P-Selective Sets, and Reducing Search to Decision vs. Self-Reducibility *

Edith Hemaspaandra†
State University of New York at Buffalo

Ashish V. Naik‡
State University of New York at Buffalo

Mitsunori Ogiwara§
University of Electro-Communications

Alan L. Selman¶
State University of New York at Buffalo

July 29, 1993

Abstract

We obtain several results that distinguish self-reducibility of a language L with the question of whether search reduces to decision for L. These include:

(i) If $\mathrm{NE} \neq \mathrm{E}$, then there exists a set L in $\mathrm{NP} - \mathrm{P}$ such that search reduces to decision for L, search does *not* nonadaptively reduces to decision for L, and L is *not* self-reducible.

*Funding for this research was provided by the National Science Foundation under grant CCR-9002292.

†Department of Computer Science, State University of New York at Buffalo, 226 Bell Hall, Buffalo, NY 14260

‡Department of Computer Science, State University of New York at Buffalo, 226 Bell Hall, Buffalo, NY 14260

§Research performed while visiting the Department of Computer Science, State University of New York at Buffalo, Jan. 1992–Dec. 1992. Current address: Department of Computer Science, University of Electro-Communications, Chofu-shi, Tokyo 182, Japan.

¶Department of Computer Science, State University of New York at Buffalo, 226 Bell Hall, Buffalo, NY 14260

(ii) If UE $\neq$ E, then there exists a language $L \in$ UP $-$ P such that search nonadaptively reduces to decision for L, but L is *not* self-reducible.

(iii) If UE $\cap$ co-UE $\neq$ E, then there is a disjunctive self-reducible language $L \in$ UP $-$ P for which search does *not* nonadaptively reduce to decision.

(iv) There is an oracle relative to which satisfiable assignments cannot be computed nonadaptively relative to SAT.

We prove that if NE $\not\subseteq$ BPE, then there is a language $L \in$ NP$-$BPP such that L is randomly self-reducible, *not* nonadaptively randomly self-reducible, and *not* self-reducible. We obtain results concerning tradeoffs in multiprover interactive proof systems, and obtain results that distinguish checkable languages from those that are nonadaptively checkable.

Many of our results depend on new techniques for constructing p-selective sets. We obtain a p-selective set that is *not* $\leq_{tt}^{\mathrm{P}}$-equivalent to any tally language, and we show that if P = PP, then every p-selective set is $\leq_{\mathrm{T}}^{\mathrm{P}}$-equivalent to a tally language. Similarly, if P = NP, then every cheatable set is $\leq_{m}^{\mathrm{P}}$-equivalent to a tally language. We construct a recursive p-selective tally set that is *not* cheatable.

Keywords: computational complexity, complexity classes, p-selective sets, self-reducible, search reduces to decision, proof systems, cheatable, left cuts

ACM Computing Reviews Subject Categories: F.1.2, F.1.3

1 Introduction

One of the early fundamental observations about SAT and other NP-complete sets is that functional versions of these problems are Turing reducible in polynomial time to their corresponding decision problems. It has long been known that the problem of finding a satisfying assignment for a formula of propositional logic is Turing reducible in polynomial time to SAT, the set of all satisfiable formulas. Loosely speaking, a *decision* problem is the problem of determining membership in a language L, and, for a language $L \in$ NP, a *search* problem is the problem of computing witnesses for membership in L. We say that *search reduces to decision* for a language $L \in$ NP if the search problem for L is Turing reducible in polynomial time to L.

[BBFG91] provides evidence that search does not reduce to decision for all of NP. It has long been felt that the structural property that causes search to be reducible to decision is disjunctive self-reducibility. It follows from [Sel88] that search reduces to decision for all disjunctive self-reducible sets and for all NP-complete sets (even though not all NP-complete sets are known to be disjunctive self-reducible). Again loosely speaking, a set is *self-reducible* if the membership question for any element can be reduced in polynomial time to the membership question for a number of shorter elements, and a set L is *disjunctive* self-reducible if there is a polynomial time computable transducer that for any element x computes a number of smaller elements $x_1, \ldots, x_n$ such that $x \in L$ if and only if at least one of the smaller elements belongs to L. Again, the classic example is SAT: SAT is disjunctive self-reducible because a Boolean formula $\phi(x_1, \ldots, x_n)$ is satisfiable if and only if at least one of the shorter formulas $\phi(0, x_2, \ldots, x_n)$ or $\phi(1, x_2, \ldots, x_n)$ is satisfiable. Disjunctive self-reducibility is the structural property that forms the basis of the work by P. Berman [Ber78], Fortune [For79], Mahaney [Mah82], and others, concerning self-reducibility and reductions to sparse sets. An obvious and well-known algorithm with SAT as an oracle uses the disjunctive self-reducibility of SAT in order to compute a satisfying assignment for any satisfiable formula.

Under a reasonable assumption about exponential time (NE $\neq$ E), we construct a language in NP for which search reduces to decision, but which is *not* self-reducible. Balcázar [Bal90] has demonstrated that search reduces to decision for a set L if and only if L is a self-1-helper, a notion defined by Ko [Ko87]. As a consequence, our result solves the principal open question in [Ko87]. Ko's theorem states that if a set L is disjunctive self-reducible, then it is a self-1-helper. As Joseph and Young write in [JY90], "One of the major open problems left by Ko's work is: What is the relationship between self-helping and self-reducibility? For instance, is the converse of the above theorem true?" Our construction shows that the converse fails under the above assumption.

We compare the properties we have been considering with the possibility of there existing a *nonadaptive* procedure for a set $L \in \mathrm{NP}$ which reduces search to decision. Under reasonable assumptions about unambiguous exponential time, we construct languages in NP which are *not* self-reducible, but for which search nonadaptively reduces to decision, and we construct languages which are disjunctive self-reducible (hence, for which search reduces to decision), but for which search does *not* nonadaptively reduce to decision. These results provide answers for the remaining open questions about self-1-helpers in [Ko87]. Also, we construct an oracle relative to which assignments for satisfying formulas cannot be computed nonadaptively from SAT.

We apply our techniques to similar issues concerning random self-reducibility [FF91, AFK89, FFLS92]. Informally, a set is randomly self-reducible if a probabilistic Turing machine can determine membership in the set by determining membership at other "random" instances. In [FFLS92] under the assumption that $\mathrm{NEEE} \not\subseteq \mathrm{BPEEE}$, the authors construct a language that is randomly self-reducible, but that is *not* randomly self-reducible by any nonadaptive procedure. We use our techniques to obtain their conclusion, and simultaneously a language that is not self-reducible, with the weaker hypothesis that $\mathrm{NE} \not\subseteq \mathrm{BPE}$.

We prove that if $\mathrm{UE} \cap \text{co-UE} \not\subseteq \mathrm{BPE}$, then there exists a language $L \in \mathrm{NP}$ which is checkable [BK89] but which is *not* nonadaptively checkable.

Lastly, we consider the natural extension of "search reduces to decision for L " to interactive proof systems. We call a prover *natural* if it is an oracle for L, and we define a *natural* proof system to be one in which all honest provers are natural. (Natural proof systems are similar to competitive proof systems defined in [BBFG91, BG].) If a language L has a natural interactive proof system how much interaction is required? We show under assumption $\mathrm{NE} \not\subseteq \mathrm{BPE}$ that there is a language $L \in \mathrm{NP} - \mathrm{BPP}$ such that L has a natural proof system but L does not have any 1 round natural proof system even if we permit a polynomial number of provers. Thus, there is no trade-off between number of rounds and number of provers.

Let us say a few words here about our techniques, for, as it turns out, this paper is also about p-selective sets [Sel79]. It is known that a set L belongs to P if and only if it is both p-selective and disjunctive self-reducible. Thus, to construct a language in $\mathrm{NP} - \mathrm{P}$ that has certain desired properties but that is not disjunctive self-reducible, a natural approach would be to construct a language that is in $\mathrm{NP} - \mathrm{P}$, that has the desired properties, but that is p-selective. We have been stymied in the past because the only method known for constructing p-selective sets has been the standard left cut construction of [Sel79, Sel82b]. In this paper we invent new methods for constructing p-selective sets that are not standard left cuts. However, every p-selective set we construct is $\leq_m^{\mathrm{P}}$-equivalent to some standard left cut, and, as the following theorem demonstrates, the question of whether *every* p-selective set is $\leq_m^{\mathrm{P}}$-equivalent to a standard left cut is subtle. We prove that if there is a p-selective set that is *not* $\leq_m^{\mathrm{P}}$-

equivalent to a standard left cut, then $\mathrm{P} \neq \mathrm{PP}$. We prove that (1) there is a p-selective set that is *not* $\leq_{tt}^{\mathrm{P}}$-equivalent to any tally set, while (2) if there is a p-selective set that is *not* $\leq_{\mathrm{T}}^{\mathrm{P}}$-equivalent to any tally set, then $\mathrm{P} \neq \mathrm{PP}$. In a similar vein, we show that if there is a cheatable set [Bei87, Bei91] that is *not* $\leq_{m}^{\mathrm{P}}$-equivalent to any tally set, then $\mathrm{P} \neq \mathrm{NP}$, and we obtain with no assumption, a recursive p-selective tally set that is *not* cheatable.

Buhrman, van Helden, and Torenvliet [BvHT93] have very recently proved that every p-selective self-reducible set belongs to P. Their result gives us an added bonus. Since it has been known that search reduces to decision for every disjunctive self-reducible set, we desired to show that the converse fails, and we obtained this result using the technique described in the previous paragraph. That is, we obtain a set in $\mathrm{NP} - \mathrm{P}$ for which search reduces to decision but that is not disjunctive self-reducible by exhibiting a p-selective set in $\mathrm{NP} - \mathrm{P}$ for which search reduces to decision. By applying the result in [BvHT93], we immediately obtain the stronger claim that there is a set in $\mathrm{NP} - \mathrm{P}$ for which search reduces to decision and that is not self-reducible at all. Throughout this paper, we will make the stronger claim whenever possible.

We shall present some preliminary notations and definitions in the next section. In section 3, we present our theorems comparing sets for which search reduces to decision with self-reducible sets, and with sets for which search nonadaptively reduces to decision. Section 4 gives our results on random self-reducibility and on natural interactive proof systems. Section 5 addresses the question of equivalence of p-selective sets with standard left cuts, tally sets, and p-cheatable sets.

2 Preliminaries

In this section we set down notation and definitions that are used throughout.

All languages considered are over the finite alphabet $\Sigma = \{0, 1\}$. The set of strings of length n is denoted by Σ^n. Σ^ω is the set of all strings of infinite length of order type ω. (Languages contains strings of finite length only.)

Let $\mathrm{E} = \cup_c \mathrm{DTIME}(2^{cn})$, and $\mathrm{NE} = \cup_c \mathrm{NTIME}(2^{cn})$. We will frequently use the well-known fact [Boo74] that $\mathrm{NP} - \mathrm{P}$ contains a tally language if and only if $\mathrm{NE} \neq \mathrm{E}$. Let $\{P_i\}_{i \in N}$ ($\{NP_i\}_{i \in N}$) denote an enumeration of deterministic (nondeterministic) polynomial time Turing machines, such that $P_i(x)$ ($NP_i(x)$) runs in time $p_i(|x|) = |x|^i$.

We use standard notation for polynomial time reductions [LLS75]. We require the following standard reductions in particular. A set A is *truth-table* reducible to a set B in polynomial time ($A \leq_{tt}^{\mathrm{P}} B$) if there exist polynomial time computable functions g and e such that on input x, $g(x)$ is a set of queries $Q = \{q_1, q_2, \ldots, q_k\}$, and $x \in A$ if and only if $e(x, B(q_1), \ldots, B(q_k)) = 1$. The function g is known as the *condition generator* and e as the *condition evaluator*. A set A is *disjunctive reducible* to a set B

($A \leq_d^{\mathrm{P}} B$) if $g(x)$ is a set of queries Q such that $x \in A \leftrightarrow Q \cap B \neq \emptyset$. It is well-known [LLS75] that $A \leq_{tt}^{\mathrm{P}} B$ if and only if there is a *nonadaptive* polynomial time reduction from A to B. That is, $A \leq_{tt}^{\mathrm{P}} B$ if and only if there is a polynomial time-bounded oracle Turing machine M and a polynomial time computable function f that maps each input word x to a set of queries $f(x) = \{q_1, q_2, \ldots, q_k\}$ such that $A = L(M, B)$ and for each input x to M, M only makes queries to B from the list $f(x)$.

We will refer to the following complexity classes of partial multivalued functions. NPMV is the set of all partial multivalued functions that are computed nondeterministically in polynomial time [BLS84], and PF is the set of all partial functions that are computed deterministically in polynomial time. $\mathrm{PF}^{\mathrm{NP}}$ is the set of all partial functions that can be computed in polynomial time relative to an oracle in NP, and $\mathrm{PF}_{tt}^{\mathrm{NP}}$ is the set of all partial functions that can be computed nonadaptively in polynomial time with access to an oracle in NP [Bei91, Sel92]. We take the point of view that a partial multivalued function is easy to compute if for each input string in the domain of the function, some value of the function is easy to compute. For this reason, we define the following technical notions. Given partial multivalued functions f and g, define g to be a *refinement* of f if $dom(g) = dom(f)$ and for all x in $dom(g)$ and all y, if y is a value of $g(x)$, then y is a value of $f(x)$. If f is a partial multivalued function and $\mathcal{G}$ is a class of partial multivalued functions, we write $f \in_c \mathcal{G}$ if $\mathcal{G}$ contains a refinement g of f, and if $\mathcal{F}$ and $\mathcal{G}$ are classes of partial multivalued functions, we write $\mathcal{F} \subseteq_c \mathcal{G}$ if for every $f \in \mathcal{F}$, $f \in_c \mathcal{G}$. This notation is consistent with our intuition that $\mathcal{F} \subseteq_c \mathcal{G}$ should entail that the complexity of the class $\mathcal{F}$ is not greater than the complexity of $\mathcal{G}$. For example, the assertion "NPMV $\subseteq_c$ PF" means that every partial multivalued function in NPMV has a refinement that can be computed efficiently by some deterministic polynomial time transducer. Notions of polynomial time Turing reducibility between partial multivalued functions are defined in [Sel92, FHOS93]. Here we need only the following special case. Let M be a deterministic oracle Turing machine transducer that accesses a set as its oracle. Given a set L, let $M[L]$ denote the single-valued partial function that M computes with oracle L. For a partial multivalued function f and set L, we define $f \leq_{\mathrm{T}}^{\mathrm{P}} L$ if for some deterministic polynomial time-bounded oracle transducer M, $M[L]$ is a refinement of f. Furthermore, $f \leq_{tt}^{\mathrm{P}} L$ if M accesses its oracle nonadaptively.

The standard definition of self-reducibility that is used in most contemporary research in complexity theory was given by Meyer and Paterson [MP79].

Definition 1 *A polynomial time computable partial order $<$ on Σ^* is* OK *if and only if,*

1. *Each strictly decreasing chain is finite and there is a polynomial p such that every finite $<$-decreasing chain is shorter than p of the length of its maximum element, and*

2. For all $x, y \in \Sigma^*$, $x < y$ implies that $|x| \leq p(|y|)$.

Definition 2 *A set $L \in \Sigma^*$ is* self-reducible *if there is an OK partial order $<$ and a deterministic polynomial time-bounded machine M such that M accepts L with oracle L and, on any input x, M asks its oracle only about words strictly less than x in the partial order. If the query machine M also provides a $\leq_d^{\mathrm{P}}$-reduction, then A is disjunctive self-reducible.*

For any language $L \in \mathrm{NP}$, if R is a polynomial time recognizable relation and p is a polynomial such that for all x,

$$x \in L \leftrightarrow \exists y[|y| = p(|x|) \text{ and } R(x, y)], \tag{1}$$

we say that R and p *define* L, and a string y that satisfies equation 1 is called a *witness* for x. For each language $L \in \mathrm{NP}$ and R and p that define L, we define the partial multivalued function $f_{R,p}$ as follows:

$$f_{R,p}(x) \mapsto y, \text{if } |y| = p(|x|) \text{ and } R(x, y).$$

Clearly, each $f_{R,p}$ belongs to NPMV.

Definition 3 *[BD76, BBFG91] Let $L \in \mathrm{NP}$.* Search reduces to decision for[1] *L if there exist R and p that define L such that*

$$f_{R,p} \leq_{\mathrm{T}}^{\mathrm{P}} L. \tag{2}$$

Search nonadaptively *reduces to decision for L if equation 2 can be replaced by $f_{R,p} \leq_{tt}^{\mathrm{P}} L$.*

In other words, search reduces to decision for a language $L \in \mathrm{NP}$ if there exist R and p that define L and a polynomial time-bounded reduction procedure with oracle L such that for each $x \in L$, the output of the reduction procedure on x is a witness for x.

Ko [Ko83] showed that all disjunctive self-reducible sets are in NP.[2] The notion of search reducible to decision applies to sets in NP only.

Schöning defined an oracle Turing machine M to be *robust* if for every oracle A, the language accepted by M^A is the same [Sch85]. Ko [Ko87] used this notion to define the notion of *self-1-helpers*. In [Bal90], Balcázar proved that a set $L \in \mathrm{NP}$ is a self-1-helper if and only if search reduces to decision for L.

[1][BD76] requires equation 2 to hold for all R and p that define L.

[2]We can show that reducing search to decision by weaker reducibilities such as $\leq_{btt}^{\mathrm{P}}$, $\leq_{pos}^{\mathrm{P}}$, or $\leq_{\log\text{-}\mathrm{T}}^{\mathrm{P}}$ is not interesting. Specifically, if $\leq_r^{\mathrm{P}}$ is one of these, then search reduces to decision for a language L by an $\leq_r^{\mathrm{P}}$ reduction only if $L \in \mathrm{P}$. (cf. [Kre88])

Definition 4 *[Ko87] A set A is a* self-1-helper *if there is a robust, deterministic oracle Turing machine M that accepts A such that for all $x \in A$, M^A on input x halts in polynomial time. A set A is a* nonadaptive self-1-helper *if M makes only nonadaptive queries to A.*

Now we define random self-reductions [FF91, AFK89, FFLS92].

Definition 5 *Let $f : \Sigma^* \mapsto \Sigma^*$, let $k(n)$ be a polynomial, and let M be a probabilistic polynomial time oracle Turing machine which has access to f as an oracle. M is a $k(n)$* random self-reduction *for f if the following properties are true.*

1. *For all $x \in \Sigma^*$, $f(x) = M^f(x)$ with probability at least 2/3.*
2. *For all $x \in \Sigma^*$, $M^f(x)$ asks at most $k(|x|)$ many queries $q_1^f(x), q_2^f(x), \ldots, q_{k(|x|)}^f(x)$ to f.*
3. *For all n and for all $x_1, x_2 \in \Sigma^n$, and all $i \in \{1, 2, \ldots, k(n)\}$, $q_i^f(x_1)$ and $q_i^f(x_2)$ are identically distributed.*

The probabilistic polynomial time oracle Turing machine M in Definition 5 is assumed to be adaptive. A language L is *randomly self-reducible* if there is a polynomial $k(n)$ and M as above such that M is a $k(n)$ random self-reduction for the characteristic function of L. *Nonadaptive random self-reduction* is a special case of an adaptive random self-reduction, where the queries to the oracle are restricted to be nonadaptive.

P-selective sets have been the focus of many researchers, [Sel79, Ko83, Tod91], and they form the basis of our work. This property was defined by Selman in [Sel79].

Definition 6 *A set L is* p-selective *if there exists a polynomial time computable function $f : \Sigma^* \times \Sigma^* \mapsto \Sigma^*$ such that*

(i) *$f(x, y) \in \{x, y\}$, and*

(ii) *if $f(x, y) = y$, then $x \in L \rightarrow y \in L$.*

The function f is called a *p-selector* for L. If L is a p-selective language with p-selector f, and Q is a finite set, we use f to define a total order $\leq_f$ on strings in Q as follows.

$$\begin{aligned} \forall x, y \in Q, x \leq_f y \quad \leftrightarrow \quad & \exists z_1, z_2, \cdots, z_m \in Q, \\ & f(x, z_1) = x, f(z_1, z_2) = z_1, \ldots, \\ & f(z_{m-1}, z_m) = z_{m-1}, f(z_m, y) = z_m. \end{aligned}$$

Let $\perp$ be a special symbol such that $\perp \leq_f x$, for all $x \in \Sigma^*$.

Given any p-selector f (for some p-selective set), every finite set Q can be ordered by $\leq_f$ in time a polynomial in the sum of the lengths of the strings in Q.

We will use the following lemma, proved by Toda in [Tod91].

Lemma 1 *Let L be a p-selective set with p-selector f, and let $Q \subseteq \Sigma^*$ be a finite set. Then, there exists a string $z \in Q \cup \{\perp\}$ such that $Q \cap L = \{y \in Q \mid y \leq_f z\}$ and $Q \cap \overline{L} = \{y \in Q \mid y \not\leq_f z\}$. The string z is called the "pivot" string.*

The classical and only previously known method for constructing p-selective sets has been the standard left cut construction of [Sel79, Sel82b]. For $r \in \Sigma^* \cup \Sigma^\omega$, the *standard left cut* of r is the set

$$L(r) = \{x \in \Sigma^+ \mid x < r\}.$$

Where $<$ is the ordinary dictionary ordering of strings with 0 less than 1. If r is infinite, then $x < r$ if and only if $x \leq r'$ for r' the prefix of r of length $|x|$. (Dictionary ordering should not be confused with lexicographic ordering. For example, the string 001 is less than the string 10 in dictionary ordering but not in lexicographic ordering.) It is obvious that every standard left cut is p-selective with selector $f(x, y) = \min\{x, y\}$.

3 On Self-Reducibilities

3.1 "Self-reducible" vs. "search reduces to decision"

Selman showed in [Sel79] that a set L is disjunctive self-reducible and p-selective if and only if $L \in \mathrm{P}$, and Buhrman, van Helden, and Torenvliet improved this by showing [BvHT93] that a set L is self-reducible and p-selective if and only if $L \in \mathrm{P}$. We obtain a similar characterization of P using sets for which search nonadaptively reduces to decision.

Theorem 1 *If $L \in \mathrm{NP}$ is p-selective and search nonadaptively reduces to decision for L, then $L \in \mathrm{P}$.*

Proof The proof is an application of Lemma 1. Let $M = \langle g, e \rangle$ be a nonadaptive reduction from search to decision for L. (Since e is a polynomial time-bounded deterministic transducer, we assume without loss of generality that e is total.) There exists a polynomial time computable relation R such that for all $x \in \Sigma^*$, if $M(x)$ outputs y, then $x \in L \leftrightarrow R(x, y)$. Let f be a p-selector for L. The following polynomial time algorithm recognizes L: On input x, simulate the condition generator g to generate the query set $Q = \{q_1, \ldots, q_k\}$. Let $Q' = \{q_0, q_1, q_2, \cdots, q_k\}$, where $q_0 = \perp$.

for $i = 0$ **to** k **do**
  begin
  { Assume that q_i is the pivot string}
  for $j = 0$ **to** k **do**
    if $q_j \leq_f q_i$ **then** $v_j := TRUE$ **else** $v_j := FALSE$;
  if $R(x, e(x, v_1, v_2, \ldots, v_k))$ **then** ACCEPT
  end;
REJECT.

The above algorithm simulates the condition evaluator e on some of the possible truth table values, and by Lemma 1, exactly one of these simulations will produce the correct set of answers. The outer for-loop executes at most a polynomial number of times since g is polynomial time-bounded. Since there are only a polynomial number of ways in which a set of queries can be answered, this algorithm runs in polynomial time. □

Thierauf [Thi92] has observed that the hypothesis of Theorem 1 can be weakened, which, for later reference, we state as a separate proposition:

Proposition 1 *If $L \in$ NP is $\leq^{\mathrm{P}}_{tt}$-reducible to a p-selective set and search nonadaptively reduces to decision for L, then $L \in$ P.*

It follows immediately from Theorem 1 and [BvHT93] that if E $\neq$ NE, then there is a set $L \in$ NP $-$ P which is *not* self-reducible and for which search does *not* nonadaptively reduce to decision. (This is so because it is known [Sel79] that E $\neq$ NE implies the existence of a p-selective set in NP $-$ P.) However, a stronger assertion than this will follow from the following theorem.

Theorem 2 *If* NE $\neq$ E, *then there exists a p-selective set $L \in$* NP $-$ P *for which search reduces to decision.*

Proof By hypothesis, there is a tally language T in NP $-$ P. For each $i \geq 1$, if $0^i \in T$, then there is a witness σ_i for $0^i \in T$, and, given $i \geq 1$ and a string σ_i, one can decide in polynomial time whether σ_i is a witness for $0^i \in T$. Let p be a polynomial so that for each $i \geq 1$, if $0^i \in T$, then each witness σ_i for $0^i \in T$ has length $|\sigma_i| = p(i)$, and for each $i \geq 1$, let $n_i = p(i)$.

For each $i \geq 1$, if $0^i \in T$, define w_i to be the lexicographically largest witness for $0^i \in T$, and if $0^i \notin T$, define $w_i = 0^{n_i}$. Define the infinite sequence $w = w_1 \cdot w_2 \cdot w_3 \cdots$.

Consider the standard left cut $L = L(w)$. Clearly L is p-selective. We need to show that $L \in$ NP $-$ P and that search reduces to decision for L.

For any string x, let $k(x)$ be the smallest length k such that $|x| \leq n_1+n_2+\cdots+n_k$. Then, $x \in L$ if and only if there exist strings $v_1, \ldots, v_{k(x)}$ such that $x \leq v_1 \cdot v_2 \cdots v_{k(x)}$ (in dictionary ordering) and each v_i is 0^{n_i} or a witness for $0^i \in T$. As the right hand side is an NP-condition, it follows that $L \in \mathrm{NP}$.

For each $i \geq 1$, the sequence $w_1 \cdot w_2 \cdots w_i$ can be computed in polynomial time in i by a binary search algorithm using L as an oracle. Thus, $L \in \mathrm{P}$ would imply $T \in \mathrm{P}$, because we could decide whether w_i is a witness for $0^i \in T$. For much the same reason, search reduces to decision for L, and this completes the proof. $\square$

Corollary 1 *If* $\mathrm{NE} \neq \mathrm{E}$*, then there exists a set L in* $\mathrm{NP}-\mathrm{P}$ *such that search reduces to decision for L, search does* not *nonadaptively reduces to decision for L, and L is* not *self-reducible.*

The corollary follows by Theorem 2, using [BvHT93] together with Theorem 1.

Corollary 2 *If* $\mathrm{NE} \neq \text{co-NE}$*, then there exists a set L in* $\mathrm{NP} - \text{co-NP}$ *such that search reduces to decision for L, search does* not *nonadaptively reduces to decision for L, and L is* not *self-reducible.*

Proof By hypothesis, there is a tally language T in $\mathrm{NP} - \text{co-NP}$, and the construction in the proof of Theorem 2 gives a p-selective set L in NP. Recalling that T is recognizable with the help of a binary search algorithm that uses L as an oracle, we have $T \in \mathrm{P}^L$. Thus, $L \in \text{co-NP}$ implies $T \in \mathrm{P}^{\mathrm{NP}\cap\text{co-NP}} \subseteq \text{co-NP}$. $\square$

Suppose that polynomial time recognizable relation S and polynomial p define the set $L \in \mathrm{NP}$ that exists by Theorem 2. Consider the partial function $f_{S,p}$, and consider the following set (cf. [GS88, Mil76])

$$R_{f_{S,p}} = \{\langle x, y\rangle \mid \exists z[y \leq z \text{ and } f_{S,p}(x) \mapsto z]\}.$$

It is easy to see that $f_{S,p}$ is reducible to $R_{f_{S,p}}$ in polynomial time by a binary search algorithm. However, under the hypothesis of Theorem 2, $f_{S,p}$ is not reducible to $R_{f_{S,p}}$ in polynomial time by any nonadaptive algorithm. Thus, we see that binary search is inherently adaptive.

In [Bal90], Balcázar proved that a set $L \in \mathrm{NP}$ is a self-1-helper if and only if search reduces to decision for L. It is easy to show that the equivalence holds for nonadaptiveness also. Thus, we may restate our corollary as a result about self-1-helpers.

Corollary 3 *If* NE $\neq$ E*, then there exists a language* $L \in$ NP $-$ P *that is a self-1-helper, that is* not *a nonadaptive self-1-helper, and is* not *self-reducible.*

Corollary 3 settles an open question in [Ko87] by showing that the converse of one of Ko's main theorems in [Ko87] is false unless NE = E.

3.2 "Self-reducible" vs. "search nonadaptively reduces to decision"

Let UE denote the class of languages accepted by nondeterministic exponential time unambiguous machines, and let co-UE $= \{L \mid \overline{L} \in \mathrm{UE}\}$.

Theorem 3 *If* UE $\neq$ E*, then there exists a language* $L \in$ UP $-$ P *such that search nonadaptively reduces to decision for* L*, but* L *is* not *self-reducible.*

Proof Assume UE $\neq$ E so that UP $-$ P contains a tally language T. Define

$$L = \{0^{\langle n,i,j\rangle} \mid 0^n \in T \text{ and } j \text{ is the } i^{th} \text{ bit of the witness of } 0^n\}.$$

Obviously, $L \in$ UP $-$ P and it follows readily that search nonadaptively reduces to decision for L. However, since L is a tally language, it cannot be self-reducible, for that would imply that $L \in$ P [Ber78]. $\square$

Corollary 4 *If* UE $\neq$ E*, then there exists a language* $L \in$ UP $-$ P *such that* L *is a nonadaptive self-1-helper, but* L *is* not *self-reducible.*

Ko defined the following concept of *unambiguous 1-helping* in [Ko87].

Definition 7 *A set* H unambiguously 1-helps *a robust machine* M *on* $A = L(M, \emptyset)$ *if there is a polynomial* p *such that for all* $x \in A$*, and all oracles* B*, if* M^B *accepts* x *in* $p(|x|)$ *many moves, then the accepting path is identical to that of* $M^H(x)$*.*

Ko proved that every unambiguous self-1-helper (i.e., a set that unambiguously 1-helps itself) is Turing equivalent to a disjunctive self-reducible set. It is easy to see that every self-1-helper in UP is an unambiguous self-1-helper. Thus, if $L \in$ UP and search reduces to decision for L, then L is $\leq^{\mathrm{P}}_{\mathrm{T}}$-equivalent to a disjunctive self-reducible set. In particular, the set L that is constructed in the proof of Theorem 3 is $\leq^{\mathrm{P}}_{\mathrm{T}}$-equivalent to a disjunctive self-reducible set. Assuming UE $\neq$ E, Corollary 4 provides an example of an unambiguous self-1-helper that is not self-reducible.

We will use the following definition in the proofs of the next two theorems.

Definition 8 *A set $C \subseteq \Sigma^*$ is a* chain *if for all elements q and q' in C, either q is a prefix of q', or q' is a prefix of q.*

Theorem 4 *If* $\mathrm{UE} \cap \text{co-UE} \neq \mathrm{E}$, *then there is a disjunctive self-reducible language* $L \in \mathrm{UP} \cap \text{co-UP} - \mathrm{P}$ *for which search does* not *nonadaptively reduce to decision.*

Proof By hypothesis, let $T \in \mathrm{UP} \cap \text{co-UP} - \mathrm{P}$ be a tally set. For each $i \geq 1$, there is a unique witness σ_i for $0^i \in T$ or for $0^i \in \overline{T}$, and, given $i \geq 1$ and a string σ_i, one can decide in polynomial time whether σ_i is the witness for $0^i \in T$ or for $0^i \in \overline{T}$.

We call the following string s_n a "witness tuple" for 0^n:

$$\begin{aligned} s_n &= \sigma_1 \sigma_2 \cdots \sigma_n, \\ &\quad \text{where for all } i \leq n,\ \sigma_i \text{ is the witness for } 0^i \in T \text{ or for } 0^i \in \overline{T}. \end{aligned}$$

Thus the witness tuple s_n encodes information about the membership of the first n tally strings in T.

Define

$$L = \{w \mid w \text{ is a prefix of a witness tuple}\}.$$

We assume without loss of generality that there exists a polynomial p such that for all $n > 0$, every witness tuple s_n has length $p(n)$. We say that a natural number m is "good" if $m = p(n)$ for some n.

It is easy to see that L is disjunctive self-reducible: For x of good length, membership in L can be determined in deterministic polynomial time, and for x not of good length, $x \in L$ if and only if $x0 \in L$ or $x1 \in L$.

To prove the other properties of L, first note that if $m \leq n$, then the witness tuple for 0^m is a prefix of the witness tuple for 0^n. It follows that L is a chain with exactly one element of each length.

Now it is easy to show that $L \in \mathrm{UP} \cap \text{co-UP} - \mathrm{P}$. Suppose that $L \in \mathrm{P}$. Then we can compute the witness tuple for 0^n in polynomial time, which implies that $T \in \mathrm{P}$. Membership in L and $\overline{L}$ for inputs of length n can be determined by guessing the unique witness tuple for 0^m, where m is the least number such that $p(m) \geq n$.

Finally, we show that if search nonadaptively reduces to decision for L, then $L \in \mathrm{P}$. Suppose $M = \langle g, e \rangle$ is a deterministic polynomial time Turing machine that nonadaptively reduces search to decision for L.

By the definition of L, Fact 1 is immediate.

Fact 1 *For $Q \subseteq \Sigma^*$, $Q \cap L$ is a chain and either $Q \cap L = \emptyset$, or $Q \cap L = \{q' \in Q \mid q' \text{ is a prefix of } q\}$ for some $q \in Q$.*

We now describe a polynomial time algorithm that recognizes L.

1. On input $x, |x| = n$, simulate M to generate all queries $Q = \{q_1, q_2, \cdots, q_{k(n)}\}$.

2. Form a collection of maximal chains over queries in Q; that is, first partition Q into disjoint chains, and then make multiple copies of queries that can be in more than one chain. Repeat the process until all chains are maximal, and each chain has at least one element that is not in any other chain. Let the chains be $C_1, C_2, \cdots, C_r$.

3. For $1 \leq i \leq r$, repeat

   - Switch the i^{th} chain "on," that is: assume that $Q \cap L \subseteq C_i$.
   - Generate the $\|C_i\|+1$ many candidate sets of answers, and for each of these answers, simulate the condition evaluator e of the "search to decision" reduction for L. If for any set of answers a witness is produced, then ACCEPT.

4. REJECT.

By Fact 1, we know that the correct set of answers would be simulated at least once during the above simulation. Further, since there are at most a polynomial number of chains, and each chain has polynomially many consistent answers, the algorithm runs in polynomial time. □

Corollary 5 *If* UE $\cap$ co-UE $\neq$ E, *then there is a disjunctive self-reducible language in* UP $-$ P *that is* not *a nonadaptive self-1-helper.*

All the well known examples of self-reducible sets in NP [Mah82, MP79, Sel88] are actually *disjunctive* self-reducible sets. In light of the results of Buhrman, van Helden, and Torenvliet [BvHT93], it is reasonable to inquire whether NP contains sets that are self-reducible but not disjunctive self-reducible. The following theorem gives an affirmative answer to this question, thereby adding import to their work.

Theorem 5 *If* UE $\cap$ co-UE $\neq$ E, *then there exists a set in* UP $\cap$ co-UP $-$ P *that is conjunctive self-reducible, but* not *disjunctive self-reducible.*

Proof The following fact is used in the proof.

Fact 2 *A set L is disjunctive self-reducible if and only if $\overline{L}$ is conjunctive self-reducible.*

If $\mathrm{UE} \cap \text{co-UE} \neq \mathrm{E}$, then there exists a tally set $T \in \mathrm{UP} \cap \text{co-UP} - \mathrm{P}$. Using T, define set L as in the proof of Theorem 4:

$$L = \{w \mid w \text{ is a prefix of a witness tuple}\}.$$

We will show that $\overline{L}$ fulfills the conditions of Theorem 5. By Theorem 4 and Fact 2, it is immediate that $\overline{L}$ is in $\mathrm{UP} \cap \text{co-UP} - \mathrm{P}$ and that $\overline{L}$ is conjunctive self-reducible.

To prove that $\overline{L}$ is not disjunctive self-reducible, it suffices to show that L is not conjunctive self-reducible. We claim that if L is conjunctive self-reducible, then $L \in \mathrm{P}$.

Suppose L is conjunctive self-reducible by a Turing machine M. Then, the following deterministic algorithm determines membership of x in L.

On input x, simulate M to obtain the set of conjunctive queries $Q = \{q_1, q_2, \ldots, q_k\}$. If Q is not a chain, then reject, else if Q is a chain, then let q_j be the maximal element in Q. By Fact 1, it follows that if $q_j \in L$ then for all $q_i \in Q$, $q_i \in L$. Simulate M on input q_j and repeat the above process. Since the queries successively generated are ordered by an OK partial order, after polynomially many simulations, M will be simulated on q^* whose membership in L can be determined in deterministic polynomial time without any queries to L. If $q^* \in L$, then accept, else reject. Thus $L \in \mathrm{P}$ which is a contradiction, which completes the proof. □

In [BvHT93], it is shown that if search reduces to decision for a language $L \in \mathrm{NP}$, then L is auto-reducible [AS84]. Let us note that the converse does not hold. Let L be a set in NP for which search does not reduce to decision [BBFG91]. Then, $L \oplus L$ is auto-reducible, but search does not reduce to decision for $L \oplus L$.

3.3 On Computing Satisfying Assignments

Corollary 2 states under a reasonable assumption that there is a set in $\mathrm{NP} - \mathrm{P}$ for which search does not nonadaptively reduce to decision. What can we say about general NP problems? That is, does search nonadaptively reduce to decision for any NP set? In order to consider this question, we will look at the standard NP-complete set SAT and its standard search problem.

Consider the partial multivalued function sat defined by $sat(x) \mapsto y$ if x is satisfiable and y is a satisfying assignment for x. Clearly, $sat \leq_{\mathrm{T}}^{\mathrm{P}} \mathrm{SAT}$. Is there a nonadaptive reduction from sat to the decision problem SAT? Next, we construct a relativized world in which the answer is no.

Recalling the definitions concerning function classes given in Section 2, we have the following proposition.

Proposition 2 *$sat \leq_{tt}^{\mathrm{P}}$ SAT if and only if $\mathrm{NPMV} \subseteq_c \mathrm{PF}_{tt}^{\mathrm{NP}}$.*

The next theorem states that there is an oracle B relative to which the right hand side of Proposition 2 holds. Of course, both the truth-table reduction and the NP-oracle should have access to the oracle B—in symbols, $\mathrm{NPMV}^B \not\subseteq_c (\mathrm{PF}^B)_{tt}^{\mathrm{NP}^B}$. Indeed the reduction might even access B adaptively. We will begin with the following lemma which demonstrates that it suffices to restrict the reduction so that the only access to B is via nonadaptive queries to the oracle in NP^B.

Lemma 2 *For all oracles A:*

$$(\mathrm{PF}^A)_{tt}^{\mathrm{NP}^A} = \mathrm{PF}_{tt}^{\mathrm{NP}^A}.$$

This lemma was first realized by Richard Beigel and Lance Fortnow, and its proof uses Buss and Hay's notion [BH88] of *multiple round* truth-table reductions to NP sets. In a k round truth-table reduction, a Turing machine asks k rounds of polynomially many nonadaptive queries to the NP oracle, where the queries for round i are formed using the answers to the queries in the previous rounds. Buss and Hay obtained the following result about the power of multiple round truth-table reductions. For our purpose, we state their result in terms of function classes.

Proposition 3 *[BH88] Let $k \geq 1$. If f is a partial function computed by a transducer T which runs in polynomial time and makes k rounds of nonadaptive queries to an* NP *set, then $f \in \mathrm{PF}_{tt}^{\mathrm{NP}}$.*

Now we will prove Lemma 2.

Proof Let $f \in (\mathrm{PF}^A)_{tt}^{\mathrm{NP}^A}$. Let $M^A = \langle g^A, e^A \rangle$ be a truth-table reduction that computes f relative to a set $L \in \mathrm{NP}^A$. Recall that $g^A(x)$ is a set of queries $Q = \{q_1, q_2, \ldots, q_k\}$ (suitably encoded as a string), and that

$$f(x) = e^A(x, L(q_1), L(q_2), \ldots, L(q_k)).$$

Define

$$\mathit{code}(e^A) = \{\langle w, i, b \rangle \mid \text{the } i^{th} \text{ bit of } e^A(w) \text{ is } b\},$$

and define

$$\mathit{code}(g^A) = \{\langle w, i, j, b \rangle \mid \text{the } i^{th} \text{ bit of the } j^{th} \text{ element of } g^A(w) \text{ is } b\},$$

where w is a string in $\{0,1\}^*$, $i \geq 1$, $j \geq 1$, and $b \in \{0,1\}$.

Since g^A and e^A are in PF^A, $\mathit{code}(g^A) \in \mathrm{P}^A$ and $\mathit{code}(e^A) \in \mathrm{P}^A$. Thus, the following language $L' = \mathit{code}(g^A) \oplus \mathit{code}(e^A) \oplus L$ is in NP^A.

Next we will define a 3-round truth-table reduction M'. M' computes f by making rounds of nonadaptive queries to L'. M' does not access the oracle A. Since

Proposition 3 relativizes, it follows that there is a one-round nonadaptive reduction M'' that computes f relative to L'. Since M' does not query A, it follows that M'' does not query A either. Thus, since $L' \in \mathrm{NP}^A$, $f \in \mathrm{PF}_{tt}^{\mathrm{NP}^A}$ follows.

The 3-round truth-table reduction M' remains to be described. On input x, let $g^A(x)$ output $Q = \{q_1, q_2, \ldots, q_k\}$. Let p_1 be a polynomial bounding the length of each element in Q, and let p_2 be another polynomial bounding the length of the string output by the evaluator on input $\langle x, L(q_1), L(q_2), \ldots, L(q_k)\rangle$. M' works as follows:

Round 1: For all $1 \le i \le p(n)$, $1 \le j \le k$, and $b \in \{0,1\}$, query $00\langle x, i, j, b\rangle$ to L'.

From the answer to the queries in round 1, it is easy to construct the strings $q_1, q_2, \ldots, q_k$.

Round 2: Query $1q_1, 1q_2, \ldots, 1q_k$ to L'.

The answers obtained from round 2 can be used to construct $L(q_1), L(q_2), \ldots, L(q_k)$.

Round 3: For all $1 \le i \le p_2(n)$, query the strings $01\langle\langle x, L(q_1), \ldots, L(q_k)\rangle, i, b\rangle$ to L'.

From the answer to these queries, it is easy to construct the string y that the condition evaluator e^A outputs on input $\langle x, L(q_1), \ldots, L(q_k)\rangle$.

Output y.

It is easy to see that M' outputs y if and only if M outputs y and M' runs in a polynomial number of steps in $|x|$.

This completes the proof of the lemma. □

Theorem 6 *There exists an oracle B such that* $\mathrm{NPMV}^B \not\subseteq_c (\mathrm{PF}^B)_{tt}^{\mathrm{NP}^B}$.

Proof We will construct an oracle B such that $\mathrm{NPMV}^B \not\subseteq_c \mathrm{PF}_{tt}^{\mathrm{NP}^B}$, for then the theorem follows immediately from Lemma 2.

For each oracle X, we will consider the following partial multivalued function f^X:

$$f^X(0^n) \mapsto y, \text{ if } y \in X, \text{ and } |y| = n; \tag{3}$$

It is easy to see that for all oracles X, $f^X \in \mathrm{NPMV}^X$. We will construct an oracle B such that $f^B \notin_c \mathrm{PF}_{tt}^{\mathrm{NP}^B}$. It suffices to diagonalize f^B over all truth-table reductions with the NP^B-complete set $K^B = \{\langle i, x, y\rangle \mid NP_i^B \text{ accepts } x \text{ in } < |y| \text{ steps}\}$ as the NP^B oracle.

Let $\{g_i\}_{i \in N}$ denote an enumeration of condition generators, and $\{e_i\}_{i \in N}$ denote an enumeration of condition evaluators in PF. Let p be a polynomial denoting the running time of a nondeterministic polynomial time Turing machine that accepts K^B. The oracle B is constructed in stages. Let $B(n)$ denote the finite set of strings added to B prior to stage n. The construction uses a parameter m which is increased at each stage; at the beginning of stage n, decisions have been made about all strings of length $\leq m$.

Initially, set $B(0) := \emptyset$, and $m := 0$. We now give the construction of $B(n+1)$ at stage n.

stage n Let $n = \langle i, j \rangle$. At this stage we consider the i^{th} condition generator g_i and the j^{th} condition evaluator e_j. Let T^Z denote the $\mathrm{PF}_{tt}^{\mathrm{NP}^Z}$ transducer described by $\langle g_i, e_j \rangle$ with K^Z as the NP^Z oracle.

Let $k > m$ be such that:

$$p(p_i(k)) \cdot p_i(k) < 2^{k/4} \tag{4}$$

$$p_{i+1}(k) \cdot (p_{i+1}(k) + 1) < 2^{k/4}. \tag{5}$$

Simulate $T^{B(n)}$ on input 0^k: If $T^{B(n)}(0^k)$ accepts, then set $B(n+1) := B(n)$ and $m := 2^k$ and go to the next stage. (Thus, 0^k will not belong to the domain of f^B, but T^B will accept 0^k.) If $T^{B(n)}(0^k)$ rejects, then it suffices to find a set $X \subseteq \Sigma^k$ such that

(a) $T^{B(n) \cup X}(0^k)$ rejects, or

(b) $T^{B(n) \cup X}(0^k)$ outputs a string y such that $y \notin X$.

We now show that such a set X can always be found. If there exists a string $w \in \Sigma^k$ such that $T^{B(n) \cup \{w\}}(0^k)$ does not output w, then setting $X := \{w\}$ satisfies one of the two conditions.

Finally, we suppose that for all $w \in \Sigma^k$, $T^{B(n) \cup \{w\}}(0^k)$ outputs w. We will demonstrate existence of a set X that satisfies condition (b). Let $Q = \{y_1, \ldots, y_q\}$ be the set of queries generated by $g_i(0^k)$. For each string w, $|w| = k$, define

$$ACCEPT(w) = \{y \in Q \mid y \in K^{B(n) \cup \{w\}}\},$$

and

$$INTERROGATE(w) = \bigcup_{y \in ACCEPT(w)} \{s \in \Sigma^k \mid s \text{ is queried along the leftmost accepting path of } K^{B(n) \cup \{w\}}(y)\}.$$

Lemma 4 to follow guarantees the existence of a set $S \subseteq \Sigma^k$ such that $\|S\| > p_i(k)$ and, for all distinct $w, w' \in S$,

$$w \notin INTERROGATE(w'), \text{ and } w' \notin INTERROGATE(w).$$

For any nonempty $W \subseteq S$, define $ACCEPT(W) = \{y \in Q \mid y \in K^{B(n)\cup W}\}$ and $OUT(W)$ to be the output of $T^{B(n)\cup W}(0^k)$. Note for any W and W' with $W \subset W' \subseteq S$, that $ACCEPT(W) \subseteq ACCEPT(W')$ and that $ACCEPT(W) = ACCEPT(W')$ implies $OUT(W) = OUT(W')$. If there is some W such that $OUT(W) \notin W$, then we are done, for, by letting X be such a W, condition (b) is satisfied. We show, by contradiction, that such a W always exists. Assume there is no such W; that is, $OUT(W) \in W$ holds for every W. Then, define $W_0, \cdots, W_{\|S\|} \subseteq S$ as follows: Initially, define $W_0 = S$. For $h = 1, \cdots, \|S\|$, define

$$W_h = W_{h-1} - \{OUT(W_{h-1})\}.$$

For every $h, 0 \leq h \leq \|S\| - 1$, $OUT(W_h) \in W_h - W_{h+1}$, and therefore, $OUT(W_h) \neq OUT(W_{h+1})$. But, since $\|Q\| < \|S\|$ and for every $h, 0 \leq h \leq \|S\| - 1$, $ACCEPT(W_h) \supseteq ACCEPT(W_{h+1})$, there is some $h, 0 \leq h \leq \|S\| - 2$, such that $ACCEPT(W_h) = ACCEPT(W_{h+1})$. For such an h, it holds that $OUT(W_h) = OUT(W_{h+1})$. This is a contradiction. Thus, we can always satisfy condition (b).

Set $B(n+1) := B(n) \cup X$. Set $m := 2^k$, and go to the next stage. This completes the construction of the oracle B. □

Only Lemma 4 needs to be proved now. To prove Lemma 4, we will use the following *pebbling game*, which was defined in [GG86] and extended in [Rub91]. An $m \times m$ *board* is an $m \times m$ matrix over $\{0,1\}$ with its m^2 squares denoted by b_{st} where $1 \leq s,t \leq m$. For any k, an $m \times m$ board is *k-weakly covered* if $b_{ss} = 1$ for all s, and if for every set $I \subseteq \{s \mid 1 \leq s \leq m\}$ with cardinality greater than $(\log m)^k$, there exists a distinct pair $s,t \in I$ such that $b_{st} = 1$. The following lemma establishes a lower bound on the number of pebbles required to k-weakly cover a board.

Lemma 3 *[Rub91] At least $\frac{m(m-1)}{z(z-1)}$ pebbles are required to k-weakly cover an $m \times m$ board, where $z = \lfloor (\log m)^k + 1 \rfloor$.*

Lemma 4 *At stage $n = \langle i,j \rangle$ in the construction, if for all $w \in \Sigma^k$, $T^{B(n)\cup\{w\}}(0^k)$ outputs w, then there exists a set $S \subseteq \Sigma^k$, such that $\|S\| > p_i(k)$, and for all distinct $w, w' \in S, w \notin INTERROGATE(w')$ and $w' \notin INTERROGATE(w)$.*

Proof We use the pebbling game to model this problem. Let $w_1, \cdots, w_{2^k}$ be an enumeration of all strings in Σ^k. Consider a $2^k \times 2^k$ board, and define a *pebble* as follows:

$$b_{st} = 1 \leftrightarrow s = t \wedge w_s \in INTERROGATE(w_t).$$

To show the existence to S, it suffices to show that it is impossible to $(i+1)$-weakly cover the board, because then, we have some set $I \subseteq \{s \mid 1 \leq s \leq 2^k\}$, such that $\|I\| \geq (\log 2^k)^{i+1} = p_{i+1}(k)$, and for all distinct $s, t \in I$, $b_{st} = b_{ts} = 0$, thus $w_s \notin INTERROGATE(w_t)$, and $w_t \notin INTERROGATE(w_s)$. Further, $\|S\| = \|I\| \geq p_{i+1}(k) > p_i(k)$.

It is easy to see that for all strings $w \in \Sigma^k$, $\|ACCEPT(w)\| \leq p_i(k)$, thus $\|INTERROGATE(w)\| \leq p(p_i(k)) \cdot p_i(k)$, so the total number of pebbles available are less than: $2^k \cdot p(p_i(k)) \cdot p_i(k) < 2^{k/4} 2^k = 2^{5k/4}$, using equation 4. By Lemma 3, the number of pebbles required to $(i+1)$-weakly cover the board is greater than

$$\begin{aligned} \frac{2^k(2^k-1)}{k^{i+1}(k^{i+1}+1)} &> \frac{2^k(2^k-1)}{2^{k/4}} \text{ (by equation 5)} \\ &> \frac{2^{2k-1}}{2^{k/4}} = 2^{(7k/4)-1} \geq 2^{5k/4}. \end{aligned}$$

Hence there are not enough pebbles to $(i+1)$-weakly cover the board, which proves the lemma. □

In contrast to our result, Watanabe and Toda [WT93] have proved that NPMV $\subseteq_c$ $\mathrm{PF}^{\mathrm{NP}}_{tt}$ holds relative to a random oracle.

4 Adaptiveness in Random Reductions

In [FFLS92], it is shown that NEEE $\not\subseteq$ BPEEE implies that there exist sets in NP $-$ P which are randomly self-reducible, but not nonadaptively randomly self-reducible. The following theorem obtains the same consequence as in [FFLS92], with the additional property that the language that we construct is *not* self-reducible.

Theorem 7 *If* NE $\not\subseteq$ BPE, *then there exists a p-selective language $L \in$ NP $-$ BPP such that L is randomly self-reducible,* not *nonadaptively randomly self-reducible, and* not *self-reducible.*

Sketch of Proof The same construction as in the proof of Theorem 2 can be used to define a language L that separates adaptive and nonadaptive random self-reductions. If follows immediately from the construction that $L \in$ NP $-$ BPP and L is p-selective.

Recall from the proof of Theorem 2 that for every string x, $k(x)$ is the smallest integer such that $|x| \leq |w_1 \cdot w_2 \cdots w_{k(x)}|$. We can decide whether $x \in L$ by a binary search algorithm that finds the string $w_1 \cdot w_2 \cdots w_{k(x)}$ in polynomial time using L as an oracle, and then by testing whether x is smaller than this string in dictionary order. This computation depends only on $|x|$. For any two strings x and y such that

$|x| = |y|$, the queries made to the oracle are identical, so by definition, the reduction is a random self-reduction.

Now let us show that if L has a nonadaptive random self-reduction, then $L \in$ BPP. Assume that M is a nonadaptive random self-reduction for L, and consider the following procedure M', which we claim demonstrates that L belongs to BPP. On input x, M' randomly selects a set of queries $Q = \{q_1, q_2, \ldots, q_m\}$, where the length of each query and the number of queries is bounded by some polynomial in the length of x. Queries in Q are called *test* queries. Order these by dictionary ordering; it follows that there are at most $m + 1$ possible consistent answers for membership of the test queries in L (cf. Lemma 1). For each such set of answers, M' obtains a candidate for the string $w_1 \cdot w_2 \cdots w_{k(x)}$ by a binary search procedure as follows: At each stage of the binary search procedure, if y denotes the string whose membership in L is to be decided, then M' simulates M on input y. The queries made by M on input y are answered consistently with the answers to the test queries (that is, if q^* is the pivot string in Q, then the simulation answers "yes" to queries q such that $q \leq q^*$, and answers "no" if $q > q^*$.) If this simulation accepts y, then M' assumes that $y \in L$, and otherwise assumes that $y \notin L$.

By performing such a binary search for all consistent sets of answers for the test queries, M' obtains $m + 1$ many candidates for $w_1 \cdot w_2 \cdots w_{k(x)}$. M' discards those that do not fulfill the defining condition that each w_i is of the form 0^{n_i} or a witness for $w_i \in T$, and chooses the largest remainder. Finally, M' accepts x if and only if x is less than or equal to this remainder.

Techniques developed in [FFLS92] can be used to show that there exists a polynomial that bounds m such that the error probability of M' is bounded. We skip the details because of the identical nature of the arguments. □

Interestingly, the same language separates self-reducible sets from randomly self-reducible sets, and randomly self-reducible sets from sets for which search nonadaptively reduces to decision.

4.1 Tradeoffs in multiprover interactive proof systems

Now we turn to interactive proof systems and show that our techniques are applicable to this area. We assume readers are familiar with interactive proof systems [GMR89, Bab85] and with multiprover interactive proof systems [BOGKW88].

In [BBFG91, BG] the authors consider competitive proof systems. An interactive proof system for a language L is *competitive* if the honest prover is a probabilistic polynomial time oracle Turing machine that accesses the language L. We consider a variation of this model. Define a prover to be *natural* if it is an oracle for L. That is, the only messages sent to a prover by the verifier are queries of the form $q \in L$?, to which the prover can answer "Yes" or "No." Define a natural proof system to be one in which all honest provers are natural. We raise the question "How much interaction is required?" in a natural proof system. We will show under plausible complexity theoretic assumption that there are languages in NP that require a polynomial number of rounds even if additional provers are allowed. Following [FRS88], let $\mathrm{IP}(j,k)$ be the class of languages accepted with no more than j provers in no more than k rounds. Let *poly* designate a polynomial. Let $\mathrm{NIP}(j,k)$ be the corresponding class of languages accepted by natural interactive proof systems. Let *const* designate a constant. Thus, for example, $\mathrm{NIP}(const, const)$ denotes the set of languages accepted by natural interactive proof systems with a constant number of provers in a constant number of rounds.

Let compIP denote the class of languages accepted by competitive proof systems. It is easy to see that $\mathrm{NIP}(1, poly) = \mathrm{compIP}$. The inclusion from left to right is easy. For the converse, construct a verifier that runs the programs of the verifier and the honest prover of the competitive proof system and that refers to its own natural prover for the queries which the competitive prover would make to the oracle.

The following proposition is easy to prove.

Proposition 4 $\mathrm{NIP}(const, const) = \mathrm{NIP}(1,0) = \mathrm{BPP}$.

In order to prove Proposition 4, observe that $\mathrm{NIP}(const, const) \subseteq \mathrm{BPP}$ and that languages in BPP have natural interactive proof systems that do not require use of the provers.

If L is any language in NP for which search reduces to decision, then L belongs to $\mathrm{NIP}(1, poly)$. Thus, as a consequence of Proposition 4, if L is any language in $\mathrm{NP} - \mathrm{BPP}$ for which search reduces to decision, then L belongs to $\mathrm{NIP}(1, poly) - \mathrm{NIP}(const, const)$. This observation stands in contrast to [FRS88] in which they show that $\mathrm{IP}(poly, poly) = \mathrm{IP}(3,2)$.

Next we show that a trade-off between number of rounds and number of provers cannot be accomplished even if a polynomial number of provers are allowed.

Theorem 8 *If L is p-selective and L belongs to* $\mathrm{NIP}(poly, const)$*, then L is in* BPP.

Proof Since it is easy to see that $\mathrm{NIP}(poly, const) = \mathrm{NIP}(poly, 1)$, assume L is p-selective and L belongs to $\mathrm{NIP}(poly, 1)$. Let $(V, P_1, P_2, \ldots, P_{k(n)})$ be an interactive proof system which recognizes L, and let f be a p-selector for L. Let x be a string of length n. Without loss of generality, we assume that if $x \in L$, then $(V, P_1, P_2, \ldots, P_{k(n)})$ rejects x with probability at most $1/2^n$, and if $x \notin L$, then $(V, P_1^*, P_2^*, \ldots, P_{k(n)}^*)$ accepts x with probability at most $1/2^n$, for all provers $P_1^*, P_2^*, \ldots, P_{k(n)}^*$.

A probabilistic, polynomial time TM M which recognizes L works as follows. On input x of length n, M simulates the verifier V to generate the messages $\mathcal{M} = \{m_1, m_2, \ldots, m_{k(n)}\}$, which are to be submitted to $P_1, \ldots, P_{k(n)}$ respectively. Order the message strings in $\mathcal{M}$ using the p-selector f. Then, for all messages $m^* \in \mathcal{M}$, simulate the remaining computation of V assuming that for all strings $m_j \in \mathcal{M}$, $P_j(m_j) = 1$ if and only if $m_j \leq_f m^*$. If in any of these simulations V accepts, then accept, else reject.

Now we show that M is a BPP machine that recognizes L.

Case 1. $x \in L$: By Lemma 1, in one of the simulations the queries are answered honestly. By assumption, this simulation accepts with probability $> (1 - 1/2^n)$, and thus, M accepts x with probability $> (1 - 1/2^n)$.

Case 2. $x \notin L$: M rejects x if and only if all simulations reject x. By assumption, every simulation rejects x with probability $> (1 - 1/2^n)$. It follows that M rejects x with probability $> (1 - 1/2^n)^{k(n)} > 1 - k(n)/2^n$.

This completes the proof of the theorem. □

Corollary 6 *If* NE $\not\subseteq$ BPE, *then there exists a language* $L \in$ NP $-$ BPP *such that* L *belongs to* NIP$(1, poly)$ $-$ NIP$(poly, const)$*. Furthermore, if the honest prover is a competitive prover, then* L *has a 1 round, 1 prover, protocol with 0-error probability.*

Proof This corollary follows from the fact that if search reduces to decision for a language L, then L has a multi-round, one prover interactive proof system in which the honest prover is just L. Thus, the corollary follows by Theorems 8 and 2. □

We conclude that the number of rounds cannot be traded off with the number of provers when honest provers are restricted to being natural. When the computational power of provers is unrestricted such a result is not possible, for every language in PSPACE has a 1 prover polynomial-round interactive proof system [LFKN92, Sha92] and every language in PSPACE can be accepted by 2 provers in 1 round [CCL91].

How do natural proof systems compare with other models in the literature? We identify four classes of languages, and for convenience designate them as I, II, III, and IV. Class I = NIP$(const, const)$ = NIP$(1, 0)$ = BPP. Class II = NIP$(poly, const)$ = NIP$(poly, 1)$. Class III = NIP$(1, poly)$ = compIP. And, class IV = NIP$(poly, poly)$. It is easy to see that I $\subseteq$ II $\subseteq$ III $\subseteq$ IV, and, by Corollary 6, II is properly included in III assuming that NE $\not\subseteq$ BPE. Let compMIP denote the class of languages having competitive multiprover proofs. It is easy to see that NIP$(poly, poly)$ = compMIP.

Furthermore, Bellare and Goldwasser [BG] demonstrate that compMIP is characterized by the class frIP of languages having function-restricted proof systems as defined by Blum and Kannan [BK89], and frIP is identical to the class Check of languages that are checkable. Thus, IV = NIP($poly, poly$) = compMIP = frIP = Check.

4.2 Nonadaptive vs. Adaptive Checkers

The following definition is due to Blum and Kannan [BK89].

Definition 9 *A function f is* checkable *if there is a probabilistic polynomial time oracle machine Ch (called a checker) such that for all programs F (that claim to compute f) and all input words x:*

1. *If for all y, $F(y) = f(y)$, then $Ch^F(x) =$ CORRECT with probability $\geq 2/3$.*
2. *If $F(x) \neq f(x)$, then $Ch^F(x) =$ FAULTY with probability $\geq 2/3$.*

A function is nonadaptively checkable if it has a checker that accesses its oracle nonadaptively. A language is (nonadaptively) checkable if its characteristic function is (nonadaptively) checkable. Whether SAT is checkable is an open question. Here we provide evidence to show that NP contains sets that are checkable but not nonadaptively checkable.

Theorem 9 *If L is nonadaptively checkable and p-selective, then $L \in$ BPP.*

Proof The proof uses techniques that are similar to those used in the proofs of Theorems 1 and 8. Let f be a p-selector for L, and let $Ch = \langle g, e \rangle$ be a nonadaptive checker for L. We assume that the error probabilities on inputs of length n are bounded by $1/2^n$. We will show that there exists a BPP machine M which recognizes L.

M operates as follows: On an input word x of length n, M simulates the condition generator g on input x to obtain a set $Q = \{q_1, q_2, \ldots, q_k\}$ of queries, where $k \leq p(n)$ for some polynomial p. M orders all strings in $Q - \{x\}$ using the p-selector f. Then, M iteratively simulates the condition evaluator e for each string $q^* \in Q \cup \{\perp\} - \{x\}$. In each iteration, M assumes that $L(x) =$"Yes," and for all strings $q \in Q - \{x\}$, M assumes that $L(q) =$"Yes" if and only if $q \leq_f q^*$. If the condition evaluator e outputs $CORRECT$ in any of the simulations, then M accepts x, and otherwise it rejects.

Note that in each simulation of Ch on input x, some program F which claims that $x \in L$ is being checked. By Lemma 1, $x \in L$ if and only if one of these simulations is done with the correct set of answers. That M is a BPP machine recognizing L follows in exactly the same way as in the proof of Theorem 8.

This completes the proof of the theorem. □

Corollary 7 *If* NE $\not\subseteq$ BPE, *then there is a language* $L \in \mathrm{NP} - \mathrm{BPP}$ *that is* not *nonadaptively checkable.*

Theorem 10 *If* $\mathrm{UE} \cap \text{co-UE} \not\subseteq \mathrm{BPE}$, *then there is a language* $L \in \mathrm{UP} - \mathrm{BPP}$ *that is checkable, but* not *nonadaptively checkable.*

Proof By hypothesis, let $T \in \mathrm{UP} \cap \text{co-UP} - \mathrm{BPP}$ be a tally set. Recall from the proof of Theorem 4 that a witness tuple is a string

$$\begin{aligned} s_n &= \sigma_1\sigma_2\cdots\sigma_n, \\ &\quad \text{where for all } i \leq n,\ \sigma_i \text{ is the witness for } 0^i \in T \text{ or for } 0^i \in \overline{T}. \end{aligned}$$

Let $\leq$ denote lexicographic ordering on Σ^*. Define

$$L = \{x \mid \exists s, |s| = |x|, s \text{ is a witness tuple, and } s \leq x\}.$$

(Languages of this form are called "step" languages in [FFLS92].)

For all strings of good length, there is a unique witness tuple which decides membership or nonmembership of $x \in L$. Since $T \in \mathrm{UP} \cap \text{co-UP}$, this witness tuple can be found nondeterministically in polynomial time. Thus, $L \in \mathrm{UP} \cap \text{co-UP}$. Also, if F is a program that claims to decide L, then the unique witness tuple can be found by a binary search algorithm using F as an oracle in polynomial time, and this witness tuple can be verified for validity in polynomial time. Thus, L is checkable. To show that L is *not* nonadaptively checkable, it suffices, by Theorem 9, to show that L is p-selective.

Now we describe a p-selector for L. We say that a natural number m is "good" if m is a good length. Recall that L consists only of strings of good length, and for every good m, L contains at least one string of length m. Note that if s_1 and s_2 are two witness tuples with $|s_1| < |s_2|$, then s_1 is a prefix of s_2. The following algorithm defines a function f, which we will then prove is a p-selector for L:

Input strings x and y. If $|x| = |y|$, then output the lexicographically larger string of x and y. Otherwise, let $x_1 = \min\{x, y\}$, and $y_1 = \max\{x, y\}$. Check whether $|x_1|$ and $|y_1|$ are good. If $|x_1|$ is not good, then output y_1, and if $|x_1|$ is good and $|y_1|$ is not good, then output x_1. If $|x_1|$ and $|y_1|$ are both good, then let v and w be such that $y_1 = vw$ with $|v| = |x_1|$. If $v \leq x_1$, then output x_1, else output y_1.

Obviously, f is polynomial time computable. To show that f is a p-selector for L, it suffices to show for every x and y that if $x \in L$ and $f(x, y) = y$, then $y \in L$.

Suppose that $x \in L$ and $f(x,y) = y$. Then $|x|$ is good, and since the algorithm outputs y, it must hold that $|y|$ is also good. Let m and n be such that $|x| = p(m)$ and $|y| = p(n)$. First suppose that $m = n$. Since $f(x,y) = y$, we have $x \leq y$, and so, by definition of L, $x \in L \rightarrow y \in L$. Since $x \in L$, this implies $y \in L$.

Next suppose that $m < n$. Let u and v be such that $y = uv$ with $|u| = p(m)$. Since $f(x,y) = y$, we have $x < u$. Since $x \in L$, the witness tuple of size $p(m)$ is $\leq x$. Obviously, the witness tuple of size $p(m)$ is a prefix of the witness tuple of size $p(n)$, so the witness tuple of size $p(n)$ is smaller than y, and thus, $y \in L$.

The case $m > n$ follows similarly.

□

5 More about P-selective Sets

5.1 P-Selective Sets, Standard Left Cuts, and Tally Languages

The classical and only previously known method for constructing p-selective sets has been the standard left cut construction of [Sel79, Sel82b]. We begin this section by comparing arbitrary p-selective sets with standard left cuts. If $L(r)$ is a standard left cut and $r \notin 0^*$, then for every length n, $L(r) \cap \Sigma^n \neq \emptyset$. The p-selective set constructed in the proof of Theorem 10 does not have this property, so it is *not* a standard left cut. However, it is not hard to see that all of the p-selective sets constructed in this paper (and in [HHO+92]) are either standard left cuts or $\leq_m^{\mathrm{P}}$-equivalent to some standard left cut. (We do not include proofs of this, but the reader may do so using the techniques of the proof of the next theorem.)

Is there a p-selective set that is not $\leq_m^{\mathrm{P}}$-equivalent to any standard left cut? The following theorem shows that obtaining such a set is as hard as showing that $\mathrm{P} \neq \mathrm{PP}$. We restrict our attention in this theorem to infinite sets because every finite set A belongs to P and, the empty set excepted, is p-selective and $\leq_m^{\mathrm{P}}$-equivalent to every standard left cut that belongs to P.

Theorem 11 *If* $\mathrm{P} = \mathrm{PP}$*, then for every infinite p-selective set* A*, there exists a standard left cut* $L(r)$ *such that* $A \equiv_m^{\mathrm{P}} L(r)$*.*

Proof Assume $\mathrm{P} = \mathrm{PP}$ and let A be p-selective with p-selector f. We assume without loss of generality that for all strings x and y, $f(x,y) = f(y,x)$.

Intuitively, we think of the relation "$f(x,y) = y$" as a weak ordering on strings, and as such, as specifying that y precedes x in this ordering. Although this is not a transitive relation, the justification for this intuition is that for every two distinct strings x and y, either y precedes x or x precedes y. Also, if $x \in A$ and y precedes x, then, since f is a p-selector, $y \in A$ follows. We use this intuition to construct a standard left cut that reflects the structure of A.

Define a function $rank$ such that for every string x, $rank(x)$ is the number of predecessors of x in $\Sigma^{|x|}$. That is,

$$rank(x) = \|\{y \mid |y| = |x| \text{ and } f(x,y) = y\}\|.$$

Note that $1 \leq rank(x) \leq 2^{|x|}$. Observe that $rank$ is computable in polynomial time by a binary search algorithm that accesses the following set C.

$$C = \{\langle x, i\rangle \mid \|\{w \in \Sigma^{|x|} \mid f(w,x) = w\}\| \geq i\}.$$

Let M be a Turing machine that on input $\langle x, i\rangle$ guesses a string y of length $|x|$ and accepts if and only if $f(x,y) = y$. Then $\langle x, i\rangle \in C$ if and only if M on input $\langle x, i\rangle$ has at least i accepting paths. It follows that C is in PP, and thus, by hypothesis, C is in P. Since $rank$ is in PF^C, $rank$ is in PF.

The function $rank$ gives the rank of x in $\Sigma^{|x|}$. (Observe however that distinct strings in $\Sigma^{|x|}$ might have the same rank.) We need to know the rank of x with respect to strings of different length as well. Thus, we define the function $rank(x,m)$ as follows:

$$rank(x,m) = \max\{rank(y) \mid |y| = m \text{ and } f(x,y) = y\}.$$

The function $rank$ has the following property.

Claim 1 *If $|x| \leq |y|$, $rank(x) \leq rank(y, |x|)$, and $y \in A$, then $x \in A$.*

Proof of Claim Suppose $|x| \leq |y|$, $rank(x) \leq rank(y, |x|)$, and $y \in A$. Let y' be a string of length $|x|$ such that $f(y,y') = y'$ and $rank(x) \leq rank(y')$. It is immediate that $y' \in A$.

Let $I = \{z \mid |z| = |x| \text{ and } f(x,z) = z\}$ and $J = \{z \mid |z| = |y'| \text{ and } f(y',z) = z\}$. By the definition of $rank$, $\|I\| \leq \|J\|$. Assume $x \notin A$. Then $x \notin J$ and for every $z \in J$, $f(x,z) = z$, and therefore, $J \subseteq I$. Since $f(x,x) = x$, $x \in I$. Thus, $\|I\| \geq \|J\| + 1$. This is a contradiction. So, $x \in A$.

□

For every i, let $r_i = \max\{rank(y) \mid y \in A, |y| = i\}$. (We define the maximum of the empty set to be 0.)

Claim 2 *$x \in A$ if and only if for every $i, 1 \leq i \leq |x|, rank(x,i) \leq r_i$.*

Proof of Claim First suppose that $x \in A$. Let i be an arbitrary integer such that $1 \leq i \leq |x|$. If $rank(x,i) = 0$, then $rank(x,i) \leq r_i$. If $rank(x,i) > 0$ then, by definition of $rank$, there exists a string y_i of length i such that $f(x, y_i) = y_i$ and $rank(y_i) = rank(x,i)$. It follows from Claim 1 that $y_i \in A$, and thus, $rank(y_i) \leq r_i$. Thus, $rank(x,i) \leq r_i$ as required.

For the converse, suppose that $rank(x) \leq r_{|x|}$. Let $y \in A$ be a string of length $|x|$ such that $rank(y) = r_{|x|}$. Applying Claim 1 gives that $x \in A$. $\square$

Let $i \geq 1$. Since there are 2^i strings of length i, we may write $rank(y)$, for each string y of length i, as a binary string of length $i+1$. (Observe that not all strings of length $i+1$ are in the range of $rank$.) Thus, r_i is a binary string of length $i+1$.

Define the sequence $r = r_1 \cdot r_2 \cdot r_3 \cdots$, and observe that r is infinite. We will show that $A \equiv^{\mathrm{P}}_m L(r)$.

Claim 3 *$A \leq^{\mathrm{P}}_m L(r)$.*

Proof of Claim Observe that $rank(x,i)$ is a binary string of length $i+1$, and define reduction g as follows:

$$g(x) = rank(x,1) \cdot rank(x,2) \cdot rank(x,3) \cdots rank(x,|x|).$$

It is immediate that g is in PF. It remains to show that $x \in A$ if and only if $g(x) < r$. This follows immediately from Claim 2: $x \in A$ if and only if for all $i, 1 \leq i \leq |x|, rank(x,i) \leq r_i$, and, since r is infinite, this is equivalent to $g(x) < r$. $\square$

Claim 4 *$L(r) \leq^{\mathrm{P}}_m A$.*

Proof of Claim Let $y = y_1 \cdots y_m$ be a string such that $|y_i| = i+1$ for $i < m$ and $1 \leq |y_m| \leq m+1$. By definition of $L(r)$, $y \in L(r)$ if and only if

$$[\forall i, 1 \leq i \leq m, y_i \leq r_i] \text{ or } [\exists i, 1 \leq i < m,\ y_i < r_i \text{ and } \forall j, 1 \leq j < i, y_j \leq r_j].$$

First we show how to reduce the following set D to A:

$$D = \{z \mid \exists i \mid |z| = i+1 \text{ and } z \leq r_i\}.$$

Let $|z| = n+1$.

- If $z \in 0^*$, then $z \in D$.

- If $z > max\{rank(x) \mid |x| = n\}$, then $z > r_n$, and thus $z \notin D$.

Under the assumption that P = PP, the condition $z > max\{rank(x) \mid |x| = n\}$ can be tested in polynomial time by a suitable binary search algorithm.

- If $z \notin 0^*$ and $z \leq max\{rank(x) \mid |x| = n\}$, then define $h(z)$ such that:
  - $|h(z)| = n$ and $z \leq rank(h(z))$,
  - for all x of length n, if $z \leq rank(x)$, then $rank(h(z)) \leq rank(x)$, and
  - $h(z)$ is the lexicographically smallest string satisfying these requirements.

We claim that the following holds for each string $z \notin 0^*$:

$$z \leq r_n \text{ if and only if } h(z) \in A.$$

For the right-to-left implication, let $h(z) \in A$. Then

$$z \leq rank(h(z)) \leq max\{rank(x) \mid x \in A \text{ and } |x| = n\} = r_n.$$

For the converse, suppose that $z \leq r_n$. Since $z \notin 0^*$, there must exist a string x such that $z \leq rank(x)$, $x \in A$, and $|x| = n$. By definition of h, $rank(h(z)) \leq rank(x)$, and it follows from Claim 1 that $h(z) \in A$.

Under the assumption that P = PP, it is easy to see that this reduction can be computed in polynomial time by a binary search algorithm that uses the following oracle set, which is in NP, and therefore by assumption in P:

$$\{\langle z, k, x\rangle \mid |x| = |z| - 1 \text{ and } \exists x'[|x'| = |x|, z \leq rank(x'), \text{ and } \langle rank(x'), x'\rangle \leq \langle k, x\rangle]\}.$$

It follows that D is many-one reducible to A in polynomial time. To reduce $L(r)$ to D, observe that for all strings z of length $n + 1$:

$$z < r_n \leftrightarrow (z \neq 1^{n+1} \text{ and } z + 1 \leq r_n).$$

If $z \neq 1^{n+1}$, we can view $z + 1$ as a binary string of length $n + 1$ and therefore:

$$z < r_n \leftrightarrow (z \neq 1^{n+1} \text{ and } z + 1 \in D).$$

It follows immediately that $L(r)$ is positive truth-table reducible to D, and therefore $L(r)$ is also positive truth-table reducible to A.

Although it is easy to transform this reduction into a many-one reduction, there is no need to do because the claim $L(r) \leq_m^{\mathrm{P}} A$ follows immediately from the fact that for any sets X and Y, [Sel82b] if Y is p-selective, and $X \leq_{ptt}^{\mathrm{P}} Y$, then $X \leq_m^{\mathrm{P}} Y$.

This completes the proof of Theorem 11. □

It is known that for every standard left cut $L(r)$, there exists a tally language T, such that $L(r) \leq_{ptt}^{\mathrm{P}} T$, and $T \leq_{\mathrm{T}}^{\mathrm{P}} L(r)$ [Sel82a]. As an immediate corollary, this holds for all p-selective sets if $\mathrm{P} = \mathrm{PP}$.

Corollary 8 *If* $\mathrm{P} = \mathrm{PP}$*, then for every p-selective set* A*, there exists a tally set* T *such that* $A \leq_{ptt}^{\mathrm{P}} T$*, and* $T \leq_{\mathrm{T}}^{\mathrm{P}} A$*.*

The following theorem demonstrates that the known relation between p-selective sets and tally languages is optimal.

Theorem 12 *There exists a standard left cut* $L(r)$ *such that for all tally sets* T*,* $L(r) \not\equiv_{tt}^{\mathrm{P}} T$*.*

Proof Let $M_1, M_2, \ldots$ be an enumeration of all polynomial time truth-table reductions, where $M_i(x)$ runs in time $p_i(|x|) = |x|^i$. We will construct a sequence r and a set $A = L(r)$ such that the following condition holds for all M_i and M_j:

(*) There does not exist a tally set T, such that M_i reduces $L(r)$ to T and M_j reduces T to A.

If this requirement is satisfied for all M_i and M_j, then A cannot be truth-table equivalent to any tally set. The following construction of r demonstrates that the requirement can always be satisfied.

Define $\mu(0) = 2^{2^2}$, $\mu(k) = 2^{2^{\mu(k-1)}}$ for $k > 0$. The sequence r (and equivalently, the set A) is constructed in stages. At stage ℓ, we will decide the $\mu(\ell-1)+1$ through $\mu(\ell)$ bits of r (in other words, we determine membership of strings of length $\mu(\ell-1)+1$ through $\mu(\ell)$ in A). Let r_ℓ denote the bits of r that have been decided prior to stage ℓ, and let A_ℓ denote the finite set of strings put into A prior to stage ℓ. Initially set $r_0 := 0^{2^{2^2}}$, $A_0 := \{0^k \mid 1 \leq k \leq 16\}$, and $m := 0$.

In stage $\ell = \langle i, j \rangle$, set $m := \mu(\ell)$. In this stage, we will show that there does not exist a tally set T such that M_i reduces A to T and M_j reduces T to A. Let $Q = \{0^k \mid 1 \leq k \leq m^i\}$. Define

$$\mathcal{S} = \{x \mid x \text{ is queried to } A \text{ by } M_j \text{ on some input } z \in Q\}.$$

Since M_i runs in time m^i, for any $x \in \Sigma^m$, the question whether $x \in A$ can be answered by determining membership of the elements of Q in some tally set T, which in turn is determined by the membership of strings in $\mathcal{S}$ in A. So, fixing the membership of strings in $\mathcal{S}$ fixes the membership of all strings of length m in A. Now, order the elements in $\mathcal{S}$ using ordinary dictionary ordering. Since A is a standard left cut, there are $\|\mathcal{S}\| + 1$-many consistent ways in which queries in $\mathcal{S}$ can be answered.

Each of these candidate answers determines the membership of all strings in Σ^m, which in turn determines a unique extension of r_ℓ of length m. Let

$$\mathcal{R} = \{r_i^* \mid r_i^* \text{ is an extension } r_\ell \text{ of length } m \text{ due to a candidate answer to queries in } \mathcal{S}\}.$$

The total number of ways in which r_ℓ can be extended is $2^{m-\mu(\ell-1)} > 2^m/m$, and since M_j runs in time m^j, $\|\mathcal{R}\| = \|\mathcal{S}\| + 1 \leq p_j(p_i(m)) + 1 = m^{ij} + 1 < 2^m/m$. Thus there exists an extension r^* of r_ℓ that is not in $\mathcal{R}$. Setting $r_{\ell+1} := r^*$ and $A_{\ell+1} := \{x \mid x \leq r^* \text{ and } |x| \leq |r^*|\}$ satisfies the requirement.

This completes the proof of the theorem. □

Whereas for every tally set T, there exists a standard left cut $L(r)$ such that $T \leq_{\mathrm{T}}^{\mathrm{P}} L(r)$ [Sel82a], it is shown in [HHO+92] that there exists a tally set T that is not $\leq_{tt}^{\mathrm{P}}$-reducible to any p-selective set. Next, under the assumption UE $\neq$ E, we easily show that such a set can be constructed in UP.

Theorem 13 *If* UE $\neq$ E*, then there exists a tally set* $T \in \mathrm{UP} - \mathrm{P}$ *such that* T *is* not $\leq_{tt}^{\mathrm{P}}$*-reducible to any p-selective set.*

Proof The theorem follows directly from Theorem 3 and Proposition 1. Namely, let UE $\neq$ E, so that UP $-$ P contains a tally language T. Define the language:

$$T' = \{0^{\langle n,i,j\rangle} \mid 0^n \in T \text{ and } j \text{ is the } i^{th} \text{ bit of the witness of } 0^n\}.$$

Since we assume that $T \in \mathrm{UP} - \mathrm{P}$, it follows that $T' \in \mathrm{UP} - \mathrm{P}$. This language is identical to the language constructed in the proof of Theorem 3, and it is shown that search nonadaptive reduces to decision for T'. By Proposition 1, if T' is $\leq_{tt}^{\mathrm{P}}$-reducible to a p-selective language, then $T' \in \mathrm{P}$, which contradicts the assumption that $T' \in \mathrm{UP} - \mathrm{P}$. □

5.2 P-selective and P-cheatable sets

The following definitions are due to Beigel [Bei87, Bei91]. For a set A and $n \geq 1$, Φ_n^A denotes the function that given $x_1, \cdots, x_n \in \Sigma^*$ outputs

$$(\chi_A(x_1), \cdots, \chi_A(x_n)).$$

For a set B and $k \geq 1$, let $\mathrm{PF}_{k\text{-}T}^B$ be the class of functions computed by a polynomial time-bounded deterministic oracle Turing machine M via k adaptive queries to B. A set A is called *k-cheatable* if $\Phi_{2^k}^A \in \mathrm{PF}_{k\text{-}T}^B$, for some set B. A set A is called *cheatable* if A is k-cheatable for some k.

Theorem 14 *If* P = NP*, then for every cheatable set A there is a tally set T such that $A \equiv_m^{\mathrm{P}} T$.*

Proof Assume P = NP. Let A be k-cheatable. There is a machine M and an oracle B such that M computes $\Phi_{2^k}^A$ within polynomial-time via k queries to B. Let $m = 2^k$. If $A \in \{\emptyset, \Sigma^*\}$, $\Phi_{2^k}^A \in \mathrm{PF}$ and A is $\leq_m^{\mathrm{P}}$-equivalent to some tally set. So, we assume that A is nontrivial. For a natural number n, let $[n]$ denote $\{1, \cdots, n\}$. Without loss of generality, for every $x_1, \cdots, x_m \in \Sigma^*$, we assume the following:

1. M on $(x_1, \cdots, x_m)$ makes k adaptive queries and outputs a $\{0,1\}^m$-vector;
2. M on $(x_1, \cdots, x_m)$ relative to B outputs $(\chi_A(x_1), \cdots, \chi_A(x_m))$; and
3. for every permutation $\pi : [n] \mapsto [n]$ and for every oracle X, if $M^X(x_1, \cdots, x_m)$ outputs $(b_1, \cdots, b_m)$, then $M^X(x_{\pi(1)}, \cdots, x_{\pi(m)})$ outputs $(b_{\pi(1)}, \cdots, b_{\pi(m)})$.

For $x_1, \cdots, x_m \in \Sigma^*$, let $V(x_1, \cdots, x_m)$ denote the set of all possible outputs of M on $(x_1, \cdots, x_m)$. Since M is making k adaptive queries to its oracle, it holds that

- $V(x_1, \cdots, x_m)$ is polynomial-time computable;
- $\|V(x_1, \cdots, x_m)\| \leq m$; and
- $(\chi_A(x_1), \cdots, \chi_A(x_m)) \in V(x_1, \cdots, x_m)$.

For $q \in [m]$, any $\{0,1\}^q$-vector $v = (b_1, \cdots, b_q)$ and $I \subseteq [q]$, $sub(v, I)$ denotes $(b_{i_1}, \cdots, b_{i_\ell})$, where $i_1, \cdots, i_\ell$ is an enumeration of all elements in I in increasing order. For $q \in [m]$, $x_1, \cdots, x_q \in \Sigma^*$ and $b_1, \cdots, b_q \in \{0,1\}$, we say that $(b_1, \cdots, b_q)$ is *feasible* for $(x_1, \cdots, x_q)$ if for every $x_{q+1}, \cdots, x_m \in \Sigma^{(\leq n)}$, there exists some $v \in V(x_1, \cdots, x_m)$ such that $sub(v, [q]) = (b_1, \cdots, b_q)$, where $n = \max\{|x_1|, \cdots, |x_q|\}$.

By definition, whether $(b_1, \cdots, b_q)$ is feasible for $(x_1, \cdots, x_q)$ is decidable by a predicate in co-NP, and thus, by our assumption P = NP, it is polynomial-time decidable. For $x_1, \cdots, x_q \in \Sigma^*, q \in [m]$, let $H(x_1, \cdots, x_q)$ denote the set of all feasible vectors for $(x_1, \cdots, x_q)$. By a discussion similar to the above, $H(x_1, \cdots, x_q)$ is polynomial-time computable. Note that for every $x_1, \cdots, x_q$,

- $(\chi_A(x_1), \cdots, \chi_A(x_q)) \in H(x_1, \cdots, x_q)$ and
- for every $\ell, 1 \leq \ell \leq q$, and for every $v \in H(x_1, \cdots, x_q)$, $sub(v, [\ell]) \in H(x_1, \cdots, x_\ell)$.

Define $D_0 = \{x \mid H(x) = \{(0)\}\}$, $D_1 = \{x \mid H(x) = \{(1)\}\}$ and $E_0 = \overline{D_0} \bigcap \overline{D_1}$. By definition, $D_0 \subseteq \overline{A}$ and $D_1 \subseteq A$, and since $H(x)$ is polynomial-time computable, $D_0, D_1, E_0 \in \mathrm{P}$. For $n \in \mathbf{N}$, let $E(n, 0) = E_0 \bigcap \Sigma^n$, and define d_n and inductively $w(n, i)$ and $E(n, i)$, $1 \leq i \leq d_n$, as follows:

(Case 1) $E(n,0) = \emptyset$: Define $d_n = 0$.

(Case 2) $E(n,0) = \{a\}$: Define $d_n = 1$ and $w(n,1) = a$.

(Case 3) $\|E(n,0)\| > 1$:

(a) For $i \geq 1$, define $w(n,i) = \min\{x \in E(n,i-1) \mid \|H(w(n,1),\cdots,w(n,i),x)\| \geq i+1\}$ and $E(n,i) = E(n,i-1) - \{w(n,i)\}$; and

(b) Define d_n to be the largest i for which $w(n,i)$ is defined.

It is not hard to see that $d_n \leq m-1$ because $H(x_1,\cdots,x_i)$ has at most m elements. Let x be an arbitrary string in $E(n,d_n)$. By definition, $\|H(w(n,1),\cdots,w(n,d_n))\| \geq d_n+1$ and $\|H(w(n,1),\cdots,w(n,d_n),x)\| \leq d_n+1$. So, $\|H(w(n,1),\cdots,w(n,d_n))\| = \|H(w(n,1),\cdots,w(n,d_n),x)\| = d_n+1$. Moreover, for every

$$v \in H(w(n,1),\cdots,w(n,d_n),x),\ sub(v,[d_n]) \in H(w(n,1),\cdots,w(n,d_n)).$$

So, there is a one-to-one onto correspondence between $H(w(n,1),\cdots,w(n,d_n),x)$ and $H(w(n,1),\cdots,w(n,d_n))$. For each $x \in E(n,0)$, let $h_n(x) = H(w(n,1),\cdots,w(n,d_n),x)$. Then there are at most $2^{d_n} \leq 2^{m-1} \leq 2^{2^k-1}$ different values of $h_n(x)$ and for every x,y with $h_n(x) = h_n(y)$, $x \in A$ if and only if $y \in A$. Now define e_n and $z(n,1)\cdots,z(n,e_n)$ so that

- For each i, $z(n,i)$ is the smallest string in $\{x \in E(n,0) \mid h_n(x) = h_n(z(n,j))$ for every $j < i\}$.
- e_n is the largest i such that $z(n,i)$ is defined.

It is not hard to see that (i) $z(n,i)$ is polynomial-time computable (ii) for every $x \in E(n,0)$, there uniquely exists an i such that $h_n(z(n,i)) = h_n(x)$ and for such i, it holds that $x \in A$ if and only if $h_n(z(n,i))$.

Now define the tally set $T = \{0^{\langle n,i\rangle} \mid z(n,i) \in A\}$. We will show that A and T are $\leq_m^{\mathrm{P}}$-equivalent. Let $a_0 \in \overline{A}$, $a_1 \in A$, $t_0 \in \overline{T}$ and $t_1 \in T$ be fixed. Define $f(x)$ to be

(i) if $x \in D_0$, then $f(x) = t_0$,

(ii) if $x \in D_1$, then $f(x) = t_1$,

(iii) if $x \in E(n,0)$, then $f(x) = z(n,i)$, where i is the number such that $h_n(x) = h_n(z(n,i))$.

By the previous discussions, f is a $\leq_m^{\mathrm{P}}$ reduction of A to T.

On the other hand, define $g(0^{\langle n,i\rangle}) = z(n,i)$ if $z(n,i)$ is defined and a_0 otherwise. Then, obviously, g is a $\leq_m^{\mathrm{P}}$ reduction of T to A. Therefore, A and T are $\leq_m^{\mathrm{P}}$-equivalent, and this proves the theorem. □

Amir, Beigel and Gasarch [ABG] showed that there is a 1-cheatable set that is not p-selective, and, conversely, that there are p-selective sets that are not cheatable. Using our new methods of constructing p-selective sets, the following theorem gives an improvement of their latter result.

Theorem 15 *There is a recursive p-selective tally set that is not cheatable.*

Proof We will construct a p-selective set A which is not k-cheatable for any k. The construction of A is done in stages. Let $M_1, M_2, \cdots$ be an enumeration of all polynomial-time computable functions. We assume that M_i runs in time n^i. Define $\mu(0) = 2^{2^2}$ and for $i \geq 1$, define $\mu(i) = 2^{2^{\mu(i-1)}}$. At stage $\ell = \langle m, i_1, \cdots, i_m\rangle$, $m = 2^k$, we construct against k and machines $M_{i_1}, \cdots, M_{i_m}$ so that Φ_m^A is not computable by these machines; more precisely, we will put some tally strings into A so that the following requirement is established:

(*) there exist tally strings $x_1, \cdots, x_m$ such that $\Phi_m^A(x_1, \cdots, x_m)$ is not outputted by $M_{i_j}(x_1, \cdots, x_m)$ for any $j, 1 \leq j \leq m$.

Suppose that the requirement is satisfied for all ℓ. Then for every m, Φ_m^A is not computable by m functions in PF. Thus, for every k, $\Phi_{2^k}^A$ is not computable by 2^k functions in PF and this implies that A is not cheatable.

The construction at stage $\ell = \langle m, i_1, \cdots, i_m\rangle$ proceeds as follows. Let A_ℓ be the set of all strings put into A prior to stage ℓ. Let $n = \mu(\ell)$. For $q, 1 \leq q \leq m$, let $x_q = 0^{n+q}$. Let $X = \{M_{i_j}(x_1, \cdots, x_m) \mid 1 \leq j \leq m\}$. Then there is some vector $1^d 0^{m-d}$ which is not in X because there are $m+1$ such strings. Let e be the smallest such d. Then we put $x_1, \cdots, x_e$ into A. Obviously, $\Phi_m^A(x_1, \cdots, x_m) \notin X$. So, the requirement (*) is established for ℓ. It is not hard to see that computing X requires $\mathcal{O}(m + \sum_{j=1}^m (n+m)^{i_j})$ operations, so it requires at most $\mathcal{O}(\mu(\ell+1))$.

In order to prove that A is p-selective, we will construct a machine D that computes a p-selector f for A. Let x and y be two given tally strings. M performs the following steps:

(1) Check whether $|x| = \mu(s) + u$, where s is of the form $\langle u, i_1, \cdots, i_u\rangle$. If this is not the case, output y.

(2) Check whether $|y| = \mu(t) + v$, where t is of the form $\langle v, i_1, \cdots, i_v\rangle$. If this is not the case, output x.

(3) If $s < t$, simulate the construction at stage s and output x if x is put into A at stage s and output y otherwise.

(4) If $t > s$, simulate the construction at stage t and output y if y is put into A at stage t and output x otherwise.

(5) If $s = t$ and $u < v$, then output x, and if $s = t$ and $u \geq v$, then output y.

Since the construction at stage s can be done in time $\mu(s+1)$, M runs in time polynomial in $|x| + |y|$. Furthermore, $f(x, y) \in A$ whenever either $x \in A$ or $y \in A$. Therefore, A is p-selective and this proves the theorem. □

6 Open Questions

We conclude with some questions that remain open:

1. If $L \in$ NP and L is self-reducible, does search necessarily reduce to decision for L?
2. Does disjunctive self-reducibility imply random self-reducibility?
3. Does search nonadaptively reduce to decision for SAT?

7 Acknowledgements

The authors gratefully acknowledge comments and suggestions by Richard Beigel, Mihir Bellare, Harry Buhrman, Joan Feigenbaum, Lance Fortnow, Silvio Micali, and Leen Torenvliet.

Some of the results in this paper were first announced in [NOS93], and the original proofs of Theorems 2 and 7 that appear in [NOS93] used the construction given in the proof of Theorem 10 and required the hypothesis NE $\cap$ co-NE $\neq$ E. The improvements given in this version and Corollary 2 are due to Richard Beigel. They appear here with his permission and with our gratitude.

References

[ABG] A. Amir, R. Beigel, and W. Gasarch. Cheatable, P-terse, and P-superterse sets. Manuscript, Dec. 7, 1989.

[AFK89] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *J. Comput. System Sci.*, 39:21–50, 1989.

[AS84] K. Ambos-Spies. P-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines, Lecture Notes in Computer Science 177*, pages 1–23. Springer-Verlag, 1984.

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

[Bal90] J. Balcázar. Self-reducibility structures and solutions of NP problems. *Rivesta Matematica*, 4(4):431–441, Dec. 1990.

[BBFG91] R. Beigel, M. Bellare, J. Feigenbaum, and S. Goldwasser. Languages that are easier than their proofs. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 19–28, 1991.

[BD76] A. Borodin and A. Demers. Some comments on functional self- reducibility and the NP hierarchy. Technical Report TR76-284, Cornell University, Department of Computer Science, Upson Hall, Ithaca, NY 14853, 1976.

[Bei87] R. Beigel. *Query-limited reducibilities.* PhD thesis, Stanford University, 1987.

[Bei91] R. Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theor. Computer Sci.*, 84(2):199–223, 1991.

[Ber78] P. Berman. Relationships between density and deterministic complexity of NP-complete languages. In *Proc. 5th Colloq. on Automata, Languages, and Programming, Lecture Notes in Computer Science*, volume 62, pages 63–71. Springer-Verlag, Berlin, 1978.

[BG] M. Bellare and S. Goldwasser. The complexity of decision versus search. *SIAM J. Comput.* In press.

[BH88] S. Buss and L. Hay. On truth table reducibility to sat and the difference hierarchy over NP. In *Procs. 3rd Annual Conference on Structure in Complexity Theory*, pages 224–233. IEEE Computer Society Press, 1988.

[BK89] M. Blum and S. Kannan. Designing programs that check their work. In *Proc. 21st Symp. ACM Symp. Theory of Computing*, pages 86–97, 1989.

[BLS84] R. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM J. Comput.*, 13(3):461–487, August 1984.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proc. 20th Annual ACM Symp. Theory of Computing*, pages 113–131, 1988.

[Boo74] R. Book. Tally languages and complexity classes. *Info. and Control*, 26:186–193, 1974.

[BvHT93] H. Buhrman, P. van Helden, and L. Torenvliet. P-selective self-reducible sets: A new characterization of P. In *Procs. 8th Annual Conference on Structure in Complexity Theory*, pages 44–51, 1993.

[CCL91] J. Cai, A. Condon, and R. Lipton. PSPACE is provable by two provers in one round. In *Proc. 6th Annual Conference on Structure in Complexity Theory*, pages 110–115. IEEE Computer Society Press, 1991.

[FF91] J. Feigenbaum and L. Fortnow. On the random-self-reducibility of complete sets. In *Proc. 6th Annual Conference on Structure in Complexity Theory*, pages 124–132, 1991. To appear in SIAM J. of Comput., 22(3), 1993.

[FFLS92] J. Feigenbaum, L. Fortnow, C. Lund, and D. Spielman. The power of adaptiveness and additional queries in random-self-reductions. In *Proc. 7th Annual Conference on Structure in Complexity Theory*, pages 338–346, 1992.

[FHOS93] S. Fenner, S. Homer, M. Ogiwara, and A. Selman. On using oracles that compute values. In *Proc. 10th Annual Symp. on Theoret. Aspects of Computer Science, Lecture Notes in Computer Science*, volume 665, pages 398–407. Springer-Verlag, 1993.

[For79] S. Fortune. A note on sparse complete sets. *SIAM J. Comput.*, 8:431–433, 1979.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proc. 3rd Annual Confference on Structure in Complexity Theory*, pages 156–161. IEEE Computer Society Press, 1988.

[GG86] J. Geske and J. Grollmann. Relativizations of unambiguous and random polynomial time classes. *SIAM J. Comput.*, 15(2):511–519, May 1986.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, Feb. 1989.

[GS88] J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–355, April 1988.

[HHO+92] L. Hemachandra, A. Hoene, M. Ogiwara, A. Selman, T. Thierauf, and J. Wang. Selectivity. Manuscript, 1992.

[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, New York, 1990.

[Ko83] K. Ko. On self-reducibility and weak P- selectivity. *J. Comput. System Sci.*, 26:209–211, 1983.

[Ko87] K. Ko. On helping by robust oracle machines. *Theor. Comput. Sci.*, 52:15–36, 1987.

[Kre88] M. Krentel. The complexity of optimization problems. *J. Comput. System Sci.*, 36:490–509, 1988.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, Oct. 1992.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theor. Comput. Sci.*, 1:103–123, 1975.

[Mah82] S. Mahaney. Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis. *J. Comput. System Sci.*, 25:130–143, 1982.

[Mil76] G. Miller. Reimann's hypothesis and tests for primality. *J. Comput. System Sci.*, 13:300–317, 1976.

[MP79] A. Meyer and M. Paterson. With what frequency are apparently intractable problems difficult? Technical Report MIT/LCS/TM-126, M.I.T., 1979.

[NOS93] A. Naik, M. Ogiwara, and A. Selman. P-selective sets, and reducing search to decision vs. self-reducibility. In *Proc. 8th Annual Conference on Structure in Complexity Theory*, pages 52–64. IEEE Computer Society Press, 1993.

[Rub91] R. Rubinstein. Self-P-printability and polynomial time Turing equivalence to a tally set. *SIAM J. Comput.*, 20(6):1021–1033, 1991.

[Sch85] U. Schöning. Robust algorithms: A different approach to Oracles. *Theor. Comput. Sci.*, 40:57–66, 1985.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Math. Systems Theory*, 13:55–65, 1979.

[Sel82a] A. Selman. Analogues of semirecursive sets and effecitve reducibilities to the study of NP complexity. *Inf. Control*, 52(1):36–51, 1982.

[Sel82b] A. Selman. Reductions on NP and P-selective sets. *Theor. Comput. Sci.*, 19:287–304, 1982.

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

[Sel92] A. Selman. A taxonomy of complexity classes of functions. *J. Comput. System Sci.*, 1992. In press.

[Sha92] A. Shamir. IP=PSPACE. *J. ACM*, 39(4):869–877, 1992.

[Thi92] T. Thierauf. Personal Communication, 1992.

[Tod91] S. Toda. On polynomial-time truth-table reducibilities of intractable sets to P-selective sets. *Math. Systems Theory*, 24(2):69–82, 1991.

[WT93] O. Watanabe and S. Toda. Structural analysis of the complexity of inverse functions. *Math. Systems Theory*, 26:203–214, 1993.