Yale University
Department of Computer Science

Improved Bounds on Coherence and Checkability

Richard Beigel
Yale University

Joan Feigenbaum
AT&T Bell Laboratories

YALEU/DCS/TR-819
September 11, 1990

Improved Bounds on Coherence and Checkability

Richard Beigel[*] Joan Feigenbaum[†]

Abstract

Sets for which membership at one point can be deduced efficiently from membership at other points are fundamental in many areas of theoretical computer science. Yao [20] recently defined *coherence* in order to capture this property. He noted that incoherent sets are not checkable in the sense of Blum and Kannan [6]. They are also not uniformly-random self-reducible in the sense of, e.g., [1, 3, 11, 19]. The goal of this work is to obtain bounds on the complexity of sets with these three important properties.

Yao [20] took a step toward this goal by showing that there is an incoherent set in DSPACE($2^{n^{\log^* n}}$). In this paper, we improve Yao's result as follows, thus obtaining the best negative result on checking to date.

Theorem: There is a set in DSPACE($n^{\log^* n}$) that is incoherent (and hence neither checkable nor uniformly-random self-reducible).

Since all sets in BPP are coherent, our space bound cannot be improved without separating BPP from PSPACE. In fact, our construction yields incoherent sets that are "just above BPP," in a sense that we will specify. We define a *strong* notion of incoherence as well and construct sets in DSPACE($n^{\log^* n}$) that are strongly incoherent.

The open questions in [20] include whether there is a "natural" complexity-theoretic assumption that implies the existence of incoherent sets in the polynomial hierarchy.

We provide the following answer to Yao's question.

Theorem: If NEEEXPTIME $\not\subseteq$ BPEEEXPTIME, then there is a set in NP that is incoherent (and hence neither checkable nor uniformly-random self-reducible).

Note that this is the first condition known to imply the existence of uncheckable sets in PSPACE; thus it partially answers an open question of Blum and Kannan [6].

Theorem: If S is complete for any of the classes Σ_i^p, Π_i^p, or Δ_i^p, $i \geq 0$, then S is coherent. In particular, all NP-complete sets are coherent.

Note that it is unknown whether NP-complete sets are checkable.

[*]Dept. of Computer Science, P.O. Box 2158, Yale Station, New Haven, CT 06520-2158 USA, beigel-richard@cs.yale.edu. Supported by in part by NSF grants CCR-8808948 and CCR-8958528.

[†]AT&T Bell Laboratories, Room 2C473, Murray Hill, NJ 07974 USA, jf@research.att.com.

1 Introduction

Suppose that A is a set and that x is a string for which we would like to test membership in A. Is there a polynomial-time algorithm that, given x and an oracle for A, can determine whether x is in A without asking the oracle "is x in A?"? That is, can membership of x in A be computed efficiently given knowledge of whether certain other strings are in A? If so, then A is said to be a *coherent* set. Examples of coherent sets include SAT, QBF, and QRES (the set of quadratic-residues modulo composites). The goal of this work is to study the properties of coherent sets and their relationship to other central notions in complexity.

Coherence was defined recently by Yao [20]. The definition in [20] is motivated by previous work on uniformly-random self-reducibility (cf. [1, 3, 11, 19]) and on efficient program-checking (cf. Blum and Kannan [6]). Uniformly-random self-reducible sets are interesting because their membership problems are as hard on average as they are in the worst case; this property is used extensively in arguments for the security of cryptographic protocols. From a complexity-theoretic point of view, checkability provides an interesting new perspective on the (presumed) difference between *computing* a value and *verifying* the result of a computation. Traditionally, this issue is represented by the (presumed) gap between P and NP$\cap$coNP. If one allows randomness and interaction to be used in the verification procedure, then the issue is represented by the (presumed) gap between BPP and "checkable sets" — the latter gap is more dramatic, because EXPTIME-complete sets are checkable (cf. [4]).

Unfortunately, neither the uniformly-random self-reducible sets nor the checkable sets have been fully characterized. Coherence captures a simple, essential property of these sets and thus provides a method for progress toward such a characterization.

Yao [20] took a step toward this characterization by showing that there is a set in DSPACE($2^{n^{\log^* n}}$) that is incoherent.[1] In this paper, we improve Yao's result by showing that there are incoherent sets "just above PSPACE;" this provides the best negative result on checking to date.

Theorem: There is a set in DSPACE($n^{\log^* n}$) that is incoherent (and hence neither checkable nor uniformly-random self-reducible).

Since all sets in BPP are coherent, our space bound cannot be improved without separating BPP from PSPACE. Our technique can also be used to show that there are incoherent sets "just above BPP" in the following sense: If $t(n)$ is superpolynomial and time-constructible, and B is BPP-hard, then there is an incoherent set in DTIME$(t(n))^B$. A different technique constructs sets in DSPACE($n^{\log^* n}$) that are strongly incoherent (defined in Section 2).

Several open questions about incoherence are stated in [20], including whether there is a "natural" complexity-theoretic assumption that implies the existence of incoherent sets in the polynomial hierarchy.

Let NEEEXPTIME denote the union, over all polynomials p, of NTIME($2^{2^{2^{p(n)}}}$), and define BPEEEXPTIME similarly. We provide the following answer to Yao's question.

[1] Both in Yao's result and in ours, the exponent $\log^* n$ can be replaced by any reasonable function of n that approaches ∞ with n.

Theorem: If NEEEXPTIME $\not\subseteq$ BPEEEXPTIME, then there is a set in NP that is incoherent (and hence neither checkable nor uniformly-random self-reducible).

Note that this is the first condition known to imply the existence of sets in PSPACE that do not have program checkers. Blum and Kannan [6] defined the class function-restricted IP (frIP) and showed that frIP $\cap$ co-frIP contains exactly the sets with program checkers. They posed the question of whether frIP is the same as IP. Our theorem implies that IP is not contained in frIP $\cap$ co-frIP if NEEEXPTIME $\not\subseteq$ BPEEEXPTIME. Conversely, frIP $\not\subseteq$ IP if PSPACE $\neq$ EXPTIME (cf. [4, 16, 18]).

Of course, characterization of the coherent sets requires positive results as well. We prove the following theorem toward that end.

Theorem: If S is complete for any of the classes Σ_i^p, Π_i^p, or Δ_i^p, $i \geq 0$, then S is coherent. In particular, all NP-complete sets are coherent.

Note that it remains open whether NP-complete sets (and, more generally, the complete sets at all levels of the polynomial hierarchy) have program checkers. It *is* known that PSPACE-complete sets and EXPTIME-complete sets are checkable and thus coherent (cf. [4, 16, 18]).

In Section 2 below, we specify the definitions and notation that we will use, and we recall some necessary results from the literature. Section 3 contains precise statements and proofs of our results. Open problems can be found in Section 4.

Most of the results in this paper first appeared in our Technical Memorandum [5].

2 Preliminaries

Throughout this paper, we consider languages that are contained in $\{0,1\}^*$. Our notation for most complexity classes is that of Hopcroft and Ullman [14]. As stated in Section 1, we use NEEEXPTIME to denote the union, over all polynomials p, of NTIME($2^{2^{2^{p(n)}}}$). BPEEEXPTIME denotes the corresponding bounded-probabilistic time class. The following definition of coherence is equivalent to Yao's.

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. 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 (cf. [2]). Similarly, weak probabilistic polynomial-time examiners are equivalent to weak deterministic polynomial-time examiners; in particular every BPP examiner is a P/poly examiner, and so every coherent set is weakly coherent.

Next we recall the definition of *checkability* given by Blum and Kannan [6]. A set A is *checkable* if there is a probabilistic, polynomial-time oracle machine C (called the *checker*) with the following properties. Let $C(O,x)$ denote the random variable computed by C with oracle O on input x. For all x, $C(A,x) = \textit{correct}$, with probability at least 3/4. For all B and all x such that $\chi_A(x) \neq \chi_B(x)$, $C(B,x) = \textit{faulty}$ with probability at least 3/4. (Note that $C(B,x)$ can be anything if $B \neq A$ but $\chi_A(x) = \chi_B(x)$.) If the oracle machine C is deterministic, then checking is equivalent to *self-helping* (cf. Schöning [17] and Ko [15]).

Finally, we repeat the definition of uniformly-random self-reducibility given by Feigenbaum, Kannan, and Nisan [11].[2] Earlier definitions appear in, for example, [1, 3, 19]. The close connection between random-self-reducibility and program checking has been demonstrated by Blum, Luby, and Rubinfeld [7, 8].

A set A is *$k(n)$-uniformly-random self-reducible* (abbreviated k-ursr) if, for all n, there are polynomial-time computable functions ϕ, $\sigma_1, \ldots, \sigma_{k(n)}$ with the following properties (here r is an element of $\{0,1\}^m$, where m is bounded by a polynomial in n):

- For all x
$$\chi_A(x) = \phi(x, r, \chi_A(\sigma_1(x,r)), \ldots, \chi_A(\sigma_{k(n)}(x,r))),$$
for at least 3/4 of all $r \in \{0,1\}^m$.

- For all n and all $x \in \{0,1\}^n$, if r is chosen uniformly from $\{0,1\}^m$, then $\sigma_i(x,r)$ is uniform over $\{0,1\}^n$, for all $1 \leq i \leq k(n)$. (In general, $\sigma_i(x,r)$ and $\sigma_j(x,r)$, for $i \neq j$, are *dependent.*)

The class poly-URSR consists of all sets that are k-ursr for any polynomial k. Intuitively, A is uniformly-random self-reducible if membership of one instance can be deduced efficiently from membership at correlated, uniformly random instances of the same length.

A *tally* set is a subset of 0^*. A *superpolynomial function* $t(n) : \mathsf{N} \to \mathsf{N}$ is one for which $n^c/t(n)$ approaches 0 as n grows, for all positive constants c. A *very fast growing function* $s(n) : \mathsf{N} \to \mathsf{N}$ is one for which $s(n+1) > p(s(n))$, for all polynomials p and all sufficiently large 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.

[2]In [11], these reductions are simply called "random-self-reductions." We use the qualification "uniformly-random" for the following reason. In the reductions studied in [11] and here, each random instance $\sigma_i(x,r)$ is *uniformly* distributed over $\{0,1\}^n$. Feigenbaum and Fortnow [10] have studied a more general notion of random-self-reducibility in which this is not the case; the unqualified term "random-self-reducible" is more appropriately used for the more general class of sets studied in [10]. Coherence is ***not*** implied by random-self-reducibility as defined in [10], but it is implied by uniformly-random self-reducibility.

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. On input x, N only queries the oracle about strings of length strictly less than $|x|$.

Fact 2.1 *If a tally set has a length-decreasing, probabilistic polynomial-time self-reduction, then it is in* BPP.

By "standard padding techniques," we mean, for example, those used by Book [9] in his study of tally sets. By "standard diagonalization techniques," we mean, for example, those used to prove the classical space- and time-hierarchy theorems (cf. [14, Chapter 12]).

Finally, note the following elementary relationships among the notions of coherence, uniformly-random self-reducibility, and checkability.

Fact 2.2 *All sets in* poly-URSR *are coherent.*

Proof: Let A be k-ursr. Using standard amplification techniques, we can reduce to 2^{-n} the probability that the function ϕ outputs a wrong answer. An examiner for A can simply choose r, query the A-oracle to obtain $a_1 = \chi_A(\sigma_1(x,r)), \ldots, a_k = \chi_A(\sigma_k(x,r))$, and output $\phi(x, r, a_1, \ldots, a_k)$. This examiner only fails if ϕ fails or at least one of the $\sigma_i(x,r)$'s is equal to x. By definition of k-ursr, this probability is at most $(k(n)+1)/2^n$, which suffices to show that A is coherent. ▮

Fact 2.3 *There are coherent sets that are not in* poly-URSR.

Proof: Standard diagonalization suffices to construct a tally set A that is not in PSPACE and, *a fortiori*, not in BPP. Then

$$A \oplus A \equiv \{0x : x \in A\} \cup \{1x : x \in A\}$$

is deterministically coherent. However, if $A \oplus A$ were in poly-URSR, then A would be poly-URSR, and hence in BPP. ▮

Fact 2.4 (Yao [20]) *All checkable sets are coherent.*

Fact 2.5 *There are coherent sets that are not checkable.*

Proof: Let A be a set that is not recursively enumerable. As in Fact 2.3, $A \oplus A$ is coherent and non-r.e. The result of Fortnow, Rompel, and Sipser [12] that all checkable sets are in NEXPTIME implies that $A \oplus A$ is not checkable. ▮

Fact 2.6 *There are sets in* poly-URSR *that are not checkable.*

Proof: There are 1-ursr sets that are non-r.e. (refer to [1] for details). These sets are not checkable, by [12]. ▮

Thus the three notions, although related, are certainly not equivalent. Furthermore, the NP-complete sets seem to distinguish them:

- All NP-complete sets are coherent.
- If any NP-complete set is uniformly-random self-reducible, then the polynomial hierarchy collapses at the third level.
- It is unknown whether NP-complete sets are checkable.

The first of these statements is proved in the following section. The second is a special case of [10, Theorem 3.1].

3 Results

In this section we construct an incoherent set in DSPACE($n^{\log^* n}$). A similar argument shows that NP contains an incoherent set unless NEEEXPTIME $\subseteq$ BPEEEXPTIME. We also show that a set that is complete for any level of the polynomial hierarchy is coherent. Finally, building on a result of Yao [20], we construct a strongly incoherent set in DSPACE($n^{\log^* n}$). Unfortunately, the techniques we use for strongly incoherent sets do not seem to have any bearing on NP.

3.1 Incoherent Tally Sets of Low Complexity

Lemma 3.1 *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 Fact 2.1.

Because A is a very sparse tally set, there is, by definition, a very fast growing, well-behaved function $s(n)$ such that $A \subseteq \{0^{s(n)} : n \in \mathbb{N}\}$. 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 $|x| \notin \mathit{Range}(s)$. If $x \in 0^*$ and $|x| \in \mathit{Range}(s)$, 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 $\mathit{Range}(s)$. 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 can now give our improvement of Yao's construction.

Theorem 3.1 *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 techniques suffice to construct a very sparse tally set A such that

$$A \in \mathrm{DSPACE}(n^{\log^* n}) - \mathrm{PSPACE} \subseteq \mathrm{DSPACE}(n^{\log^* n}) - \mathrm{BPP}.$$

By Lemma 3.1, A is incoherent. ∎

A stronger result about incoherent sets just above PSPACE is given in Section 3.3 below.

Corollary 3.1 *There is a set in* DSPACE($n^{\log^* n}$) *that is not checkable.*

Corollary 3.2 *There is a set in* DSPACE($n^{\log^* n}$) *that is not in* poly-URSR.

Corollary 3.2 is a direct improvement of Feigenbaum, Kannan, and Nisan's construction in [11] of a set in DSPACE(2^n) that is not in poly-URSR.

Lemma 3.1 can also be used to show that there are incoherent tally sets "just above BPP."

Theorem 3.2 *If $t(n)$ is superpolynomial and time-constructible, and B is* BPP*-hard, then there is an incoherent set in* DTIME$(t(n))^B$.

Proof: Because $t(n)$ is superpolynomial and time-constructible, standard diagonalization techniques suffice to construct a very sparse tally set in DTIME$(t(n))^B$ – P^B, for any oracle B. If B is BPP-hard, then this tally set must be incoherent. This follows directly from Lemma 3.1 and the fact that BPP $\subseteq$ P^B. ∎

Corollary 3.3 *There is an incoherent set in* DTIME$(n^{\log^* n})^{\Sigma_2^p}$.

Proof: The function $n \mapsto n^{\log^* n}$ is superpolynomial and time-constructible, and complete sets for Σ_2^p are BPP-hard. ∎

Corollary 3.4 *There is a set in* DTIME$(n^{\log^* n})^{\Sigma_2^p}$ *that is not checkable.*

Corollary 3.5 *There is a set in* DTIME$(n^{\log^* n})^{\Sigma_2^p}$ *that is not in* poly-URSR.

Finally, we use Lemma 3.1 to derive a "structural hypothesis" that implies the existence of incoherent tally sets in NP. This answers Yao's question (a) (see [20, Section 6]).

Theorem 3.3 *If* NEEEXPTIME $\not\subseteq$ BPEEEXPTIME, *then there is an incoherent set in* NP.

Proof: By Lemma 3.1, it suffices to show that the hypothesis implies that there is a very sparse tally set in NP − BPP. Standard padding techniques suffice. We give the construction for those unfamiliar with Book [9].

First note that the hypothesis implies that there is a set in NTIME($2^{2^{2^{n^2}}}$) that is not in BPEEEXPTIME. To see this, take a set S in NTIME($2^{2^{2^{n^c}}}$) − BPEEEXPTIME and let $S' = \{x10^{|x|^{c/2}} : x \in S\}$.

Throughout this proof, we use the symbol $1x$, where $x \in \{0,1\}^*$, to denote both a string and an integer written in binary.

Let A be a set in NTIME($2^{2^{2^{n^2}}}$) − BPEEEXPTIME, and consider the tally set

$$A' \equiv \{0^{1x} : x \in A\}.$$

Because the time needed to decide whether 0^{1x} is in A' is exactly the time needed to decide whether x is in A, and because $|x| \leq \log_2(|0^{1x}|)$, we have that

$$A' \in \text{NTIME}(2^{2^{2^{(\log n)^2}}}).$$

Now let

$$s(1x) = 1x10^{2^{2^{2^{|x|^2}}}}$$

and

$$A'' \equiv \{0^{s(1x)} : 0^{1x} \in A'\}.$$

Note that s is very fast growing and well-behaved, and hence A'' is a very sparse tally set. Furthermore,

$$A'' \in \text{NTIME}(n) \subset \text{NP}.$$

So it remains to show that A'' is not in BPP.

If A'' were in BPTIME(n^c), then A' would be in

$$\text{BPTIME}((n+1+2^{2^{2^{(\log n)^2}}})^c) \subseteq \text{BPTIME}(2^{2^{2^{(\log n)^2}+c\log n}}).$$

This would imply that A was in

$$\text{BPTIME}(2^{2^{2^{(n^2+cn)}}}) \subseteq \text{BPEEEXPTIME},$$

thus contradicting the hypothesis. ∎

Corollary 3.6 *If* NEEEXPTIME $\not\subseteq$ BPEEEXPTIME, *then there is a set in* NP *that is not checkable.*

As discussed in Section 1, our next corollary provides a partial answer to an open question of Blum and Kannan [6].

Corollary 3.7 *If* NEEEXPTIME $\not\subseteq$ BPEEEXPTIME, *then* IP $\not\subseteq$ frIP $\cap$ co-frIP.

Corollary 3.8 *If* NEEEXPTIME $\not\subseteq$ BPEEEXPTIME, *then there is a set in* NP *that is not in* poly-URSR.

3.2 NP-Complete Sets are Coherent

Theorem 3.4 *If A is* NP*-complete, then A is deterministically coherent.*

Proof: We use $cook_A(x)$ to denote Cook's reduction from A to SAT (cf. [13, Chapter 2]); any polynomial-time many-one reduction would work as well. We use the notation y_0 (resp. y_1) to denote the boolean formula obtained by replacing the first variable in boolean formula y with FALSE (resp. TRUE).

Let x be an element of $\{0,1\}^*$, and let $y = cook_A(x)$. Thus

$$x \in A \iff (y_0 \in \text{SAT} \lor y_1 \in \text{SAT}).$$

Because A is NP-complete, there is a polynomial-time computable function r such that, for all $z \in \{0,1\}^*$,

$$z \in \text{SAT} \iff r(z) \in A.$$

Figure 1 contains a deterministic examiner M that accepts the set A when it uses oracle A. The notation $A(w)$ is used to denote the answer to the oracle query "is w in A?"

The recursive depth of M, on input (A, x), is at most $|cook_A(x)|$, because each call to test-SAT passes a formula with one fewer variable than that passed in the previous call. The recursive width is 1. Thus M is a deterministic, polynomial-time oracle TM that computes membership in A with oracle A and never queries the oracle about membership of the original input string. ▮

A related but more complicated technique yields the following.

Theorem 3.5 *Let $\mathcal{C}$ 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 $\mathcal{C}$. Then A is coherent (resp. deterministically coherent).*

Proof: Let K be a polynomial-time Turing complete set for $\mathcal{C}$, and let r_{KK} be a length-decreasing self-reduction for K. Let r_{AK} (resp. r_{KA}) be a polynomial-time Turing reduction from A to K (resp. from K to A). An examiner for A proceeds as follows.

M starts by simulating r_{AK} on input x. Whenever it reaches a point at which it must know whether, say, y is in K, it starts to simulate r_{KA} on input y. Recall that M has access to an A-oracle. Thus, if r_{KA} can decide whether y is in K without asking whether x is in

```
M(A, x)
{
    If $x = r(\text{TRUE})$, then return(TRUE);
    If $x = r(\text{FALSE})$, then return(FALSE);
    $y \leftarrow cook_A(x)$;
    If $r(y) \neq x$, then return($A(r(y))$)
        Else return(test-SAT($y, x$));
}

test-SAT($y, x$) /* apply self-reduction to $y$ */
{
    If $y =$ TRUE, then return(TRUE);
    If $y =$ FALSE, then return(FALSE);
    If $r(y_0) = x$, then return(test-SAT($y_0, x$))
        Else $b_0 \leftarrow A(r(y_0))$;
    If $r(y_1) = x$, then return(test-SAT($y_1, x$))
        Else $b_1 \leftarrow A(r(y_1))$;
    Return($b_0 \vee b_1$);
}
```

Figure 1: A deterministic examiner M that accepts A with oracle set A.

A, then the examiner M has made progress and can proceed with the simulation of r_{AK} on input x. Otherwise, M can try both possible values for $\chi_A(x)$. If the value obtained for $\chi_K(y)$ is the same in both cases, then M knows that this answer for $\chi_K(y)$ is correct and once again proceeds with the simulation of r_{AK} on input x.

The only other possibilities are $x \in A \leftrightarrow y \in K$ or $x \in A \leftrightarrow y \notin K$, and thus it suffices for M to determine $\chi_K(y)$. In this case, M aborts the simulation of r_{AK}. Instead, it applies r_{KK} to y. Once again, M has made progress, because all subsequent values $\chi_K(y')$ that it tries to determine satisfy $|y'| < |y|$. ▮

Corollary 3.9 *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.*

Note that Theorem 3.5 also implies the coherence of PSPACE-complete sets. However, as explained in Section 1, this already follows from the fact that those sets are checkable (cf. [18]).

3.3 Strongly Incoherent Sets

The incoherent set given by Theorem 3.1 is a tally set; thus it is in P/*poly* and is weakly coherent. In this section, we give an alternative construction of an incoherent set just above PSPACE that does not have this drawback.

First, we recall a combinatorial result due to Yao [20]. 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 3.1 (Yao [20]) *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 3.6 *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 e^{th} oracle Turing machine, running in time $t(n)$, and let $p(n)$ be an advice string 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 $N(n) = \left\lfloor n^{\log^* n} \right\rfloor$. 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 $A \in \mathrm{DSPACE}(n^{\log^* n})$. ∎

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

4 Open Problems

Open questions include:

(1) Are NP-Complete sets checkable?

(2) Is there a natural structural hypothesis that implies that there are strongly incoherent sets in the polynomial hierarchy?

Refer to Feigenbaum and Fortnow [10] for open questions about random-self-reducibility.

5 Acknowledgements

We are very grateful to Andy Yao for his suggestion that we look at coherence and the polynomial hierarchy. We also thank Eric Allender, Debby Joseph, Jeremy Kahn, Yishay Mansour, Dana Randall, and Peter Shor for helpful discussions.

References

[1] M. Abadi, J. Feigenbaum, and J. Kilian. On Hiding Information from an Oracle, *J. Comput. System Sci.* 39 (1989), 21–50.

[2] L. Adleman. Two Theorems on Random Polynomial Time, *Proc. of the 19th FOCS* (1978), IEEE, 75–83.

[3] D. Angluin and D. Lichtenstein. Provable Security of Cryptosystems: A Survey, YALEU/DCS/TR-288, 1983.

[4] L. Babai, L. Fortnow, and C. Lund. Nondeterministic Exponential Time has Two-Prover Interactive Protocols, *Proc. of the 31st FOCS* (1990), IEEE.

[5] R. Beigel and J. Feigenbaum. On the Complexity of Coherent Sets, AT&T Bell Laboratories Technical Memorandum, February 19, 1990.

[6] M. Blum and S. Kannan. Designing Programs that Check Their Work, *Proc. of the 21st STOC* (1989), ACM, 86–97.

[7] M. Blum, M. Luby, and R. Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems, *Proc. of the 22nd STOC* (1990), ACM, 73–83.

[8] M. Blum, M. Luby, and R. Rubinfeld. Program Result Checking Against Adaptive Programs and in Cryptographic Settings, *Proc. of the DIMACS Workshop on Distributed Computing and Cryptography* (1989), AMS.

[9] R. Book. Tally Languages and Complexity Classes, *Inf. and Control* 26 (1974), 186–193.

[10] J. Feigenbaum and L. Fortnow. On the Random-Self-Reducibility of Complete Sets, University of Chicago Technical Report 90-22, Computer Science Department, August 20, 1990.

[11] J. Feigenbaum, S. Kannan, and N. Nisan. Lower Bounds on Random-Self-Reducibility, *Proc. of the 5th Structures* (1990), IEEE, 100–109.

[12] L. Fortnow, J. Rompel, and M. Sipser. On the Power of Multiprover Interactive Protocols, *Proc. of the 3rd Structures* (1988), IEEE, 156–161.

[13] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[14] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, 1979.

[15] K. Ko. On Helping by Robust Oracle Machines, *Theor. Comput. Sci.* 52 (1987), 15–36.

[16] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic Methods for Interactive Proof Systems, *Proc. of the 31st FOCS* (1990), IEEE.

[17] U. Schöning. Robust Algorithms: A Different Approach to Oracles, *Theor. Comput. Sci.* 40 (1985), 57–66.

[18] A. Shamir. IP = PSPACE, *Proc. of the 31st FOCS* (1990), IEEE.

[19] M. Tompa and H. Woll. Random Self-Reducibility and Zero-Knowledge Interactive Proofs of Possession of Information, *Proc. of the 28th FOCS* (1987), IEEE, 472–482.

[20] A. C. Yao. Coherent Functions and Program Checkers, *Proc. of the 22nd STOC* (1990), ACM, 84–94.