decision and the power index of an NP language. However, it is not necessarily the case that having a low power index implies that search reduces to decision efficiently, or even in polynomial time at all. Let EE stand for DTIME$[2^{2^{O(n)}}]$, and NEE for its nondeterministic counterpart. The classes EE and NEE were recently considered by Beigel, Bellare, Feigenbaum, and Goldwasser [BBFG91], and there are reasons for believing it unlikely that NEE = EE.

Theorem 4.6. *Suppose* NEE $\neq$ EE. *Then for all $k > 0$ there is a tally language in* NP *whose power index is at most $1/k$, but for which search does not reduce to decision in polynomial time.*

Proof Sketch. We use the techniques developed by [BBFG91]. Let T be the tally set constructed in [BBFG91] such that search does not reduce to decision for T in polynomial time, unless NEE = EE. There exists a polynomial p such that for all $x \in T \cap \Sigma^n$, some witness for x is of length $p(n)$. Now, define:

$$T^k = \{0^{p(n)^k} \mid 0^n \in T\}.$$

It is easy to see that T^2 has power index at most $1/k$, since an exhaustive search for witnesses recognizes T^k in time $2^{n^{1/k}}$. However if search reduces to decision in polynomial time for T^k, then the same holds for T, which is a contradiction. $\square$

Finally, it is interesting to ask whether there are length-preserving 1-1 functions f which are computable in *qlin* time but not invertible in *qlin* time. Homer and Wang [HW89] construct, for any $k \geq 1$, functions computable in quadratic time which are not invertible in time $O(n^k)$, but their methods seem not to apply for *qlin* time or f length-preserving. If DQL $\neq$ UQL, then such "quasilinear one-way" functions exist, but unlike the polynomial case (assuming P $\neq$ UP), the converse is not known to hold. We look toward further research which might show that length-preserving functions with certain "pseudorandom" properties cannot be inverted in *qlin* time, unless unlikely collapses of quasilinear classes occur.

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 Trans. Info. Thy.*, 38(2):509–512, March 1992.

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

[AMV88] G. Agnew, R. Mullin, and S. Vanstone. Fast exponentialtion in $GF(2^n)$. In *Proceedings, Advances in Cryptology: Eurocrypt '88*, volume 330 of *LNCS*, pages 251–255, 1988.

[Bar92] D. Mix Barrington. Quasipolynomial size circuit classes. In *Proc. 7th Structures*, pages 86–93, 1992.

[BBFG91] R. Beigel, M. Bellare, J. Feigenbaum, and S. Goldwasser. Languages that are easier than their proofs. In *Proc. 32nd FOCS*, pages 19–28, 1991.

[BCGL89] S. Ben-David, B. Chor, O. Goldreich, and M. Luby. On the theory of average-case complexity. In *Proc. 21st STOC*, pages 204–216, 1989.

[BD76] A. Borodin and A. Demers. Some comments on functional self-reducibility and the NP hierarchy. Technical Report TR 76-284, Cornell Univ. Comp. Sci. Dept., 1976.

[BFLS91] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. 23rd STOC*, pages 21–31, 1991.

[BG93] J. Buss and J. Goldsmith. Nondeterminism within P. *SIAM J. Comp.*, 22:560–572, 1993.

[CR73] S. Cook and R. Reckhow. Time bounded random access machines. *J. Comp. Sys. Sci.*, 7:354–375, 1973.

[CRS93] S. Chari, P. Rohatgi, and A. Srinivasan. Randomness-optimal unique element isolation, with applications to perfect matching and related problems. In *Proc. 25th STOC*, pages 458–467, 1993.

[CW79] J. Carter and M. Wegman. Universal classes of hash functions. *J. Comp. Sys. Sci.*, 18:143–154, 1979.

[GS89] Y. Gurevich and S. Shelah. Nearly-linear time. In *Proceedings, Logic at Botik '89*, volume 363 of *LNCS*, pages 108–118. Springer-Verlag, 1989.

[Gup93] S. Gupta. On isolating an odd number of elements and its applications to complexity theory. Technical Report OSU-CISRC-6/93-TR24, Dept. of Comp. Sci., Ohio State University, 1993.

[HNOS93] E. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. P-selective sets, and reducing search to decision versus self-reducibility. Technical Report 93–21, Computer Science Dept., SUNY at Buffalo, Buffalo, NY 14260, 1993.

[HS90] H. Hunt III and R. Stearns. The complexity of very simple Boolean formulas, with applications. *SIAM J. Comp.*, 19:44–70, 1990.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison–Wesley, Reading, MA, 1979.

[HW89] S. Homer and J. Wang. Absolute results concerning one-way functions and their applications. *Math. Sys. Thy.*, 22:21–35, 1989.

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

[Jus72] J. Justesen. A class of constructive asymptotically good algebraic codes. *IEEE Trans. Info. Thy.*, IT-18:652–656, September 1972.

[JY90] D. Joseph and P. Young. Self-reducibility: the effects of internal structure on computational complexity. In A. Selman, editor, *Complexity Theory Retrospective*, pages 82–107. Springer-Verlag, 1990.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Inf. Proc. Lett.*, 14:39–43, 1982.

[LL76] R. Ladner and N. Lynch. Relativization of questions about log-space computability. *Math. Sys. Thy.*, 10:19–32, 1976.

[LV90] M. Li and P. Vitányi. Applications of Kolmogorov complexity in the theory of computation. In A. Selman, editor, *Complexity Theory Retrospective*, pages 147–203. Springer-Verlag, 1990).

[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 *Proc. 22nd STOC*, pages 213–223, 1990.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comp.*, 22:838–856, 1993.

[NOS93] A. Naik, M. Ogiwara, and A. Selman. P-selective sets, and reducing search to decision vs. self-reducibility. In *Proc. 8th Structures*, pages 52–64, 1993.

[PZ83] C. H. Papadimitriou and S. Zachos. Two remarks on the power of counting. In *The 6th GI Conference on Theoretical Computer Science*, Lecture Notes in Computer Science No. 145, pages 269–276. Springer-Verlag, 1983.

[Rab80] M. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comp.*, pages 273–280, 1980.

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

[Sch78] C. Schnorr. Satisfiability is quasilinear complete in NQL. *J. ACM*, 25:136–145, 1978.

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

[Sel88] A. Selman. Natural self-reducible sets. *SIAM J. Comp.*, 17:989–996, 1988.

[SH86] R. Stearns and H. Hunt III. On the complexity of the satisfiability problem and the structure of NP. Technical Report 86–21, Dept. of Comp. Sci., SUNY at Albany, 1986.

[SH90] R. Stearns and H. Hunt III. Power indices and easier hard problems. *Math. Sys. Thy.*, 23:209–225, 1990.

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

[Sti90] D. Stinson. Some observations on parallel algorithms for fast exponentiation in GF(2^n). *SIAM J. Comp.*, 19:711–717, 1990.

[Sto77] L. Stockmeyer. The polynomial time hierarchy. *Theor. Comp. Sci.*, 3:1–22, 1977.

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

[Tod89] S. Toda. On the computational power of PP and $\oplus$P. In *Proc. 30th FOCS*, pages 514–519, 1989.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comp.*, 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.

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

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theor. Comp. Sci.*, 3:23–33, 1977.

[Wra78] C. Wrathall. Rudimentary predicates and relative computation. *SIAM J. Comp.*, 7:194–209, 1978.

[WW86] K. Wagner and G. Wechsung. *Computational Complexity.* D. Reidel, 1986.