A Note on P-selective sets and on Adaptive versus Nonadaptive Queries to NP

Ashish V. Naik* Alan L. Selman†

Abstract

We study two properties of a complexity class $\mathcal{C}$—whether there exists a truth-table hard p-selective language for $\mathcal{C}$, and whether polynomially-many nonadaptive queries to $\mathcal{C}$ can be answered by making $O(\log n)$-many adaptive queries to $\mathcal{C}$ (in symbols, is $\mathrm{PF}_{tt}^{\mathcal{C}} \subseteq \mathrm{PF}^{\mathcal{C}}[O(\log n)]$). We show that if there exists a NP-hard p-selective set under truth-table reductions, then $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$. As a consequence, it follows that if there exists a tt-hard p-selective set for NP, then for all $k > 0$, $SAT \in DTIME[2^{n/\log^k n}]$. We show that if $\mathcal{C} \supseteq \mathrm{ZPP}^{\mathrm{NP}}$, then these two properties are equivalent. Also, we show that if there exists a truth-table complete standard-left cut in NP*, then the polynomial hierarchy collapses to* P^{NP}.

We prove that P = NP follows if for some $k > 0$, the class $\mathrm{PF}_{tt}^{\mathrm{NP}}$ is effectively included in $\mathrm{PF}^{\mathrm{NP}}[k\lceil \log n \rceil - 1]$.

1 Introduction

As NP-hard problems continue to resist polynomial time solutions, researchers in computational complexity have investigated sets for which some partial information can be extracted in polynomial time. The class of p-selective sets, introduced

*Computer Science Department, University of Chicago, Chicago, IL 60637. Research conducted in part at State University New York at Buffalo and supported in part by NSF grant CCR 92-53582. Email: `naik@cs.uchicago.edu`

† Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Research supported in part by the NSF under grant NSF-INT-9123551 and NSF-CCR-9400229. Email: `selman@cs.buffalo.edu`

by Selman [Sel79], is one such example. A set A is p-selective if there is a polynomial time computable function that, given two strings x and y as input, selects one of the two strings such that if any of the input strings is in A, then the selected string is in A.

One important line of research on p-selective sets has been to determine the strongest consequence of NP sets reducing to a p-selective set under various reductions [LLS75]. Selman [Sel82] showed that if there exists a p-selective set that is NP-hard under *positive truth-table* reductions, then $\mathrm{P} = \mathrm{NP}$. Buhrman, Torenvliet and van Emde Boas [BTvEB94] generalized this to show that if there exists a p-selective that is NP -hard under *positive Turing reductions*, then $\mathrm{P} = \mathrm{NP}$. Thierauf, Toda and Watanabe [TTW94] showed that if every set in NP is *bounded truth-table* reducible to a p- selective set, then $\mathrm{NP} \subseteq DTIME[2^{n^{O(1/\sqrt{\log n})}}]$. Agrawal and Arvind [AA94], Beigel, Kummer and Stephan [BKS94], and Ogihara [Ogi94] independently have proved that the existence of a *btt*-hard p-selective set for NP implies $\mathrm{P} = \mathrm{NP}$. Toda [Tod91] proved that if there is a p-selective set that is *truth-table* hard for NP, then $\mathrm{P} = \mathrm{FewP}$ and $\mathrm{RP} = \mathrm{NP}$. Let hypothesis **A** denote the assertion that some p-selective set is truth-table hard for NP.

Hypothesis A. There exists a tt-hard p-selective set for NP.

It is not yet known whether hypothesis **A** implies $\mathrm{P} = \mathrm{NP}$. We study the connection of hypothesis **A** with the following question about function classes [Sel94]: Is the class of functions computable in polynomial time by nonadaptive access to NP included in the class of functions computable by at most $O(\log n)$ queries to some set X as an oracle? Selman [Sel94] showed that if $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$, then $\mathrm{P} = \mathrm{FewP}$ and $\mathrm{RP} = \mathrm{NP}$. Let hypothesis **B** denote the assertion that $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$

Hypothesis B. $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$.

It is not yet known whether hypothesis **B** implies $\mathrm{P} = \mathrm{NP}$. Observe that the known consequences, $\mathrm{P} = \mathrm{FewP}$ and $\mathrm{RP} = \mathrm{NP}$, of hypotheses **A** and **B** are identical. This is not a coincidence, for we prove the following assertion:

Theorem. Hypothesis **A** implies hypothesis **B**[1].

[1]This result has been obtained independently by T. Thierauf [Thi94]. Also, this assertion can be obtained by strengthening results of Beigel [Bei87, Bei88].

Using a result of Jenner and Toran [JT93] and the above theorem, it follows that if every set in NP truth-table reduces to a p-selective set, then for all $k > 0$, $SAT \subseteq DTIME[2^{n/\log^k n}]$.

We do not know whether Hypothesis **B** implies Hypothesis **A**. It is not known, for example, whether assuming both hypotheses implies that P = NP. However, we show that the following assumption, which is presumably stronger than hypothesis B, implies P = NP.

> **Theorem**. If hypothesis **B** is true and there exists a truth-table complete tally set in NP, then $\mathrm{P} = \mathrm{NP}$.

Next, we strengthen hypothesis **A** to obtain a stronger collapse of the polynomial hierarchy than the collapse known by Toda's theorem [Tod91]. That is, recall that $\mathrm{RP} = \mathrm{NP}$ implies a collapse of the polynomial hierarchy to $\mathrm{ZPP}^{\mathrm{NP}}$ because RP has polynomial size circuits [Adl78, KL80, KW94]. In the following theorem we obtain a collapse of the polynomial hierarchy to P^{NP}.

A standard left-cut is a special kind of a p-selective set and is defined in the next section.

> **Theorem**. If there exists a standard left-cut L in NP such that $SAT \leq_{tt}^{\mathrm{P}} L$, then the polynomial hierarchy collapses to P^{NP}.

Hypotheses **A** and **B** are equivalent for higher complexity classes—and equivalent to $\mathrm{NP} = \mathrm{P}$.

> **Theorem**. The following are equivalent.
>
> **(i)** There exists a p-selective that is tt-hard for $\mathrm{ZPP}^{\mathrm{NP}}$.
>
> **(ii)** For all $L \in \mathrm{ZPP}^{\mathrm{NP}}$, there exists a set X such that $\mathrm{PF}_{tt}^{L} \subseteq_c \mathrm{PF}^{X}[O(\log n)]$.
>
> **(iii)** $\mathrm{P} = \mathrm{NP}$

Our final result reports progress on the question of whether hypothesis **B** implies that $\mathrm{P} = \mathrm{NP}$. Krentel showed that if $c < 1$ is an arbitrary constant and if $\mathrm{PF}^{\mathrm{NP}}[m(n)] \subseteq \mathrm{PF}^{\mathrm{NP}}[m(n) - 1]$, for all functions m such that $m(n) < c \log n$, then $\mathrm{P} = \mathrm{NP}$. Beigel, Kummer, and Stephan [BKS94], and Ogihara [Ogi94] independently improved this result by showing that the collapse $\mathrm{P} = \mathrm{NP}$ can be obtained by assuming the weaker hypothesis of $\mathrm{PF}^{\mathrm{NP}}_{m(n)\text{-}tt} \subseteq \mathrm{PF}^{\mathrm{NP}}[m(n) - 1]$ for all $m(n) < c \log n$ for some $c < 1$. We obtain this consequence for a more general class of functions, albeit with a stronger hypothesis. Namely, we assume that the class of partial functions $\mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}}$ is "effectively" included in the class $\mathrm{PF}^{\mathrm{NP}}[k\lceil\log n\rceil - 1]$ for some fixed $k > 0$, and from this assumption we conclude that P = NP. Informally, our assumption stipulates existence of a polynomial time-bounded functional that for every witness to a function belonging to the class on the left-hand side produces a witness to the fact that the function belongs to the class on the right-hand side.

One significance of this result is the novel proof technique that we introduce. The proof techniques of Beigel, Kummer, and Stephan, and Ogihara involve two main steps. First, they use the hypothesis to construct a transducer T that can be used to eliminate possible characteristic vectors for a given set. However this transducer has the restriction that the number of vectors that it outputs is sublinear (that is, $\leq n^c$ for $c < 1$). Then, the transducer T is used to prune the disjunctive self-reducibility tree of *SAT* in polynomial time. The obvious generalization of their technique to remove the restriction that $c < 1$ causes the tree-pruning algorithm to be exponential due to an increase in the input size at each stage of the pruning. In our result, we use a *uniformity* argument that handles this issue by pruning the self-reduction tree without letting the input to each stage grow in size. The effectiveness that we impose is crucial for controlling the uniformity argument. We believe that such a uniformity argument will be essential in proving that either hypothesis **A** or **B** implies that $\mathrm{P} = \mathrm{NP}$.

2 Preliminaries and Notation

All languages are defined over the finite alphabet $\Sigma = \{0,1\}$. We denote by Σ^n the set of all binary strings of length n. We consider $\uparrow$ and # to be special symbols, and assume that there is a polynomial time computable encoding of $\{0,1,\uparrow,\#\}$ into Σ^*. We denote by $\langle,\rangle$ any standard pairing function that is computable and invertible in polynomial time, and by $\cdot$ the concatenation operator. Suppose $S = \{x_1, x_2, \ldots, x_k\}$ is an ordered finite set, and A is any language. Then, $A(S)$ is an abbreviation for the binary string $A(x_1) \cdot A(x_2) \cdots A(x_k)$.

We assume that the reader is familiar with the complexity classes P and NP and with the standard polynomial time reductions among classes [LLS75]. All reducibilities in this paper are assumed to be polynomial time reducibilities. Recall that a language A is *truth-table* reducible to a set B in polynomial time ($A \leq^{\mathrm{P}}_{tt} B$) 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 $x \in A \leftrightarrow e(x, B(q_1), \ldots, B(q_k)) = 1$.

Definition 1 *A set L is* p-selective *[Sel79] 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.

Standard left-cuts are perhaps the most natural examples of p-selective sets [Sel79, HNOS93]. Given a real number r in *dyadic notation* (that is, as an infinite binary string, $r = r_1 r_2 \cdots$ is interpreted as $r = 0.r_1 \cdot r_2 \cdots$), the standard left-cut $L(r)$ of r is defined as the set,

$$L(r) = \{x \mid x \leq r\},$$

where $\leq$ denotes the standard dictionary ordering. It is easy to see that $L(r)$ is p-selective, since a function that, given strings x and y, outputs the smaller string in $\{x,y\}$ according to the dictionary ordering is a selector function for $L(r)$.

We will be referring to the following classes of functions [Sel94, Bei88].

Definition 2 *Suppose $m : N \mapsto N$ and $f : \Sigma^* \mapsto \Sigma^*$ are functions.*

(i) We say that $f \in \mathrm{PF}^{\mathrm{NP}}[m(n)]$ if there exist a polynomial time transducer T and a set $A \in \mathrm{NP}$ such that for all strings x, $T(x)$ computes $f(x)$ by making at most $m(|x|)$ queries to A. We say that $f \in \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$ if there exists a function m such that $m(n) = O(\log n)$ and $f \in \mathrm{PF}^{\mathrm{NP}}[m(n)]$.

(ii) We say that $f \in \mathrm{PF}^{\mathrm{NP}}_{m(n)\text{-}tt}$, if there exist polynomial time computable functions g and e and a set $A \in \mathrm{NP}$ such that for all x, $g(x)$ is a set of queries $Q = \{q_1, q_2, \ldots, q_{m(|x|)}\}$, and $f(x) = e(x, A(q_1), A(q_2), \ldots, A(q_{m(|x|)}))$. We say that $f \in \mathrm{PF}^{\mathrm{NP}}_{tt}$ if there exists a polynomially bounded function m such that $f \in \mathrm{PF}^{\mathrm{NP}}_{m(n)\text{-}tt}$.

3 On *tt*-hard p-selective sets

In this section, we will prove our theorems on the consequence of the existence of truth-table hard p-selective sets. We will utilize the following properties of p-selective sets.

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. For all $x, y \in Q$:

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

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. The following lemma was proved by Toda [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 following lemma is a consequence of Lemma 1.

Lemma 2 *Suppose $A \leq_{tt}^{\mathrm{P}} L$, L is p-selective and $S = \{x_1, x_2, \ldots, x_k\}$ is a finite ordered set of strings. Then, there exists a set $G \subseteq \Sigma^k$ such that G can be computed in polynomial time in $\sum_{x \in S} |x|$ and $A(S) \in G$.*

Proof See Appendix. $\square$

Theorem 3 *If there exists a p-selective set A that is tt-hard for* NP*, then*

$$\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)].$$

Proof Let L be a p-selective set such that for all $X \in \mathrm{NP}$, $X \leq_{tt}^{\mathrm{P}} L$. Let $h \in \mathrm{PF}_{tt}^{\mathrm{NP}}$ and let g and e be polynomial time computable functions such that for each x, $g(x) = \{q_1, q_2, \ldots, q_r\}$ is a set of queries and $h(x) = e(x, A(q_1), A(q_2), \ldots, A(q_r))$. It suffices to show the existence of a polynomial time transducer M and a set B in NP such that M computes h by making at most $O(\log n)$ queries to B.

Since $A \in \mathrm{NP}$, $A \leq_{tt}^{\mathrm{P}} L$. Let $|g(x)|$ denote $|q_1| + |q_2| + \cdots |q_r|$. By Lemma 2, there exists a set $G(x) \subseteq \Sigma^r$ such that $G(x)$ is computable in polynomial time in $|g(x)|$ and $A(g(x)) \in G(x)$. Order $G(x)$ and for each string v in $G(x)$, let $id(v)$ denote the index of v in $G(x)$. We say that a string $v = v_1 v_2 \ldots v_r$ in $G(x)$ is valid if for all $i, 1 \leq i \leq r$

$$v_i = 1 \rightarrow q_i \in A.$$

We are now ready to define B, which is in NP since A is in NP:

$$B = \{ \langle x, g(x), G(x), i\rangle \mid \exists v \in G(x) \\ [id(v) \geq i \text{and } v \text{ is valid}] \}$$

Next, we describe M. On input x, M computes $g(x)$, $G(x)$ and orders the strings in $G(x)$ lexicographically. Next, M performs a binary search procedure by querying to B the strings $\langle x, g(x), G(x), i\rangle$ for $1 \leq i \leq r$. The binary search procedure outputs the largest index ℓ such that for some $v^* \in G(x)$, $id(v^*) = \ell$ and v^* is a valid string. M now computes $e(x, v^*)$ and outputs this value.

We now show that M computes h by making at most $O(\log n)$ many queries to B. By Lemma 2 and the fact that g is a polynomial time nonadaptive oracle transducer, there exists a polynomial p such that $\|G(x)\| \leq p(|x|)$. Since M performs binary search over the range $1 \leq i \leq p(|x|)$, M asks at most $\log(p(|x|)) = O(\log n)$ queries to B. The fact that M computes h follows by the following claim:

Claim 1 *For all x, $A(g(x)) = v^*$.*

Proof of Claim Suppose $v' = A(g(x))$ and that $v' \neq v^* = v_1^* v_2^* \cdots v_k^*$. By Lemma 2, $v' \in G(x)$. We have the following two cases: Firstly, suppose that $v' > v^*$. Since v' is a valid string, the binary search procedure will output v' instead of v^*, which is a contradiction. Now suppose that $v^* > v'$. Then, there exists an index i such that $v_i^* > A(q_i)$, that is, $v_i^* = 1$ and $q_i \notin A$. This implies that v^* is not a valid string, which is a contradiction. □

This completes the proof of the theorem. □

Jenner and Toran [JT93] proved that $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$ implies that for all $k > 0$, $SAT \subseteq DTIME[2^{n/\log^k n}]$.

Corollary 4 *If there exists a p-selective set that is truth-table hard for* NP*, then for all $k > 0$, $SAT \subseteq DTIME[2^{n/\log^k n}]$.*

Next, we show that a somewhat stronger hypothesis than **B** implies a collapse of NP to P.

Theorem 5 *If there exists a tt-complete tally set in* NP *and* $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$, *then* $\mathrm{P} = \mathrm{NP}$.

Proof Let $T \in \mathrm{NP}$ be a tally set such that $SAT \leq_{tt}^{\mathrm{P}} T$ via a reduction f that runs in time $p(n)$. Consider the function h defined as follows: $h(0^n) \mapsto T(0) \cdot T(0^2) \cdots T(0^{p(n)})$. $h \in \mathrm{PF}_{tt}^{\mathrm{NP}}$ and hence there exists a polynomial-time oracle TM M such that M computes h by making at-most $O(\log n)$ adaptive queries to an NP oracle. The set set-M$(x) = \{v \mid v$ is output along some computation path of $M\}$ is computable in polynomial time.

A polynomial-time algorithm A for SAT works as follows: On input x, A computes set-$M(x)$. Then, for all $y \in$set-$M(x)$, A assumes that $h(x) = y$ and simulates the reduction from SAT to T on input x. Since the correct value of $h(x)$ must appear at least once in the simulation, at least one of the simulations will be correct. The algorithm A uses the self-reducibility of SAT to check the correct simulation. For each $y \in$ set-$M(x)$, A will use y to traverse the self-reducibility tree for x. If, at the end of the traversal, A obtains a satisfying assignment of x, then $x \in SAT$. Else, either $x \notin SAT$ or y is the incorrect value of $h(0^n)$. On repeating this process for every y and accepting x if and only if at least one of the simulations generates a satisfying assignment of x, it follows that A accepts x if and only if $x \in SAT$. $\square$

What if we strengthen hypothesis **A** to assume that the tt-hard p-selective set is in NP? In this case, we get an improved collapse of the polynomial hierarchy.

Theorem 6 *If there exists a tt-complete standard left-cut in* NP*, then the polynomial hierarchy collapses to* P^{NP}.

Proof Let $L \in \mathrm{NP}$ be such that L is a standard left-cut and $SAT \leq_{tt}^{\mathrm{P}} L$. Let L be the left-cut of the real number $r = r_0 \cdot r_1 \cdots$. Now, define the tally set T as: for all i, $0^i \in T \iff r_i = 1$. It is easy to see that $L \leq_{tt}^{\mathrm{P}} T$ and $T \leq_{T}^{\mathrm{P}} L$, hence $T \in \mathrm{P}^{\mathrm{NP}}$. It follows by a result of Kadin [Kad87] that if there exists an NP-hard tally set in P^{NP}, then $\mathrm{PH} = \mathrm{P}^{\mathrm{NP}}$, thus the theorem follows. $\square$

Next, we consider hypotheses **A** and **B** for higher complexity classes in the polynomial hierarchy. If we consider classes that contain $\mathrm{ZPP}^{\mathrm{NP}}$, then the hypotheses are equivalent, and, using Toda's theorem [Tod91], equivalent to $\mathrm{NP} = \mathrm{P}$.

Theorem 7 *The following are equivalent.*

(i) *For all languages* $L \in \mathrm{ZPP}^{\mathrm{NP}}$, *there exists a set* X *such that* $\mathrm{PF}_{tt}^{L} \subseteq \mathrm{PF}^{X}[O(\log n)]$.

(ii) *There exists a p-selective set that is tt-hard for* $\mathrm{ZPP}^{\mathrm{NP}}$.

(iii) $\mathrm{P} = \mathrm{NP}$.

Proof Toda showed [Tod91] that (ii) and (iii) are equivalent. Also, it is easy to see that (iii) implies (i). It remains to be seen that (i) implies (ii).

Let us assume that (i) holds. Then, it follows by [Sel94] that $\mathrm{NP} = \mathrm{RP}$, hence $\mathrm{NP} \subseteq \mathrm{P/poly}$, and every set in NP truth-table reduces to some tally set T [KL80]. It is implicit in the Karp and Lipton [KL80] proof that $T \in \Sigma_2^{\mathrm{P}}$. Köbler and Watanabe [KW94] showed that if $\mathrm{NP} \subseteq \mathrm{P/poly}$, then $\mathrm{PH} = \mathrm{ZPP}^{\mathrm{NP}}$, hence $T \in \mathrm{ZPP}^{\mathrm{NP}}$. Let $SAT \leq_{tt}^{\mathrm{P}} T$ via a reduction f that runs in time $p(n)$. Consider the function h defined as follows: for all n, $h(0^n) \mapsto T(0) \cdot T(0^2) \cdots T(0^{p(n)})$. Note that $h \in \mathrm{PF}_{tt}^{\mathrm{ZPP}^{\mathrm{NP}}}$, so by hypothesis, there exists a poly-time oracle TM M and a set X such that h can be computed by M making at-most $O(\log n)$ queries to X as an oracle.

As before, all elements in set-M(x), defined as

$$\{v \mid v \text{ is output along some computation path of } M\}$$

are computable in polynomial time.

We will use Selman's construction [Sel79] of a standard left-cut $L(r)$ such that $T \equiv_T^{\mathrm{P}} L(r)$. Consider the real number $r = T(0) \cdot T(0^2) \cdots$ and the left cut $L(r)$ associated with r, $L(r) = \{x \mid x \leq r\}$, where $\leq$ denotes the standard dictionary order.

We now describe a truth-table reduction g from SAT to $L(r)$. On input x of length n, the reduction g first computes set-$M(x)$. Then it queries all the elements in set-$M(x)$ to $L(r)$. Let $y \in$ set-$M(x)$ be the lexicographically largest string in $L(r)$. Then g simulates the reduction of SAT to T using the string y to answer the queries to T (that is, it assumes that $h(0^n) = y$) and accepts x if and only if the reduction accepts x. We claim that $x \in SAT$ if and only if g accepts x. Ths claim follows by the fact that set-M(x) contains the correct value of $h(0^n)$. More importantly, observe by definition of $L(r)$ that the value of $h(0^n)$ is the lexicographically largest string of length $p(n)$ in $L(r)$. Hence, y is the correct value of $h(x)$, and the simulation is correct. Hence $SAT \leq_{tt}^{\mathrm{P}} L(r)$. □

4 On m Nonadaptive versus $\log m - 1$ Adaptive Queries to NP

The question of whether $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$ implies $\mathrm{P} = \mathrm{NP}$ is still open. In this section, we report progress on this question by showing that if for some constant $k \geq 0$, $\mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}}$ is "effectively" included in $\mathrm{PF}^{\mathrm{NP}}[k\lceil \log n \rceil - 1]$, then $\mathrm{P} = \mathrm{NP}$. Let us consider what this statement means. Let $f \in \mathrm{PF}^{\mathrm{NP}}_{n^k-tt}$. The hypothesis $\mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}} \subseteq \mathrm{PF}^{\mathrm{NP}}[k\lceil \log n \rceil - 1]$ asserts that for every oracle Turing machine M that witnesses $f \in \mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}}$ there is another oracle Turing machine N that witnesses the fact that $f \in \mathrm{PF}^{\mathrm{NP}}[k\lceil \log n \rceil - 1]$. Informally, our assertion is that there is a polynomial time computable effective process T that on input M produces N. More exactly, T is a Turing transducer. Input to T is a pair consisting of code $< M >$ for an oracle Turing machine M and an input string x. M comprises a query generator g and evaluator e. It suffices however to assume that $< M >$, the input to T, is code for the generator only. For, recall that once M knows the correct value of $SAT(q_1) \cdots SAT(q_{n^k})$, where $g(x) = \{q_1, \ldots, q_{n^k}\}$, it can then compute $f(x)$ in polynomial time without further use of its oracle.

The machine N that makes $k \log n - 1$ adaptive queries can output at most $n^k/2$ values over all computation paths. One might think of each of these paths as an attempt to compute the correct value of $f(x)$. However, as we just recalled it suffices to think of each of these paths as an attempt to compute the correct value of the sequence $SAT(q_1) \cdots SAT(q_{n^k})$. Thus, instead of defining T to output N (or all possible output values of N on x) it suffice to define T so that output of T is a set of strings S such that $||S|| \leq n^k/2$ and such that $SAT(q_1) \cdots SAT(q_{n^k}) \in S$.

Finally, we want T to be computable in polynomial time. That is, there is a fixed polynomial p so that for each pair of input strings x and $< M >$, the running time of T is $T_M(x) + p(|x| + |T_M(x)|)$, where T_M is the running time of M. It is necessary to include $T_M(x)$ in order to give T the opportunity to run M once on input x in order to compute the sequence $q_1, \ldots, q_n$.

Thus, we arrive at the following formulation of our theorem.

Theorem 8 *Suppose there exists a constant k and a deterministic transducer T that, given a string x and an encoding $\langle M \rangle$ of a Turing machine as input, runs in time $p(|x| + |\langle M \rangle|) + q_M(|x|)$, where $p(\cdot)$ is an arbitrary fixed polynomial, and $q_M(|x|)$ is the running time of M on input x. The transducer T outputs a set S of strings, such that $||S|| \leq (|x| + |\langle M \rangle|)^k/2$ and the characteristic vector of the output of $M(x)$ in SAT is in S. Then,* $\mathrm{P} = \mathrm{NP}$.

Proof Sketch Let k and T be as described in the hypothesis. We will show that there exists a polynomial time algorithm GENSAT that accepts SAT. On input x, GENSAT generates a satisfying assignment for x (if one exists) by pruning the self-reducibility tree of x. Before presenting the algorithm, we describe the following essential preliminaries.

Let $prefixSAT$ denote the following set in NP:

$$\begin{aligned} prefixSAT &= \{x\#v \mid x \in SAT \text{ and } v \text{ is a prefix} \\ &\quad \text{of a satisfying assignment of } x\} \end{aligned}$$

For every string x (that encodes a satisfiable formula), we will assume without loss of generality that the length of each satisfying assignment of x is $\leq |x|$.

We define the function $next$ for all strings x and v such that $|v| \leq |x|$ by

$$next(x,v) = \begin{cases} 0, & \text{if } x\#v0 \in prefixSAT; \\ 1 & \text{if } x\#v1 \in prefixSAT \\ & \quad \text{and } x\#v0 \notin prefixSAT; \\ \uparrow & \text{if } x\#v \notin prefixSAT. \end{cases}.$$

Suppose $x \in \Sigma^*$ and $S = \{v_1, v_2, \ldots, v_{\|S\|}\}$ is an ordered finite set such that for all $v \in S$, $|v| \leq |x|$. We define the function H by

$$H(x,S) = next(x,v_1) \cdot next(x,v_2) \cdots next(x,v_{\|S\|}).$$

At this point we will sketch the pruning algorithm that is at the heart of our proof. The formal proof is given in the appendix. To simplify this informal description, we assume that $|x|$ is a power of 2 so that $\log |x|$ is an integer. Let $a = k \log |x| - 1$, and let $S_1 = \Sigma^a$. Clearly, if x is satisfiable, then S_1 contains a prefix of a satisfying assignment of x. Observe that $\|S_1\| = |x|^k/2$ and that $H(x, S_1)$ can be computed by making $|x|^k$ nonadaptive queries to NP. Using the hypothesis, on simulating T on inputs x and an encoding of a TM that outputs the set $Y = \{x\#vb \mid b \in \{0,1\}, v \in S_1\}$, T outputs a set of strings S_1' such that $\|S_1'\| \leq |x|^k/2$ such that S_1' contains the characteristic vector of Y. Each string in S_1' is a candidate value of $H(x, S_1)$.

Consider the following procedure $PRUNE$ that takes as input a finite set S and a set of strings S' such that $S \subseteq \Sigma^*$ and $S' \subseteq \{0,1,\uparrow\}^{\|S\|}$, and outputs a finite set $S'' \subseteq \Sigma^*$ such that $\|S''\| \leq \|S'\|$. Assuming that S and S' are ordered, let v_i denote the i^{th} element of S, r^j denote the j^{th} element of S', and r_i^j denote the i^{th} bit of r^j.

begin PRUNE(S, S')

$S'' := \emptyset$;
for $j = 1$ **to** $\|S'\|$ **do**
  if $r^j \notin \{\uparrow\}^*$ **then**
    begin
    find the smallest index ℓ such that $r^j_\ell \neq \uparrow$;
    $S'' := S'' \cup \{v_\ell \cdot r^j_\ell\}$
    end
end PRUNE

Run $PRUNE(S_1, S'_1)$ and let S_2 be the finite set that is output. Then, $\|S_2\| \leq \|S'_1\| \leq n^k/2$. We claim that S_2 contains a prefix of a satisfying assignment of x, if x is satisfiable. To see this, assume that x is satisfiable, let $H(x, S_1) = r$, and note that $r \in S'_1$. Let ℓ be the smallest index such that $v_\ell \in S_1$ is a prefix of a satisfiable assignment of x. Then, $r_\ell \neq \uparrow$, ℓ is the least index such that $r_\ell \neq \uparrow$, and $x\#v_\ell \cdot r_\ell \in prefixSAT$. The procedure $PRUNE$ places $x\#v_\ell \cdot r_\ell$ into S_2. Thus, indeed, S_2 contains a prefix of a satisfying assignment, and this prefix is one bit longer than the strings in S_1. Since $\|S_2\| \leq n^k/2$, we use the hypothesis again, this time on inputs x and an encoding of a Turing machine that outputs S_2, and continue in this manner iteratively until we have obtained a set of strings of length $|x|$, which we then accept if and only if the final set contains a satisfying assignment.

Thus, we see that GENSAT makes iterative calls to PRUNE and then to the transducer T. At each iteration, the input to T is a description of a Turing machine M whose output is the result of the last call to PRUNE. The danger is that the size of the sets S_i grow in size. Therefore, these machines might grow in size, and therefore, so might their descriptions. If this were so, then, even though T runs in time polynomial in the length of its input, T would not run in time polynomial in x. We now show that we can control the size of the descriptions of the Turing machines that are the successive inputs to T. From this it follows that our algorithm for accepting SAT runs in polynomial time.

On input $\langle x, \langle M\rangle\rangle$, let T run in time $p(|\langle x, \langle M\rangle\rangle|) + q_M(|x|)$, where $p(\cdot)$ is a polynomial and $q_M(|x|)$ denotes the running time of M on input x. Let $\langle M\rangle$ denote a description of a transducer M.

We will use the following sequence $\{M_i\}$ of transducers as inputs to T. We define the sequence $\{M_i\}$ by induction. M_0 is a transducer that on input a string y, outputs the finite set $S_1 = \Sigma^a$, where $a = k\lfloor \log |y| \rfloor - 1$. Now we define M_i for $i \geq 1$.

begin description of M_i

input y;
$S'_i := T(y, \langle M_{i-1} \rangle)$;
$S_{i+1} := PRUNE(M_{i-1}(y), S'_i)$;
output S_{i+1}
end description

Of course, the finite control of M_i does not store M_{i-1}. Rather, as with the implementation of any recursive procedure, the finite control of M_i only needs to store the depth of recursion i and the calling procedure. For the latter, the finite control of M_i needs to be able to simulate the procedure *PRUNE* and the transducer T.

Claim 2 *For all i,* $|\langle M_i \rangle| = O(1 + \log i)$.

Proof of Claim Clearly, the machine M_0 executes an algorithm whose description is of length $O(1)$; that is, $|\langle M_1 \rangle| = O(1)$.

For $i \geq 1$, observe that other than the number of recursive calls, the computations performed by M_i and M_{i-1} are identical. Thus, a description of M_i only contains the number i and a code of fixed length. Since i in binary uses $O(\log i)$ bits, the claim follows. □

Let $b = |x| + 1 - (k\lceil \log |x| \rceil)$. It follows immediately from Claim 2 that for all i, $0 \leq i \leq b$, $|\langle M_i \rangle| = O(\log |x|)$ (the constant term in the above expression is independent of i). Now, we are ready to describe the polynomial time algorithm GENSAT for *SAT*. Input to GENSAT is a string x. If for some integer m, $2^m < |x| < 2^{m+1}$, then assume that x is replaced with a logically equivalent string x' so that $|x'| = 2^{m+1}$. Clearly x is satisfiable if and only if x' is satisfiable. Furthermore, by padding machines with "no operation" instructions and by Claim 2, we assume that $|\langle x, \langle M_0 \rangle \rangle| = |\langle x, \langle M_1 \rangle \rangle| = \cdots = |\langle x, \langle M_b \rangle \rangle| = 2|x|$. Thus, for all i, $0 \leq i \leq b$, $|\langle x, \langle M_i \rangle \rangle|$ is a power of 2.

begin GENSAT
input x;
$b := |x| - (k \log |x| - 1)$;
Compute $\langle M_1 \rangle, \langle M_2 \rangle, \ldots, \langle M_b \rangle$ such that
$|\langle x, \langle M_0 \rangle \rangle| = \cdots = |\langle x, \langle M_b \rangle \rangle| = 2|x|$;
$S := M_b(x)$;
for all strings $v \in S$ **do**

if v is a satisfying assignment of x **then**
ACCEPT;
else REJECT
end GENSAT

We claim that GENSAT runs in polynomial time and that GENSAT accepts x if and only if x is satisfiable.

Claim 3 *For all i such that* $0 \leq i \leq |x| - k\log|x| + 1$*,* $M_i(x)$ *runs in time bounded by a polynomial in* $|x|$ *and i.*

Proof of Claim Since M_0 on input x prints all strings of length $k\log|x| - 1$, it is easy to see that M_1 runs in $O(|x|^{2k})$ steps.

Let $t_i(\cdot)$ denote the running time of M_i, and recall that T on input $\langle x, \langle M_{i-1}\rangle\rangle$ runs in time $p(|\langle x, \langle M_{i-1}\rangle\rangle|) + t_{i-1}(|x|)$. Recall that $|\langle x, \langle M_i\rangle\rangle| = 2|x|$. Also, it is easy to see that *PRUNE* runs in time $O(\sum_{v\in S}|v| + \sum_{u\in S'}|u|)$ on input $\langle S, S'\rangle$. It follows by the definition of M_i that

$$\begin{aligned} t_i(|x|) &\leq O(|x|) + p(2|x|) + t_{i-1}(|x|) + \\ & \quad O(\sum_{v\in S_i}|v| + \sum_{u\in S'_i}|u|), \end{aligned}$$

where $S_i = M_{i-1}(x)$ and S'_i is the output of T on input $\langle x, \langle M_{i-1}\rangle\rangle$. (The $O(|x|)$ term in the expression is the time taken to compute $\langle M_{i-1}\rangle$.) To estimate the above inequality, we need to obtain upper bounds on $\sum_{v\in S_{i-1}}|v|$ and $\sum_{u\in S'_i}|u|$.

Recalling that $b = |x| - k\log|x| + 1$, we compute $||S_i||$ and $||S'_i||$ for all i such that $1 \leq i \leq b$. First, for the special case $i = 1$, it holds that $||S_1|| = |x|^k/2$. Now consider the case that $i \geq 2$. By Claim 2, it follows that the length of the input to T is $2|x|$, which is a power of 2, hence $T(x, \langle M_{i-1}\rangle)$ outputs at-most $(2|x|)^k/2$ many candidates. Thus,

$$||S'_i|| \leq (2|x|)^k/2. \tag{1}$$

But $||S_{i+1}|| \leq ||S'_i$, hence for all i, $1 \leq i \leq b$,

$$||S_i|| \leq (2|x|)^k/2. \tag{2}$$

Next, we bound the lengths of strings in S_i and S'_i. Note that the length of each string in S_{i+1} is less than the number of strings in S'_i. By definition of T, the number of strings in S'_i is bounded by $|\langle x, \langle M_i\rangle\rangle|^k/2 \leq (2|x|)^k/2$. Finally, we observe that for all u in S_i, $|u| = i + k\log|x| - 1$.

By substituting from Equations 1 and 2, we have

$$\begin{aligned}\sum_{v\in S_i} |v| &\leq (2|x|)^k/2 \times (i + k\log|x| - 1) \\ &= O((i + k\log|x|)\cdot|x|^k) \end{aligned} \tag{3}$$

$$\begin{aligned}\sum_{u\in S'_i} |u| &\leq (2|x|)^k/2 \times (2|x|)^k/2 \\ &= O(|x|^{2k}) \end{aligned} \tag{4}$$

We now substitute for $t_{i-1}, t_{i-2}, \ldots, t_1$ and p in the relation for t_i.

$$\begin{aligned} t_i(|x|) \leq\ & (i-1)\cdot p(2|x|) + t_1(|x|) + O(i\cdot|x|) + \\ & O\left(\sum_{j=2}^{i}\sum_{v\in S_{j-1}} |v| + \sum_{j=2}^{i}\sum_{u\in S'_j} |u|\right). \end{aligned}$$

On substituting the bound for t_1, and from Equations 3, 4, and noting that $i \leq |x|$, we have:

$$\begin{aligned} t_i(|x|) &\leq |x|\cdot(i-1)\cdot p(2|x|) + O(|x|^{2k}) + \\ &\quad O(|x|^{k+2}) + O(|x|^{2k+1}) + O(|x|^2) \\ &< |x|^{2k}\cdot p(|x|)^2 \end{aligned}$$

Thus for all $1 \leq i \leq b$, $t_b(x)$ is bound by a polynomial in $|x|$. □

Next, we prove the correctness of GENSAT. The proof is identical to that given in the earlier informal description. Before proceeding, observe that by Claim 2 (and by use of padding), it follows for all strings x and for all i that the input $\langle x, \langle M_i\rangle\rangle$ is not infeasible.

Claim 4 *If $x \in SAT$ and $|x|$ is a power of 2, then for all i, $1 \leq i \leq |x| - k\log|x| + 2$ there exists a string v in S_i such that $x\#v \in prefixSAT$.*

Proof of Claim The proof is by induction on i. For $i = 1$, since S_1 contains all strings of length $k \log |x| - 1$, the claim follows trivially. Assume as induction hypothesis that the claim holds for some $i \geq 1$, $S_i = \{v_1, v_2, \ldots, v_t\}$, and let $H(x, S_i) = r$. It follows that $r \in$ set-T$(x, \langle M_i \rangle)$. Let ℓ be the smallest index such that $x \# v_\ell \in prefixSAT$. Then, $r_\ell \neq \uparrow$, ℓ is the least index such that $r_\ell \neq \uparrow$, and $x \# v_\ell \cdot r_\ell \in prefixSAT$. To complete the proof, observe that the procedure *PRUNE* places $x \# v_\ell \cdot r_\ell$ into S_{i+1}. □

Observe that $S_{|x|+2-k\log|x|} \subseteq \Sigma^{|x|}$. By combining Claim 4 with this observation, it follows that if $x \in SAT$, then GENSAT accepts x. If $x \notin SAT$, then a satisfying assignment for x does not exist, and hence the algorithm GENSAT will reject x.

This completes the proof of the theorem. □

5 Acknowledgments

The authors thank Richard Beigel, Mitsunori Ogihara and Ken Regan for helpful discussions and comments. We especially acknowledge Jin-Yi Cai, who suggested using a uniformity argument to obtain a correct proof of Theorem 8.

References

[AA94] M. Agrawal and V. Arvind. Polynomial time truth-table reductions to P-selective sets. In *Proceedings of 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 24–30, 1994.

[Adl78] L. Adleman. Two theorems on random polynomial time. In *Proceedings of 19th IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[Bei87] R. Beigel. A structural theorem that depends quantitavely on the complexity of sat. In *Proceedings of 2nd Annual IEEE Structure in Complexity Theory Conference*, pages 28–34, 1987.

[Bei88] R. Beigel. NP-hard sets are P-superterse unless R = NP. Technical Report 88-04, Department of Computer Science, The Johns Hopkins University, 1988.

[BKS94] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. In *Proceedings of 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 12–23, 1994.

[BTvEB94] H. Buhrman, L. Torenvliet, and P. van Emde Boas. Twenty Questions to a P-selector. *Information Processing Letters*, 48(4), 1994.

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

[JT93] B. Jenner and J. Toran. Computing functions with parallel queries to NP. In *Proceedings of 8th Annual Conference on Structure in Complexity Theory*, pages 280–291, 1993.

[Kad87] J. Kadin. $\mathrm{p}^{\mathrm{np}}[\log n]$ and sparse Turing-complete sets for NP. In *Proceedings of Struct. in Complexity Second Annual Conference*, pages 33–40, 1987.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of 12th ACM Symposium on Theory of Computing*, pages 302–309, 1980. An extended version has also appeared as: Turing machines that take advice, *L'Enseignement Mathématique*, 2nd series 28, 1982, pages 191–209.

[KW94] J. Köbler and O. Watanabe. New collapse consequences of NP having small circuits. Technical Report 94-11, Universität Ulm, November 1994. To appear in ICALP 95.

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

[Ogi94] M. Ogihara. Polynomial-time membership comparable sets. In *Proceedings of 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 2–11, 1994.

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

[Sel82] A. Selman. Analogues of semirecursive sets and effecitve reducibilities to the study of NP complexity. *Information and Control*, 52(1):36–51, 1982.

[Sel94] A. Selman. A taxonomy of complexity classes of functions. *Journal of Computer and System Sciences*, 48(2):357–381, 1994.

[Thi94] T. Thierauf. Private Communication, 1994.

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

[TTW94] T. Thierauf, S. Toda, and O. Watanabe. On sets bounded truth-table reducible to p-selective sets. In *Proceedings of 11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 427–438, 1994.

Appendix

Here we present a proof of Lemma 2.

Lemma 2 *Suppose $A \leq_{tt}^{\mathrm{P}} L$, L is p-selective and $S = \{x_1, x_2, \ldots, x_k\}$ is a finite ordered set of strings. Then, there exists a set $G \subseteq \Sigma^k$ such that G can be computed in polynomial time in $\sum_{x \in S} |x|$ and $A(S) \in G$.*

Proof Suppose f is a p-selector for L and $A \leq_{tt}^{\mathrm{P}} L$ via a truth-table reduction $\langle g, e \rangle$, where g is a condition generator and e is a condition evaluator. For all strings x, $g(x)$ outputs a set $B(x)$ of queries to L. Construct the set $B = \cup_{x \in S} B(x) = \{b_1, b_2, \ldots, b_r\}$, where for each $1 \leq i < j \leq r$, $b_i \leq_f b_j$. Let $B' = \{b_0, b_1, b_2, \cdots, b_r\}$, where $b_0 = \perp$.

The idea is very simple. There are $r+1$ possible pivot strings. Each choice of pivot string determines a possible value of $A(S)$; we define G to be the set of these values. The following algorithm provides the details.

begin
$G := \emptyset$.
for $i = 0$ **to** r **do**
  begin
  { Assume that b_i is the pivot string}
  for $j = 0$ **to** r **do**
    begin
    if $j \leq i$ **then** $v_j := TRUE$;
      else $v_j := FALSE$;

end
for $\ell = 1$ **to** k **do**
begin
$u_\ell = e(x_\ell, v_1, v_2, \ldots, v_r)$.
end
$G := G \cup \{u_1 \cdot u_2 \cdots u_k\}$.
end;
Output G.
end

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 value of $A(S)$. The elements of G are the result of these simulations on elements in S, and hence, $A(S) \in G$. Since g is polynomial-time bounded, the number $\sum_{q \in B} |b|$ is polynomial in $\sum_{x \in S} |x|$, and hence the above algorithm runs in polynomial time. $\square$