Languages that are Easier than their Proofs*

Richard Beigel[†] Mihir Bellare[‡] Joan Feigenbaum[§] Shafi Goldwasser[¶]

Abstract

A basic question about NP is whether or not search reduces in polynomial time to decision. We indicate that the answer is negative: under a complexity assumption (that deterministic and nondeterministic double-exponential time are unequal) we construct a language in NP for which search does *not* reduce to decision.

These ideas extend in a natural way to interactive proofs and program checking. Under similar assumptions we present languages in NP for which it is harder to prove membership interactively than it is to decide this membership. Similarly we present languages where checking is harder than computing membership.

Each of the following properties — checkability, random-self-reducibility, reduction from search to decision, and interactive proofs in which the prover's power is limited to deciding membership in the language itself — implies coherence, one of the weakest forms of self-reducibility. Under assumptions about triple-exponential time, we construct incoherent sets in NP. Finally, without any assumptions, we construct incoherent sets in DSPACE($n^{\log^* n}$), yielding the first uncheckable and non-random-self-reducible sets in DSPACE($n^{\log^* n}$).

[†] Yale University Computer Science Department, P.O. Box 2158 Yale Station, New Haven, CT 06520. e-mail: beigel-richard@cs.yale.edu. Supported in part by NSF grants CCR-8808949 and CCR-8958528.

[‡] IBM T.J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598. e-mail: mihir@watson.ibm.com. Work done while author was at MIT supported in part by NSF grant No. CCR-8719689 and DARPA grant No. N00014-89-J-1988.

[§]AT&T Bell Laboratories, Room 2C473, 600 Mountain Avenue, Murray Hill, NJ 07974. e-mail: jf@research.att.com.

[¶]MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139. e-mail: shafi@theory.lcs.mit.edu. Supported in part by NSF grant No. CCR-8657527, DARPA grant No. DAAL03-86-K-0171 and grant No. 89-00312 from the United States - Israel Binational Science Foundation (BSF), Jerusalem, Israel.

* This paper is a combination of the results by Beigel and Feigenbaum [BF] and Bellare and Goldwasser [BG]. Some proofs have been omitted because of space limitations.

1 Introduction

The work on interactive proofs brought back to light an old question: how powerful does a prover need to be to convince a verifier of membership in a language L? Clearly, the prover needs at least the power to decide the language for himself. But is this enough?

For the interactive proofs known for complete problems in NP, $P^{\#P}$ and PSPACE it is sufficient for the prover to be able to decide membership in the language. Such power is also sufficient for many of the languages in IP that have been looked at closely (graph Isomorphism, graph Non-Isomorphism, quadratic non-residuosity). One striking exception is the language of quadratic residuosity, for which all the known interactive proofs require the prover to do more than decide quadratic residuosity (see §3.2). Of course the complete languages in co-NP also constitute an exception: All known proof systems require the prover to be at least as powerful as $\mathrm{MOD}_k\mathrm{P}$ for some k (cf. [BaF]) and hence at least as powerful as PH, though it is not known whether even PH provides sufficient power.

This is essentially a generalization of the old question of whether search problems reduce to their decision counterpart for NP. Namely, is computing a witness for membership in $L \in \mathrm{NP}$ any harder than establishing the existence of such a witness? For SAT and other NP-complete problems it is well known that the answer is no: given an oracle for membership a witness can be computed in polynomial time (self-reducibility). But for general $L \in \mathrm{NP}$ the problem remains open.

In this paper, we use natural complexity assumptions to indicate that proving membership may be harder than deciding it. As a first example we look at NP-provability. We then turn to the power of the prover in interactive proofs and to program checking. Finally we present various negative results on the subject of *coherence* [Ya] and use these to derive negative results on checking and random-self-reducibility.

Let us proceed to describe our results in detail.

1.1 Decision Versus Search in NP

The definition of the class NP is in terms of decision problems: decide whether or not formula ϕ is satisfiable, decide whether or not graph G has a Hamiltonian cycle. But typically one wants more than just this yes/no answer: one wants to find the satisfying assignment of ϕ (if one exists) or find a Hamiltonian cycle of G (if one exists). These are the so-called *search* problems (also sometimes called optimization problems) associated with the decision problem.

For NP-complete problems search reduces to decision — sometimes called *self-reducibility* this is one of the most basic and well known facts in the theory of computation. But we do not know whether this is true in general (i.e., does search reduce to decision for *every* $L \in$ NP?).

Let us now take a moment to recall more precisely what we mean by reducing search to decision for a general NP problem.

DEFINITION. Suppose L is a language in NP. Then there exists a polynomial time computable binary predicate $\rho(\cdot,\cdot)$ and a constant $c \geq 0$ such that $x \in L$ if and only if there is a *witness* $w \in \{0,1\}^{n^c}$ such that $\rho(x,w) = 1$ (where $n = |x|$). The decision problem on input x is to determine whether or not there exists a witness for x, while the search problem is to find the witness if it exists. We say that *search reduces to decision* for L if there exist ρ and c as above such that there is a polynomial time procedure (called a "witness finder") for finding w given an oracle for L.

Note that we say search reduces to decision for L as long as there is *some* ρ and c with respect to which a witness finder exists. This corresponds to the fact that there are many different predicates ρ which could define the same language L, and we want as liberal a notion of the reduction as possible.

OUR RESULT. Let EE $= \bigcup_{c \geq 0} \text{DTIME}(2^{2^{c+n}})$ and NEE $= \bigcup_{c \geq 0} \text{NTIME}(2^{2^{c+n}})$. Then we have the following

Theorem 1.1 *Suppose* EE $\neq$ NEE. *Then there is a language in* NP *for which search does not reduce to decision.*

(Note that an unconditional proof would imply P $\neq$ NP).

PREVIOUS AND RELATED WORK. Karp, Upfal and Wigderson [KUW] have studied the complexity of *parallel* reduction of search to decision for many problems. Impagliazzo and Naor [IN] present oracles under which P = NP $\cap$ co-NP but there are search problems not solvable in polynomial time. Impagliazzo and Tardos [IT] have relativized results for search versus decision in non-deterministic *exponential* time. More recently, Impagliazzo and Sudan [IS] have constructed NP languages for which search does not reduce to decision under the assumption that NE $\neq$ co-NE. They have also shown that under the assumption E $\neq$ NE there is a (particular) polynomial time relation $\rho(x,y)$ such that the search problem for ρ does not reduce to the corresponding decision problem for ρ.

1.2 Competitive Proof Systems: the Natural Extension

NP represents the simplest kind of proof system — the interaction is restricted to a single message from the prover to the verifier and both parties are deterministic. In this setting the question of whether or not search reduces to decision for $L \in$ NP is seen to capture the computational difficulty of the prover's task. For we observe that search reduces to decision for $L \in$ NP if and only if L has an NP proof system with a prover who is polynomial time given an oracle for L. So our result of Theorem 1.1 indicates that there is a language L to give an NP proof of which any prover must use power over and above that necessary to decide L.

NP-proof systems, however, are very restrictive. Would the prover's task be alleviated if the parties were allowed interaction and the proof was now only required to be correct with high probability? In other words, we now consider interactive proofs. Let us call an interactive proof system *competitive* if the prover[1] runs in probabilistic polynomial time given access to L as an oracle. We say that L is harder to prove than to decide if it does not have a competitive IP.

Are there languages in NP (and more generally in IP) which are harder to prove than to decide? In other words, does the extra leeway provided by interaction and randomness reduce the burden on the prover, or does the discrepancy between proving and deciding remain even if coins and interaction are allowed?

Quadratic residuosity again provides a telling example. Let $QR = \{ \underline{(x,N)} : \exists y \in Z_N^* \text{ s.t. } x \equiv y^2 \pmod{N} \}$, and $QNR = \overline{QR}$. Search is not known to reduce to decision for QNR — all known NP-proofs that $(x,N) \in QNR$ imply the ability to factor N, and factoring is not known to be reducible to quadratic residuosity. Yet, we do know of interactive proofs for $(x,N) \in QNR$ where it suffices for the prover to be

[1] The prover here refers, of course, to the "honest" prover; the soundness condition of the proof system is as usual required to hold with respect to any computationally unbounded prover.

able to tell membership in QR (i.e., QNR does have competitive interactive proofs).

On the other hand, there is no known interactive proof for QR where it suffices for the prover to be able to decide membership in QR (i.e QR is not known to have a competitive interactive proof). Our results indicate that this is no accident. Letting BPEE denote the class of languages recognized in bounded error by a probabilistic TM running in time $2^{2^{c+n}}$ for some constant $c \geq 0$ we have the following

Theorem 1.2 *If* NEE $\not\subseteq$ BPEE *then there is a language in* NP *which is harder to prove than to decide.*

Function-restricted interactive proof systems, introduced by Blum and Kannan [BK], also make the restriction that the honest prover only compute membership in L. But, in contrast to competitive IP, they also assume that the dishonest prover is a predetermined function (and thus its messages are independent of the history). Clearly comp-IP $\subseteq$ fr-IP. Function-restricted IPs were introduced by [BK] in order to relate program checking to interactive proofs. We introduce competitive IP to address the question of how much power is necessary for the honest prover to prove membership interactively, whence it is imperative to not weaken the definition of IP by making any assumptions on the power of the dishonest prover. Our techniques do extend to show (under the same assumption) the existence of a language in NP which is not in fr-IP.

One could define comp-MIP similarly to comp-IP but this is identical to fr-IP (cf. [FRS]).

1.3 Program Checking

Blum and Kannan [BK] introduced the notion of program checkers (see §4 for full definitions). Since checkability implies membership in MIP and co-MIP (cf. [FRS]) and MIP = NEXP (cf. [BFL]), all sets outside of NEXP $\cap$ co-NEXP are uncheckable. We give further negative results here.

First, under the same assumptions as above we present a language of reasonable complexity which is not checkable:

Theorem 1.3 *If* NEE $\not\subseteq$ BPEE *then there is a language in* NP *that does not have a checker.*

We also prove the following unconditional result.

Theorem 1.4 *There is a language in* DSPACE($n^{\log^* n}$) *that does not have a checker.*

Recently Spielman [Sp] constructed an uncheckable set in Σ_2^P under the assumption that $\Sigma_2^E \neq \Pi_2^E$.

1.4 Incoherence and its Consequences

Informally, a language L is coherent [Ya] if the membership of x in L can be decided in probabilistic polynomial time and bounded error by a machine (called the *examiner*) that has access to L as an oracle but is allowed to query this oracle only on points different from x. Extending [Ya], we say that the language is deterministically coherent if the examiner is deterministic and weakly coherent if the examiner is nonuniform; sets that are not weakly coherent are called strongly incoherent.

Yao [Ya] showed that if L has a checker then it is coherent. Similarly one can show that, if L has a competitive (or even function-restricted) proof system or is uniformly-random self-reducible, then it is coherent. These implications motivate the construction of incoherent sets, as a tool to show negative results. We give more details in Section 5.

Yao's construction yielded an incoherent set in DSPACE($2^{n^{\log^* n}}$). Here we draw a connection between incoherence and sparseness, and we use it to construct incoherent sets of much lower complexity. Let NEEEXP denote the union, over all polynomials p, of NTIME($2^{2^{2^{p(n)}}}$) and let BPEEEXP be the analogous bounded-probabilistic time class. We first show

Theorem 1.5 *If* NEEEXP $\not\subseteq$ BPEEEXP *then there is an incoherent set in* NP.

Note that the assumption required is stronger than in our theorems about checking. Indeed, the uncheckable set of Theorem 1.3 is not known to be incoherent. This is to be expected, because coherence is weaker than any of the other properties discussed above. For example, reductions of search to decision, competitive proof systems, and program checkers all involve a notion of "proof" that x is in L. Coherence does not involve a notion of proof: If the examiner is run with some oracle $A \neq L$, then it may give incorrect answers.

Without any assumptions we prove the following:

Theorem 1.6 *There is a (strongly) incoherent set in* DSPACE($n^{\log^* n}$).

The unconditional consequences for checking and random-self-reducible sets are immediate. In particular we get a set in DSPACE($n^{\log^* n}$) that is not uniformly-random self-reducible, improving Feigenbaum, Kannan and Nisan [FKN].

2 Search Versus Decision in NP

In this section we present a simple construction of a language in NP for which search does not reduce to decision, assuming that EE $\neq$ NEE. In later sections we will extend the argument to interactive proofs and program checking. Let us begin with a more precise definition of what it means for search to reduce to decision.

Definition 2.1 Suppose $L \subseteq \{0,1\}^*$. We say that *search reduces to decision* for L if there exists a polynomial time computable binary predicate $\rho(\cdot,\cdot)$, a polynomial time oracle Turing Machine $W(\cdot)$, and a constant $c \in \mathbb{N}$ such that for all n and all $x \in \{0,1\}^n$ it is the case that

(1) $x \in L$ if and only if there exists a $w \in \{0,1\}^{n^c}$ such that $\rho(x,w)=1$

(2) If $x \in L$ then $W^L(x) \in \{0,1\}^{n^c}$ and $\rho(x, W^L(x)) = 1$.

($W^L(x)$ denotes the output of W^L on input x.) A string $w \in \{0,1\}^{n^c}$ such that $\rho(x,w)=1$ is called a *witness* for x, and W is called the witness extractor.

Notice that we have chosen a liberal definition of the reduction of search to decision. In particular we say search reduces to decision if there is *any* predicate ρ with respect to which there exists a witness extractor. Moreover, the only limitation on the witness extractor is that it be polynomial time. It can ask any polynomial number of questions, and the lengths of these question may be different from the input length n (while of course remaining polynomially bounded).

Since our results will be negative all this only makes them stronger.

Let μ_A denote the census function of a language A (i.e., $\mu_A(n)$ is the number of strings in A which have length $\leq n$). Sparse languages ($\mu_A(n) \leq n^{O(1)}$) have been extensively studied in the literature [BH],[Be],[Ma],[Fo]. [HSI] consider languages of logarithmic sparseness ($\mu_A(n) \leq O(\log n)$); they call them *super-sparse*. Here we will use something a little stronger: languages which combine logarithmic sparseness with the property that it be possible to efficiently identify a logarithmic sized superset of the strings below any given length. More formally we have the following

Definition 2.2 We say that a language L is *uniformly log-sparse* if there exists a language $C \in$ P such that $L \subseteq C$ and $\mu_C(n) = O(\log n)$ (C is called a *candidate selector* for L).

The interest of uniformly log-sparse languages is that they form a class for which the problem of reducing search to decision is particularly hard. In fact, the reduction is only possible in the trivial case where the language is already in P:

Lemma 2.3 *Suppose L is a uniformly log-sparse language for which search reduces to decision. Then $L \in$ P.*

Proof: By assumption, there exists a polynomial time computable predicate ρ, a witness extractor W, and a constant $c \geq 0$ such that the conditions of Definition 2.1 hold. We will construct a polynomial time machine M which decides L.

We can assume that there is a constant $d > 0$ such that on any input $x \in \{0,1\}^n$ the machine W will halt in $\leq dn^d$ steps and output a string of length n^c, regardless of how the oracle queries of W are answered. Let C be the candidate selector for L which satisfies the conditions of Definition 2.2. On input $x \in \{0,1\}^n$ the machine M behaves as follows:

(1) M runs W on input x. Each time W makes an oracle query q, the machine M provides a response as follows:
   (1) If $q \notin C$ then M responds with 0.
   (2) Else it continues by trying in parallel both possible answers 0 and 1. That is, M branches into two parallel computations. In the first it lets the response to q be 0 and in the second it lets the response to q be 1. It then continues to run W in each computation.

   In this manner M generates a number of parallel computations. After dn^d steps all of these computations have halted and each has yielded an n^c bit output (the output of W).

(2) M now examines the set of outputs from the previous step. It accepts if at least one of these outputs w satisfies $\rho(x,w)=1$, and rejects otherwise.

This completes the description of the machine M. The fact that it works should be clear, but for completeness let us spell it out.

First, to see that M accepts x if and only if $x \in L$, it suffices to check that on at least one of the parallel computations all oracle queries are correctly (that is, according to L) answered. But Step (1) is obviously correct by definition of the candidate selector and in step (2) everything is being tried, so one of the runs will certainly end up having all the right query answers.

The next thing to check is that M runs in polynomial time. For this note that any query q has length at most the running time dn^d of W, and branching only occurs when $q \in C$. The number of branches on any path is thus $\leq \mu_C(dn^d) = O(\log n)$, so the total number of parallel computations undertaken by M is $n^{O(1)}$.

This completes the proof. ■

Hartmanis, Sewelson and Immerman [HSI] call a language L *super-sparse* if $\mu_L(n) = O(\log n)$, and show that there is a super-sparse language in NP − P if EE $\neq$ NEE[2]. Super-sparseness is weaker than uniform log-sparseness in that no candidate selector is required, but it is easy to see that a uniformly log-sparse language in NP − P nonetheless exists under the same assumption.

Lemma 2.4 *If* EE $\neq$ NEE *then there is a uniformly log-sparse language in* NP − P.

Proof: A standard "downward separation" padding argument. Suppose $L^* \in$ NEE − EE. Define $L = \{ y.0^{g(|y|)-|y|} : y \in L^* \}$, where $g(k) = 2^{2^k}$. It is easy to see that L is uniformly log-sparse and in NP − P. ■

We can now put the pieces together to state the result:

Theorem 2.5 *If* EE $\neq$ NEE *then there exists a language in* NP *for which search does not reduce to decision.*

Proof: By Lemma 2.4 there exists a uniformly log-sparse language $L \in$ NP − P. By Lemma 2.3, search cannot reduce to decision for L. ■

Note that the fact that search does not reduce to decision for L implies that L is not NP-complete. The existence of a non NP-complete language in NP − P can however be established assuming only P $\neq$ NP (cf. [La, GJ]).

3 Competitive Interactive Proofs

In a conventional interactive proof system [GMR] the prover's computational power is not restricted. Soundness is only enhanced by this; this is an asset we do not want to forgo. It is however desirable to make the actual (honest) prover as efficient as possible.

How efficient can the honest prover be? Certainly he would need at least the ability to decide the language himself. We define a competitive interactive proof system as one where the honest prover is allowed no more than this. Specifically, he must run in probabilistic polynomial time given access to L as an oracle. Thus,

[2] [HSI] claimed the converse as well, but Allender [Al] points out that their proof is flawed and the theorem cannot be proved using techniques that relativize.

Definition 3.1 Let P be a probabilistic polynomial time oracle interactive TM and V a probabilistic polynomial time interactive TM. We say that (P,V) is a competitive interactive proof system for a language L if

- For every $x \in L$ the probability that (P^L, V) accepts x is $\geq \frac{2}{3}$
- For every $x \notin L$ and every interactive TM $\widehat{P}$, the probability that $(\widehat{P}, V)$ accepts x is $\leq \frac{1}{3}$.

The first condition is the completeness condition and the second is the soundness condition. We call P a competitive prover and denote by comp-IP the class of languages possessing competitive interactive proof systems.

Observe that search reduces to decision for L if and only if L has a competitive NP proof system. In this sense competitive interactive proofs are indeed the natural extension of the problem of search versus decision.

We say L is *harder to prove than to decide* if it does not have a competitive interactive proof.

3.1 A language in NP which is Harder to Prove than to Decide

We saw in §2 that reducing search to decision for uniformly log-sparse languages was hard. In particular, this means that non-trivial uniformly log-sparse languages do not have competitive NP proof systems. The truth, however, is that proving membership in a uniformly log-sparse language remains hard even when interaction and randomness are allowed — we will show here that non-trivial uniformly log-sparse languages are harder to prove than to decide.

Lemma 3.2 *Suppose L is uniformly log-sparse and has a competitive interactive proof system. Then $L \in$* BPP.

Proof: The first step is to construct a probabilistic polynomial time oracle TM D such that on input $x \in \{0,1\}^n$,

- if $x \in L$ then D^L accepts with probability $\geq 1 - 2^{-n}$
- if $x \notin L$ then the probability that D^A accepts is $\leq 2^{-n}$ for all oracles A.

D is easily constructed given a competitive interactive proof system (P, V) for L — given an oracle for A, the machine D, on input x, simply samples in probabilistic polynomial time the space of conversations between P^A and V on input x and accepts if and only if the conversation was accepting. The error probability is reduced to the desired amount by standard techniques.

We can now conclude the proof by extending the ideas of the proof of Lemma 2.3. The machine D will play the role that the witness finder played in that proof. Briefly, a BPP machine M to decide L works as follows on input x. It runs D on input x and answers its oracle queries according to the same rules as those used in the proof of Lemma 2.3. M accepts if and only if D accepts on at least one of the parallel computations. Since one of these parallel computations is the correct one, M accepts with probability $\geq 1 - 2^{-n}$ if $x \in L$. For the case of $x \notin L$ we can use the log-sparseness to define a class of $n^{O(1)}$ oracles such that any computation on any sequence of coin tosses can be viewed as the operation of D with some oracle from this class. It follows that the error probability is $\leq n^{O(1)} 2^{-n} = o(1)$. ■

Let BPEE denote the class of languages accepted in time $2^{2^{c+n}}$ for some constant $c \geq 0$ by a probabilistic machine with bounded error. By the same argument as in the proof of Lemma 2.4 we can show that if NEE $\not\subseteq$ BPEE then there exists a uniformly log-sparse language in NP $-$ BPP. Combining this with Lemma 3.2 we obtain

Theorem 3.3 *If* NEE $\not\subseteq$ BPEE *then there exists a language in* NP *which is harder to prove than to decide.*

These techniques also yield, under the same assumptions, a language in NP which is not in fr-IP. Thus if NEE is not contained in BPEE then IP is not contained in fr-IP, answering an open question of [BK]. Conversely, Babai, Fortnow, and Lund [BFL] have shown that fr-IP is not contained in IP, unless EXP = PSPACE.

Finally, one can place the language L in other complexity classes by making other assumptions. For example, if EESPACE $\neq$ BPEE then there exists a language in PSPACE = IP which is harder to prove than to decide. In general, a language in a class C which is harder to prove than to decide can be constructed by assuming that the "double-exponential counterpart" of C is different from BPEE.

3.2 Quadratic Residuosity: A Natural Candidate?

Clearly, it would be most interesting to exhibit an example of a *natural* problem in NP for which there are no competitive interactive proofs. A candidate example — as we indicated in §1 — is the quadratic residuosity problem. Let us consider the interactive proofs known for membership both in QR and QNR in more detail.

First for membership in QNR, the following protocol is repeated k times. The verifier tosses a coin $c \in \{0,1\}$. If $c = 1$ then the verifier sends the prover $z = xr^2 \bmod N$ for random $r \in Z_N^*$, else the verifier sends the prover $z = r^2 \bmod N$ for random $r \in Z_N^*$. The prover is then asked to the guess the value of c. If the prover guesses correctly in each repetition of the protocol, then the verifier accepts. Clearly, if $(x, N) \in QNR$ then $c = 1$ if and only if $(z, N) \notin QR$. Thus, it is sufficient for the prover to be able to tell membership in QR in order to guess c and the prover is competitive. On the other hand, if $(x, N) \notin QR$ then the probability that the verifier accepts is no greater than 2^{-k}.

How about proving that $(x, N) \in QR$? A simple proof would be the factorization of N or a $y \in Z_N^*$ such that $x = y^2 \bmod N$. In fact, all known interactive proofs of this fact require the ability to factor N. As we do not know whether factoring reduces to deciding quadratic residuosity, it remains an intriguing open problem whether membership in QR can be interactively proved by a probabilistic polynomial time prover with access to a QR oracle. We note that this can be done for special N. For example, one can show that on input N a product of t primes and $x \in Z_N^*$, it is possible to prove that x is a square mod N given only probabilistic polynomial time and an oracle for QR.

4 Program Checking

Following Blum and Kannan [BK], we say that a set A is *checkable* if there is a probabilistic, polynomial-time oracle machine C (called the *checker*) with the following properties. On input x, the first thing C does is query the oracle about membership of x. Thereafter, C's goal is to "check" whether the oracle answered this first question correctly. Let $C(O, x)$ denote the random variable computed by C with oracle O on input x. For all x, $C(A, x) =$ *correct*, with probability at least 3/4. For all B and all x such that $\chi_A(x) \neq \chi_B(x)$, $C(B, x) =$ *faulty* with probability at least 3/4. (Note that $C(B, x)$ can be anything if $B \neq A$ but $\chi_A(x) = \chi_B(x)$.) The checker's error probability can be made exponentially small using standard amplification techniques.

The definition is close in spirit to that of (competitive) interactive proofs, but there is a fundamental difference: a cheating prover is history dependent, which checking does not guard against. Indeed, sets that are complete for exponential time are checkable [BFL], and since IP = PSPACE (cf. [LFKN, Sh]) the two notions must be different if PSPACE $\neq$ EXP. Finally, [BK] point out that L is checkable if and only if both L and $\overline{L}$ are in fr-IP. From the results of §3 it follows that

Theorem 4.1 *If* NEE $\not\subseteq$ BPEE *then there exists a language in* NP *which is not checkable.*

Similarly if EESPACE $\neq$ BPEE then there exists a language in PSPACE which is not checkable.

An unconditional negative result on checking is given in the next section.

5 Incoherence and its Consequences

In this section, we continue the study of incoherent sets begun by Yao [Ya]. We establish a connection between incoherence (and all its consequences, such as uncheckability) and sparseness. We use this connection to construct the first nontrivial example of a set that is provably uncheckable, without relying on any complexity-theoretic assumptions.

5.1 Definitions and Basic Facts

An *examiner* M is an oracle Turing Machine that, on input x, is not permitted to query whether x belongs to the oracle set. A set A is *coherent* if there exists a bounded-error probabilistic polynomial-time (BPP) examiner using oracle A that recognizes A (cf.[Ya]). A is *deterministically coherent* if there exists a deterministic polynomial-time examiner using oracle A that recognizes A. A is *weakly coherent* if there exists a P/poly examiner (also called a *weak* examiner) using oracle A that recognizes A. If A is not weakly coherent then A is called *strongly incoherent.* Recall that BPP/poly = P/poly, because randomness can be incorporated into nonuniform advice using standard techniques [Ad]. Hence, there is no need to consider weak probabilistic examiners, because, for a fixed oracle, they are equivalent to weak deterministic examiners. In particular, for a fixed oracle, every BPP examiner is a P/poly examiner, and so every coherent set is weakly coherent.

Following [FF], we say that a set A is *random-self-reducible* (abbreviated rsr) if there is a probabilistic, polynomial-time oracle machine ϕ with the following properties. Let $q_i^B(x,r)$ denote the i^{th} query asked by ϕ when the oracle is B, the input is x, and the random tape holds r. Let $p(n)$ be an upper bound on the number of queries asked by ϕ when $|x| = n$.

(1) For all x, for at least 3/4 of all r, $\chi_A(x) = \phi^A(x,r)$, the result of running ϕ with oracle A, input x, and random tape r.

(2) For all x and y such that $|x| = |y|$, for all i such that $1 \leq i \leq p(|x|)$, the random variables $q_i^A(x,r)$ and $q_i^A(y,r)$ are identically distributed if r is a uniform random variable.

Note that we do not require that $q_i^B(x,r)$ and $q_i^B(y,r)$ be identically distributed if $B \neq A$. That is, condition (2) may not hold if incorrect oracle answers are given to any of queries 1 through $i-1$.

A set A is *uniformly-random self-reducible* (abbreviated ursr) if it has a random-self-reduction in which each random variable $q_i^A(x,r)$ is uniformly distributed over $\{0,1\}^{|x|}$. Uniformly-random self-reducible sets are treated at length in [FKN].

Yao [Ya] showed that if L has a checker then it is coherent, and thus reduced the problem of proving negative results on checking to that of constructing incoherent languages. Similarly, it is not hard to see the following

Theorem 5.1 *If search reduces to decision for L then L is deterministically coherent. If L has a competitive (or even function-restricted) interactive proof system then L is coherent. All uniformly-random self-reducible sets are coherent. All random-self-reducible sets are weakly coherent.*

However, coherence does not imply any of these properties. For any set A, there is a (deterministically) coherent set A' that is equivalent to A under polynomial-time, many-one reductions: Simply let $A' = \{0x : x \in A\} \cup \{1x : x \in A\}$ be the join of A with itself. Using this construction, it is straightforward to construct coherent sets that are not checkable, that are harder to prove than to decide, etc. Because coherence is a weaker property it is natural to use a stronger assumption (i.e., one about triple-exponential classes, as opposed to double) to construct incoherent sets in NP. Whether a stronger assumption is necessary remains open.

5.2 Sparse Incoherent Sets

We begin with some elementary facts. A *length-decreasing, probabilistic (resp. deterministic) self-reduction* for A is a BPP (resp. P) oracle machine N with the following properties. With oracle A, the set accepted by N is A. N, on input x, only queries the oracle about strings shorter than x.

A *tally* set is a subset of 0^*. A *superpolynomial function* $t(n) : \mathbb{N} \rightarrow \mathbb{N}$ is one for which $\lim_{n\rightarrow\infty} t(n)/n^c =$

∞, for all positive constants c. A *very fast growing function* $s(n) : \mathsf{N} \longrightarrow \mathsf{N}$ is one such that for all polynomials p, for all sufficiently large n, $s(n+1) > p(s(n))$. For example, the function

$$s(n) = 2^{2^{n^2}}$$

is very fast growing. A function s is called *well-behaved* if $s(x)$ is computable in time linear in the length of $s(x)$, range(s) is decidable in linear time, and s^{-1} is computable in linear time on range(s), where we use the standard binary string representation for integers. A *very sparse tally set* is a subset of $\{0^{s(n)} : n \in \mathsf{N}\}$, where $s(n)$ is very fast growing and well-behaved.

Proposition 5.2 *If a tally set has a length decreasing, probabilistic (resp. deterministic) polynomial time self-reduction, then it is in* BPP *(resp.* P*).*

Lemma 5.3 *If A is a very sparse tally set and A is coherent, then $A \in$* BPP*.*

Proof: It suffices to show that the hypothesis implies that A has a length-decreasing, probabilistic polynomial-time self-reduction. The conclusion then follows from Proposition 5.2.

Because A is a very sparse tally set, there is, by definition, a very fast growing, well-behaved function $s(n)$ such that $A \subseteq \{0^{s(n)} : n \in \mathsf{N}\}$. Because A is coherent, there is a BPP examiner M using oracle A that recognizes A. We use s and M to construct r, an appropriate self-reduction for A.

On input x, the reduction r first rejects x if $x \notin 0^*$ or $|x| \notin range(s)$. If $x \in 0^*$ and $|x| \in range(s)$, then r simulates M on input x. Let $\langle y_1, \ldots, y_m \rangle$ be the sequence of strings for which M, on input x, asks the oracle "is this string in A?" Because M is polynomial-time bounded and s is very fast growing, each query y_i such that $|y_i| > |x|$ is not in A, because its size falls between two elements of range(s). Because A is a tally set, each query y_i such that $|y_i| = |x|$ is not in A (by definition, the examiner M does not query the oracle about x, and all other strings of length $|x|$ are not in 0^*). Thus r probabilistically reduces membership of x in A to membership in A of a polynomial-length sequence of strings, each of which is shorter than x. ■

We first use this basic lemma to answer Yao's question (a) (see [Ya, Section 6]).

Theorem 5.4 *If* NEEEXP $\not\subseteq$ BPEEEXP *then there is an incoherent set in* NP.

Theorem 5.4 follows from Lemma 5.3 by a straightforward padding argument.

Theorems 5.4 and 5.1 can of course be used to construct a set in NP that is uncheckable, harder to prove than to decide, etc. under the assumption that NEEEXP $\not\subseteq$ BPEEEXP, but these results are subsumed by the constructions of the previous sections in which the assumption is NEE $\not\subseteq$ BPEE.

Next, we use Lemma 5.3 to construct an incoherent set just above PSPACE; this improves Yao's incoherence bound by an exponential. Oded Goldreich has informed us that an alternative proof of the following theorem was obtained independently by Hugo Krawczyk [Go].

Theorem 5.5 *There is an incoherent set in* DSPACE$(n^{\log^* n})$.

Proof: Because $n \mapsto n^{\log^* n}$ is superpolynomial and space-constructible, standard diagonalization techniques suffice to construct a very sparse tally set A such that $A \in$ DSPACE$(n^{\log^* n})$ $-$ PSPACE $\subseteq$ DSPACE$(n^{\log^* n})$ $-$ BPP. By Lemma 5.3, A is incoherent. ■

Corollary 5.6

(1) *There is a set in* DSPACE$(n^{\log^* n})$ *that is not checkable.*
(2) *There is a set in* DSPACE$(n^{\log^* n})$ *that is not uniformly-random self-reducible.*
(3) *There is a set in* DSPACE$(n^{\log^* n})$ *that is not in* fr-IP.

Corollary 5.6 gives the first nontrivial example of an uncheckable set. In his original paper on incoherence [Ya], Yao constructs an incoherent set that is just above exponential space (specifically, in DSPACE$(2^{n^{\log^* n}})$). This is a nontrivial example of incoherence, but the set is trivially uncheckable: It follows from results in [FRS] that all checkable sets are in NEXP $\cap$ co-NEXP, and hence everything above exponential space is uncheckable.

Corollary 5.6 also gives a direct improvement on Feigenbaum, Kannan, and Nisan's construction of a set in DSPACE(2^n) that is not uniformly-random self-reducible [FKN].

Finally, we use Lemma 5.3 to construct an incoherent set "just above BPP."

Theorem 5.7 *If $t(n)$ is superpolynomial and time-constructible and O is* BPP*-hard, then there is an incoherent set in* DTIME$(t(n))^O$.

Proof: Because $t(n)$ is superpolynomial and time-constructible, standard diagonalization techniques suffice to construct a very sparse tally set in

$\mathrm{DTIME}(n^{\log^* n})^O - \mathrm{P}^O$, for any oracle O. If O is BPP-hard, then this tally set must be incoherent. This follows directly from Lemma 5.3 and the fact that $\mathrm{BPP} \subseteq \mathrm{P}^O$. ■

For example, we can take O to be a set that is complete for Σ_2^p.

5.3 Strongly Incoherent Sets

Whereas in Section 5.2 we constructed a sparse, incoherent set, we will now construct an uncheckable set that is not polynomial-time reducible to any sparse set. Strong incoherence is the key.

First, we recall a combinatorial result due to Yao [Ya]. Consider the following game between two players called P_1 and P_2, with parameters N, m, and t.

(1) Player P_1 chooses a sequence of N bits, $b_1, \ldots, b_N$.

(2) Player P_2 inspects $b_1, \ldots, b_N$ and stores m bits of information, which we call her *pad p*. Player P_2 may not remember anything else about $b_1, \ldots, b_N$.

(3) Player P_1 chooses i between 1 and N and asks P_2 for the value of b_i.

(4) Player P_2 refers to her pad p and inspects b_j for t values of j other than i. Player P_2 then answers P_1's question. (Player's P_2 strategy may be adaptive, but must be deterministic.)

Player P_2 *wins* if she answers correctly. We say that player P_2's strategy is *a winning strategy for N, m, t* if she wins no matter how P_1 plays.

Fact 5.8 [Ya] *Player P_2 has a winning strategy if and only if $(t+1)m \geq N$.*

Player P_2's pad corresponds to advice in the proof below.

Theorem 5.9 *There is a strongly incoherent set in* $\mathrm{DSPACE}(n^{\log^* n})$.

Proof: Let $m(n) = t(n) = \left\lfloor n^{\frac{1}{2}\log^* n} \right\rfloor - 1$. Note that $m()$ and $t()$ dominate every polynomial on all but finitely many points. Let M_e denote the eth oracle Turing machine, running in time $t(n)$ with advice of length $m(n)$. If A is weakly coherent, then there exist e and p such that M_e, using oracle A and advice p, decides $x \in A$ correctly for all but finitely many x, without actually querying the oracle about x.

Let $N(n) = \lfloor n^{\log^* n} \rfloor$. Then $(t(n)+1)m(n) < N(n)$. Let $\langle \cdot, \cdot \rangle$ denote a pairing function on the natural numbers. We construct a strongly incoherent $A \subseteq \{0,1\}^*$ by the following initial segment argument.

Stage −1: Let $A = \emptyset$. Let $n = 3$.

Stage $\langle e, i \rangle$: Let $n = \max(n, t(n)) + 1$. Let X be the set containing the first $N(n)$ strings of length n. Find a subset A_n of X such that for every advice string p of length $m(n)$, there exists $x \in X$ such that on input x with advice p and oracle $A \cup A_n$

- M_e queries x, or
- M_e accepts x and $x \notin A_n$, or
- M_e rejects x and $x \in A_n$.

If such a set A_n exists at every stage, then clearly the construction guarantees that A differs infinitely often from every weakly coherent language, so A is strongly incoherent. But if A_n did not exist then player P_2 would have a winning strategy in Yao's game above with parameters $N(n), t(n), m(n)$. Finally we note that the set A_n can be found by exhaustive search using space $N(n) + t(n) + m(n) = O(n^{\log^* n})$, so $A \in \mathrm{DSPACE}(n^{\log^* n})$. ■

Corollary 5.10 *There is a set in* $\mathrm{DSPACE}(n^{\log^* n})$ *that is not random-self-reducible.*

In fact, the language A constructed above is in $(\mathrm{DTIME}(n^{\log^* n}))^{\Sigma_2^p}$.

5.4 Coherence and Completeness

Theorem 5.11 *Let C be a complexity class that has a polynomial-time Turing complete set that is length-decreasing, probabilistically (resp. deterministically) self-reducible. Let A be polynomial-time Turing complete for C. Then A is coherent (resp. deterministically coherent).*

Corollary 5.12 *If A is polynomial-time Turing complete for any class Σ_i^p, Π_i^p, Δ_i^p, $i \geq 0$, then A is deterministically coherent. In particular, all* NP*-complete sets are deterministically coherent.*

Note that Theorem 5.11 also implies the coherence of sets complete for PSPACE. However, this follows from the fact that those sets are checkable [Sh].

Acknowledgements

The first and third authors thank Lance Fortnow, Peter Shor, and Andy Yao for helpful discussions. The second and fourth authors thank Muli Safra, Jack Lutz, Lance Fortnow, Juris Hartmanis, Eric Allender, Satish Thate, and Oded Goldreich for helpful discussions, and Silvio Micali for suggesting the term "competitive proof systems."

References

[Ad] L. Adleman. Two Theorems on Random Polynomial Time, FOCS 78.

[Al] E. Allender. Limitations of the Upward Separation Technique, *Mathematical Systems Theory* **24**, 53–67 (1991).

[BaF] L. Babai and L. Fortnow. A Characterization of #P by Arithmetic Straight-Line Programs, FOCS 90.

[BFL] L. Babai, L. Fortnow and C. Lund. Nondeterministic Exponential Time has Two-Prover Interactive Protocols, FOCS 90.

[BF] R. Beigel and J. Feigenbaum. Improved Bounds on Coherence and Checkability, YALEU/DCS/TR-819 (September 11, 1990).

[BG] M. Bellare and S. Goldwasser. The Complexity of Decision versus Search, MIT-LCS Technical Memo. TM-444 (April 1991).

[BGKW] M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multiprover Interactive Proof Systems: How to Remove Intractability Assumptions, STOC 88.

[Be] L. Berman. Relationship between Density and Deterministic Complexity of NP-complete Languages, *Proceedings of ICALP 78,* Lecture Notes in Compter Science 62, Springer-Verlag (1978).

[BH] L. Berman and J. Hartmanis. On Isomorphisms and Density of NP and other Complete Sets, *SIAM J. Computing* **6**, 305-322 (1977).

[BK] M. Blum and S. Kannan. Designing Programs that Check their Work, STOC 89.

[FF] J. Feigenbaum and L. Fortnow. On the Random-Self-Reducibility of Complete Sets, *Proceedings of the 6th Structures*, IEEE (1991).

[FKN] J. Feigenbaum, S. Kannan and N. Nisan. Lower Bounds on Random-Self-Reducibility, *Proceedings of the 5th Structures*, IEEE (1990).

[FRS] L. Fortnow, J. Rompel and M. Sipser. On the Power of Multiprover Interactive Protocols, *Proceedings of the 3rd Structures*, IEEE (1988).

[Fo] S. Fortune. A Note on Sparse Complete Sets, *SIAM J. Computing* **8**, 431-433 (1979).

[GJ] M. Garey and D. Johnson. Computers and Intractability, W.H. Freeman and Co., New York (1979).

[Go] O. Goldreich. Private communication (July 1991).

[GMW] O. Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity, FOCS 86.

[GMR] S. Goldwasser, S. Micali and C. Rackoff. The Knowledge Complexity of Interactive Proofs, *SIAM J. Computing* **18**(1), 186–208 (1989).

[HSI] J. Hartmanis, V. Sewelson and N. Immerman. Sparse Sets in NP-P: EXPTIME versus NEXPTIME, STOC 83.

[IN] R. Impagliazzo and M. Naor. Decision Trees and Downward Closures, *Proceedings of the 3rd Structures*, IEEE (1988).

[IS] R. Impagliazzo and M. Sudan. Private communication (May 1991).

[IT] R. Impagliazzo and G. Tardos. Decision Versus Search Problems in Super-Polynomial Time, FOCS 89.

[KUW] R. Karp, E. Upfal and A. Wigderson. The Complexity of Parallel Search, *J. Computer and System Sciences* **36**, 225–253 (1988).

[La] R. Ladner. On the Structure of Polynomial Time Reducibility, *J. Assoc. Comput. Mach.* **22**, 155-171 (1975).

[LFKN] C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic Methods for Interactive Proof Systems, FOCS 90.

[Ma] S. Mahaney. Sparse Complete Sets for NP: Solution of a Conjecture of Berman and Hartmanis, FOCS 80.

[Sh] A. Shamir. IP=PSPACE, FOCS 90.

[Sp] D. Spielman. Private communication (June 1991).

[Ya] A. Yao. Coherent Functions and Program Checkers, STOC 90.