On P-selective sets and Adaptive versus Nonadaptive Queries to NP

Jin-Yi Cai * Ashish V. Naik† Alan L. Selman ‡
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260

February 2, 1994

Abstract

We show that if there exists a p-selective set that is NP-hard under truth-table reductions, then every function that is computable in polynomial time by truth-table access to an NP oracle is computable in polynomial time by making at most $O(\log n)$ queries to an NP oracle. As a consequence, it follows that if there exists a tt-hard p-selective set for NP, then for all $k > 0$, $\mathrm{NP} \subseteq DTIME[2^{n/\log^k n}]$.

Reporting progress on the question of whether every function that is computable in polynomial time by truth-table access to an NP oracle is computable in polynomial time by making at most $O(\log n)$ queries to an NP oracle, we show that if there is a constant k such that

$$\mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}} \subseteq \mathrm{PF}^{\mathrm{NP}}[k\lceil \log n\rceil - 1],$$

then P = NP.

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 by Selman [Sel79], is an example of such a class of sets. 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.

*Supported by the National Science Foundation under grant CCR-9057486.
†Supported by the National Science Foundation under grant CCR-9002292.
‡Supported by the National Science Foundation under grant CCR-9002292.

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 P = NP. Buhrman, Torenvliet and van Emde Boas [BTvEB93] generalized this to show that if there exists a p-selective that is NP-hard under *positive Turing reductions*, then P = NP. Toda [Tod91] proved that if there is a p-selective set that is *truth-table* hard for NP, then P = FewP and R = NP. He asked if this assumption would imply the stronger collapse of P = NP. Thierauf, Toda and Watanabe [TTW93] showed that if every set in NP is *bounded truth-table* reducible to a p-selective set, then NP $\subseteq DTIME[2^{n^{O(1/\sqrt{\log n})}}]$. Agrawal and Arvind [AA93], Beigel, Kummer and Stephan [BKS93], and Ogiwara [Ogi93] independently have proved that the existence of a *btt*-hard p-selective set for NP implies P = NP.

Our first result connects the question of whether there exist p-selective tt-hard sets with the following question about function classes [Sel92]: Is the class of functions computable in polynomial time by nonadaptive access to an NP oracle included in the class of functions computable by at most $O(\log n)$ queries to an NP oracle? (in notation, is $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$?). It was shown by Selman [Sel92] that if $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$, then P = FewP and R = NP. Jenner and Toran [JT93] showed that if $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$, then for all $k > 0$, NP $\subseteq DTIME[2^{n/\log^k n}]$. The question of whether $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$ implies P = NP is still open.

Observe that some of the known consequences of the existence of a *tt*-hard p-selective set and $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$ are identical. This is not a coincidence, for we prove the following assertion:

> If there is a p-selective set that is truth-table hard for NP, then $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$[1].

As a corollary, it follows [JT93] that if every set in NP truth-table reduces to a p-selective set, then for all $k > 0$, NP $\subseteq DTIME[2^{n/\log^k n}]$.

We also report our progress on the question of whether $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$ implies P = NP. Beigel, Kummer and Stephan [BKS93], and Ogiwara [Ogi93] independently showed that for all functions $m(n)$ such that $m(n) < c \log n$ for some $c < 1$, if $\mathrm{PF}_{m(n)\text{-}tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[m(n) - 1]$, then P = NP. We obtain this consequence for a more general class of functions, albeit with a stronger hypothesis:

> If there is a constant k such that $\mathrm{PF}_{n^k\text{-tt}}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[k\lceil \log n \rceil - 1]$, then P = 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 $\langle , \rangle$

[1] This result has been obtained independently by T. Thierauf [Thi94]

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 ($A \leq_{tt}^{\mathrm{P}} B$) 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 $x \in A \leftrightarrow e(x, B(q_1), \ldots, B(q_k)) = 1$.

Definition 1 *A set L is* p-selective *(Selman,[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.

We will be referring to the following classes of functions [Sel92, 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 theorem 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:

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

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. 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.*

In our proof, we use the following lemma that 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 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 := ∅.
for i = 0 to r do
      begin
      { Assume that b_i is the pivot string}
      for j = 0 to r do
            begin
            if j ≤ i then v_j := TRUE else v_j := FALSE;
            end
      for ℓ = 1 to k do
            begin
            u_ℓ = e(x_ℓ, v_1, v_2, ..., v_r).
            end
      G := G ∪ {u_1 · u_2 ··· 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. $\square$

Theorem 1 *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. $\square$

This completes the proof of the theorem. $\square$

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

Corollary 1 *If there exists a p-selective set that is truth-table hard for NP, then for all* $k > 0$, $\mathrm{NP} \subseteq DTIME[2^{n/\log^k n}]$.

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

In this Section, we report our progress on the question of whether $\mathrm{PF}_{tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[O(\log n)]$.

Theorem 2 *If there is a constant* k *such that* $\mathrm{PF}_{n^k\text{-}tt}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[k\lceil \log n\rceil - 1]$, *then* $\mathrm{P} = \mathrm{NP}$.

Proof Let k be such that $\mathrm{PF}_{n^k\text{-tt}}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[k\lceil \log n\rceil - 1]$. We will show that there exists a polynomial time algorithm $\mathbf{A}$ that accepts SAT. On input x, $\mathbf{A}$ 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:

$$prefixSAT = \{x\#v \mid x \in SAT \text{ and } v \text{ is a prefix of a satisfying assignment of } x\}.$$

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

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

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 explain the pruning algorithm that is at the heart of our proof. Although the description we give here is not technically correct, it is the essence of the correct proof that will follow. Furthermore, this description will help the reader to understand the reason for the various modifications that we will make. 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. Let us use the hypothesis that $\mathrm{PF}_{n^k\text{-tt}}^{\mathrm{NP}} \subseteq \mathrm{PF}^{\mathrm{NP}}[k\log n - 1]$ in order to infer existence of a transducer T that computes $H(x, S_1)$ by making $k\log n - 1$ queries to NP. (In fact, this is incorrect usage of our hypothesis, for we know only that H on the specific inputs x and S_1 can be computed by making $|x|^k$ nonadaptive queries to NP. We do not know the complexity of computing H on other input pairs. Nevertheless, we ask the readers indulgence, for we will correct this flaw later.) Let S_2' denote the set of output values, ranging over all choices of oracles, of the leaves of the entire computation tree of T on input $\langle x, S_1\rangle$ and observe that $\|S_2'\| \leq 2^{k\log n - 1} = n^k/2$. Consider the following procedure *PRUNE* that takes as input two finite sets S and S' such that $S \subseteq \Sigma^*$ and $S' \subseteq \{0, 1, \uparrow\}^{\|S\|}$, and outputs a finite set S'' 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 $r_\ell^j \neq\uparrow$;
        $S'' := S'' \cup \{v_\ell \cdot r_\ell^j\}$
        end
end *PRUNE*

Run $PRUNE(S_1, S_2')$ and let S_2 be the finite set that is output. Then, $\|S_2\| \leq \|S_2'\| \leq n^k/2$. 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 $r \in S_2'$. 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 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.

There are two problems with the sketch we have just provided. The first problem is that it is not the case that the function $H(x, S)$ is in the class $\mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}}$, where S ranges over all finite sets. Secondly there is a uniformity issue here. It is conceivable that when we iterate the above process, a different T results each time, and the aggregate running time of these T's is no longer polynomial in $|x|$. This requires the following more subtle and indirect approach, which will concern us for the remainder of the proof. Nevertheless, we will use the procedure *PRUNE*.

Let $\langle M\rangle$ denote a description of a transducer M. For a string x and transducer M, we say that the pair $\langle x, \langle M\rangle\rangle$ is *infeasible* if any of the following conditions hold:

(i) $|x|$ is not a power of 2, or

(ii) $|\langle x, \langle M\rangle\rangle|$ is not a power of 2, or

(iii) M does not halt on x in $< |x|^{3k}$ steps, or

(iv) $M(x)$ outputs a set S such that

- $\|S\| > (|x| + |\langle M\rangle|)^k/2$, or
- $|v| > |x|$, for some $v \in S$.

The function F, defined as follows, takes as inputs a string x and a description of a machine M. First $F(x, \langle M\rangle)$ simulates M on x to produce a finite set of queries S, and then it computes $H(x, S)$.

$$F(x, \langle M\rangle) = \begin{cases} 0, & \text{if } \langle x, \langle M\rangle\rangle \text{ is infeasible;} \\ H(x, M(x)), & \text{otherwise.} \end{cases}$$

It is easy to see that $F \in \mathrm{PF}^{\mathrm{NP}}_{n^k\text{-tt}}$, where n is the length of the input to F. Thus, by hypothesis, there exists a transducer T that witnesses $F \in \mathrm{PF}^{\mathrm{NP}}[k\lceil\log n\rceil - 1]$. (Of course, in case $\langle x, \langle M\rangle\rangle$ is not infeasible, n is a power of 2.) Formally, input to T is a pair $\langle x, \langle M\rangle\rangle$, but for readability we will write $T(x, \langle M\rangle)$ rather than $T(\langle x, \langle M\rangle\rangle)$. Let *set-*$T(x, \langle M\rangle)$ denote the set of output values, ranging over all choices of oracle answers to queries of T), of the leaves of the entire computation tree of T on input $(x, \langle M\rangle)$.

We will use the following sequence $\{M_i\}$ of transducers as inputs to F. We define the sequence $\{M_i\}$ by induction. M_1 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 $i > 1$.

begin description of M_i
  input y;
  $S_i' :=$ set-T$(y, \langle M_{i-1}\rangle)$;
  $S_i := PRUNE(M_{i-1}(y), S_i')$;
  output S_i
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_1 executes an algorithm whose description is of length $O(1)$; that is, $|\langle M_1\rangle| = O(1)$.

For $i > 1$, 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 $b = |x| + 2 - (k\lceil\log|x|\rceil)$. It follows immediately from Claim 2 that for all i, $1 \le i \le b$, $|\langle M_i\rangle| = O(\log|x|)$ (the constant term in the above expression is independent of i). Now, we are ready to describe the polynomial time algorithm **A** for SAT. Input to **A** 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 $|\langle x, \langle M_1\rangle\rangle| = |\langle x, \langle M_2\rangle\rangle| = \cdots = |\langle x, \langle M_b\rangle\rangle| = 2|x|$. Thus, for all i, $1 \leq i \leq b$, $|\langle x, \langle M_i\rangle\rangle|$ is a power of 2.

begin A
input x;
$b := |x| - (k \log |x| - 2)$;
Compute $\langle M_1\rangle, \langle M_2\rangle, \ldots, \langle M_b\rangle$ such that
$\quad\quad |\langle x, \langle M_1\rangle\rangle| = |\langle x, \langle M_2\rangle\rangle| = \cdots = |\langle x, \langle M_b\rangle\rangle| = 2|x|$;
$S := M_b(x)$;
for all strings $v \in S$ **do**
$\quad\quad$ **if** v is a satisfying assignment of x **then** ACCEPT;
REJECT
end A

The proof that **A** runs in polynomial time and that **A** accepts x if and only if x is satisfiable follows from the following claims.

Claim 3 *For all strings x such that $|x|$ is a power of 2 and for all i such that $1 \leq i \leq |x| - k\log|x| + 2$, $M_i(x)$ runs in time bounded by a polynomial in $|x|$ and i.*

Proof of Claim Since M_1 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 $t'(\cdot)$ denote the time taken to simulate all computing paths of T. 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

$$t_i(|x|) \leq O(|x|) + t'(2|x|) + t_{i-1}(|x|) + O(\sum_{v\in S_{i-1}} |v| + \sum_{u\in S'_i} |u|),$$

where $S_{i-1} = M_{i-1}(x)$ and $S'_i = \text{set-T}(x, \langle M_{i-1}\rangle)$. (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 $\sum_{u\in S'_i}|u|$.

We compute $\|S_i\|$ for all i such that $1 \leq i \leq |x| - k\log|x| + 2$, and $\|S'_i\|$ for all i such that $2 \leq i \leq |x| - k\log|x| + 2$. First, for the special case $i = 1$, it holds that $\|S_1\| = |x|^k/2$. Now consider the case that $i \geq 2$. Since the length of the input to T is a power of 2, it follows by hypothesis that $\|\text{set-T}(x, \langle M_{i-1}\rangle)\| \leq (2|x|)^k/2$. Thus,

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

By definition of *PRUNE*, for i such that $2 \leq i \leq |x| - k\log|x| + 2$, $\|S_i\| \leq \|S'_i\|$. Thus, for all i, $2 \leq i \leq |x| - k\log|x| + 2$,

$$\|S_i\| \leq \|S'_i\| \leq (2|x|)^k/2.$$

So, for all i, $1 \le i \le |x| - k \log |x| + 2$,

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

Next, we bound the lengths of strings in S_i and S'_i. Observe that for all u in S_i, $|u| = (i-1) + k \log |x| - 1$. We claim that for all $v \in S'_i$, $|v| = O(\|S_{i-1}\|)$. This follows from the fact that the j^{th} bit in v, (which could be $0, 1$ or $\uparrow$) denotes a value of $next(x\#u_j)$, where u_j is the j^{th} element in S_{i-1}. Hence the number of bits in v is bound by the number of elements in S_{i-1}. (Without loss of generality, assume that T is restricted so that on an input pair $\langle x, \langle M\rangle\rangle$ that is not infeasible, T will not write a string on its output tape that is longer than $\|M(x)\|$). By substituting from Equations 1 and 2, we have

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

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

We now substitute for $t_{i-1}, t_{i-2}, \ldots, t_1$ and t' in the relation for t_i.

$$t_i(|x|) \le (i-1) \cdot t'(2|x|) + t_1(|x|) + O\left(\sum_{j=2}^{i} \sum_{v \in S_{j-1}} |v| + \sum_{j=2}^{i} \sum_{u \in S'_j} |u|\right) + O(i \cdot |x|).$$

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

$$t_i(|x|) \le |x| \cdot t'(2|x|) + O(|x|^{2k}) + O(|x|^{k+2}) + O(|x|^{2k+1}) + O(|x|^2).$$

Since T runs in polynomial time and asks at most $O(\log |x|)$ queries, the time taken t' to simulate all paths of T is bound by a polynomial in $|x|$ as well, and the claim follows. $\square$

By combining Claim 3 with the fact that the input to **A** is padded to at most twice its length, we can now assert that **A** runs in polynomial time. Next, we prove the correctness of **A**. 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, $1 \le i \le |x| - k \log |x| + 2$ 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 **A** accepts x. If $x \notin$ SAT, then a satisfying assignment for x does not exist, and hence the algorithm **A** will reject x.

This completes the proof of the theorem. □

5 Acknowledgments

The authors thank Mitsunori Ogiwara and Ken Regan for helpful discussions and comments.

References

[AA93] M. Agrawal and V. Arvind. Polynomial time truth-table reductions to p-selective sets. manuscript, 1993.

[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.

[BKS93] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. manuscript, 1993.

[BTvEB93] H. Buhrman, L. Torenvliet, and P. van Emde Boas. Twenty questions to a p-selector. manuscript, 1993.

[Thi94] T. Thierauf. Personal Communication.

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

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

[Ogi93] M. Ogiwara. Membership comparable sets. manuscript, 1993.

[Sel79] A. Selman. P-selective sets, tally languages, and the behavior of polynomial time reducibilities on NP. *Math. 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.

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

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

[TTW93] T. Thierauf, S. Toda, and O. Watanabe. On sets bounded truth-table reducible to p-selective sets. Technical Report 93-03, Universitat Ulm, Germany, Juni 93. To be presented at STACS 94.