Electronic Colloquium on Computational Complexity, Report No. 63 (2013)

Non-autoreducible Sets for NEXP

Dung T. Nguyen* Alan L. Selman †

April 19, 2013

Abstract

We investigate autoreducibility properties of complete sets for NEXP under different polynomial reductions. Specifically, we show under some polynomial reductions that there is are complete sets for NEXP that are not autoreducible. We obtain the following results:

- There is a $\leq^p_{tt}$-complete set for NEXP that is not $\leq^p_{btt}$-autoreducible.
- For any positive integers s and k such that $2^s - 1 > k$, there is a $\leq^p_{s-T}$-complete set for NEXP that is not $\leq^p_{k-tt}$-autoreducible.
- There is a Turing complete set for NEXP that is not $\leq^p_{tt}$-autoreducible.
- For any positive integer k, there is a $\leq^p_{k-tt}$-complete set for NEXP that is not weakly $\leq^p_{k-tt}$-autoreducible.
- There is a $\leq^p_{3-tt}$-complete set for NEXP that is not $\leq^p_{3-tt}$-autoreducible, given that the autoreduction cannot be allowed to ask a query too short or too long.
- Relative to some oracle, there is a $\leq^p_{2-T}$-complete set for NEXP that is not $\leq^p_T$-autoreducible.
- Relative to some oracle, there is a $\leq^p_m$-complete set for NEXP that is not $\leq^p_{NOR-tt}$-autoreducible.

We will show that settling the question whether every $\leq^p_{dtt}$-complete set for NEXP is $\leq^p_{NOR-tt}$-autoreducible either positively or negatively would lead to major results about the exponential time complexity classes.

1 Introduction

Autoreducibility was first introduced by Trakhtenbrot [9] in a recursion theoretic setting. A set A is autoreducible if A is reducible to A via an oracle Turing machine M such that M never queries x on input x. Ambos-Spies [1] introduced the polynomial-time variant of autoreducibility, where the oracle Turing machine now runs in polynomial time. In this paper, we focus on the autoreducibility in this setting. Moreover, each notion of reduction induces the corresponding notion of autoreducibility.

The question of whether complete sets for various classes are polynomial-time autoreducible has been studied extensively. Over many years, many results about the autoreducibility of complete sets of different classes have been discovered. Glaßer et al. [5] showed that all m-complete sets of the following complexity classes are many-one autoreducible: NP, PSPACE, EXP, NEXP, $\Sigma^{\mathrm{P}}_{\mathrm{k}}$, $\Pi^{\mathrm{P}}_{\mathrm{k}}$, and $\Delta^{\mathrm{P}}_{\mathrm{k+1}}$ for $k \geq 1$. Beigel and Feigenbaum [8] showed that Turing complete sets for the classes that form the polynomial-time hierarchy, $\Sigma^{\mathrm{P}}_{\mathrm{k}}$, $\Pi^{\mathrm{P}}_{\mathrm{k}}$, and $\Delta^{\mathrm{P}}_{\mathrm{k}}$, are Turing

*University at Buffalo, The State University of New York, dtn3@buffalo.edu
†University at Buffalo, The State University of New York, selman@buffalo.edu

ISSN 1433-8092

autoreducible. Also, all Turing complete sets for NP are Turing autoreducible. So studying autoreducibility is an interesting research area and in some cases, it turns out that resolving some open questions would lead to major class separation results. Buhrman et al. [2] proved various autoreducibility results for many different complexity classes and demonstrated strong evidence that studying the structural property of the complete sets, especially the autoreducibility property, might be an important tool to separate the complexity classes. For example, if there exists a Turing complete set of NEXP that is not Turing autoreducible, then EXP is different from NEXP.

With this motivation in mind, we study autoreducibility questions for NEXP. Buhrman et al. [2] extensively studied autoreducibility for EXP. Especially, for EXP, it is known that under many-one, 1-tt, 2-tt, and Turing reductions, all complete sets for EXP are autoreducible. Also for any $k \geq 3$, under $\leq^p_{k-tt}$-reduction, there exists a complete set for EXP that is not autoreducible. For NEXP, it is known that all many-one complete sets are autoreducible. Moreover, Glaßer et al. [4] took a next step to show that under 2-tt, disjunctive-truth-table, and conjunctive-truth-table reductions, all complete sets for NEXP are autoreducible. We make progress in this paper by proving non-autoreducibility of complete sets for NEXP under some polynomial-time reductions. In particular, we will show that

- there is a $\leq^p_{tt}$-complete set for NEXP that is not $\leq^p_{btt}$-autoreducible.
- there is a $\leq^p_T$-complete set for NEXP that is not $\leq^p_{tt}$-autoreducible.
- there is a $\leq^p_{3-tt}$-complete set for NEXP that is not poly-honest $\leq^p_{3-tt}$-autoreducible.[1]
- for any positive integer k, there is a $\leq^p_{k-tt}$-complete set for NEXP that is not weakly $\leq^p_{k-tt}$-autoreducible.

This paper is organized as follows. Section 2 contains notation and definitions about many different polynomial-time reductions and autoreducibilities. In section 3, the diagonalization technique is extensively exploited together with other tricks to obtain the non-autoreducibility results for many different complete sets in NEXP. We prove in Section 4 that settling the question whether every $\leq^p_{dtt}$-complete for NEXP is $\leq^p_{NOR-tt}$-autoreducible would lead to important results about the exponential time complexity classes. In section 5, we will show negative results in relativized worlds for some open questions.

2 Preliminaries

Most of the notations and definitions are standard[7]. Strings are elements of $\{0,1\}^*$. For every string x, denote $|x|$ to be a length of x. For every Turing machine M, $L(M)$ denotes the language accepted by the machine M. We denote M^A to be an oracle Turing machine M that accesses to the oracle B. Also for every input x, denote $M(x)$ to be the outcome of the computation of M on input x; i.e., $M(x) = 1$ if and only if M accepts input x. We assume that the pairing function $\langle \ldots \rangle$ is a one-to-one, polynomial computable function that can take any finite number of inputs and its range does not intersect with 0^*. For every set A, the characteristic function of A is denoted by A; that is, $A(x) = 1$ if $x \in A$ and $A(x) = 0$ otherwise. Also $|A|$ denotes the cardinality of A.

For any two sets A and B, A is Turing-reducible to B in polynomial time, $A \leq^p_T B$, if there exists a deterministic polynomial-time-bounded oracle Turing machine M such that $A = L(M^B)$. In this paper, if we do not mention explicitly the running time of a reduction, then

[1] All definitions to follow.

that reduction is a polynomial time reduction. The reduction is nonadaptive, $A \leq^p_{tt} B$, if the queries are independent of the oracle and so they do not depend on the answers to the previous queries. Other notions of reductions are also considered. A set A is k-truth-table-reducible to B, $A \leq^p_{k-tt}$, if there exists a nonadaptive oracle Turing machine M^B that accepts A such that for any input x, the computation of M^B on input x asks no more than k queries. A set A is bounded-truth-table-reducible to B, $A \leq^p_{btt} B$, if there exists some integer k such that $A \leq^p_{k-tt} B$. A set A is disjunctive-truth-table reducible to B in polynomial time, $A \leq^p_{dtt} B$, if there exists a polynomial computable function f such that for any x, $f(x) = \langle q_1, \ldots, q_k \rangle$, and $x \in A \iff A(q_1) \vee \cdots \vee A(q_k) = 1$. Similarly, a set A is conjunctive-truth-table reducible to B in polynomial time, $A \leq^p_{ctt} B$ if there exists a polynomial computable function f such that for any x, $f(x) = \langle q_1, \ldots, q_k \rangle$, and $x \in A \iff A(q_1) \wedge \cdots \wedge A(q_k) = 1$. Other notions $\leq^p_{k-dtt}$ and $\leq^p_{k-ctt}$ are defined analogously. For any k-ary Boolean function α, a set A is α-truth-table reducible to B in polynomial time, $A \leq^p_{\alpha tt} B$, if there exists a polynomial time computable function f such that for any x, $f(x) = \langle q_1, \ldots, q_k \rangle$, and $x \in A \iff \alpha(A(q_1), \ldots, A(q_k)) = 1$.

For any oracle Turing machine M^B, let $Q(M^B, x)$ denote the set of all queries of the computation of M^B on input x.

EXP $= \bigcup\{\text{DTIME}(2^{p(n)}) | p \text{ is a polynomial}\}$ is the class of languages that can be decided by a deterministic Turing machine in exponential time

NEXP $= \bigcup\{\text{NTIME}(2^{p(n)}) | p \text{ is a polynomial}\}$ is the class of languages that can be decided by a nondeterministic Turing machine in exponential time

Definition 2.1. Autoreducibility *For any reduction $\leq$, a set A is $\leq$-autoreducible if $A \leq A$ via an oracle Turing machine M^A such that for any x, $x \notin Q(M^A, x)$. We call M an autoreduction of A by $\leq$-reduction. The reduction $\leq$ can apply to any reductions, specifically, all those that we mention above, such as $\leq^p_T, \leq^p_{tt}, \leq^p_{k-tt}, \leq^p_{dtt}$, etc.*

Definition 2.2. Polynomial-honest reduction *Given any two sets A and B and an arbitrary number c, we define a truth-table polynomial honest reduction $\leq^p_{tt-h-c}$ as follows: $A \leq^p_{tt-h-c} B$ if there exists a nonadaptive Turing machine M with the oracle B such that M^B accepts x if and only if $x \in A$ and for any input x, all queries q made to the oracle B have length satisfying $|x|^{1/c} \leq |q| \leq |x|^c$*

Definition 2.3. NOR-reduction *Given any two sets A and B, we define a NOR-truth-table reduction $\leq^p_{NOR-tt}$ as follows: $A \leq^p_{NOR-tt} B$ if there exists a nonadaptive Turing machine M with the oracle B such that for any input x, letting $q_1, \ldots, q_k$ be all queries of M^B on input x, then $x \in A \iff q_1 \notin B \wedge \cdots \wedge q_k \notin B$.*

Definition 2.4. Weak-reduction *Given any two sets A and B, we define a weak truth-table reduction $\leq^p_{tt-w}$ as follows: $A \leq^p_{tt-w} B$ if and only if there exist two polynomial computable functions f and g such that for any input x, $f(x) = \langle q_1, \ldots, q_k \rangle$, $g(x) = h(\alpha_1, \ldots \alpha_k)$ is a boolean function such that h is neither an OR nor a NOR boolean function, and $x \in A \iff h(B(q_1), \ldots, B(q_k)) = 1$.*

3 Non-autoreducible sets for NEXP

Buhrman and Torenvliet [3] showed that there is a $\leq^p_{tt}$-complete set for EXP that is not $\leq^p_{btt}$ autoreducible. Homer [6] and Watanabe[10] obtained several results about separating the notions of reductions in complexity classes EXP and NEXP. Based on their ideas, we have the following autoreducibility results for NEXP.

Theorem 3.1. *There is a $\leq^p_{tt}$-complete set for* NEXP *that is not $\leq^p_{btt}$ autoreducible.*

Algorithm 1 Thm 3.1. B's construction at stage n. It will encode all strings x of length between $y_{n-1}^{n-1}+1$ and y_n^n such that $x \in K$ into B. Also diagonalize against M_m

Compute m and k such that $n = \langle m, k \rangle$
Compute $Q \leftarrow Q(M_m^B, 0^{y_n})$
$f \leftarrow$ true ▷ Boolean variable to determine whether the diagonalization step is needed here
if $|Q| > k$ **then**
    $Q \leftarrow \emptyset$
    $f \leftarrow$ false ▷ Because the number of queries is more than k, no diagonalization is needed
for every string x, $y_{n-1}^{n-1} < |x| \leq y_n^n$ **do** ▷ Encoding K into B
    for every string j, $|j| = \lceil \log |x| \rceil$ **do**
        if $\langle x, j \rangle \notin Q$ and $x \in K$ **then**
            $B \leftarrow B \cup \{\langle x, j \rangle\}$
if f is true **then**
    if (M_m^B rejects 0^{y_n}) **then**
        $B \leftarrow B \cup \{0^{y_n}\}$ ▷ Diagonalize against M_n using the string 0^{y_n}

Proof.

We assume a polynomial-time computable one-to-one pairing function that can take any finite number of inputs such that its range does not intersect with 0^*.

Let $\{M_j\}_{j\geq 1}$ be an enumeration of all polynomial-time bounded truth-table reductions.

Let $\{\text{NEXP}_i\}_{i\geq 1}$ be an enumeration of all nondeterministic exponential time oracle Turing machines.

For each $j \geq 1$, assume that the computations of M_j and NEXP_j on input x have the running times that are bounded by $|x|^j$ and $2^{|x|^j}$, respectively.

Let $K = \{\langle i, x, l \rangle \mid \text{NEXP}_i \text{ accepts input } x \text{ within } l \text{ steps}\}$ be a canonical complete set for NEXP.

We will construct a set B with the following property

$x \in K \iff$ there exists a string $j, |j| = \lceil \log |x| \rceil$ and $\langle x, j \rangle \in B$

which ensure that $K \leq_{tt}^p B$, and we need $B \in \text{NEXP}$. So B is $\leq_{tt}^p$-complete set for NEXP.

We also need a set B such that for any $n \geq 1$, the following property holds

$0^{y^n} \in B \iff M_m^B$ rejects input 0^{y^n} (value of y_n and m will be chosen later in the proof) which ensures that M_n is not an autoreduction of B. Then we can conclude that B is not autoreducible for NEXP.

We will construct the set B in stages. In each stage, we will encode K into B and diagonalize all $\leq_{btt}^p$-reductions using the string 0^{y_n} simultaneously to obtain those above two properties.

Before going into detail of how the set B is constructed, let's define the sequence $\{y_n\}_{n\geq 0}$ such that $y_0 = 1$ and $y_{n+1} = 2^{y_n^n} + 1$ for every $n \geq 0$.

The set B is constructed in each stage as follows.

- Initially we set $B = \emptyset$.
- At stage n, suppose that the set B has been already constructed such that all strings x of length up to y_{n-1}^{n-1} and $x \in K$ are all encoded into B appropriately. In this stage, we will encode all strings of length between $y_{n-1}^{n-1}+1$ and y_n^n. The Algorithm 1 describes how B is constructed in this stage.

Lemma 3.1. $K \leq_{tt}^p B$.

Proof.

Refer to the Algorithm 1, for all strings x such that $x \notin K$, for every string j of length $\lceil \log |x| \rceil$, the string $\langle x, j \rangle$ is not added to B. So to prove $K \leq^p_{tt} B$, we just need to prove that for every string $x \in K$, one of strings $\langle x, j \rangle$, $|j| = \lceil \log |x| \rceil$, is added to B.

For any string x in K, denote n be the integer such that $y^{n-1}_{n-1} < |x| \leq y^n_n$. We have the following facts:

$2^{\lceil \log |x| \rceil} \geq |x| > y^{n-1}_{n-1} \geq n \geq k \geq |Q|$

So there exists a string j of length $\lceil \log |x| \rceil$ such that $\langle x, j \rangle \notin Q$. Refer to the line 9 in the Algorithm 1, the string $\langle x, j \rangle$ is added to B and then satisfies the truth-table reduction from K to B

So $K \leq^p_{tt} B$. □

Lemma 3.2. $B \in$ NEXP

Proof.

The Algorithm 2 decides B correctly and it is consistent to the B's construction in the Algorithm 1.

Now we will analyze the running time of the Algorithm 2

Running-time Analysis:

Notice that in the Algorithm 2, most of the steps can easily be done in exponential time. We will consider some interesting steps below:

- Line 17: Notice that checking whether $q \in B$ can be done nondeterministically in $2^{|x|}$ where $q = \langle x, j \rangle$. So it can be checked deterministically in no more than $2^{2^{|x|}}$. Also $|x| \leq y^{n-1}_{n-1}$, so the running time will be at most $2^{2^{y^{n-1}_{n-1}}} < 2^{y_n} = 2^{|b|}$.
- Line 30 and Line 38: It can be done nondeterministically in exponential time in terms of $|x|$ by simulating the Turing machine to accept K on input x.

So $B \in$ NEXP. □

Lemma 3.3. *B is not $\leq^p_{btt}$-autoreducible.*

Proof. Suppose that B is $\leq^p_{btt}$-autoreducible, then there is some number m such that $B \leq^p_{btt} B$ by the autoreduction M^B_m. Because M_m is $\leq^p_{btt}$-reduction, denote k be a number such that for any input x, the number of queries made in the computation of M^B_m on input x is no more than k. Then consider the computation of M^B_m on input 0^{y_n}, where $n = \langle m, k \rangle$. By B's construction in the Algorithm 1, M^B_m accepts 0^{y_n} if and only if $0^{y_n} \notin B$. But that contradicts to the fact that M_m is the reduction from B to itself.

So B is not $\leq^p_{btt}$-autoreducible. □

Conclusion: By Lemma 3.1, Lemma 3.2, and Lemma 3.3, B is the $\leq^p_{tt}$-complete set for NEXP that is not $\leq^p_{btt}$-autoreducible. □

As we can see the strategy used in the above proof, for every string x in K that we want to encode into B, we can encode by one of the strings $\langle x, 0 \rangle$, $\langle x, 1 \rangle$, etc. It is easy to see that in the above proof we have more options to encode than the number of queries of M_n on input 0^{y_n}; i.e. there exists an i such that $\langle x, i \rangle$ is not a query and then we can encode x in K by $\langle x, i \rangle$ in B and so it does not affect the computation of M^B_n on input 0^{y_n}. In the following theorem,

Algorithm 2 Thm 3.1. Algorithm to decide B

Require: An input string b
Ensure: Return YES if $b \in B$. Otherwise, return NO

if $(b = 0^*)$ **then**
  Compute n such that $|b| = y_n$
  if no n exists **then**
    Return NO
  else
    Compute integers m and k such that $n = \langle m, k \rangle$
    Compute $Q \leftarrow Q(M_m^B, 0^{y_n})$
    if $|Q| > k$ **then**
      Return NO
    else
      Simulate the Turing machine M_m^B on input 0^{y_n}
      for every query q **do**
        Compute x and j such that $q = \langle x, j \rangle$. ▷ If no x and j are found, the answer is NO
        if $|x| > y_{n-1}^{n-1}$ **then**
          Answer NO
        else
          Call recursively this function to check whether $q \in B$.
      if M_m^B accepts 0^{y_n} **then**
        Return NO
      else
        Return YES
else
  Compute x and j such that $b = \langle x, j \rangle$, $|j| = \lceil \log |x| \rceil$
  if no such x and j **then**
    Return NO
  Compute n such that $y_{n-1}^{n-1} < |x| \leq y_n^n$
  Compute integers m and k such that $n = \langle m, k \rangle$.
  Compute $Q \leftarrow Q(M_m^B, 0^{y_n})$
  if $|Q| > k$ **then**
    if $x \in K$ **then**
      Return YES
    else
      Return NO
  else
    if $b \in Q$ **then**
      Return NO
    else
      if $x \in K$ **then**
        Return YES
      else
        Return NO

we have a slightly different situation in which options to encode are adaptive, so if we want to construct B such that $K \leq^p_{2-T} B$, we can encode every x in K by using two options out of three possible options to encode into B.

Theorem 3.2. *There is a $\leq^p_{2\text{-}T}$-complete set for* NEXP *that is not $\leq^p_{2\text{-}tt}$-autoreducible.*

Proof.

Let $\{M_j\}_{j\geq 1}$ be an enumeration of all $\leq^p_{2\text{-}tt}$ reductions.

Let $\{\text{NEXP}_i\}_{i\geq 1}$ be an enumeration of all nondeterministic exponential time oracle Turing machines.

For each $j \geq 1$, assume that the running times of computations of M_j and NEXP_j on input x are bounded by $|x|^j$ and $2^{|x|^j}$, respectively.

Let $K = \{\langle i, x, l\rangle \mid \text{NEXP}_i \text{ accepts input } x \text{ within } l \text{ steps } \}$ be a canonical complete set for NEXP.

We will construct a set B such that $K \leq^p_{2\text{-}T} B$ but B is not $\leq^p_{2\text{-}tt}$-autoreducible.

Algorithm 3 Thm 3.2. $\leq^p_{2-T}$ Algorithm that reduces K to B

Require: An input string x
Ensure: Return YES if $x \in K$. Otherwise, return NO

```
1: if ⟨x, 00⟩ ∈ B then
2:     if ⟨x, 01⟩ ∈ B then
3:         Return YES
4:     else
5:         Return NO
6: else
7:     if ⟨x, 10⟩ ∈ B then
8:         Return YES
9:     else
10:        Return NO
```

The $\leq^p_{2\text{-}T}$ reduction from K to B is described in the Algorithm 3. During the B's construction, we try to maintain that property to make sure that this $\leq^p_{2-T}$-reduction from K to B is correct. At the same time, we need to diagonalize all $\leq^p_{2-tt}$-reductions M_n to make B not autoreducible.

Let's define the sequence $\{y_n\}_{n\geq 0}$ such that $y_0 = 1$ and $y_{n+1} = 2^{y_n^n} + 1$ for every $n \geq 0$.

The set B is constructed in each stage as follows.

- Initially we set $B = \emptyset$.
- At stage n, suppose that the set B has been constructed in the way that all strings of length up to y_{n-1}^{n-1} are all encoded into B appropriately to make the reduction in Algorithm 3 works. In this stage, we will encode all strings of length between $y_{n-1}^{n-1} + 1$ and y_n^n into B. The construction is described in the Algorithm 4.

Refer to the B's construction in the Algorithm 4, to see that $K \leq^p_{2-T} B$, we will show that the property that is described in the Algorithm 3 is maintained:

For any x, consider the following cases in the Algorithm 4

- Line 7 : Because $\langle x, 00\rangle$ is not put into B and then $\langle x, 10\rangle \in B \iff x \in K$. So the reduction in Algorithm 3 is correct with this input.

Algorithm 4 Thm 3.2. B's construction at stage n.

Ensure: Encoding all strings x such that $y_{n-1}^{n-1} + 1 \leq |x| \leq y_n^n$ into B to make the reduction algorithm 3 correct. Also diagonalize M_n using the string 0^{y_n}

Compute $Q \leftarrow Q(M_n^B, 0^{y_n})$
Compute $P \leftarrow \{x || x| > y_{n-1}^{n-1}$ and $\langle x, a\rangle \in Q$ for some $a \in \{00, 01, 10\}\}$
for every $x \in P$ **do**
    Compute $P^x \leftarrow \{\langle x, a\rangle | x \in P$ and $a \in \{00, 01, 10\}\}$
for every x in P **do**
    if $P^x = \{q_1, q_2\}$ **then**
        if $q_1 = \langle x, 00\rangle$ and $q_2 = \langle x, 01\rangle$ **then**
            if $x \in K$ **then**
                $B \leftarrow B \cup \{\langle x, 10\rangle\}$ ▷ Don't put $\langle x, 00\rangle$ and $\langle x, 01\rangle$ into B, then $x \in K \iff \langle x, 10\rangle \in B$
        if $q_1 = \langle x, 00\rangle$ and $q_2 = \langle x, 10\rangle$ **then**
            $B \leftarrow B \cup \{\langle x, 00\rangle\}$
            if $x \in K$ **then**
                $B \leftarrow B \cup \{\langle x, 01\rangle\}$
        if $q_1 = \langle x, 01\rangle$ and $q_2 = \langle x, 10\rangle$ **then**
            $B \leftarrow B \cup \{\langle x, 01\rangle\}$
            if $x \in K$ **then**
                $B \leftarrow B \cup \{\langle x, 00\rangle\}$
    if $P^x = \{q\}$ **then**
        if $q = \langle x, 00\rangle$ or $q = \langle x, 01\rangle$ **then**
            if $x \in K$ **then**
                $B \leftarrow B \cup \{\langle x, 10\rangle\}$
        if $q = \langle x, 10\rangle$ **then**
            $B \leftarrow B \cup \{\langle x, 00\rangle\}$
            if $x \in K$ **then**
                $B \leftarrow B \cup \{\langle x, 01\rangle\}$
for every string x such that $y_{n-1}^{n-1} + 1 \leq |x| \leq y_n^n$ and $x \notin P$ **do** ▷ Encoding all remaining strings
    if $x \in K$ **then**
        $B \leftarrow B \cup \{\langle x, 10\rangle\}$ ▷ Because $\langle x, 00\rangle \notin B$, $x \in K \iff \langle x, 10\rangle \in B$
if M_n^B rejects 0^{y_n} **then** ▷ diagonalization step
    $B \leftarrow B \cup \{0^{y_n}\}$

- Line 14: If $x \in K$ then $\langle x, 00\rangle$ is put into B. And also $\langle x, 01\rangle \in B$. So it is correct. If $x \notin K$, then $\langle x, 00\rangle$ is not put into B; that also means, $\langle x, 10\rangle$ is used to determine $x \in K$ or not. And by the construction, $\langle x, 10\rangle \notin B$. So it is correct.
- Line 26: Because $\langle x, 00\rangle$ is not put into B, $\langle x, 10\rangle$ is used to encode whether $x \in K$. And the construction in this case reflects correctly.

The following lemma claims the time complexity of B

Lemma 3.4. $B \in$ NEXP.

Proof. Obviously B only contains elements of the following forms: 0^{y^n}, $\langle y, 00\rangle$, $\langle y, 01\rangle$, and $\langle y, 10\rangle$.

Given an input b, to decide whether $b \in B$, we consider the following cases:

- $b = 0^{y^n}$: Simulate the Turing machine M_n on input 0^{y^n} while resolving all queries the same way as how B is constructed in the above algorithm.

  Accept $b \iff M_n$ rejects 0^{y^n}.
- b is of the forms $\langle x, 00\rangle$, $\langle x, 01\rangle$, or $\langle x, 10\rangle$: Reverse direction from how B is constructed.

All operations can be done deterministically in exponential time; except checking whether x is in K that can be done nondeterministically in exponential time.

So $B \in$ NEXP. □

So B is the $\leq^p_{2-T}$-complete set for NEXP. And also by the way B is constructed, it is easy to see that B is not $\leq^p_{2-tt}$-autoreducible.

□

Notice that Glaßer et al. [4] showed that every $\leq^p_{2-tt}$-complete set for NEXP is $\leq^p_{2-tt}$-autoreducible. So this theorem is somehow "tight" and also we have the following corollary that separates the notions of $\leq^p_{2\text{-}T}$ and $\leq^p_{2\text{-}tt}$.

Corollary 3.1. *There is a $\leq^p_{2\text{-}T}$-complete set for* NEXP *that is not $\leq^p_{2\text{-}tt}$ complete.*

By using a similar technique, Theorem 3.2 can be generalized to show the following:

Theorem 3.3. *For any positive integers s and k such that $2^s - 1 > k$, there is a $\leq^p_{s\text{-}T}$-complete set for* NEXP *that is not $\leq^p_{k\text{-}tt}$-autoreducible.*

Proof.

Let $\{M_j\}_{j\geq 1}$ be an enumeration of all $\leq^p_{k\text{-}tt}$ reductions.

Let $\{\text{NEXP}_i\}_{i\geq 1}$ be an enumeration of all nondeterministic exponential time oracle Turing machines.

For each $j \geq 1$, assume that the running time of M_j and NEXP_j are bounded by n^j and 2^{n^j}, respectively.

Let $K = \{\langle i, x, l\rangle \mid \text{NEXP}_i \text{ accepts input } x \text{ within } l \text{ steps }\}$ be a canonical complete set for NEXP.

We will construct a set B such that $K \leq^p_{s\text{-}T} B$ but B is not $\leq^p_{k\text{-}tt}$-autoreducible.

The $\leq^p_{s\text{-}T}$ reduction from K to B will be as follows: we build a full binary tree of height s. This tree has exactly $2^s - 1$ nodes. We number the nodes from top to down, left to right by using numbers $0, 1, \ldots, 2^s - 1$; i.e. the root node will be numbered 0, then its two children nodes will be 1 and 2, etc. Then for any string x, each node i will be labeled by the pair $\langle x, i\rangle$.

From now on, for every such x, we denote $\mathcal{T}(x)$ is such a query tree and for every node $\mathcal{N}$, $\mathcal{N}$ is referred as a node itself or its label interchangeably. Also for any two nodes $\mathcal{N}_1$ and $\mathcal{N}_2$ such that one node is an ancestor of another node, denote $\mathcal{P}(\mathcal{N}_1, \mathcal{N}_2)$ be a unique path from $\mathcal{N}_1$ to $\mathcal{N}_2$. For every node $\mathcal{N}$, denote the left path $\mathcal{L}(\mathcal{N})$ be a path from $\mathcal{N}$ to a leaf node by just traversing left. The right path $\mathcal{R}(\mathcal{N})$ is defined similarly. Those labels are possible queries that can be asked to the oracle B by this reduction. Specifically, start at the root node, and if the current query is the node $\mathcal{N}$, if the answer is YES, $\mathcal{N} \in B$, then the next query will be $\mathcal{N}$'s left child node; otherwise the right child node will be asked. The reduction accepts if and only if the last query (certainly, it is one of the leaf nodes) belongs to B.

So now we will try to construct such a set B that satisfies the above reduction. At the same time, we want to diagonalize all M_n such that M_n accepts 0^{y_n} if and only if $0^{y_n} \notin B$. y_n will be defined formally as follows:

Define the sequence $\{y_n\}_{n\geq 0}$ such that $y_0 = 1$ and $y_{n+1} = 2^{y_n^n} + 1$ for every $n \geq 0$.

The set B is constructed in each stage as follows.

Initially we set $B = \emptyset$.

At stage n, suppose that the set B has been constructed such that all strings of length up to y_{n-1}^{n-1} have already been encoded into B appropriately to make the above reduction work. We will encode all strings of length between $y_{n-1}^{n-1} + 1$ and y_n^n into B in this stage.

Compute Q that is the set of all queries q of M_n on input 0^{y_n} such that $q = \langle x, i\rangle$, $i \leq 2^s - 1$, and $y_{n-1}^{n-1} + 1 \leq |x| \leq y_n^n$. Denote P be the set of all x such that $\langle x, i\rangle \in Q$ for some $0 \leq i \leq 2^s - 1$. And for each $x \in P$, denote P^x be the set of all $\langle x, i\rangle$ such that $\langle x, i\rangle \in Q$ and $0 \leq i \leq 2^s - 1$.

For each x in P, consider the set P^x. Notice that $|P^x| \leq k < 2^s - 1$. Consider the query tree $\mathcal{T}(x)$

- **Case 1:** If all leaf nodes are in P^x then there are some internal nodes such that they are not in P_x. Denote $\mathcal{N}$ be the smallest node in the set of those nodes. Put $\mathcal{N}$ into B if and only if $x \in K$. Also for every node $\mathcal{N}'$ in $\mathcal{L}(\mathcal{N})$ and $\mathcal{N}' \neq \mathcal{N}$, add $\mathcal{N}'$ to B. Finally for every node $\mathcal{N}'$ in the path $\mathcal{P}(Root, \mathcal{N})$, add $\mathcal{N}'$ to B if and only if its left child node is in the path.

- **Case 2:** If there are some leaf nodes that are not in P^x. Denote $\mathcal{N}$ be the smallest node in the set of those nodes. Add $\mathcal{N}$ to B if and only if $x \in K$. For every node $\mathcal{N}'$ in $\mathcal{P}(Root, \mathcal{N})$, add $\mathcal{N}'$ to B if and only if its left child is in that path

For every $x \notin P$ such that $y_{n-1}^{n-1} + 1 \leq |x| \leq y_n^n$, put $\langle x, 2^s - 1\rangle$ into B if and only if $x \in K$.

After all those steps are done, put 0^{y_n} into B if and only if M_n^B rejects 0^{y_n}

That is all how B is constructed. It is straightforward to see that the construction satisfies two properties: $K \leq_{s\text{-}T}^p B$ and B is not $\leq_{k\text{-}tt}^p$-autoreducible. The following lemma claims the time complexity of B

Lemma 3.5. $B \in$ NEXP

Proof. Notice that all elements of B have one of two forms 0^* and $\langle x, i\rangle$ where $0 \leq i \leq 2^s - 1$. So for any input of different forms, it just rejects immediately.

Given an input b, consider the following cases:

- If $b = 0^{y_n}$ for some n (otherwise, $b \notin B$). Then by the construction above, $0^{y_n} \in B \iff M_n^B$ rejects 0^{y_n}. So if we know how to resolve all queries made to the oracle B then it is easy to determine whether M_n^B accepts 0^{y_n} in exponential time. Now notice that in the above construction, for every query q, it can be resolved by considering the query tree and it does not depend on the membership of some x in K. So it can be done deterministically in exponential time.

- If $b = \langle x, i \rangle$ for some $0 \leq i \leq 2^s - 1$. Then just by considering the query tree $\mathcal{T}(x)$, there are two cases
  - The membership of b in B can be determined straightforward, based on the above construction, and not depend on the fact that $x \in K$ or not.
  - $b \in B \iff x \in K$. In this case, we can simulate the machine to accept K on an input x. Notice that x and b have the same polynomial length, so it can be done nondeterministically in exponential time.

So $B \in$ NEXP
□

So B is $\leq^p_{s-tt}$-complete set for NEXP that is not $\leq^p_{k-tt}$-autoreducible.
□

It has been known that there is a Turing complete set for EXP that is not $\leq^p_{tt}$-autoreducible [3]. By a little trick to the proof in Theorem 3.3, we show that it also holds for NEXP.

Theorem 3.4. *There is a Turing complete set for* NEXP *that is not* $\leq^p_{tt}$*-autoreducible.*

Proof. Using the same technique in Theorem 3.3 with a little trick, we can obtain the above result. Notice that in this case, the M_n autoreduction will not ask just k queries on input 0^{y_n}, but it can ask up to y_n^n queries, because its running time on input 0^{y_n} is bounded by y_n^n. Another modification is that the reduction from K to B will now need to ask more queries, saying $|x|^2$ adaptive queries; that also means the query tree will have height $|x|^2$. With this trick in mind, in the construction algorithm of B at stage n, for every x in P, $|P^x| \leq y_n^n = 2^{y_{n-1}^{n-1} n} < 2^{y_{n-1}^{2(n-1)}} < 2^{|x|^2}$. So the number of nodes in the query tree $\mathcal{T}(x)$ will be bigger than the number of queries of M_n on input 0^{y_n}, so in two cases, 1 and 2, the construction will work similarly. □

Now we consider the question whether every $\leq^p_{3-tt}$-complete set for NEXP is $\leq^p_{3-tt}$-autoreducible. Notice that the above technique cannot be used because the number of options to encode every x in K into B is no more than the number of queries of M_n^B on input 0^{y_n}; both are equal 3 in this case. This difficulty arises because we have no "room" for the encoding and diagonalization at the same time. We need to use a different trick and the following lemma will help in the proof:

Lemma 3.6. *Let f be any boolean function of 3 variables a_1, a_2, b_1. Then at least one of the following statements must be true:*

- *There exists a matrix A of size 4×3 that will be described below. Each entry is 0 or 1. Columns are labeled a_1, a_2, b_1 from left to right. The submatrix $A[1..4, 1..2]$ contains 4 distinct binary rows. For every $k = 1, \ldots, 4$, the row k satisfies the condition $f(a_1, a_2, b_1) = 1$. Also either one of the following statements must be true:*
  - *$A[b_1]$ is constant*
  - *$A[b_1] = A[c_1] \wedge A[c_2]$ or $A[b_1] = A[c_1] \vee A[c_2]$ for some $c_1, c_2 \in \{a_1, a_2\}$. Note that c_1 can be equal to c_2. Here $A[b_1]$ denotes the column b_1.*
- *There exists a matrix A of size 2×3 that will be described below. Each entry is 0 or 1. Columns are labeled b_1, a_1, a_2 from left to right. The submatrix $A[1..2, 1..1]$ contains 2 distinct binary rows. For every $k = 1, 2$, the row k satisfies the condition $f(a_1, a_2, b_1) = 0$. Also for every column a_i, $i = 1, 2$, $A[a_i]$ is constant or $A[a_i] = A[b_1]$.*

The following are illustrations of possible matrices, where β_i and α_j are 0 or 1.

$$\begin{array}{c} \\ row_1 \\ row_2 \\ row_3 \\ row_4 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} a_1 & a_2 & b_1 \end{array} \\ \begin{pmatrix} 0 & 0 & \beta_1 \\ 0 & 1 & \beta_2 \\ 1 & 0 & \beta_3 \\ 1 & 1 & \beta_4 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ row_1 \\ row_2 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & \alpha_1 & \alpha_2 \\ 1 & \alpha_3 & \alpha_4 \end{pmatrix} \end{array}$$

Proof.

Given any boolean function f of 3 variables a_1, a_2, and b_1, if $\exists a_1 \exists a_2 \forall b_1 f(a_1, a_2, b_1) = 0$ then we easily have the matrix of form 2. So now we assume that $\forall a_1 \forall a_2 \exists b_1 f(a_1, a_2, b_1) = 1$. For every a_1 and a_2, denote $g(a_1, a_2)$ be the smallest b_1 such that $f(a_1, a_2, b_1) = 1$. The following 9 matrices will show most of the cases.

Fig. 1, 2, 3:

$$\begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array} \qquad \begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array} \qquad \begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array}$$
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Fig. 4, 5, 6:

$$\begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array} \qquad \begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array} \qquad \begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array}$$
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Fig. 7, 8, 9:

$$\begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array} \qquad \begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array} \qquad \begin{array}{ccc} a_1 & a_2 & g(a_1,a_2) \end{array}$$
$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Consider the following cases:

- Case 1,2,4,6,8: Correspond to Fig. 1,2,4,6,8. Straightforward.
- Case 3: Correspond to Fig. 3. If $f(1,1,1) = 1$ then matrix we need is in Fig. 4. If $f(1,1,1) = 0$, then the following matrix is what we need:

$$\begin{array}{c} \\ row_1 \\ row_2 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array}$$

- Case 5: Correspond to Fig. 5. Then obviously $f(0,1,0) = 0$. If $f(1,1,1) = 1$ then Fig. 6 is what we need. Otherwise, if $f(1,1,1) = 0$ then the following matrix will satisfy the form 2:

$$\begin{array}{c} \\ row_1 \\ row_2 \end{array}\!\!\begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \end{array}$$

- Case 7: Correspond to Fig. 7. Then $f(1,0,0) = 0$. If $f(1,1,1) = 1$ then Fig. 8 is what we need. Otherwise, if $f(1,1,1) = 0$, then the following matrix will satisfy the form 2:

$$\begin{array}{c} \\ row_1 \\ row_2 \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array}$$

- Case 9,...,16: In these cases, $f(0,0,0) = 0$.
  - If $f(0,1,1) = 0$, then we have the following satisfied matrix:
  $$\begin{array}{c} \\ row_1 \\ row_2 \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array}$$
  - Else if $f(1,0,1) = 0$, then we have the following satisfied matrix:
  $$\begin{array}{c} \\ row_1 \\ row_2 \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$
  - Else if $f(1,1,1) = 0$, then we have the following satisfied matrix:
  $$\begin{array}{c} \\ row_1 \\ row_2 \end{array} \begin{array}{c} \begin{array}{ccc} b_1 & a_1 & a_2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \end{array}$$
  - Else we have the following satisfied matrix:
  $$\begin{array}{c} \\ row_1 \\ row_2 \\ row_3 \\ row_4 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & a_2 & b_1 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \end{array}$$

□

Theorem 3.5. *For any number c, there is a $\leq^p_{2-T}$-complete set for* NEXP *that is not $\leq^p_{3-tt-h-c}$-autoreducible.*

Proof Sketch. The basic idea is to use the diagonalization technique to diagonalize the string 0^m against the autoreduction M_n. At the same time, we need to encode the complete set K into B. In this proof, we will encode K into B by 2 tracks: for every $x \in K$, we will encode $\langle 0, x\rangle$ if $0^m \in B$; otherwise we will encode $\langle 1, x\rangle$ into B. The diagonalization part will ensure that we do not add 0^m to B if M_n^B accepts 0^m. Here notice that we need to determine whether M_n^B accepts 0^m and this task can affect the encoding and vice versa. For example, if the queries of M_n^B on input 0^m are $\langle 1, q_1\rangle$, $\langle 1, q_2\rangle$, and $\langle 0, q_3\rangle$. Then the value of $M_n^B(0^m)$ will depends on $B(\langle 1, q_1\rangle)$, $B(\langle 1, q_2\rangle)$, and $B(\langle 0, q_3\rangle)$. It is possible to have the case when setting $B(\langle 1, q_1\rangle) = 1$ will make M_n^B accepts 0^m, and then in this case the encoding needs to take place in $\langle 1, q_1\rangle$. But if $q_1 \notin K$ then it will cause the contradiction and break the $\leq^p_{3-tt}$-hardness of B. To resolve this issue, we can consider the truth-table of M_n^B on input 0^m. If we have the case when whatever the values of $B(\langle 1, q_1\rangle)$ and $B(\langle 1, q_2\rangle)$ are, there exists a value of $B(\langle 0, q_3\rangle)$ such that M_n^B accepts 0^m. Then in this case, we can set $B(\langle 1, q_1\rangle)$ and $B(\langle 1, q_2\rangle)$ to the appropriate values depending on their membership in K. And this way will make the diagonalization part and encoding part consistent. But there is a catch here: The value of $B(\langle 0, q_3\rangle)$ will depend on $B(\langle 1, q_1\rangle)$ and $B(\langle 1, q_2\rangle)$, and because NEXP is not known to be closed under complement, it will make the complexity of the set B go beyond the NEXP class. To resolve this problem, we try to find a way to make the value of $B(\langle 0, q_3\rangle)$ depend positively on $B(\langle 1, q_1\rangle)$ and/or $B(\langle 1, q_2\rangle)$; i.e. $B(\langle 0, q_3\rangle) = 1 \iff B(\langle 1, q_1\rangle) = 1$. By this way, it will force B to belong to NEXP.

□

Proof.

Let $\{M_i\}_{i\geq 1}$ be a standard enumeration of all $\leq^p_{3-tt-h-c}$ autoreductions clocked such that M_i runs in time n^i.

Let K be a canonical complete set for NEXP. We will construct the $\leq^p_{2-T}$ complete set A for NEXP incrementally in each stage and diagonalize all autoreductions M_i.

We assume a polynomial-time computable one-to-one pairing function that can take any finite number of inputs such that its range does not intersect with 0^*.

We define the sequence $\{y_n\}_{n\geq 1}$ recursively as follows: $y_1 = 1$ and $y_{n+1} = \max(y_n^n, y_n^{c^2}) + 1$ for all $n \geq 1$.

In each stage, we will construct the set B such that the following procedure is the $\leq^p_{2-T}$ reduction that reduces K to B. Given any input x, ask a query 0^m to the oracle B, where m is a number that is bounded by some polynomial of $|x|$. If the answer is YES, then accept x if and only if $\langle 0, x\rangle \in B$. If the answer is NO, then accept x if and only if $\langle 1, x\rangle \in B$. Obviously if B satisfies this condition, then B is $\leq^p_{2-T}$-hard for NEXP.

The detail of how B is constructed will be as follows.

- Initially $B = \emptyset$
- Suppose at stage n, the set B is constructed up to length $y_n - 1$. In this stage n, we will add appropriate strings of length between y_n and $y_{n+1} - 1$ to accomplish two things: encoding K into B and diagonalize the string $0^{y_n^c}$ against the autoreduction M_n that asks no more than 3 queries. So in the following steps, if M_n asks more than 3 queries then the diagonalization task will be skipped to the next stage.

Algorithm 5 Thm 3.5. Handle-Case-1: B's construction algorithm when all queries of M_n^B on input $0^{y_n^c}$ are of the forms $\langle 1, q\rangle$.

▷ In this case, $Q^+ = \{\langle 1, q_1\rangle, \langle 1, q_2\rangle, \langle 1, q_3\rangle\}$
▷ f : Boolean truth-table function of M_n^B on input $0^{y_n^c}$

if f is constant 1 **then**
  for every string x, $y_n \leq |x| \leq y_{n+1} - 1$ **do**
    if $x \in K$ **then**
      $B \leftarrow B \cup \{\langle 1, x\rangle\}$ ▷ $0^{y_n^c} \notin B$, so encoding each x in K by $\langle 1, x\rangle$.
else ▷ Encoding each x in K by $\langle 0, x\rangle$
  Compute $(\alpha_1, \alpha_2, \alpha_3) \leftarrow$ lexicographically smallest value satisfying $f(\alpha_1, \alpha_2, \alpha_3) = 0$
  for $i = 1$ to 3 **do**
    if $\alpha_i = 1$ **then**
      $B \leftarrow B \cup \{\langle 1, q_i\rangle\}$
  for every string x, $y_n \leq |x| \leq y_{n+1} - 1$ **do**
    if $x \in K$ **then**
      $B \leftarrow B \cup \{\langle 0, x\rangle\}$
  $B \leftarrow B \cup \{0^{y_n^c}\}$ ▷ Diagonalization. M_n^B rejects $0^{y_n^c}$ in this case

---

The Algorithm 8 describes how to construct B at the stage n. We will show that the construction of B maintains the property that for any x, if $0^{y_n^c} \in B$ then $x \in K \iff \langle 0, x\rangle \in B$. Otherwise, if $0^{y_n^c} \notin B$, then $x \in K \iff \langle 1, x\rangle \in B$. So this reflects the $\leq^p_{2-T}$ from K to B as we mentioned before.

Algorithm 6 Thm 3.5. Handle-Case-2: B's construction algorithm when all queries of M_n^B on input $0^{y_n^c}$ are of the forms $\langle 0, q\rangle$.

▷ In this case, $Q^- = \{\langle 0, q_1\rangle, \langle 0, q_2\rangle, \langle 0, q_3\rangle\}$
▷ f: Boolean truth-table function of M_n^B on input $0^{y_n^c}$

1: **if** f is constant 0 **then** ▷ Encoding each x in K by $\langle 0, x\rangle$
2: **for** every string x, $y_n \leq |x| \leq y_{n+1} - 1$ **do**
3: **if** $x \in K$ **then**
4: $B \leftarrow B \cup \{\langle 0, x\rangle\}$
5: $B \leftarrow B \cup \{0^{y_n^c}\}$ ▷ Diagonalization here. M_n^B rejects $0^{y_n^c}$ in this case
6: **else** ▷ Encoding each x in K by $\langle 1, x\rangle$
7: **Compute** $(\alpha_1, \alpha_2, \alpha_3) \leftarrow$ lexicographically smallest value satisfying $f(\alpha_1, \alpha_2, \alpha_3) = 1$
8: **for** $i = 1$ to 3 **do**
9: **if** $\alpha_i = 1$ **then**
10: $B \leftarrow B \cup \{\langle 0, q_i\rangle\}$
11: **for** every string x, $y_n \leq |x| \leq y_{n+1} - 1$ **do**
12: **if** $x \in K$ **then**
13: $B \leftarrow B \cup \{\langle 1, x\rangle\}$

Consider the Algorithm 5. In line 1, M_n^B accepts $0^{y_n^c}$. So in this case, we need to encode every $x \in K$ by $\langle 1, x\rangle$ in B. It is done by line 2-4. And notice that by adding those strings to B does not affect the fact that M_n^B accepts $0^{y_n^c}$ because f is constant. In another case, consider line 5, if we choose whether or not to add $\langle 1, q_i\rangle$ to B appropriately, then M_n^B rejects $0^{y_n^c}$. And so we encode every x in K by $\langle 0, x\rangle$ in B. It is reflected in line 10-12. The Algorithm 6 will do the similar job.

Consider the Algorithm 7. In line 2, we can make f equal to 1, then we can encode every $x \in K$ by $\langle 1, x\rangle$ in B. By doing that, for every query $\langle 0, r_i\rangle$, we need to add it to B or not depend on the other queries $\langle 1, q_j\rangle$. All of those operations are reflected in line 7-14. In line 15, we have the similar reasoning.

So it can be verified that the B's construction in the Algorithm 8 makes sure that B is $\leq_{2-T}^p$-hard for NEXP and B is not autoreducible because of the diagonalization. We just need to prove that $B \in$ NEXP.

Lemma 3.7. $B \in$ NEXP

Proof. We will show that given any input x, determining whether $x \in B$ can be done nondeterministically in exponential time. Given an input x, consider the following cases:

- If $x = 0^*$: if $|x| \neq y_n^c$ for all n then reject. Otherwise, consider all queries of M_n on input $0^{y_n^c}$ and the boolean truth-table f on that input, we have the following cases:
  - If all queries are of the form $\langle 1, q\rangle$
    - If f is constant 1, then reject
    - Otherwise, accept
  - If all queries are of the form $\langle 0, q\rangle$, then accept if and only if f is constant 0
  - If all queries are of the form $\langle 1, q_1\rangle, \ldots, \langle 1, q_s\rangle, \langle 0, r_1\rangle, \ldots, \langle 0, r_t\rangle$. If the matrix A is of the form 1, then reject. Otherwise, accept
- If $x = \langle 1, x_1\rangle$: determine n such that $y_n \leq |x| < y_{n+1}$. Consider the k queries of M_n on input $0^{y_n^c}$ and the boolean truth-table f on that input. We will show some interesting cases; other cases are straightforward from the construction of B or similar.

Algorithm 7 Thm 3.5. Handle-Case-3: B's construction algorithm when all queries of M_n^B on input $0^{y_n^c}$ are of the forms $\langle 1, q_1\rangle, \ldots, \langle 1, q_s\rangle, \langle 0, r_1\rangle, \ldots, \langle 0, r_t\rangle$.

▷ The boolean truth-table function of M_n^B on input $0^{y_n^c}$ is $f(a_1, \ldots, a_s, b_1, \ldots, b_t)$
▷ M_n^B accepts $0^{y_n^c} \iff f(B(\langle 1, q_1\rangle), \ldots, B(\langle 1, q_s\rangle), B(\langle 0, r_1\rangle), \ldots, B(\langle 0, r_t\rangle)) = 1$

Compute matrix A in the Lemma 3.6
if A is of the form 1 **then** ▷ $0^{y_n^c} \notin B$, so encoding each $x \in K$ by $\langle 1, x\rangle$
  for every string x, $y_n \leq |x| \leq y_{n+1} - 1$ **do**
    if $x \in K$ **then**
      $B \leftarrow B \cup \{\langle 1, x\rangle\}$
  for $i = 1$ to t **do**
    if $A[b_i]$ is a constant 1 **then**
      $B \leftarrow B \cup \{\langle 0, r_i\rangle\}$
    if $A[b_i] = A[a_{i_1}] \vee A[a_{i_2}]$ for some $1 \leq i_1, i_2 \leq s$ **then**
      if $(q_{i_1} \in K)$ or $(q_{i_2} \in K)$ **then**
        $B \leftarrow B \cup \{\langle 0, r_i\rangle\}$
    if $A[b_i] = A[a_{i_1}] \wedge A[a_{i_2}]$ for some $1 \leq i_1, i_2 \leq s$ **then**
      if $(q_{i_1} \in K)$ and $(q_{i_2} \in K)$ **then**
        $B \leftarrow B \cup \{\langle 0, r_i\rangle\}$
if A is of the form 2 **then** ▷ $0^{y_n^c} \in B$, so encoding each $x \in K$ by $\langle 0, x\rangle$.
  for every string x, $y_n \leq |x| \leq y_{n+1} - 1$ **do**
    if $x \in K$ **then**
      $B \leftarrow B \cup \{\langle 0, x\rangle\}$
  for $i = 1$ to s **do**
    if $A[a_i]$ is a constant 1 **then**
      $B \leftarrow B \cup \{\langle 1, q_i\rangle\}$
    if $A[a_i] = A[b_{i_1}] \vee A[b_{i_2}]$ for some $1 \leq i_1, i_2 \leq t$ **then**
      if $(r_{i_1} \in K)$ or $(r_{i_2} \in K)$ **then**
        $B \leftarrow B \cup \{\langle 1, q_i\rangle\}$
    if $A[a_i] = A[b_{i_1}] \wedge A[b_{i_2}]$ for some $1 \leq i_1, i_2 \leq t$ **then**
      if $(r_{i_1} \in K)$ and $(r_{i_2} \in K)$ **then**
        $B \leftarrow B \cup \{\langle 1, q_i\rangle\}$
  $B \leftarrow B \cup \{0^{y_n^c}\}$ ▷ M_n^B rejects $0^{y_n^c}$, adding $0^{y_n^c}$ to B to diagonalize against M_n^B

Algorithm 8 Thm 3.5. B's construction algorithm at stage n

Ensure: Encoding all strings of length between y_n and $y_{n+1} - 1$ to make $K \leq^p_{2-T} B$ correct. Also diagonalize against M_n^B using the string $0^{y_n^c}$

1: **Compute** $Q \leftarrow Q(M_n^B, 0^{y_n^c})$
2: **Compute** $f \leftarrow$ boolean truth-table function of M_n^B on input $0^{y_n^c}$
3: **Compute** $Q^+ \leftarrow \{\langle 1, x\rangle | \langle 1, x\rangle \in Q\}$
4: **Compute** $Q^- \leftarrow \{\langle 0, x\rangle | \langle 1, x\rangle \in Q\}$
5: **if** $Q^+ = \{\langle 1, q_1\rangle, \langle 1, q_2\rangle, \langle 1, q_3\rangle\}$ **then**
6: **Call** Handle-Case-1 ▷ It is described in the Algorithm 5
7: **if** $Q^- = \{\langle 0, q_1\rangle, \langle 0, q_2\rangle, \langle 0, q_3\rangle\}$ **then**
8: **Call** Handle-Case-2 ▷ It is described in the Algorithm 6
9: **if** $Q = \{\langle 1, q_1\rangle, \ldots, \langle 1, q_s\rangle, \langle 0, r_1\rangle, \ldots, \langle 0, r_t\rangle\}$ **then** ▷ s+t = 3
10: **Call** Handle-Case-3 ▷ it is described in the Algorithm 7

- If all queries are of the form $\langle 1, q\rangle$ then :
  - If f is constant 1, then accept if and only if $x_1 \in K$.
  - Else
    - If x is one of the queries $\langle 1, q_i\rangle$, then compute $\alpha_1, \ldots, \alpha_k$, the lexicographically first value satisfying $f(\alpha_1, \ldots, \alpha_k) = 0$. Accept if and only if $\alpha_i = 1$
    - Else reject
- If all queries are of the form $\langle 0, q\rangle$: similar to the previous case.
- If all queries are of the form $\langle 1, q_1\rangle, \ldots, \langle 1, q_s\rangle, \langle 0, r_1\rangle, \ldots, \langle 0, r_t\rangle$, compute the matrix A.
  - If the matrix A is of the form 1, then accept if and only if $x_1 \in K$.
  - If the matrix A is of the form 2, then
    - If x is not one of those queries, then reject
    - else if $x = \langle 1, q_i\rangle$
    - If $A[a_i]$ is a constant c, then accept if and only if $c = 1$
    - Else if $A[a_i] = A[b_{i_1}] \vee A[b_{i_2}]$, then accept if and only if $r_{i_1} \in K$ or $r_{i_2} \in K$. To do this, we just run the Turing machine to accept K simultaneously for 2 inputs r_{i_1} and r_{i_2}. If one of those 2 paths terminates in YES, then accept. Otherwise, reject.
    - Else if $A[a_i] = A[b_{i_1}] \wedge A[b_{i_2}]$, then accept if and only if $r_{i_1} \in K$ and $r_{i_2} \in K$. To do this, we just run the Turing machine to accept K simultaneously for 2 inputs r_{i_1} and r_{i_2}. If both 2 paths terminates in YES, then accept. Otherwise, reject.

- If $x = \langle 0, x_2\rangle$: similar to the case $\langle 1, x_1\rangle$

Now we will analyze the running time it takes to do the above tasks. All expensive tasks are as follows:

- Determining the membership of y in K. Because M_n is a poly-honest reduction, it can be done nondeterministically in exponential time in terms of input's length.
- Compute the queries and truth-table f of M_n on input $0^{y_n^c}$. It takes y_n^{nc}, which is at most $2^{y_n c}$.

So $B \in$ NEXP
□

So B is $\leq^p_{2-T}$-complete set for NEXP that is not $\leq^p_{3-tt-h-c}$-autoreducible.
□

We note that Lemma 3.6 cannot be generalized to a boolean function of 4 variables a_1, a_2, b_1, b_2 or more because we found a counterexample in that case. So the proof of Theorem 3.5 cannot be generalized to work with $\leq^p_{k-tt}$-reductions for $k \geq 4$. Nevertheless, the following theorem will show non-autoreducibility for $\leq^p_{k-tt}$-reductions if we reduce the power of the $\leq^p_{k-tt}$-autoreduction by not allowing the truth-table function to be an OR or a NOR.

Theorem 3.6. *For any positive integer k, there is a $\leq^p_{k-tt}$-complete set for* NEXP *(*EXP*) that is not weakly $\leq^p_{k-tt-w}$-autoreducible.*

Proof. We assume a polynomial-time computable one-to-one pairing function that can take any finite number of inputs such that its range does not intersect with 0^*.

Let $\{M_j\}_{j\geq 1}$ be an enumeration of polynomial-time weak $\leq^p_{k-tt}$ autoreductions.

Let $\{\text{NEXP}_i\}_{i\geq 1}$ be an enumeration of all nondeterministic exponential time oracle Turing machines.

For each $j \geq 1$, suppose that n^j bounds the running time of M_j and 2^{n^j} bounds the running time of NEXP_j

Let $K = \{\langle i, x, l\rangle \mid \text{NEXP}_i \text{ accepts input } x \text{ within } l \text{ steps }\}$ be a canonical complete set for NEXP.

Denote $\alpha_1, \ldots, \alpha_k$ be the lexicographically first k strings of length $\lceil \log k \rceil$.

We will construct a set B with the following property

$x \in K \iff$ there exists a j, $1 \leq j \leq k$, and $\langle \alpha_j, x\rangle \in B$

which ensure that $K \leq^p_{k-tt} B$, and then B is $\leq^p_{k-tt}$-hard for NEXP.

We also need the set B such that for any $n \geq 1$, the following property holds

$0^{y_n} \in B \iff M^B_n$ rejects input 0^{y_n} (value of y_n will be chosen later in the proof)

which ensures that M_n is not an autoreduction of B. Then we can conclude that B is not autoreducible for NEXP.

We will construct the set B in stage. In each stage, we will encode K into B and diagonalize all weak $\leq^p_{k-tt}$-reduction using the string 0^{y_n} simultaneously to obtain those above two properties.

Before going into detail of how the set B is constructed, let's define the sequence $\{y_n\}_{n\geq 0}$ such that $y_0 = 1$ and $y_{n+1} = 2^{y_n^n} + 1$ for every $n \geq 0$.

The set B is constructed in each stage as follows.

Initially we set $B = \emptyset$.

At stage n, suppose that the set B is already constructed up to strings of length y^{n-1}_{n-1}. We will encode all strings of length between $y^{n-1}_{n-1} + 1$ and y^n_n.

The construction in this stage will be as follows:

Denote Q be the set of all queries q of M_n on input 0^{y_n} such that $|q| > y^{n-1}_{n-1}$

Denote $P = \{x|$ there exists a $1 \leq j \leq k$ such that $\langle \alpha_j, x\rangle \in Q\}$

For every $x \in P$, denote $P^x = \{\langle \alpha_j, x\rangle | \langle \alpha_j, x\rangle \in Q\}$.

Consider the following cases:

- If $|P^x| < k$ for all x, then for every $x \in P$, denote t be the smallest number such that $\langle \alpha_t, x\rangle \notin Q$. Put $\langle \alpha_t, x\rangle$ into B if and only if $x \in K$. Also for every $x \notin P$, put $\langle \alpha_1, x\rangle$ into B if and only if $x \in K$. Finally, put 0^{y_n} into B if and only if M^B_n rejects 0^{y_n}.

- If $|P^x| = k$ for some x, consider the truth-table function g of M_n on input 0^{y_n}, we have two following cases:
  - If $g(0, 0, \ldots, 0) = 0$, then denote $c_1, \ldots, c_k$ be the lexicographically smallest non-zero value such that $g(c_1, \ldots, c_k) = 0$. For every c_i such that $c_i = 1$, put $\langle \alpha_i, x \rangle$ into B if and only if $x \in K$. Also for every $x \notin P$, put $\langle \alpha_1, x \rangle$ into B if and only if $x \in K$. Finally put 0^{y_n} into B.
  - If $g(0, 0, \ldots, 0) = 1$, then denote $c_1, \ldots, c_k$ be the lexicographically smallest non-zero value such that $g(c_1, \ldots, c_k) = 1$. For every c_i such that $c_i = 1$, put $\langle \alpha_i, x \rangle$ into B if and only if $x \in K$. Also for every $x \notin P$, put $\langle \alpha_1, x \rangle$ into B if and only if $x \in K$.

That is all how B is constructed. The following lemma claims the time complexity of B.

Lemma 3.8. $B \in \text{NEXP}$

Proof. Given an input b, one of the following cases can happen:

- **Case 1:** If b has the form 0^*: if $|b| \neq y_n$ for all n then reject. Otherwise, compute the set Q of all queries when running a Turing machine M_n^B on input 0^{y_n}. The notations of P and P^x are defined similarly to the B's construction above.
  - If $|P^x| < k$ for all x, then simulate the Turing machine M_n^B on input 0^{y_n}. Whenever a query q is asked, the answer from the oracle B will be resolved as follows:
    - If the length of q is greater than y_{n-1}^{n-1} then answer is NO.
    - Otherwise, check whether $q \in B$ recursively.
  - If $|P^x| = k$ for some x. Then denote g be a boolean truth-table function of M_n^B on input 0^{y_n}.
    - If $g(0, \ldots, 0) = 0$ then accept.
    - Otherwise, reject.
- **Case 2:** $b = \langle \alpha_i, x \rangle$ for some α_i (if $b \neq \langle \alpha_j, x \rangle$ for all j then just reject)

  Compute the number n such that $y_{n-1}^{n-1} < |b| \leq y_n^n$.

  Consider the set Q, P, and P^x as above. We have the following cases:
  - If $|P^x| < k$ for all x, then the following cases can happen. If $b \in Q$ then reject. Otherwise, accept if and only if $i = 1$ and $x \in K$.
  - If $|P^x| = k$ for some x. Denote g be a boolean truth-table function of M_n on input 0^{y_n}. Consider two following cases:
    - If $g(0, \ldots, 0) = 0$. Denote $c_1, \ldots, c_k$ be the lexicographically smallest non-zero value such that $g(c_1, \ldots, c_k) = 0$. If $b \in Q$, then accept if and only if $c_i = 1$ and $x \in K$. Otherwise, accept if and only if $i = 1$ and $x \in K$.
    - If $g(0, \ldots, 0) = 1$. Denote $c_1, \ldots, c_k$ be the lexicographically smallest non-zero value such that $g(c_1, \ldots, c_k) = 1$. If $b \in Q$, then accept if and only if $c_i = 1$ and $x \in K$. Otherwise, accept if and only if $i = 1$ and $x \in K$.

Now we will analyze the running time of the above tasks. Most expensive tasks will be described as follows:

- The number n can be determined in polynomial time in terms of length of input b.

- The query set Q and the truth-table function g can be computed in time y_n^n, which is no more than $O(2^{|b|^2})$.
- In case 1, recursively check whether the query q of length smaller than y_{n-1}^{n-1} belongs to B or not deterministically takes time $2^{2^{y_{n-1}^{n-1}}}$, which is no more than $2^{y_n} = 2^{|b|}$
- Determining whether x belongs to K can be done nondeterministically in $2^{|x|} < 2^{|b|}$

So $B \in$ NEXP
□

Lemma 3.9. $K \leq^p_{k-tt} B$

Proof. Notice in the construction algorithm of B, for every x that is in K, we encode at least one of the following strings $\langle \alpha_1, x\rangle \dots \langle \alpha_k, x\rangle$ into B. If $x \notin K$, no those strings are encoded into B. So $B \leq^p_{k-tt} K$.
□

Lemma 3.10. *B is not weakly $\leq^p_{k-tt}$-autoreducible.*

Proof. Straightforward from the B's construction.
□

By Lemma 3.8, Lemma 3.9, and Lemma 3.10, B is $\leq^p_{k-tt}$-complete for NEXP that is not weakly $\leq^p_{k-tt}$-autoreducible. □

The above proof also yields the following corollary:

Corollary 3.2. *For any positive integer k, there is a $\leq^p_{k-dtt}$-complete set for* NEXP *(*EXP*) that is not weakly $\leq^p_{k-tt}$-autoreducible.*

4 Implications

We begin with the following theorem.

Theorem 4.1. *Every $\leq^p_{dtt}$-complete set for* EXP *is $\leq^p_{NOR-tt}$-autoreducible.*

Glaßer et al. [4] also showed that every $\leq^p_{dtt}$-complete set for EXP is $\leq^p_{dtt}$-autoreducible. So by Theorem 4.1, the Corollary 3.2 is somehow "tight" for EXP.

Proof.

Let A be a $\leq^p_{dtt}$-complete set for EXP. We will show that A is also $\leq^p_{NOR-tt}$-autoreducible.

Let $\{M_i\}_{i\geq 1}$ be a standard enumeration of all $\leq^p_{dtt}$ reduction such that M_i runs in time $p_i(n) = n^i$ on inputs of size n.

Consider the set B containing elements of the form $\langle 0^i, x\rangle$ which are decided by the following nondeterministic algorithm in exponential time:

Given an input $\langle 0^i, x\rangle$,

- Compute Q, the set of all queries of M_i on input $\langle 0^i, x\rangle$.
- If $x \notin Q$ then Accept $\langle 0^i, x\rangle \iff x \notin A$.

- Otherwise, Reject $\langle 0^i, x\rangle$

Obviously $B \in$ EXP.

Since A is the $\leq^p_{dtt}$-complete set for EXP, $B \leq^p_{dtt} A$ by some disjunctive truth-table reduction M_j.

For any x, if x is one of the queries of M_j on input $\langle 0^j, x\rangle$, then $\langle 0^j, x\rangle \notin B$. This fact implies that for all queries q, including x, $q \notin A$. So $x \notin A$. If x is not one of the queries $q_1, \ldots, q_k$ of M_j on input $\langle 0^j, x\rangle$, then $x \in A \iff \langle 0^j, x\rangle \notin B \iff q_i \notin A$ for all i

Based on that observation, we have the following autoreduction algorithm for A:

Given input x,

- Compute $Q = \{q_1, \ldots, q_k\}$, the set of all queries of M_j on input $\langle 0^j, x\rangle$
- If $x \notin Q$, then accept $\iff q_1 \notin A \wedge \cdots \wedge q_k \notin A$
- Otherwise, reject x.

Observe that the above autoreduction is $\leq^p_{NOR-tt}$-autoreduction. So A is $\leq^p_{NOR-tt}$-autoreducible.

□

Recall that every $\leq^p_{k-dtt}$-complete set for NEXP is $\leq^p_{k-dtt}$-autoreducible [4]. Also every $\leq^p_{k-dtt}$-complete set for EXP is both $\leq^p_{k-dtt}$-autoreducible [4] and $\leq^p_{NOR-k-tt}$-autoreducible. So we want to know whether the same holds for NEXP; that is, whether every $\leq^p_{k-dtt}$-complete set for NEXP is also $\leq^p_{NOR-k-tt}$-autoreducible. Settling this question would lead to important complexity class results.

Theorem 4.2. *For any positive integer k,*

- *If every $\leq^p_{k-dtt}$-complete set for* NEXP *is $\leq^p_{NOR-k-tt}$-autoreducible, then* NEXP = coNEXP.
- *If there is a $\leq^p_{k-dtt}$-complete set for* NEXP *that is not $\leq^p_{NOR-k-tt}$-autoreducible, then* NEXP $\neq$ EXP.

Proof.

If there is a $\leq^p_{k-dtt}$-complete set for NEXP that is not $\leq^p_{NOR-k-tt}$-autoreducible, then by Theorem 4.1 , NEXP $\neq$ EXP.

Otherwise, suppose every $\leq^p_{k-dtt}$-complete set for NEXP is $\leq^p_{NOR-k-tt}$-autoreducible. Notice that K, the canonical complete set of NEXP, is also $\leq^p_{k-dtt}$-complete. So by the assumption, K is $\leq^p_{NOR-k-tt}$-autoreducible.

Denote f be the autoreduction of K. That is, for every x, $f(x) = \langle q_1, \ldots, q_k\rangle$, $x \neq q_i$ for all i, and $x \in K \iff q_1 \notin K \wedge \cdots \wedge q_k \notin K$.

By that, we have the following fact:

$x \in \overline{K} \iff x \notin K \iff q_1 \in K \vee \cdots \vee q_k \in K$. So $\bar{K} \leq^p_{k-dtt} K$

The following lemma shows that $\overline{K} \in$ NEXP.

Lemma 4.1. *If $A \leq^p_{k-dtt} B$ and $B \in$ NEXP then $A \in$ NEXP.*

Proof.

Suppose that $B \in$ NEXP. Denote N be a nondeterministic Turing machine to accept B in exponential time.

Denote f be the reduction from A to B by a disjunctive truth-table. That is, for every x, $f(x) = \langle q_1, \ldots, q_k\rangle$, and $x \in A \iff q_1 \in B \vee \cdots \vee q_k \in B$.

For every x, determining whether $x \in A$ is as follows:

- Compute $f(x) = \langle q_1, \ldots, q_k \rangle$.
- Simultaneously simulating the machine N on k inputs $q_1, \ldots, q_k$. If one of them turns out to be an accepting path, then accept. Otherwise reject.

So $A \in$ NEXP. $\square$

By this lemma, we have $\overline{K} \in$ NEXP. So NEXP = coNEXP. $\square$

In the following section, we will show a partial result about NOR-autoreducibility for a $\leq^p_{dtt}$-complete set for NEXP in the relativized world.

5 Relativization

While the question whether every $\leq^p_{dtt}$-complete set for NEXP is $\leq^p_{NOR-tt}$-autoreducible is still open, we can prove that it does not hold in the relativized world.

Theorem 5.1. *Relative to some oracle B, there is a $\leq^{p^B}_m$-complete set for* NEXP^B *that is not $\leq^{p^B}_{NOR-tt}$-autoreducible.*

Proof. We assume a polynomial-time computable one-to-one pairing function that can take any finite number of inputs such that its range does not intersect with 0^*.

Let $\{M^B_j\}_{j\geq 1}$ be an enumeration of polynomial-time $\leq^{p^B}_{NOR-tt}$ autoreductions. Notice that M^B_j can now access the oracle B.

Let $\{\text{NEXP}^B_i\}_{i\geq 1}$ be an enumeration of all nondeterministic exponential time oracle Turing machines.

For each $j \geq 1$, suppose that n^j bounds the running time of M^B_j and 2^{n^j} bounds the running time of NEXP^B_j

Let $K^B = \{\langle i, x, l\rangle \mid \text{NEXP}^B_i \text{ accepts input } x \text{ within } l \text{ steps }\}$ be a canonical complete set for NEXP^B.

We will construct sets A and B with the following property

$x \in K^B \iff \langle 0, x\rangle \in A$

which ensure that $K^B \leq^p_m A$, and then A is $\leq^p_m$-hard set for NEXP^B.

We also need the sets A and B such that for any $n \geq 1$, the following property holds

$0^{y^n} \in A \iff M^{B,A}_n$ rejects input 0^{y^n} (value of y_n will be chosen later in the proof)

which ensures that M^B_n is not an autoreduction of A. Then we can conclude that A is not autoreducible for NEXP^B.

We will construct the sets A and B together in stage. In each stage, we will encode K^B into A and diagonalize all $\leq^{p^B}_{NOR-tt}$-reductions using the string 0^{y_n} simultaneously to obtain those above two properties.

Before going into detail of how those sets are constructed, let's define the sequence $\{y_n\}_{n\geq 0}$ such that $y_0 = 1$ and $y_{n+1} = y^n_n + 1$ for every $n \geq 0$.

Suppose that at stage n, the set A has already been constructed up to length $y_n - 1$. In this stage, we will construct the set A for strings of length between y_n and $y_{n+1} - 1$.

Consider all queries q of M^B_n on input 0^{y_n} made to the oracle A when $|q| >= y_n$ and $q = \langle 0, x\rangle$ for some j. Denote Q be the set of all such queries q.

Consider the following cases:

1. If there is a query q' such that $|q'| < y_n$ and $q' \in A$. Then put 0^{y_n} to A and $\langle 0^{2^{y_n}}, 0^{y_n}\rangle$ to B. Finally, for all strings $s = \langle 0, x\rangle$ and $y_n \leq |s| < y_{n+1}$, put s into A if and only if $x \in K^B$.

2. Otherwise, ignore all queries of length smaller than y_n. For every $q' = \langle 0, x\rangle \in Q$ such that $x \in K^B$, choose any accepting path of K^B on input x and denote $Q^{q'}$ be the set of all queries made in that path. Consider the following cases:

   (a) If no such q' exists, then for all strings $s = \langle 0, x\rangle$ and $y_n \leq |s| < y_{n+1}$, put s into A if and only if $x \in K^B$.

   (b) Otherwise, denote P be the union of $Q^{q'}$ for all such q'. Notice that there are no more than y_n^n such q', and for every q', $|Q^{q'}| \leq 2^{|x|} < 2^{y_n^n}$. So $|P| < y_n^n 2^{y_n^n} < 2^{2^{y_n}}$. So there exists a string t of length 2^{y_n} such that $t \notin P$. Put $\langle t, 0^{y_n}\rangle$ into B. Put 0^{y_n} into A. Finally, for all strings of $s = \langle 0, x\rangle$ and $y_n \leq |s| < y_{n+1}$, put s into A if and only if $x \in K^B$.

That is all how two sets A and B are constructed.

Now we will briefly show that the set A belongs to NEXP^B. To determine the membership of an input 0^y, we just need guess one string t of length 2^y and ask one query $\langle t, 0^y\rangle$ to the oracle B; accept if and only if the answer is YES. For other input of the form $\langle 0, x\rangle$, accept if and only if $x \in K^B$. So $A \in \mathrm{NEXP}^B$.

To see that A is not reduced to itself by any $\leq^{p^B}_{NOR-tt}$ autoreduction, we will show that for any M_n, $M_n^{A,B}$ accepts 0^{y_n} if and only if $0^{y_n} \notin A$. In case 1), because there is one query q' such that $q' \in A$, so by NOR-tt reduction, M_n rejects 0^{y_n}. And notice that putting $\langle 0^{2^{y_n}}, 0^{y_n}\rangle$ into B does not affect the membership of q' in A. In case 2a), M_n accepts 0^{y_n} and in this case 0^{y_n} is not put into A, and then it makes M_n not reduce A to itself. In case 2b), M_n does not accept 0^{y_n} and notice that putting $\langle t, 0^{y_n}\rangle$ into B does not affect the memberships of all q' in K^B. And finally 0^{y_n} is added to A to make M_n not reduce A to itself.

Finally it is easy to see that $K^B \leq^p_m A$ because we encode all strings $x \in K^B$ by $\langle 0, x\rangle$ into A and nothing else, except the strings of form 0^*.

So A is many-one complete for NEXP^B that is not $\leq^{p^B}_{NOR-tt}$-autoreducible. □

We note that Theorem 4.1 actually relativizes; that is, for any set B, relative to the oracle B, every $\leq^{p^B}_{dtt}$-complete set for EXP is $\leq^{p^B}_{NOR-tt}$-autoreducible. So we have the following familiar corollary:

Corollary 5.1. *There is a set B such that relative to the oracle B, $\mathrm{EXP}^B \neq \mathrm{NEXP}^B$.*

Buhrman et al. [2] showed that relative to some oracle, there is a $\leq^p_{2\text{-}T}$-complete set for EXP that is not Turing autoreducible. Their technique also works for NEXP. So we have the following theorem:

Theorem 5.2. *Relative to some oracle, there is a $\leq^p_{2\text{-}T}$-complete set for* NEXP *that is not Turing autoreducible.*

Acknowledgements

Our thanks to Nils Wisiol and Benedikt Budig for useful discussions and feedback on drafts of this paper.

References

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

[2] H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet. Using autoreducibility to separate complexity classes. *SIAM Journal on Computing*, 29(5):1497–1520, 2000.

[3] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Structure in Complexity Theory Conference, 1994., Proceedings of the Ninth Annual*, pages 118–133, 1994.

[4] C. Glaßer, D. Nguyen, C. Reitwießner, A. Selman, and M. Witek. Autoreducibility of complete sets for log-space and polynomial-time reductions. ICALP, 2013.

[5] C. Glaßer, M. Ogihara, A. Pavan, A. Selman, and L. Zhang. Autoreducibility and mitoticity. *SIGACT News*, 2009.

[6] Steven Homer. Structural properties of nondeterministic complete sets. In *Structure in Complexity Theory Conference*, pages 3–10, 1990.

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

[8] J. Feigenbaum R. Beigel. On being incoherent without being very hard. *Computational Complexity*, 2:1–17, 1992.

[9] B. Trahtenbrot. On autoreducibility. *Dokl. Akad. Nauk SSSR*, 192, 1970. Translation in Soviet Math. Dokl. 11: 814– 817, 1970.

[10] Osamu Watanabe. A comparison of polynomial time completeness notions. *Theoretical Computer Science*, 54:249–265, 1987.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de