Probabilistic Autoreductions

Liyu Zhang[1] *, Chen Yuan[3], and Haibin Kan[2]

[1] Departmente of Computer Science, University of Texas Rio Grande Valley, Brownsville, Texas, 78520, USA. liyu.zhang@utrgv.edu
[2] School of Physical Mathematical Sciences, Nanyang Technological University, Singapore. yuan0064@e.ntu.edu.sg
[3] Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, China. hbkan@fudan.edu.cn

Abstract. We consider autoreducibility of complete sets for the two common types of *probabilistic* polynomial-time reductions: RP reductions containing one-sided errors on positive input instances only, and BPP reductions containing two-sided errors. Specifically, we focus on the probabilistic counterparts of the deterministic many-one and truth-table autoreductions. We prove that non-trivial complete sets of NP are autoreducible for the *RP many-one reduction*. This extends the result by Glaßer et al. (2007) that complete sets of NP are autoreducible for the deterministic many-one reduction. We also prove that complete sets of classes in the *truth-table Polynomial Hierarchy*, which is the polynomial hierarchy defined using the truth-table reduction instead of the general Turing reduction, are autoreducible with respect to the *BPP truth-table reductions*. This generalizes the result by Buhrman et. al. (2000) that truth-table-complete sets for NP are probabilistically truth-table autoreducible to multiple classes of higher complexity although for a weaker reduction.

Keywords: computational complexity, complete sets, probabilistic polynomial-time autoreductions, probabilistic many-one and truth-table reductions.

1 Introduction

Let r be a reduction between two languages as defined in computational complexity such as the common *many-one* and *Turing* reductions. We say that a language L is r*-autoreducible* if L is reducible to itself via the reduction r where the reduction does not query on the same string as the input. In case that r is the many-one reduction, we require that r outputs a string different from the input in order to be an autoreduction. Researchers started investigating on autoreducibility as early as 1970's [18] although much of the work done then was in the recursive setting. Ambos-Spies [1] translated the notion of autoreducibility to the polynomial-time setting, and Yao [21] considered autoreducibility in the probabilistic polynomial-time setting, which he called *coherence*.

* Research supported in part by NSF grant CCF-1218093

More recently polynomial-time autoreducibilities, which correspond to polynomial-time reductions, gained attention due to its candidacy as a structural property that can be used in the "*Post's program for complexity theory*" [5] that aims at finding a structural/computational property that complete sets of two complexity classes don't share, hereby separating the two complexity classes. Autoreducibility is believed to be possibly one of such properties that will lead to new separation results in the future [3]. Autoreducibility certainly is also an interesting topic in its own right as knowing whether a language is autoreducible or not helps us better understand its computational complexity/structure. This is especially valuable when the language is a complete set for a complexity class for then autoreducibility of that language informs about the intrinsic computational properties that *all* languages in that class might have since they are all reduced to the complete set [4].

During the past two decades or so many exciting results have been obtained regarding autoreducibility of complete sets of common complexity classes. In particular we know now that many-one/Turing complete sets for most natural complexity classes including P, NP, Polynomial Hierarchy, PSPACE and EXP, are many-one/Turing autoreducible [2,3,9]. We refer the reader to Glaßer et al. [8] for a survey. There were also several more recent papers that investigate (non-)autoreducibility of complete sets for high-complexity classes such as NEXP [16] or more restricted reductions such as log-space reductions [11]. Most of those results, however, concern only autoreducibilities under *deterministic* reductions.

In this paper we consider autoreducibilities under *probabilistic* reductions and attempt to see whether and how the probabilistic counterparts of deterministic autoreductions might exhibit different or similar behaviors. Towards that goal we examined two common types of probabilistic reductions, the RP-type reductions, which have one-sided errors on positive input instances only, and BPP-type reductions, which have two-sided errors. We were able to prove that complete sets for NP are autoreducible for the *RP many-one reduction*, the RP-type probabilistic version of the common many-one reduction. This extends the result by Glaßer et al. [9] that complete sets of NP are autoreducible for the (deterministic) polynomial-time many-one reduction, to its RP-type probabilistic counterpart. We also proved that all complete sets of classes in the *truth-table Polynomial Hierarchy*, which is the polynomial hierarchy defined in terms of the polynomial-time truth-table reductions instead of the general Turing reduction, are autoreducible for the *BPP truth-table reduction*, the BPP-type probabilistic version of the common polynomial-time truth-table reduction. This generalizes the result by Buhrman et. al. [3] that truth-table-complete sets for NP are RP truth-table autoreducible to multiple classes of higher complexity but for a weaker reduction (BPP reduction instead of RP reduction). Wagner [20] introduced Θ^{P}*-levels* to the standard Polynomial Hierarchy, which coincides with the Δ^{P}-levels in the truth-table Polynomial Hierarchy. Hence, our result also indicates that all truth-table complete sets in the Θ^{P}-levels are truth-table autoreducible.

We give necessary definitions and notations in Section 2 and present our results in sections 3 and 4.

2 Definitions and Notations

We assume familiarity with basic notions in complexity theory and particularly, common complexity classes such as P, RP, NP, PH and BPP, and polynomial-time reductions including many-one ($\leq_m^p$), truth-table ($\leq_{tt}^p$) and Turing reductions ($\leq_T^p$) [14,13]. Without loss of generality, we use the alphabet $\Sigma = \{0, 1\}$ and all sets we referred to are languages over Σ. We also use *Turing machines and algorithms interchangeably*. Following Glaßer et al. [10], we define a non-trivial set to be a set L where both L and $\overline{L}$ contain at least two distinct elements. This allows us present our results in a simple and concise way. All reductions used in this paper are *polynomial-time computable* unless otherwise specified. A language L is *complete* for a complexity class $\mathcal{C}$ for a reduction r if every language in $\mathcal{C}$ is reducible to L via r. For any algorithm $\mathcal{A}$, we use $\mathcal{A}(x)$ to denote both the execution and output of $\mathcal{A}$ on input x, i.e., "$\mathcal{A}(x)$ accepts" has the same meaning as "$\mathcal{A}(x) = accept$". We use $\mathcal{A}^B(x)$ for the similar meaning if the algorithm/Turing machine $\mathcal{A}$ has oracle access to a set B.

We consider two types of probabilistic reductions, those with one-sided errors on positive input instances only (RP-type) and those with two-sided errors (BPP-type), that correspond to the common deterministic *many-one* and *truth-table* reductions. The truth-table reduction is often also called *non-adaptive Turing reduction* in the literature.

Definition 1 ([19]). *Define a language A to be* RP (randomized polynomial-time) many-one reducible *($\leq_m^{rp}$) to a language B, if there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ and a polynomial q such that the following hold for every $x \in \Sigma^*$:*

- *If $x \in A$, then* $\Pr[\mathcal{A}(x) \in B] \geq \frac{1}{q(|x|)}$.
- *If $x \notin A$, then* $\Pr[\mathcal{A}(x) \in B] = 0$.

Note that we cannot use a fixed polynomial or constant in the definition of RP many-one reductions for otherwise the reduction will not be transitive.

Definition 2. *Define a language A to be* BPP (bounded-error probabilistic and polynomial-time) truth-table reducible *($\leq_{tt}^{bpp}$) to a language B, if there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ with oracle access to B and a (deterministically) polynomial-time computable function g such that the following hold for every $x \in \Sigma^*$:*

- *On input x, $g(x)$ outputs all queries $\mathcal{A}$ will make to B.*
- *If $x \in A$, then* $\Pr[\mathcal{A}^B \text{ accepts } x] \geq \frac{2}{3}$.
- *If $x \notin A$, then* $\Pr[\mathcal{A}^B \text{ accepts } x] \leq \frac{1}{3}$.

Note that in the above definition Algorithm $\mathcal{A}$ is a Monte-Carlo algorithm [15,6], i.e., always runs in polynomial time on all inputs regardless of the probabilistic execution. In addition, Algorithm $\mathcal{A}$ on any fixed input makes the same set of queries to B that can be computed from the input deterministically in polynomial time by the function g. For the latter, we also say that algorithm $\mathcal{A}$ has *truth-table oracle access* to the language B.

Using the standard probability amplification technique we immediately obtain the following:

Corollary 1. *A language A is BPP truth-table reducible to a language B if and only if there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ with truth-table oracle access to B such that the following hold for every $x \in \Sigma^*$:*

- *If $x \in A$, then* $\Pr[\mathcal{A}^B$ *accepts* $x] \geq 1 - \frac{1}{2^{|x|}}$.
- *If $x \notin A$, then* $\Pr[\mathcal{A}^B$ *accepts* $x] < \frac{1}{2^{|x|}}$.

With the above definition of the BPP truth-table reduction, the notion of *BPP truth-table hard* sets can be defined accordingly, which will be used for the proof of our results in Section 4.

Definition 3. *A language A is* BPP truth-table hard *for a complexity class $\mathcal{C}$ if every language $B \in \mathcal{C}$ is BPP truth-table reducible to A.*

Again using the standard probability amplification technique we immediately have the following:

Corollary 2. *A language A is BPP truth-table hard for a complexity class $\mathcal{C}$ if and only if for every language $B \in \mathcal{C}$, there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ with truth-table oracle access to A that decides B with error probability 2^{-n}, where n is the input size.*

Now that we have defined RP many-one reductions and BPP truth-table reductions, the definition of the corresponding autoreductions follow straightforwardly.

3 RP Many-one Autoreductions

In this section we extend the result by Glaßer et al. [9] that many-one complete sets for NP are many-one autoreducible to the RP many-one reduction. Although the overall proof strategy is similar to that of the previous result, the additional part that argues why the adapted autoreduction output a correct value with the desired probability is not trivial at all.

Theorem 1. *If L is a non-trivial $\leq_m^{rp}$-complete set for* NP*, then L is $\leq_m^{rp}$-autoreducible.*

Proof. Let L be a nontrivial $\leq_m^{rp}$-complete set for NP. Then there exist strings a_1 and a_2 that belong to L, and b_1 and b_2 that belong to $\overline{L}$. Let N be a nondeterministic polynomial-time Turing machine that accepts L in time $p(n)$ for some polynomial p. Without loss of generality, we assume that all computation paths of N are of length $m = p(n)$ on inputs of length n. Define the following "left-set" [17] for L:

$$left(L) = \{\langle x, u\rangle \mid x \in L, \text{ and there is an accepting path } v \text{ of } N \text{ on } x \text{ where } u \leq v.\}$$

Here $\leq$ represents the common dictionary order among strings. Clearly $left(L) \in$ NP. Hence, there is a RP many-one reduction f from $left(L)$ to L such that a polynomial q exists, where

- for every $\langle x, y\rangle \in left(L)$, $\Pr[f(\langle x, y\rangle) \in L] \geq \frac{1}{q(|x|)}$, and
- for every $\langle x, y\rangle \notin left(L)$, $\Pr[f(\langle x, y\rangle) \in L] = 0$.

Now define a probabilistic polynomial-time computable function f' as follows: On input $\langle x, y\rangle$, where $|x| = n$ and $|y| = p(n)$, run f on $\langle x, y\rangle$ for $2q(n)\ln p(n)$ times and output x if at least one of the $2q(n)\ln p(n)$ runs of f outputs x, and output z if otherwise, where $z \neq x$ is one of the outputs by the $2q(n)\ln p(n)$ runs of f.

Consider the following function g:

```
1       Input x
2       Let m ← p(|x|) and z ← f'(⟨x, 0^m⟩)
3       If z ≠ x then output z
4       If f'(⟨x, 1^m⟩) = x then
5          If N(x) accepts along the path 1^m then
6             Output a string in {a1, a2} − {x}
7          Else
8             Output a string in {b1, b2} − {x}
9       Determine w of length m such that
              f'(⟨x, w⟩) = x ≠ f'(⟨x, w + 1⟩) = y
10      If N(x) accepts along w then
              output a string in {a1, a2} − {x}
11      Else
12            Run f' on ⟨x, w + 1⟩ again and yield output y'
13            Output y' if y' ≠ x, or {b1, b2} − {x} otherwise
```

Line 9 of the above function g can be executed in polynomial time since f' is a polynomial-time computable function and a standard binary search can be used to determine w as specified. Hence, the function g is polynomial-time computable. It is also clear that $g(x)$ always outputs a string $s \neq x$. Now it remains to show that $L \leq_m^{rp} L$ via g.

Assume $x \notin L$. Then for every $w \in \Sigma^m$, $\langle x, w\rangle \notin left(L)$ and hence, $f(\langle x, w\rangle) \notin L$. Therefore, the function g on input x outputs $z = f'(\langle x, w\rangle) \notin L$ in line 3, or output a string not in L in either line 8 or line 13.

Now assume $x \in L$. Let us define a computation path u (of length m) to be *good* if $\Pr[f(\langle x, u\rangle) = x] \leq 1/2q(n)$, and *bad* otherwise. We immediately observe the following claim:

Claim 1. For every $u \in \Sigma^m$ and $s = f'(\langle x, u\rangle)$,

i. if u is good, then
   (a) $\Pr[s \neq x] \geq \frac{1}{2p(n)}$, and
   (b) $\langle x, u\rangle \in left(L) \Rightarrow \Pr[s \in L \mid s \neq x] \geq \frac{1}{r(n)}$ for some polynomial $r(n)$.
ii. if u is bad, then $\Pr[s = x] > 1 - \frac{2}{p(n)}$.

Due to space limit, we omit the proof of Claim 1. Now we consider the following cases:

Case 1: 0^m is good. Let $z = f'(\langle x, 0^m\rangle)$. By i(a) of Claim 1, we have $\Pr[z \neq x] \geq \frac{1}{2p(n)}$. Hence, with probability at least $\frac{1}{2p(n)}$ the function g outputs $z = f'(\langle x, 0^m\rangle) \neq x$ in line 3. Furthermore, the probability that $z \in L$ in this case is $\Pr[z \in L \mid z \neq x] \geq 1/r(n)$ for some polynomial r, by i(b) of Claim 1. Therefore, the function g outputs a string $z \in L$ where $z \neq x$ with probability at least $\frac{1}{2p(n)} \cdot \frac{1}{r(n)}$ in case 1.

Case 2: Both 0^m and 1^m are bad. By ii of Claim 1,

$$\Pr[f'(\langle x, 0^m\rangle) = x] > 1 - \frac{2}{p(n)}, \text{ and}$$

$$\Pr[f'(\langle x, 1^m\rangle) = x] > 1 - \frac{2}{p(n)}.$$

Hence, with probability at least $(1 - \frac{2}{p(n)})^2 > 1 - \frac{4}{p(n)}$ the function g reaches line 5. Since $x \in L$, it follows that $\langle x, 1^m\rangle \in left(L)$ and so 1^m is an accepting path of N on x. Consequently g outputs in line 6 a string in $\{a1, a2\} - \{x\}$. Therefore, the function g outputs a string in L that does not equal to x with probability at least $1 - \frac{4}{p(n)}$ in case 2.

Case 3: 0^m is bad and 1^m is good. By Claim 1 again, we have

$$\Pr[f'(\langle x, 0^m\rangle) = x] > 1 - \frac{2}{p(n)}, \text{ and}$$

$$\Pr[f'(\langle x, 1^m\rangle) \neq x] \geq \frac{1}{2p(n)}.$$

Hence, with probability at least $(1 - \frac{2}{p(n)})\frac{1}{2p(n)}$ function g reaches line 9. Note that using a standard binary search to determine a string $w \in \Sigma^m$, where $f'(\langle x, w\rangle) = x$ and $f'(\langle x, w + 1\rangle) \neq x$, requires computing $f'(\langle x, u\rangle)$ for $m - 1$ strings u in Σ^m in addition to 0^m and 1^m. We denote those strings by $u_1, \ldots, u_{m-1}$ and also let $u_0 = 0^m$ and $u_m = 1^m$. Now consider the following statement S_i for each $i \in \{0, 1, \ldots, m\}$:

$$S_i : u_i \text{ is bad} \Rightarrow f'(\langle x, u_i\rangle) = x.$$

Assume that the execution of g has reached line 9. Then $f'(\langle x, u_0\rangle) = x$ and $f'(\langle x, u_m\rangle) \neq x$ as computed in line 2 and 4 of function g, respectively. Hence, both S_0 and S_m hold since u_0 is bad and u_m is good. Now for each $i \in \{1, \ldots, m-1\}$, let $z_i = f'(\langle x, u_i\rangle)$. Then Statement S_i fails with the following probability:

$$\Pr[\overline{S_i}] = \Pr[z_i \neq x \text{ and } u_i \text{ is bad}] \leq \Pr[z_i \neq x \mid u_i \text{ is bad}] < \frac{2}{p(n)}.$$

Therefore, all S_i's hold with probability at least $(1-2/p(n))^{p(n)-1} \geq 1/(2e^2)$. Now assume that all S_i's hold. Let w be the string determined in line 9 of function g. If w is an accepting path of N on x, then function g output a string in $L-\{x\}$ in line 10. Otherwise, it holds that $\langle x, w\rangle \in left(L)$ since $f'(\langle x, w\rangle) = x \in L$, and hence it must be the case that $\langle x, w+1\rangle \in left(L)$. Also, $w+1$ must be good since $f'(\langle x, w+1\rangle) \neq x$. Thus, for the string y' produced in line 12, $\Pr[y' \in L \mid y' \neq x] \geq \frac{1}{r(n)}$ for some polynomial r by i(b) of Claim 1. It follows in this case that the function g outputs a string in L that does not equal x with probability at least that all the following events occur:

- The function g reaches line 9,
- All statements S_i hold for $i \in \{0, 1, 2, \ldots, m\}$, and
- The function g outputs a correct string in line 10 or line 13 given that the string $w+1$ determined in line 9 is good.

By our previous discussion, the probability referred to above is at least

$$((1 - \frac{2}{p(n)})\frac{1}{2p(n)}) \cdot \frac{1}{2e^2} \cdot ((1 - \frac{1}{2q(n)})\frac{1}{r(n)}) \geq \frac{1}{16e^2 p(n) r(n)}.$$

We have shown in all cases that the function g on an input $x \in L$ produces a string $y \neq x$ where $y \in L$ with probability no less than $\frac{1}{q'(n)}$ for some polynomial $q'(n)$. This finishes the proof of Theorem 1.

□

We believe that similar results to Theorem 1 would hold for RP versions of those reductions such as 1-tt and dtt, as well as for RP many-one complete sets of all the complexity classes as listed in Glaßer et al. [9] for which deterministic autoreducibilities of complete sets have been proved. However, we have not considered all the proof details for those sets yet.

4 BPP Truth-table Autoreductions

In this section we consider *BPP truth-table reductions*. We prove that complete sets of classes in the *truth-table Polynomial Hierarchy* ($\mathrm{PH}^{\mathrm{tt}}$), which is defined below, are autoreducible for the BPP truth-table reductions. This generalizes the result by Buhrman et. al. [3] that truth-table complete sets for NP are probabilistically (in fact, RP) truth-table autoreducible.

Given a complexity class $\mathcal{C}$, we use $\mathrm{P}^{tt[\mathcal{C}]}$ ($\mathrm{NP}^{tt[\mathcal{C}]}$) to denote the class of languages decidable by a (nondeterministic) polynomial-time Turing machine with truth-table oracle access to a language in $\mathcal{C}$.

Definition 4.

$$\Sigma_0^{\mathrm{P,tt}} = \Pi_0^{\mathrm{P,tt}} = \Delta_0^{\mathrm{P,tt}} = \mathrm{P}$$

For $k \geq 1$,

$$\Sigma_{\mathrm{k}}^{\mathrm{P,tt}} = \mathrm{NP}^{tt[\Sigma_{\mathrm{k-1}}^{\mathrm{P,tt}}]},$$

$$\Pi_{\mathrm{k}}^{\mathrm{P,tt}} = \mathrm{co}\Sigma_{\mathrm{k}}^{\mathrm{P,tt}} = \{L \mid \overline{L} \in \Sigma_{\mathrm{k}}^{\mathrm{P,tt}}\}, \textit{ and}$$

$$\Delta_{\mathrm{k}}^{\mathrm{P,tt}} = \mathrm{P}^{tt[\Sigma_{\mathrm{k-1}}^{\mathrm{P,tt}}]}$$

$$\mathrm{PH}^{\mathrm{tt}} = \bigcup_{k \geq 0} \Sigma_{\mathrm{k}}^{\mathrm{P,tt}} = \bigcup_{k \geq 0} \Pi_{\mathrm{k}}^{\mathrm{P,tt}}$$

Clearly, $\mathrm{PH}^{\mathrm{tt}}$ as defined above is the same as the standard PH except thats truth-table reductions are used instead of the general Turing reductions.

Towards of the goal of proving the main result of this section, we first observe that Valiant and Vazirani's result [19] can be extended to *relativized CNF formulas* in a straightforward manner.

Definition 5 ([7]). *For any language A, a CNF formula* relative to A, *ϕ^A, is a CNF formula with each clause of the following form*

$$x_{i_1} \vee x_{i_2} \cdots \vee x_{i_u} \vee y_{i_1} \vee y_{i_2} \vee \cdots \vee y_{i_v},$$

where x_{i_j}'s are literals, and each y_{i_j} is a predicate of the form $A(w)$ or $\overline{A}(w)$ for some string w consisting of literals, 0's, 1's and other predicates of the form $A(w')$ or $\overline{A}(w')$.

Definition 6. *A relativized formula is a formula* truth-table relative *to a language A, $\phi^{tt[A]}$, if the following conditions hold:*

1. *$\phi^{tt[A]}$ is of the form $x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k} \wedge \mathrm{TRUE} \wedge F$, where x_{i_1}, x_{i_2}, ..., x_{i_k} are literals and F is a formula relative to A.*
2. *Every variable appearing inside the predicate A in F does not appear outside predicate A unless it is one of those x_{i_j}'s ($1 \leq j \leq k$) or its negation as stated in 1.*

If a CNF formula ϕ is one relative to some language A, we also say that ϕ is a *relativized (CNF) formula*. In case ϕ is truth-table relative to to A, ϕ is also called a *truth-table relativized (CNF) formula*, or simply *truth-table formula*.

Note that in order to satisfy a truth-table formula $x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k} \wedge \mathrm{TRUE} \wedge F$, all x_{i_j}'s must be true. This induces a polynomial-time algorithm that on a truth-table relativized formula $\phi^{tt[A]}$, outputs all queries to A that are needed for evaluating $\phi^{tt[A]}$ under a satisfying assignment and do not depend on any particular assignment of $\phi^{tt[A]}$.

With the above definitions, we say that a relativized formula ϕ^A is *satisfiable* if $\phi^A(a)$ evaluates to true for some assignment a, where for each occurrence of a predicate $A(w)$, $A(w) = 1$ if and only if $w \in A$ with the value of w determined by a and values of other predicates of the form $A(u)$ that appears in w. In addition, we say that ϕ^A has a unique satisfying assignment a if $\phi^A(a)$ evaluates to true and for every other a' where $\phi^A(a')$ is true, a and a' coincides on variables appearing outside of the predicate A.

Definition 7. *We define the following languages.*

- SAT *is the set of satisfiable CNF formulas.*
- SAT^A *(*$\mathrm{SAT}^{tt[A]}$*) is the set of satisfiable CNF formulas (truth-table) relative to* A*.*
- USAT^A *(*$\mathrm{USAT}^{tt[A]}$*) is the set of CNF formula (truth-table) relative to* A *that have unique satisfying assignments.*

Let NP^A ($\mathrm{NP}^{tt[A]}$) denote the class of languages decidable by nondeterministic polynomial-time Turing machines with (truth-table) oracle A. Goldsmith and Joseph [12] proved that for every language A, SAT^A is complete for NP^A via a many-one reduction that does not use any oracle. A straightforward adaption of their proof yields a similar result for $\mathrm{SAT}^{tt[A]}$.

Lemma 1. *For any language* A*,* $\mathrm{SAT}^{tt[A]}$ *is complete for* $\mathrm{NP}^{tt[A]}$ *via a many-one reduction that does not use any oracle.*

Theorem 2 ([19]). SAT *is reducible to* USAT *via a RP many-one reduction* r *such that* $r(\phi) \notin \overline{\mathrm{SAT}}$ *with probability* 1 *if* $\phi \notin \mathrm{SAT}$*, and* $r(\phi) \in \mathrm{USAT}$ *with probability at least* $1/(4|\phi|)$ *otherwise.*

Note that the proof of Theorem 2 is relativizable and hence we observe the following corollary immediately.

Corollary 3. *For any language* A*,* $\mathrm{SAT}^{tt[A]}$ *is reducible to* $\mathrm{USAT}^{tt[A]}$ *via a RP many-one reduction* r *such that* $r(\phi) \in \overline{\mathrm{SAT}^{tt[A]}}$ *with probability* 1 *if* $\phi \in \overline{\mathrm{SAT}^{tt[A]}}$*, and* $r(\phi) \in \mathrm{USAT}^{tt[A]}$ *with probability at least* $1/(4|\phi|)$ *if* $\phi \in \mathrm{SAT}^{tt[A]}$*. In addition, the reduction* r *does not use any oracle or change* w *for each occurrence of* $A(w)$ *or* $\overline{A}(w)$ *in* ϕ*.*

Buhrman et al. [3] proved that truth-table complete sets for NP are autoreducible for the RP truth-table reductions. The key element of the proof is a probabilistic algorithm that utilizes Theorem 2 and decides the satisfiability of a CNF formula with oracle access to a truth-table complete set for NP where a particular query is avoided. With Corollary 3, we are able to prove a result similar to Buhrman et al.'s for all complete sets in $\mathrm{PH}^{\mathrm{tt}}$.

Our proof uses the following languages consisting of relativizable propositional formulas.

Definition 8. – $\mathrm{SAT}^{(1),tt} = \mathrm{SAT}$*. For every* $k \geq 2$*,* $\mathrm{SAT}^{(k),tt} = \mathrm{SAT}^{tt[\mathrm{SAT}^{(k-1),tt}]}$.

– $\mathrm{USAT}^{(1),tt} = \mathrm{USAT}$. *For every* $k \geq 2$, $\mathrm{USAT}^{(k),tt} = \mathrm{USAT}^{tt[\mathrm{SAT}^{(k-1),tt}]}$.
– $\mathrm{F}^{(1)} = \mathrm{F}^{(1),tt}$ *is the set of CNF propositional formulas. For every* $k \geq 2$.
– $\mathrm{F}^{(k)}$ *(*$\mathrm{F}^{(k),tt}$*) is the set of CNF propositional formulas (truth-table) relative to* $\mathrm{SAT}^{(k-1),tt}$.

Theorem 3. *For every* $k \geq 1$ *and every BPP truth-table hard set* L_k *for* $\Sigma_{\mathrm{k}}^{\mathrm{P,tt}}$, *there is a probabilistic algorithm* $\mathcal{A}_k$ *that on input* $\langle \phi, y, 0^n \rangle$ *runs in polynomial time in* $|\phi|$ *and* n, *where* $\phi \in \mathrm{F}^{(k),tt}$, *and decides the satisfiability of* ϕ *with error probability at most* $2^{-\max(n,|\phi|)}$. *In addition,* $\mathcal{A}_k$ *makes only truth-table queries to* L_k *and does not query on* y.

Proof. We prove the theorem by induction. Theorem 4.10 in Buhrman et al. [3] essentially established the proof for the base case $k = 1$.

Now we prove the induction step. Let L be a BPP truth-table hard set for $\Sigma_{\mathrm{k}}^{\mathrm{P,tt}}$ where $k > 1$. Clearly, L is also BPP truth-table hard for $\Sigma_{\mathrm{i}}^{\mathrm{P,tt}}$ for $1 \leq i \leq k-1$. Let $\mathcal{A}_1, \mathcal{A}_1, \cdots, \mathcal{A}_{k-1}$ be the probabilistic algorithms that satisfy the properties as stated in the lemma for $1 \leq i \leq k-1$ and use L as the oracle.

Define

$$T = \left\{ \langle \phi, 0^i \rangle \;\middle|\; \begin{array}{l} \phi \in \mathrm{SAT}^{(k),tt} \text{ has a satisfying assignment} \\ \text{where the } i\text{-th variable is true.} \end{array} \right\}$$

Since $T \in \Sigma_{\mathrm{k}}^{\mathrm{P,tt}}$, there is a BPP truth-table reduction g from T to L. Now let r be the RP many-one reduction of $\mathrm{SAT}^{(k),tt}$ to $\mathrm{USAT}^{(k),tt}$ as stated in Corollary 3. Hence, if $\phi \in \overline{\mathrm{SAT}^{(k),tt}}$, then $r(\phi) \in \mathrm{USAT}^{(k),tt}$ with probability at least $1/4|\phi|$. Also, $r(\phi) \in \overline{\mathrm{SAT}^{(k),tt}}$ with probability 1 if $\phi \in \overline{\mathrm{SAT}^{(k),tt}}$.

Consider the following algorithm $\mathcal{B}$:

```
1       Input ⟨φ, y, 0^n⟩
2       If φ ∉ F^(k),tt, REJECT
3       ψ := r(φ)
4       Use g to determine the memberships of ⟨ψ, 0^i⟩ in T for
        1 ≤ i ≤ m, where m is the number of variables in ψ,
        with oracles L − {y} and L ∪ {y}, respectively.
5       Let a_0 and a_1 be the two assignments induced by the two
        sets of memberships of ⟨ψ, 0^i⟩ in T, respectively,
        as determined in Line 4.
6       Evaluating ψ(a_0) and ψ(a_1) by calling A_{k−1} on
        ⟨q, y, 0^max(|φ|,n)⟩ for each membership query SAT^(k−1),tt(q).
7       If the above ψ(a_0) or ψ(a_1) evaluate to true, ACCEPT.
8       REJECT.
```

We again omit some part of the proof here due to space limit, but just state that one can show that the algorithm $\mathcal{B}$ has the following properties on input $\langle \phi, y, 0^n \rangle$:

– runs in polynomial time in $|\phi|$ and n,

– makes nonadaptive queries only to L, none of which is y,
– accepts ϕ with probability at most ϵ_1 if $\phi \notin \mathrm{SAT}^{(k),tt}$, and
– accepts ϕ with probability at least ϵ_2 if $\phi \in \mathrm{SAT}^{(k),tt}$,
– where $\epsilon_1 = o(1/\max(|\phi|, n))$ and $\epsilon_2 = \Omega(1/\max(|\phi|, n))$ for sufficiently large ϕ and n.

Then we use the standard amplification technique to obtain an algorithm $\mathcal{A}_k$ that consists of multiple runs of $\mathcal{B}'$ and has the error probability as stated in the lemma.

□

Corollary 4. *Every BPP truth-table-complete set of every class in* $\mathrm{PH}^{\mathrm{tt}}$ *is BPP truth-table autoreducible.*

Proof. The corollary is trivially true for $\Sigma_0^{\mathrm{P,tt}} = \Pi_0^{\mathrm{P,tt}} = \Delta_0^{\mathrm{P,tt}} = \mathrm{P}$ since every language in P can be decided by a deterministic polynomial-time Turing machine without any oracle access.

Now let L_a be a BPP truth-table-complete set for $\Sigma_{\mathrm{k}}^{\mathrm{P,tt}}$ for $k \geq 1$. Then L_a reduces to $\mathrm{SAT}^{(k),tt}$ via a many-one reduction f that does not use any oracle. Now let $\mathcal{A}_k$ be the probabilistic polynomial-time algorithm as stated in Theorem 3 that makes nonadaptive queries to L_a. Consider the following algorithm $\mathcal{A}_a$: On input x, compute $\phi = f(x)$. Then run $\mathcal{A}_k$ on $\langle \phi, x, 0^{|x|} \rangle$ and accept if and only if $\mathcal{A}_k$ accepts. It is not hard to show that $\mathcal{A}_a$ is a BPP truth-table autoreduction for L_a. It also follows immediately that a BPP truth-table-complete set L_b for $\Pi_{\mathrm{k}}^{\mathrm{P,tt}}$ is BPP truth-table autoreducible since $\overline{L_b}$ is a BPP truth-table-complete set for $\Sigma_{\mathrm{k}}^{\mathrm{P,tt}}$ and a set is autoreducible if and only if the complement of the set is autoreducible for the same reduction.

Now consider a BPP Truth-table-complete set L_c for $\Delta_{\mathrm{k}}^{\mathrm{P,tt}}$ for $k \geq 1$. Then there exists a deterministic polynomial-time Turing machine M that decides L_c with truth-table oracle access to $\mathrm{SAT}^{(k-1),tt}$. Let $\mathcal{A}_{k-1}$ be the probabilistic polynomial-time algorithm as stated in Theorem 3 that makes nonadaptive queries to L_c. Now consider the following algorithm $\mathcal{A}_c$: On input x, run M on x and resolve each query on q by running $\mathcal{A}_{k-1}$ on $\langle q, x, 0^{|x|} \rangle$; accept x if and only if M accepts x. Using the properties that $\mathcal{A}_{k-1}$ has according to Theorem 3 we can show that $\mathcal{A}_c$ is a BPP truth-table autoreduction for L_c.

□

Wagner [20] defined the Θ^{P}-levels by $\Theta_0^{\mathrm{P}} = \mathrm{P}$ and $\Theta_{\mathrm{k+1}}^{\mathrm{P}} = L^{\Sigma_{\mathrm{k}}^{\mathrm{P}}}$, where L denotes a log-space oracle Turing machine, and recommended including Θ^{P}-levels in the standard PH. He also proved for every $k \geq 1$ that $\Theta_{\mathrm{k}}^{\mathrm{P}} = \mathrm{P}^{tt[\Sigma_{\mathrm{k-1}}^{\mathrm{P}}]} = \Delta_{\mathrm{k}}^{\mathrm{P,tt}}$. Hence, we immediately have the following corollary.

Corollary 5. *For every* $k \geq 0$*, every truth-table complete set for each class in* $\Theta_{\mathrm{k}}^{\mathrm{P}}$ *is probabilistically truth-table autoreducible.*

References

1. Ambos-Spies, K.: On the structure of the polynomial time degrees of recursive sets. Habilitationsschrift, Zur Erlangung der Venia Legendi Für das Fach Informatik an der Abteilung Informatik der Universität Dortmund (September 1984)
2. Beigel, R., Feigenbaum, J.: On being incoherent without being hard. Computation Complexity 2(1), 1–17 (1992)
3. Buhrman, H., Fortnow, L., van Melkebeek, D., Torenvliet, L.: Using autoreducibility to separate complexity classes. SIAM Journal on Computing 29(5), 1497–1520 (2000)
4. Buhrman, H., Torenvliet, L.: On the structure of complete sets. In: Proceedings 9th Structure in Complexity Theory. pp. 118–133 (1994)
5. Buhrman, H., Torenvliet, L.: A Post's program for complexity theory. Bulleting of the EATCS 85, 41–51 (2005)
6. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. The MIT Press, 3e edn. (2009)
7. Fortnow, L.: The role of relativization in complexity theory. Bulletin of the European Association for Theoretical Computer Science 52, 52–229 (1994)
8. Glaßer, C., Ogihara, M., Pavan, A., Selman, A., Zhang, L.: Autoreducibility and mitoticity. ACM SIGACT News 40(3), 60–76 (2009)
9. Glaßer, C., Ogihara, M., Pavan, A., Selman, A.L., Zhang, L.: Autoreducibility, mitoticity, and immunity. Journal of Computer and System Sciences 73, 735–754 (2007)
10. Glaßer, C., Pavan, A., Selman, A., Zhang, L.: Splitting NP-complete sets. SIAM Journal on Computing 37(5), 1517–1535 (2008)
11. Glaßer, C., Witek, M.: Autoreducibility and mitoticity of logspace-complete sets for NP and other classes. In: Proceedings 39th International Symposium on Mathematical Foundations of Computer Science, part II. pp. 311–323. Budapest, Hungary (August 2014)
12. Goldsmith, J., Joseph, D.: Three results on the polynomial isomorphism of complete sets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science. pp. 390–397. IEEE, New York (1986)
13. Hemaspaandra, L., Ogihara, M.: The Complexity Theory Companion. Springer (2002)
14. Homer, S., Selman, A.: Computability and Complexity Theory. Texts in Computer Science, Springer, New York, 2e edn. (December 2011)
15. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press (1995)
16. Nguyen, D., Selman, A.: Non-autoreducible sets for NEXP. In: Proceedings 31st Symposium on Theoretical Aspects of Computer Science. pp. 590–601. LIPICS, Lyon, France (March 2014)
17. Ogiwara, M., Watanabe, O.: On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. SIAM Journal of Computing 20(3), 471–483 (1991)
18. Trakhtenbrot, B.: On autoreducibility. Dokl. Akad. Nauk SSSR 192(6), 1224–1227 (1970), translation in Soviet Math. Dokl. 11(3): 814C817, 1790
19. Valiant, L., Vazirani, V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47, 85–93 (1986)
20. Wagner, K.: Bounded query classes. SIAM Journal on Computing 19(5), 833–846 (October 1990)
21. Yao, A.: Coherent functions and program checkers. In: Proceedings of the 22nd Annual Symposium on Theory of Computing. pp. 89–94 (1990)