# Membership Comparable and p-selective Sets

Richard Beigel[*]
Lance Fortnow[†]
A. Pavan[‡]

**Abstract**

We construct a 2-membership comparable set that is not btt-reducible to any p-selective set. This is a rare example of an unconditional separation in computational complexity. More generally, for every $c > 0$, we construct a 2-membership comparable set that is not $n^c$-tt-reducible to any p-selective set.

Our construction exploits the fact that the p-selector may be asked to examine inputs of vastly different lengths. We show that this generality is in fact necessary. Consider lengthwise comparators and lengthwise selectors which may only be asked to compare strings of equal length. We show that, given P = PSPACE, every lengthwise 2-membership comparable set is 2-tt reducible to a lengthwise p-selective set. We also show that if P = NP, then every 1-cheatable set is 1-tt reducible to a p-selective set.

## 1 Introduction

Though a language may have difficult instances, we may in some cases efficiently compute relationships between different inputs. Membership comparability, p-selectivity, and cheatability take this approach. Informally, a set $A$ is p-selective, if there is a polynomial-time algorithm that, given two strings $x$ and $y$, tells which string is more likely to be in $A$. A set $A$ is $k$-membership comparable, if given $k$ strings $x_1, x_2, \cdots, x_k$, we can eliminate, in polynomial time one of the $2^k$ possible choices for $A(x_1)\cdots A(x_k)$. A set $A$ is 1-cheatable if given two strings $x$ and $y$, we can output two strings such that one of them is $A(x)A(y)$. P-selective sets are a special case of 2-membership comparable sets; we always exclude one of the two possibilities $(1, 0)$ or $(0, 1)$. Also, every 1-cheatable set is 2-membership comparable. This raises some interesting and fundamental questions — "How broad is the class of 2-membership sets?"; 'What are the relations among 2-membership comparable sets, 1-cheatable sets, and p-selective sets?";

This paper studies these questions. Amir, Beigel, and Gasarch [ABG03] constructed a set that is 1-cheatable but not p-selective; in particular that set is 2-membership comparable but not p-selective. We improve the latter by constructing, for every $c > 0$, a 2-membership comparable set that is not $n^c$-truth table reducible to any p-selective set. It will be hard to improve the former, however, because we show that if P = NP, then every 1-cheatable set is 1-tt reducible to a p-selective set. Thus, showing that 1-cheatable sets form a much larger class than p-selective sets would be difficult.

[*]Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122. Email: beigel@cis.temple.edu

[†]Department of Computer Science, University of Chicago, Chicago, IL 60637. Email: fortnow@cs.chicago.edu

[‡]Department of Computer Science, Iowa State University, Department of Computer Science, 226 Atanasoff Hall, Ames, IA 50011. pavan@cs.iastate.edu

Our construction of 2-membership comparable set exploits the fact that the p-selector may be asked to compare strings of different lengths. We consider *lengthwise* p-selective sets, for which the p-selector is only asked to compare strings of same length. We show that a similar separation between lengthwise p-selective sets and 2-membership comparable sets would imply P $\neq$ PSPACE. Thus we will find it very difficult to prove that there is a lengthwise 2-membership comparable set that is not btt-reducible to any lengthwise p-selective set.

## 1.1 Previous Work

Selman [Sel79] defined p-selective sets as a polynomial-time analogue of semirecursive sets [Joc68]. Beigel [Bei87a] while studying bounded query classes defined $k$-membership comparable sets using the terminology superterse. Beigel, Kummer and Stephan [BKS95] called these sets approximable sets. A set is $k$-membership comparable if and only if it is $k$-approximable if and only if it is not $(k+1)$-superterse. Amir, Beigel, and Gasarch [ABG03] and Ogihara [Ogi95] considered membership comparability for nonconstant functions. From now, we write *$k$-mc sets* for $k$-membership comparable sets.

Membership comparable sets and p-selective sets are studied extensively in the literature [Sel79, Sel82, Ko83, Bei87a, Bei87b, Bei88, ABG03, Tod91, BvHT96, BKS95, Ogi95, AA96, HNOS96]. Ko [Ko83] showed that p-selective sets have polynomial-size circuits. This result was improved by Amir, Beigel, and Gasarch [ABG03] who showed that $O(1)$-mc sets are in P/poly. They also showed that $O(\log n)$-mc sets are in NP/poly and poly-mc sets are in $\Sigma_2$/poly. Recently Beigel [Bei05] has shown that $O(\log n)$-mc sets are in P/poly. Ogihara [Ogi95] claimed that poly-mc sets are in P/poly, but he has since retracted that claim [Ogi05].

Membership comparable sets and p-selective sets have played an important role in understanding the structure of NP. For example, Selman [Sel79] showed that SAT is not p-selective, unless P = NP. Ogihara [Ogi95], Beigel, Kummer, and Stephan [BKS95], and Agrawal and Arvind [AA96] showed that if SAT is $c \log n$-mc, for $c < 1$, then P = NP. This implies that SAT is $n^{o(1)}$-truth-table reducible to a p-selective set if and only if P = NP. It is not known whether the same collapse can be achieved under the hypothesis SAT is $O(\log n)$-mc or under the hypothesis that SAT is truth-table reducible to a p-selective set. Sivakumar [Siv91] showed that if SAT is $O(\log n)$-mc, then UniqueSAT is in P, i.e., there exists a polynomial-time algorithm that distinguishes the cases of a formula having no satisfying assignments from a formula having exactly one satisfying assignment. Valiant and Vazirani [VV86] showed that if UniqueSAT is in P, then NP = RP. Questions regarding reductions to p-selective sets and membership comparable sets have connections with several interesting questions about function classes. See the survey by Buhrman, Fortnow, and Torenvliet [BFT97] for details. Beigel [Bei90] showed that if NP contains a P-bi-immune set then, NP $-$ P contains a 1-cheatable set.

For every 2-mc set, given $n$ strings $n$ strings $x_1, x_2, \cdots x_n$, we can, in polynomial time, compute a set of $n+1$ vectors from $\{0,1\}^n$ that contains $A(x_1)\cdots A(x_n)$ [Bei87a]. Since every p-selective set is 2-mc, p-selective sets also have this property.

It is known that that membership comparable sets form a proper hierarchy, i.e, the class of $k$-mc sets is a proper subset of the class of $(k+1)$-mc sets [ABG03, Ogi95]. Tantau [Tan02] improved this result by showing that for $k \geq 2$, there exists a $(k+1)$-mc set that is not truth-table reducible to any $k$-mc set. His proof uses the fact that the collection of languages consistent with a particular $k$-mc comparator has VC dimension $k-1$. His technique cannot be used to show a similar result about 2-mc set and p-selective sets, because the collection of languages consistent with a particular p-selector also has VC dimension 1.

The rest of the paper is organized as follows. Section 2 contains basic definitions and notation. In Section 3, we show the separation result between 2-mc sets and p-selective sets. In Section 4, we consider lengthwise-selectors, lengthwise-comparators, and 1-cheatable sets.

## 2 Preliminaries

A language $L$ is *p-selective* if there exists a polynomial-time computable function $f : \Sigma^* \times \Sigma^* \to \Sigma^*$ such that for all $x$ and $y$, $f(x, y) \in \{x, y\}$ and if either $x \in L$ or $y \in L$, then $f(x, y) \in L$. We call $f$ a *selector* for $L$.

Well known examples of p-selective are *left-cut sets*. Given an infinite binary sequence $r$, the left cut of $r$ is the set

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

where $<$ is the ordinary dictionary ordering of strings with 0 less than 1. Every left cut is p-selective with selector $f(x, y) = \min(x, y)$.

Let $L$ be a p-selective set with selector $f$. Given any finite set $Q$, we define a "total preorder" $\leq_f$ on $Q$ ($\leq_f$ is total, reflexive and transitive, but is not necessarily asymmetric) as follows:

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

A *p-selective ordering* on a set is an order that is total, reflexive, and transitive that can be computed in polynomial time.

Note that for any finite set $Q$ we can order the elements of $Q$ in time polynomial of $||Q||$. Let $\perp$ be a special symbol such that $\perp \leq_f x$ for all strings $x$. The following property of p-selective sets is useful.

**Lemma 1.** *Let $L$ be a p-selective set with selector $f$. For any finite set $Q$ there exists a string $z \in Q \cup \{\perp\}$ such that $Q \cap \overline{L} = \{y \in Q \mid y \leq_f z\}$ and $Q \cap L = \{y \in Q \mid y \not\leq_f z\}$. The string $z$ is called "cut".*

Let $f$ be a function mapping positive integers to positive integers. A function $g$ is called an $f$-comparator for a set $A$ if for every $x_1, \cdots, x_m$ with $m \geq f(\max\{|x_1|, \cdots, |x_m|\})$,

$$g(x_1, \cdots, x_m) \in \{0, 1\}^m$$

and

$$A(x_1) \cdots A(x_m) \neq g(x_1, \cdots x_m).$$

A set $A$ is called $f$-mc if it has a polynomial-time computable $f$-comparator.

A set $A$ is *1-cheatable* if there exists a polynomial-time computable function $f$ such that, for every $x$, $y$, $f(x,y)$ is a 2-element set that contains $A(x)A(y)$. Note that the value of $f(x, y)$ determines the answer to one of the following questions: $x \in A$?, $y \in A$?, or $(x \in A) \oplus (y \in A) = 1$?.

A set $A$ is *lengthwise p-selective* if there exists a polynomial-time computable function $f$ such that for every $n$, given two strings $x$ and $y$ of length $n$, $f$ eliminates one of two possibilities $(0, 1)$ or $(1, 0)$ for $A$. Similarly, a set is *lengthwise 2-mc* if the comparator $f$ is required to compare strings of same length, i.e., given $x$ and $y$ of same length, $f$ eliminates of the four possibilities for $A(x, y)$.

The characteristic sequence $\chi_L(S)$ of a set $L$ on an ordered set of strings $S = \{x_1, \ldots, x_m\}$ is the binary sequence $a_1 \ldots a_m$ such that $a_i$ is 1 when $x_i$ is in $L$.

We will use the following definitions of Kolmogorov complexity.

Fix a universal Turing Machine $\Phi$. For any string $x$ in $\Sigma^*$, the *Kolmogorov complexity of $x$* is defined as

$$K(x) = \min\{|p| \mid \Phi(p) = x\}.$$

We sometimes consider relative Kolmogorov complexity. $K(x|y)$ is the *Kolmogorov complexity of $x$ relative to a string $y$* where we replace $\Phi(p)$ with $\Phi(p, y)$ in the above definition.

## 3 Separation

In this section we show that 2-mc form a much broader class than p-selective sets. We first consider bounded truth-table reductions.

**Theorem 1.** *There exists a 2-mc set that is not bounded truth-table reducible to any p-selective set.*

*Proof.* We construct the 2-mc language $L$ in stages. Let $f_1, f_2, \cdots$ be an enumeration of p-selectors, and $g_1, g_2, \cdots$ be an enumeration of bounded truth-table reductions. Let $f_i$ can be computed in time $n^i$, and $g_j$ can be computed in time $n^j$. During stage $n = \langle i, j\rangle$, we ensure that $L$ does not reduce to any p-selective set, that has $f_i$ as a selector, via $g_j$.

At any length, we place at most one string in $L$. Initially, at stage 0, L is empty. Fix a pairing function $\langle ., .\rangle$. We define functions $l$ and $r$ as follows: $l(0) = 0, r(0) = 0, l(n) = 2^{r(n-1)}$, , and $r(n) = l(n)^{ij} + 1$, where $n = \langle i, j\rangle$.

*Stage $n = \langle i, j\rangle$ :* $L$ has been defined on all strings of length less than $l(n)$ before this stage. Let $g_j$ be a $k$-tt reduction. Let $S = \{a_1, a_2, \cdots, a_s\}$ be the set of first $s$ strings of length $l(n)$ and $T = \{e_1, e_2, \cdots, e_t\}$ be the set of first $t$ strings of length $r(n)$. We will set the values of $s$ and $t$ later. Let $S'$ be the set of queries generated by $g_j$ on $S$ and $T'$ be the set of queries generated by $g_j$ on $T$.

Let $Q = S' \cup T'$, assume that $Q$ is ordered via the $p$-selector function $f_i$. A *cut* is either a string in $Q$ or the special string $\perp$. Given a cut $c$ define as set $L_c$ as follows:

$$L_c = \{x \in Q \mid x > c\},$$

where $>$ is the order defined by the $p$-selector $f_i$.

Let $g_j = (h_j, m_j)$, where $h_j$ is the query generator and $m_j$ is the truth-table evaluator. Given a cut $c$, and a string $a \in S$,

$$\chi_s^c(a) = m_j(s, L_c(h_j(s))).$$

Similarly define, $\chi_t^c(a)$ for a string $a \in T$. Let $\chi_s^c = \chi_s^c(a_1)\chi_s^c(a_2)\cdots\chi_s^c(a_s)$. Similarly define $\chi_t^c$.

**Remark.** We will view $L_c$ as a p-selective set, and $\chi_s^c$ as the (sub)characteristic sequence of a language defined on $S$ using the reduction $g_j$ along with $L_c$. Similarly $\chi_t^c$ is the characteristic sequence of a language defined on $T$ using the reduction $g_j$ and $L_c$. We will construct $L$ such that for every $c$, either $\chi_L(S) \neq \chi_s^c$ or $\chi_L(T) \neq \chi_t^c$. This ensures that $L$ is not does not reduce to any p-selective set, that has $f_i$ as $p$-selector, via $g_j$. During the construction of the language $L$, we will also ensure that $|L \cap S| = |L \cap T| = 1$.

To define the language $L$ we need a function $f : S \to T$ with following property.

**Property 1.** *i. $f$ can be computed in time polynomial in $r(n)$. Recall that $r(n)$ is the length of strings from $T$.*

*ii. There exist $a \in S$ and $e \in T$ such that $f(a) = e$, and for every cut $c$, if both $\chi_s^c$ and $\chi_t^c$ are vectors with exactly one 1, then at least one of $\chi_s^c(a), \chi_t^c(e)$ is 0.*

We will later show that such a function exists. Assuming such a function exists, we can define $L$ as follows. Let $a, e$ be the smallest pair that satisfies condition 2. Place $a$ and $e$ in L and all other strings of $S$ and $T$ in $\overline{L}$. This completes stage $n$.

We first argue that $L$ is not reducible to any p-selective set $L'$ that has $f_i$ as selector, via $g_j$. Let $L'$ be a p-selective set with $f_i$ as selector. This set $L'$ defines a particular cut $c$ for $Q$. Since, both $\chi_L(S)$ and $\chi_L(T)$ have exactly one 1, if either $\chi_s^c$ or $\chi_t^c$ is not a vector with exactly one 1, then $L$ is not reducible to $L'$ via $g_j$. Assume that bothe $\chi_s^c$ and $\chi_t^c$ are vectors wit exactly one 1. Now, by the second property of $f$, there exists $a$ and $e$ such that either $\chi_s^c(a)$ or $\chi_t^c(e)$ is zero. However, both $\chi_L(a)$ and $\chi_L(e)$ are 1. Thus in this case also $L$ does not reduce to $L'$ via $g_j$.

Next we argue that $L$ is 2-membership comparable. Observe that $L$ has at most one string at any length, and has exactly one string at lengths $l(n)$ and $r(n)$. The membership comparator for $L$ is: If both the strings are of same length output "11", if one string $a$ is of length $l(n)$ and the other string $e$ is of length $r(n)$ then output "10" if $f(a) = e$, and output "11" if $f(a) \neq e$. Note that this can be done in polynomial time as $f$ is computable in polynomial in $|e|$. If one string is of length $l(n)$ (or $r(n)$) and the other string is of length $l(n-1)$ or $r(n-1)$ ( or $r(n-1)$ or $l(n-1)$, respectively), then we can simulate the construction of $L$ and decide the membership of the smaller string in polynomial time. This is possible because $l(n-1)$ and $r(n-1)$ are logarithmic in $l(n)$ and $r(n)$, and the entire construction can be simulated in exponential time.

Now our job is to show that the function $f$ with desired properties exist. Rest of the proof is devoted to this. We first describe a two person game, and then use this game to define $f$.

**Two person game.** Let $S = \{1, 2, \cdots, s\}$ and $T = \{1, 2, \cdots, t\}$. Let $R$ be the real line. We view each member of $S$ and $T$ as a color.

1. Player I partitions $R$ into at most $ks$ disjoint intervals. For each color $i$ in $S$, colors at most $k$ intervals with the color $i$.
2. Player II picks a function $f : S \to T$.
3. Player I partitions $R$ into at most $kt$ disjoint intervals. These intervals could intersect with the intervals picked in the first move. For each color $i$ in $T$, colors at most $k$ intervals with color $i$.
4. Player II picks a color $i$ in $S$.
5. Player I picks a point $p_i$ on the real line $R$.

Player I wins if the $p_i$ is colored with $i$ in the first move and is colored with $f(i)$ in the third move.

We show that for every $k$, there exists appropriate choices of $s$ and $t$ such that the player II has a winning strategy. The proof uses Kolmogorov complexity.

Let $s = am$ and $t = m$. We can assume that the intervals have endpoints at integers from $\{1, 2, \cdots, 2akm\}$. Consider a coloring of the intervals by Player I in the first move. This coloring can described by a string $C_1$ of length $2akm \log am$: for each interval $[i, i+1]$ specify it color. Let $f$ be a string of length $am \log m$ that is Kolmogorov random string *with respect to* $C_1$, i.e, $K(f|C_1) \geq am \log m$.

Assume that Player I has a winning strategy. Thus there exist a coloring of the intervals by Player I, in the third move, such that, for every $i$ in $S$ there exists a point $p_i$ and $p_i$ gets color $i$ in the first move and gets color $f(i)$ in the third move. We can use this to give a shorter description of $f$.

We can describe the coloring of the intervals, in the third move, with $2km \log 2akm$ bits: for each color $i$ in $T$ specify the all intervals that have $i$ as color. Each interval can be described by its both end points, each color appears at most $k$ times and there are $m$ colors. Thus the total number of bits needed are $2km \log 2akm$. Let $C_2$ be string that describes this coloring.

For each $i$ we can compute $f(i)$ if we know $C_2$ and the point $p_i$ that lies in the interval whose color is $i$, in the first move. That is because, since Player I has a wining strategy this point $p_i$ get the color $f(i)$ in third move.

Given $C_1$, we can describe $p_i$ by giving an index among all intervals that are colored $i$. Every color is used for at most $k$ intervals. Thus, we can describe $p_i$ by $\log k$ bits, given $C_1$. Thus the total description of $f$ is at most $2km \log 2akm + am \log 2k$. Note that $f$ is a random string of length $am \log am$. If we choose $a = 8k$ and $m = k^2$, then $f$ can be described by a string whose length is less than $am \log am$. This is a contradiction. Thus Player II has a winning strategy.

Now we use this game to show the existence of the function with Property 1. We recall the earlier scenario here. $S$ is the set of first $8k^3$ strings at length $l(n)$ and $T$ is the set of first $k^2$ strings at length $r(n)$. $S'$ is the set of queries produced by $g_j$ on $S$, and $T'$ is the set of queries produced by $g_j$ on $T$, and $Q = S' \cup T'$. Cardinality of $Q$ is at most $k(8k^3 + k^2)$. Our goal is to exhibit a function $f : S \to T$ and a string $a \in S$ such that for every cut $c$, if both $\chi_s^c$ and $\chi_t^c$ are sequences with exactly one 1, then either $\chi_s^c(a)$ or $\chi_t^c(f(a))$ is zero.

Let $Q = \{a_1, \cdots a_m\}$ where $a_i \leq_{f_i} a_{i+1}$. View strings in $Q$ as point on real line with $a_i$ being integer $i$. Player 1's coloring is now defined as follows: Consider an interval $[b, b+1]$. If there is an unique string $i \in S$ such that $\chi_s^b(i) = 1$, then color the interval $[b, b+1]$ with color $i$. If two adjacent intervals get the same color, then merge them, similarly two adjacent intervals remain uncolored then merger them. Observe that this process defines at most $8k^3 \times k$ intervals. This is also a partial coloring. Suppose an interval $[b, c]$ gets color $i$ and the interval $[c+1, d]$ gets a different color, the it must be the case that $c$ is a query produced on $i$ by $g_j$. Since $g_j$ can produce at most $k$ queries, this partial coloring has the property that any color is used at most $k$ times. Now extend this coloring to a complete coloring, by coloring uncolored intervals appropriately. Since there are at most $8k^4$ intervals and there are $8k^3$ colors available, we can ensure that no color is used for more than $k$ intervals.

Similar process defines Player 1's coloring in Move 3, this time the set $T$ is used instead of set $S$.

Since Player II has a winning strategy, there is a function $f$ and a color $a \in S$ such that for no point $p$, it is colored with color $a$ in first move and color $f(a)$ in third move.

Now consider a cut $c$. Suppose both $\chi_s^c$ and $\chi_t^c$ have exactly one 1. Let $\chi_s^c(i) = 1$ and $\chi_t^c(j) = 1$. This means that there is an interval $[c, d]$ that is colored with color $i$ in first move and an interval $[c, e]$ that is colored with color $j$ in third move. Thus there is a point $p_i$ that gets color $i$ in first move and color $j$ in third move. By the definition of the winning strategy, either $i \neq a$ or $f(i) \neq j$. If $i \neq a$, then $\chi_s^c(a) \neq 1$. This is because $\chi_s^c$ has exactly one 1, and $\chi_s^c(i) = 1$. Similarly, if $i = a$, then $\chi_t^c(f(a)) \neq 1$. Thus $\chi_s^c(a)$ and $\chi_t^c(f(a))$ both are not one.

Now we show that $f$ is computable in polynomial time in $r(n)$. Observe that we can compute the coloring of Player I in the first move by running the reduction $g_j$ on $S$ and running the selector on strings from $S'$. Each string in $S$ is of length $l(n)$, since $g_j$ is $n^j$-time bounded each string in $S'$ is of length $l(n)^j$. There are $8k^3$ strings in $S$, and at most $8k^4$ strings in $S'$. Thus the total time taken to compute this coloring is $8k^3 \times l(n)^j \times 8k^4 \times l(n)^{ij}$. By our choice of $r(n)$, this quantity is less than $r(n)$. Since Player II has a winning strategy, there is a function $f : S \to T$ that beats every move of Player I in step 3. Since $|S| = 8k^3$ and $|T| = k^2$, there are at most $k^{16k^3}$ functions from $S$ to $T$. In the third move, Player 1 uses $k$ colors to color at most $8k^4$ intervals. There at most $k^{8k^4}$ such colorings. Since $k$ is a constant the number of functions as well as the number of colorings are constant. We can cycle through all functions and colorings and pick a function that beats all colorings in the third move. This process again takes constant amount of time. Thus the total time taken by $f$ is bounded by polynomial in $r(n)$.

Thus there exists a function $f$ with Property 1.

□

The above proof can be extended to $n^c$-truth-table reductions.

**Theorem 2.** *For any* $c > 0$*, there exists a 2-mc set that is not* $n^c$*-truth-table reducible to any p-selective set.*

The proof is runs similar to the previous theorem. The main difference is, in the first moves of the game Player I picks $l(n)^c s$ intervals and each color is used for $l(n)^c$ intervals. In the third move, Player I picks $r(n)^c t$ disjoint intervals. As before, there exists a function, a Kolmogorov random function, for which Player II has a winning strategy. However, we cannot cycle through all function from $S$ to $T$ as there are too many functions. However, it can be shown that a greedy algorithm for $f$ works.

## 4 Collapse

In this section, we consider lengthwise-selectors and lengthwise-comparators. We show that the proof of the previous section cannot be extended to the case of lengthwise-selectors, without separating P from PSPACE.

**Theorem 3.** *If* P = PSPACE*, then every lengthwise 2-mc set is 2-tt reducible to a lengthwise p-selective set.*

To prove Theorem 3 we first we 1-tt reduce 2-mc sets to an intermediate type we call tree-selective and then show that every tree-selective set 2-tt reduces to a p-selective set.

Call a set $S$ *tree-selective* if $S$ has a 2-comparator $g$ such that for all $x$ and $y$, $g(x, y) \neq 00$.

The proof of Theorem 3 follows immediately from the following two lemmas.

**Lemma 2.** *If P = PSPACE then every lengthwise 2-mc sets is 1-tt reducible to a lengthwise tree-selective set.*

**Lemma 3.** *If P = PSPACE, then every lengthwise tree-selective set* $V$ *is 2-tt reducible to a lengthwise p-selective set.*

**Proof of Lemma 2:**

Let $L$ be a lengthwise 2-mc set and $f$ be the 2-comparator. Our goal is to construct a lengthwise tree-selective set $V$ such that $L$ 1-tt reduces to $V$.

Fix an input length $n$ and an ordering of the inputs of length n. Our 1-tt reduction will just be the identity function. We will use Boolean variables $u_x$ for each $x$ that will indicate whether the reduction is positive or negative, i.e.,

1. If $u_x$ is true then $x$ is in $L$ if and only if $x$ is in $V$.

2. If $u_x$ is false then $x$ is in $L$ if and only if $x$ is not in $V$.

For a setting of the $\vec{u}$ we define $V_{\vec{u}}$ to fulfill the above items. The $\vec{u}$ also gives a 2-comparator for $V$, $g_{\vec{u}}(x_1, x_2)$, defined as follows. Let $f(x_1, x_2) = b_1b_2$. We define $g_{\vec{u}}(x_1, x_2) = a_1a_2$ by $a_i = b_i$ if $u_{x_i}$ is true and $a_i = 1 - b_i$ if $u_{x_i}$ is false.

Ideally we would like to find a $\vec{u}$ such that $g_{\vec{u}}$ is a tree-selector, i.e., $g_{\vec{u}}(x, y) \neq 00$ for all $x$ and $y$ of length $n$. Unfortunately this is not possible in general. Instead we will find a $\vec{u}$ such that an easily computable modification of $g_{\vec{u}}$ will be a tree-selector. We show, given P = PSPACE, how to do this in polynomial-time.

Consider a graph of $2^{n+1}$ nodes labeled $x$ and $\overline{x}$ for all $x$ of length $n$. We add directed edges to the graph according the the possibilities that the comparator eliminates.

1. If $f(x, y) = 00$ then add $\overline{x} \to y$ and $\overline{y} \to x$.

2. If $f(x, y) = 01$ then add $\overline{x} \to \overline{y}$ and $y \to x$.

3. If $f(x, y) = 10$ then add $x \to y$ and $\overline{y} \to \overline{x}$.

4. If $f(x, y) = 11$ then add $x \to \overline{y}$ and $y \to \overline{x}$.

These arrows indicate implications derived from $f$.

We define the set $T$ as all of the strings $x$ of length $n$ that fulfill one of the following

1. There is a path from $x$ to $\overline{x}$. ($x$ must not be in $L$)

2. There is a path from $\overline{x}$ to $x$. ($x$ must be in $L$)

3. There is a $y < x$ such that there is a path from $x$ to $y$ and from $y$ to $x$. ($x$ is in $L$ if and only if $y$ is in $L$)

4. There is a $y < x$ such that there is a path from $x$ to $\overline{y}$ and from $\overline{y}$ to $x$. ($x$ is in $L$ if and only if $y$ is not in $L$)

Let $T_i$ be the set of $x$ that fulfill case $i$ but no earlier case. $T_1$, $T_2$, $T_3$ and $T_4$ partition $T$. The elements of $T_i$ can be computed in PSPACE and thus in P by assumption. Let $S$ be $\Sigma^n - T$.

**Claim 1.** *There is a setting of the $u_x$ for $x$ in $S$ such that $V_{\vec{u}}$ is tree-selective via $g_{\vec{u}}$ on the elements of $S$.*

*Proof.* Initially set the $u_x$ to be true for all $x$ in $S$. Repeat the following until we have fulfilled Claim 1: Pick the smallest $x$ such that there is a $y < x$ with $g_{\vec{u}}(y, x) = 00$. Set $u_x$ to be false.

Suppose that when we set $u_x$ to be false to get $\vec{u'}$, we now have $g_{\vec{u'}}(z, x) = 00$ for some $z < x$. By case analysis of $g_{\vec{u}}(yz)$ we show this cannot happen.

1. $g_{\vec{u}}(yz) = 00$: This violates the fact that $x$ was smallest.

2. $g_{\vec{u}}(yz) = 01$: This forces $y$ in $V_{\vec{u}}$ which forces $y$ either in $L$ if $u_y$ is true and $y$ out of $L$ if $u_y$ is false. Either way we have $y$ in $T$ contradicting the fact that $y$ is in $S$.

3. $g_{\vec{u}}(yz) = 10$: This forces $z$ in $V_{\vec{u}}$ leading to a similar contradiction.

4. $g_{\vec{u}}(yz) = 11$: This forces $V_{\vec{u}}(y) \neq V_{\vec{u}}(z)$. This means we will either have $L(y) = L(z)$ (if $u_y \neq u_z$) or $L(y) \neq L(z)$ (if $u_y = u_z$). Either way we will have the greater of $y$ and $z$ in $T$.

□

Define the set $U$ as the set of $x$ in $S$ such that $u_x$ is true. The set $U$ is 2-mc: Given $x$ and $y$ in $S$, we know that $g_{\vec{u}}(xy) \neq 00$ and this eliminates a possibility for the $u_x$ and $u_y$.

Amir, Beigel, and Gasarch [ABG03] showed that every $k$-mc set has polynomial-size circuits. So there must be some polynomial-size circuit $C$ that computes $U$.

In PSPACE we can search all possible circuits to find one that gives $U$ such that $g_{\vec{u}}(x, y) \neq 00$ for all $x$ and $y$ in $S$. We can now define the 1-tt reduction via $\vec{u}$ and our tree-selector function $g_{\vec{u}}(x, y)$. For $x$ in $S$ we use the $\vec{u}$ we found from the circuit. If $x$ is in $T_1$ set $u_x$ be true and if $x$ is in $T_2$ set $x$ to be false.

If $x$ is in $T_3$ or $T_4$ let $w_x$ be the smallest $w$ such that either

1. There is a path from $x$ to $w$ and from $w$ to $x$, or

2. There is a path from $x$ to $\overline{w}$ and from $\overline{w}$ to $x$.

Note that $w_x < x$ and $w_x$ is not in $T_3$ or $T_4$ or $w_x$ would not be smallest. Let $u_x = u_{w_x}$ if the first case occurs, otherwise let $u_x = \neg u_{w_x}$.

We define the tree-selector $g(x, y)$ for $x < y$ as follows.

1. If $x$ and $y$ are in $S$, define $g(x, y) = g_{\vec{u}}(x, y)$, otherwise

2. If $x$ or $y$ is in $T_1$ or $T_2$ define $g(x, y) = 11$, otherwise

3. If $x$ is in $T_3$ or $T_4$ define $g(x, y) = g(w_x, y)$, otherwise

4. If $y$ is in $T_3$ or $T_4$ define $g(x, y) = g(x, w_y)$.

□

**Proof of Lemma 3:** Again, we concentrate on strings of length $n$. For every $x$, we define $x \to x$. A pair $(x, y)$ is a *maximal nand* if $nand(x, y)$, and if $x \to x'$ and $y \to y'$, provided either $x \neq x'$ or $y \neq y'$, then we do not have $nand(x', y')$. If $x \to x'$ we say $x$ is *below* $x'$ and $x'$ is *above* $x$.

Note that if there is a "nand", then there is a maximal "nand". If there is no "nand", then we can easily define a p-selective ordering among strings of length $n$.

In the following assume $(x, y)$ is a maximal nand.

**Claim 2.** *If $z$ is above $x$, then $z$ is above $y$.*

*Proof.* Let $x \to z$ and $nand(x, y)$. Consider the relation between $y$ and $z$. If $z \to y$, then we have $x \to y$. Since $x \to y$ and $nand(x, y)$ implies $x$ is false, this cannot have happened. Hence, assume $nand(y, z)$. However, by definition $y \to y$. Thus if $nand(y, z)$, then $(x, y)$ cannot be a maximal nand. The only remaining possibility is $y \to z$. □

Consider the following sets.

$$W = \{w \mid (x, w) \text{ is a maximal nand}\} \cup \{x\},$$

$$\forall w \in W, Below(w) = \{z \mid z \to w\} - \{w\},$$

and

$$Above = \{z \mid y \to z\} - \{y\}.$$

**Claim 3.** *There is a p-selective ordering among strings from Above.*

*Proof.* Consider two strings $z_1$ and $z_2$ from *Above*. By definition, they are above $y$. Every element above $y$ is also above $x$, by Claim 2. Since $(x, y)$ is a maximal nand, we do not have $nand(z_1, z_2)$. Thus either $z_1 \to z_2$ or $z_2 \to z_1$. This gives a p-selective ordering. □

Now we are ready to define a p-selective ordering on $\{0,1\}^n$. Strings in $W$ are ordered in arbitrary manner, we can chose lexicographic ordering. Strings in *Above* are ordered as per the existing p-selective ordering. Order the elements of $Below(w)$, for each $w$, recursively. This process defines a tree, where $w$ is ancestor of strings of $Below(w)$, and $w$ is a descendant of the smallest string in *Above*. For $w_1, w_2 \in W$, if $w_1 < w_2$ then the subtree containing $w_1$ is to the left of the subtree containing $w_2$. Now, consider a post-order traversal on the tree. This is our p-selective ordering. Let $v$ be the least string in the order that belongs to the tree-selective set $V$. Then the p-selective set is $V' = \{v' \mid v \leq v'\}$. Note that given two elements, they can be compared in polynomial time as the tree traversal can be done in logarithmic space in the size of the tree, and because of the assumption that P = PSPACE. Consider the following 2-tt reduction from $V$ to $V'$: First note that, if $w$ is an ancestor of $w'$ and $w'$ belongs to $V$, then $w$ also belongs to $V$. So, if $w$ belongs to $V$, either $w$ or some descendant of $w$ belong to $V$. Thus, $w$ belongs to $V'$. Hence we can conclude that if $w$ does not belong to $V'$, then $w$ does not belong to $V$. On the other hand, it is possible that that $w$ belongs to $V'$, but $w$ does not belong to $V$. This could happen only if the left sibling of $w$ belongs to $V'$. Thus a second query checks whether the left sibling of $w$ belong to $V'$.

This proves the lemma. □

We next consider 1-cheatable sets.

**Theorem 4.** *If* P = NP*, every one cheatable set is 1-tt reducible to a p-selective set.*

*Proof.* Let $L$ be the given 1-cheatable set. There exists a comparator $f$, for any two strings $x$ and $y$, by looking at the output of $f(x, y)$ we can either decide the membership of $x$ in $L$, or decide the membership of $y$ in $L$ or compute $(x \in L) \oplus (y \in L)$.

We first decide the membership of all strings that can be decided easily, under the assumption P = NP: Given $x$ we ask whether there exists a $y$ such that $f(x, y) \in \{\langle 00, 01\rangle, \langle 10, 11\rangle\}$. And also ask whether there exists $y$ such that $f(y, x) \in \{\langle 00, 10\rangle, \langle 01, 11\rangle\}$. If the answer to either of the questions is "Yes", then we can easily decide the membership of $x$.

At each length let $S_n \subseteq \Sigma^n$ be the set of the strings whose membership cannot be easily decided by the above process. If $S_n$ is empty at that length we are done. Otherwise let $x_n$ be the lexicographically least string of $S_n$. Define the p-selective set $S$ to be the left cut of the infinite sequence $L(x_1)L(x_2)\cdots$. We now define the reduction from $L$ to $S$: On input $x$, of length $n$, first decide whether the membership in $L$ can be decided easily. If not using P = NP compute $x_0, x_1, \cdots x_n$. If we know $L(x_n)$, then we can easily decide the membership of $x$ in $L$ by computing $f(x, x_n)$. Amir, Beigel, and Gasarch [ABG03] showed that if $L$ is 1-cheatable then given any $n$ strings $x_0, \cdots x_n$, in polynomial time, we can compute two strings $r$ and $s$ such that one of them is equal to $L(x_0)L(x_1)\cdots L(x_n)$. Let $p$ be the longest common prefix of $r$ and $s$. We can decide the correct sequence, by asking whether $p1$ belongs to $S$. This tells the value of $L(x_n)$. □

## 5 Acknowledgments

We thank Leen Torenvliet for many helpful comments on an earlier draft.

## References

[AA96] M. Agrawal and V. Arvind. Quasi-linear truth-table reductions to P-selective sets. *Theoretical Computer Science*, 158:361–370, 1996.

[ABG03] A. Amir, R. Beigel, and W. Gasarch. Some connections between bounded query classes and non-uniform complexity. *Information and Computation*, 186:104–139, 2003.

[Bei87a] R. Beigel. *Query-limited reducibilities.* PhD thesis, Stanford University, 1987.

[Bei87b] R. Beigel. A structural theorem that depends quantitatively on the complexity of sat. In *Proc. 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.

[Bei90] R. Beigel. Bi-immunity results for cheatable sets. *Theoretical Computer Science*, 73(3):249–263, 1990.

[Bei05] R. Beigel. On greed, apprximation, and approximable sets. Submitted to CCC 2006.

[BFT97] H. Buhrman, L. Fortnow, and L. Torenvliet. Six hypotheses in search of a theorem. In *Proceedings of 12th Annual Conference on Computational Complexity*, pages 2–12, 1997.

[BKS95] R. Beigel, M. Kummer, and F. Stephan. Approximable sets. *Information and Computation*, 120(8):304–314, 1995.

[BvHT96] H. Buhrman, P. van Helden, and L. Torenvliet. P-selective self-reducible sets: A new characterization of P. In *Journal of Computer and System Sciences*, 53(2): 44–51, 1996.

[HNOS96] E. Hemaspaandra, A. Naik, M. Ogiwara, and A. Selman. P-selective sets and reducing search to decision vs. self-reducibility. *Journal of Computer and System Sciences*, 53(2):194–209, 1996.

[Joc68] C. Jockusch, Jr. Semirecursive sets and positive reducibility. *Trans. of the Amer. Math. Soc.*, 131(2):420–436, 1968.

[Ko83] K. Ko. On self-reducibility and weak P-selectivity. *Journal Computer and System Sciences*, 26:209–211, 1983.

[Ogi95] M. Ogihara. Polynomial-time membership comparable sets. *SIAM Journal on Computing*, 24(5):1168–1181, 1995.

[Ogi05] M. Ogihara. Personal Communication.

[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 effective reducibilities to the study of NP complexity. *Information and Control*, 52(1):36–51, 1982.

[Siv91] D. Sivakumar. On membership comparable sets. *Journal of Computer and System Sciences*, 59(2):270–280, 1991.

[Tan02] T. Tantau. A note on the power of extra queries to membership comparable sets. Technical Report TR02-004, ECCC, 2002.

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

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.