On one query self-reducible sets

Antoni Lozano
Mitsunori Ogiwara

Report LSI–91–16

Resum

Estudiem els conjunts auto-reduïbles per mots decreixents amb una pregunta (1-md auto-reduïbles), introduïts per Lozano i Torán. Es defineixen com els auto-reduïbles usuals amb la peculiaritat que la màquina que calcula l'auto-reducció només pregunta un mot i aquest és més petit que l'entrada en ordre lexicogràfic. En primer lloc demostrem que per a totes les classes de comptatge definides per un predicat sobre el nombre de camins acceptadors, existeixen conjunts complets 1-md auto-reduïbles. Fent servir aquest fet, demostrem que per a qualsevol classe K presa de $\{\mathrm{NP}, \mathrm{PP}, \mathrm{C_{=}P}, \mathrm{MOD_2P}, \mathrm{MOD_3P}, \cdots\}$ es compleix que (1) si existeix un conjunt espars $\leq^{\mathrm{P}}_{\mathrm{btt}}$-difícil per a K llavors $K = \mathrm{P}$, i (2) si existeix un conjunt espars $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-difícil per a K llavors $K \subseteq \mathrm{NP} \cap \mathrm{co\text{-}NP}$. El resultat principal també implica que es compleixen les mateixes propietats per a la classe PSPACE. Això generalitza un resultat recent d'Ogiwara i Watanabe a totes les classes de complexitat esmentades.

Abstract

We study one word-decreasing self-reducible sets, which were introduced by Lozano and Torán. These are usual self-reducible sets with the peculiarity that the self-reducibility machine makes at most one query and this is a word lexicographically smaller than the input. We show first that for all counting classes defined by a predicate on the number of accepting paths there exist complete sets which are one word-decreasing self-reducible. Using this fact we can prove that for any class K chosen from $\{\mathrm{NP}, \mathrm{PP}, \mathrm{C_{=}P}, \mathrm{MOD_2P}, \mathrm{MOD_3P}, \cdots\}$ it holds that (1) if there is a sparse $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard set for K then $K = \mathrm{P}$, and (2) if there is a sparse $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-hard set for K then $K \subseteq \mathrm{NP} \cap \mathrm{co\text{-}NP}$. The main result also shows that the same facts hold for the class PSPACE. This generalizes a recent result from Ogiwara and Watanabe to the mentioned complexity classes.

On One Query Self-Reducible Sets

Mitsunori Ogiwara

Dept. of Comp. Science and Information Math.
University of Electro-Communications
1-5-1 Chofugaoka, Chofu-shi
Tokyo 182, Japan
Email: ogiwara@cso.cs.uec.ac.jp

Antoni Lozano *

Department of Software (L.S.I.)
Universitat Politècnica de Catalunya
Pau Gargallo 5
08028 Barcelona, Spain
Email: lozano@lsi.upc.es

Abstract

We study one word-decreasing self-reducible sets, which are introduced by Lozano and Torán [21]. These are usual self-reducible sets with the peculiarity that the self-reducibility machine makes at most one query to a word lexicographically smaller than the input. We show first that for all counting classes defined by a predicate on the number of accepting paths there exist complete sets which are one word-decreasing self-reducible. Using this fact we can prove that for any class K chosen from {NP, PP, $C_=P$, MOD_2P, MOD_3P, $\cdots$} it holds that (1) if there is a sparse $\leq^P_{btt}$-hard set for K then $K = P$, and (2) if there is a sparse $\leq^{SN}_{btt}$-hard set for K then $K \subseteq NP \cap co\text{-}NP$. The main result also shows that the same facts hold for the class PSPACE. This generalizes the result from Ogiwara and Watanabe [24] to the mentioned complexity classes.

1 Introduction

One of the central roles in the study of structural complexity theory resides in finding structural differences or similarities among complexity classes. Since almost every complexity class is defined by using some resource bounded computational model, finding relationships among such classes sometimes requires us to specify different computational models, and therefore, it seems tremendously hard to find such relationships. Above all, as subclasses of PSPACE, there have been introduced many complexity classes [4, 9, 12, 25, 28, 35]. For example, Gill, and independently Simon defined PP as the class of sets having probabilistic polynomial-time acceptors with error probability $< 1/2$ [12, 28]. Papadimitriou and Zachos defined $\oplus P$ as the class of sets for which there is a polynomial-time nondeterministic Turing machine such that a string is in the set if and only if the machine has an odd number of accepting computation paths for the string [25]. Based on this definition, Beigel, Gill, and Hertrampf, and independently, Cai and Hemachandra defined MOD_kP as the class of sets for which there exists a polynomial-time nondeterministic Turing machine such that for every x, x is in the set if and only if the number of accepting computation paths of the machine on input x is not a multiple of k, where $k \geq 2$ [4, 9]. Wagner introduced $C_=P$ (respectively, CP) as the class of sets for which there exist a polynomial-time computable function and a polynomial-time nondeterministic Turing machine such that a string x is in the set if and only if the number of accepting computation paths of the machine on the input x is equal to (respectively, larger than or equal to) the value of the function for x [35]. Also, he showed that CP is equal to PP. So, concerning the classes that are located between NP and PSPACE, the most important unsolved questions are the following:

1. Does the polynomial-time hierarchy have infinite levels?
2. Is any class defined above included in the polynomial-time hierarchy?

Some results that settle these questions partially have been obtained in [5, 6, 18, 29, 31, 32, 33]. Nevertheless, until now, neither of the above questions is solved.

On the other hand, it is widely known that we can classify sets into some categories by using different reducibilities to sets of small density [8, 16, 17]. Especially,

*This research was partially supported by ESPRIT-II Basic Research Actions Program of the European Community under contract No. 3075 (project ALCOM).

a set having a census function bounded above by some polynomial is called sparse. Relative to this notion, the following questions have been considered by many researchers [7, 11, 14, 15, 22, 24, 34, 37, 39, 40].

1. For a class K and a reducibility $\leq_r$, is there any sparse set to which every set in K is $\leq_r$-reducible?
2. Suppose that every set in K is $\leq_r$-reducible to some sparse set. Will then any unexpected inclusions follow?

As a matter of fact, after Berman and Hartmanis conjectured that all $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete sets for NP are P-isomorphic, and thereby conjectured that there are neither sparse nor co-sparse $\leq^{\mathrm{P}}_{\mathrm{m}}$-hard sets for NP [7], reducibilities of NP sets to sparse sets have been considered for a long time [11, 22, 23, 34, 37, 40]. And the question whether NP having sparse $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard sets implies $\mathrm{P} = \mathrm{NP}$ or not had been left open for a long time until Ogiwara and Watanabe solved the question affirmatively [24].

In order to settle this question, they introduced the notion of 'left-sets', which is a certain type of self-reducible structure, and showed that every 'left-set' in NP which is $\leq^{\mathrm{P}}_{\mathrm{btt}}$-reducible to sparse sets is already in P. The proof technique takes advantage of the structure of left-sets and of the structure resulting from the reduction to a sparse set, to show that any set with these two characteristics can be decided deterministically in polynomial time. Furthermore, Ogiwara extended the notion of left-sets and showed that the existence of $\leq^{\mathrm{P}}_{\mathrm{btt}}$ (respectively, $\leq^{\mathrm{SN}}_{\mathrm{btt}}$) hard sets for PP implies $\mathrm{PP} = \mathrm{P}$ (respectively, $\mathrm{PP} = \mathrm{NP}$) [23]. So, in order to extend the result to more classes, it can seem useful to consider sets with a more rich internal structure than that of left-sets. We consider, for this goal, one word-decreasing self-reducible sets, which have the desired properties: its time complexity decreases if they are reducible to sparse sets, and we can find self-reducible sets of this type that are complete for the classes NP, PP, $\mathrm{MOD}_k\mathrm{P}$, $\mathrm{C}_=\mathrm{P}$, and PSPACE.

One word-decreasing self-reducible sets were introduced by Lozano and Torán [21] as a variation of Balcázar's wdq self-reducible sets [2], and they also constitute a generalization of left-sets [23, 24]. A set A is one word-decreasing self-reducible if there exists a polynomial-time deterministic oracle Turing machine which accepts A with oracle A itself in such a way that for every input x, the machine queries at most one string to its oracle and the query string is lexicographically smaller than x. We define strict one word-decreasing self-reducible sets as a restriction of these sets. In section 3, we will consider the existence of one-word decreasing self-reducible sets in NP, $\mathrm{C}_=\mathrm{P}$, PP, $\mathrm{MOD}_k\mathrm{P}$, and show that

1. PP and $\mathrm{C}_=\mathrm{P}$ have a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set which is one word-decreasing self-reducible, and
2. NP, $\mathrm{MOD}_2\mathrm{P}$, $\mathrm{MOD}_3\mathrm{P}$, $\cdots$, have a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set which is strictly one word-decreasing self-reducible.

As for the reducibility, we will consider polynomial-time bounded truth-table reducibility as well as strong nondeterministic polynomial-time bounded truth-table reducibility. The last one was introduced by Adleman and Manders [1] as a reducibility having the computational power of $\mathrm{NP} \cap \mathrm{co\text{-}NP}$, and extended and studied by Long [19, 20]. In section 4, we will prove the following theorems:

Theorem 4.5 *If a set A is 1-wd self-reducible and $\leq^{\mathrm{SN}}_{\mathrm{btt}}$ reducible to some sparse set, then A is in $\mathrm{NP} \cap \mathrm{co\text{-}NP}$.*

Theorem 4.8 *If a set A is strictly 1-wd self-reducible and $\leq^{\mathrm{P}}_{\mathrm{btt}}$ reducible to some sparse set, then A is in P.*

Then, by combining results in sections 3 and 4, we will prove that for any class K chosen from $\{\mathrm{NP}, \mathrm{PP}, \mathrm{C}_=\mathrm{P}, \mathrm{MOD}_2\mathrm{P}, \mathrm{MOD}_3\mathrm{P}, \cdots\}$, it holds that

1. if there is a sparse $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard set for K then K is in P, and
2. if there is a sparse $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-hard set for K then K is in $\mathrm{NP} \cap \mathrm{co\text{-}NP}$.

Also, as PSPACE has 1-wd self-reducible complete sets, Theorem 4.5 implies that

3. if there is a sparse $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-hard set for PSPACE then $\mathrm{PSPACE} = \mathrm{NP}$.

Finally, as the property of having sparse $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard sets propagates to the smaller classes, it follows that

4. if there is a sparse $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard set for PSPACE then $\mathrm{PSPACE} = \mathrm{P}$.
5. [36] there are not sparse $\leq^{\mathrm{P}}_{\mathrm{btt}}$-hard sets for E.

2 Preliminaries

We fix now some of the notation that will be used throughout this paper. We use the alphabet $\Sigma = \{0, 1\}$, and define the basic notation about sets and words as it is done in [3]. By 'polynomials' we mean monotone

nondecreasing polynomials. We will denote the cardinality of a set A with $\#A$. We assume the canonical lexicographic order on Σ^*. A string x is less than y (write $x < y$) if either (1) $|x| < |y|$ or (2) $|x| = |y|$ and there exist strings $u, v, w \in \Sigma^*$ such that $x = u0v$ and $y = u1w$. For a string $x \in \Sigma^* \setminus \{\lambda\}$, $\mathrm{pred}(x)$ denotes the predecessor of x; that is, $\mathrm{pred}(x)$ is $\max\{y : y < x\}$. Also, for a string $x \in \Sigma^*$, $\mathrm{suc}(x)$ denotes the successor of x; that is, $\mathrm{suc}(x)$ is $\min\{y : y > x\}$. For a string x, $\mathrm{ord}(x)$ denotes $\#\{y : |y| = |x| \text{ and } y < x\}$. It is worth noting that $\mathrm{ord}(\cdot)$ is computable within time polynomial in the length of the string. $\langle \cdot, \cdot \rangle$ denotes a natural encoding of two strings into one string. We assume that this function is polynomial-time computable and invertible. We also assume that for every x, x', y and y' in Σ^* with $|x| = |x'|$ and $|y| = |y'|$, (i) $|\langle x, y \rangle| = |\langle x', y' \rangle|$, (ii) if $y' = \mathrm{pred}(y)$, then $\mathrm{pred}(\langle x, y \rangle) = \langle x, y' \rangle$, and (iii) if $x' < x$, then $\langle x, y \rangle > \langle x', y' \rangle$. For simplicity, for $k \geq 2$ and k strings $y_1, y_2, \cdots, y_k$, $\langle y_1, y_2, \cdots, y_k \rangle$ denotes $\langle \langle \cdots \langle y_1, y_2 \rangle, \cdots \rangle, y_k \rangle$. Furthermore, N denotes the set of natural numbers.

Our computational model is the polynomial-time Turing machine. We can assume w.l.o.g. that our nondeterministic Turing machines have polynomial clocks, that they have exactly $2^{p(|x|)}$ computation paths on input x, and any of these paths can be uniquely encoded by a word of $\Sigma^{=p(|x|)}$. For each $x \in \Sigma^*$, $\mathrm{acc}_N(x)$ and $\mathrm{rej}_N(x)$ denotes the set of all strings in $\Sigma^{=p(|x|)}$ representing accepting computation paths of N on x and rejecting computation paths of N on x, respectively. It is worth noting that for every $x \in \Sigma^*$, $\{\langle x, y \rangle : y \in \mathrm{acc}_N(x)\}$, and $\{\langle x, y \rangle : y \in \mathrm{rej}_N(x)\}$ are in P. Furthermore, for $x \in \Sigma^*$ and for $y \in \Sigma^{=p(|x|)}$, $\mathrm{acc}_{N,\leq y}(x)$ (respectively, $\mathrm{rej}_{N,\leq y}(x)$) denotes $\{z \in \mathrm{acc}_N(x) : z \leq y\}$ (respectively, $\{z \in \mathrm{rej}_N(x) : z \leq y\}$).

In this paper, we will consider bounded truth-table reducibility of sets to sparse sets. We start by defining the notion of sparse sets.

Definition 2.1 *A set S is sparse if there exists a polynomial p such that for every natural number n, $\#S^{\leq n} \leq p(n)$.*

For a polynomial-time deterministic Turing transducer M and for a string $x \in \Sigma^*$, $M(x)$ denotes the output of M on x. For a polynomial-time nondeterministic Turing transducer N and for a string $x \in \Sigma^*$, $N(x)$ denotes the set of nonempty strings which some computation path of N outputs on input x. Next we define the notion of one word-decreasing self-reducibility [21].

Definition 2.2 *A set A is one word-decreasing self-reducible (1-wd self-reducible, for short) if there exists a polynomial-time Turing transducer M such that for every $x \in \Sigma^*$, the following conditions are satisfied:*

(1) $M(x)$ is of one of the following forms: either true, false, (id, y), or $(\neg, y)$, where id *(respectively, $\neg$) is the identity function (respectively, negation) of one argument, $y \in \Sigma^*$ and $y < x$, and*

(2) if $M(x) \in \{true, false\}$, then $x \in A$ iff $M(x) = true$ and if $M(x) = (\alpha, y)$, then $x \in A$ iff $\alpha(\chi_A(y)) = true$.

Next we define a restriction of 1-wd self-reducible sets. Consider the self-reducibility chain from an input x, consisting of x, the string printed by the self-reducibility machine on input x, and so on, until we arrive to a string that the machine decides directly (it writes *true* or *false*). In a strictly 1-wd self-reducible set, every string in a self-reducibility chain has two components: the first one is the predecessor (in lexicographical order) of the first component of the previous string in the chain, and the second one is an extra information with only a polynomial number of possible values. This structure will allow to make a binary search in the chains in the proof of Theorem 4.8.

Definition 2.3 *A set A is strictly one word-decreasing self-reducible (strictly 1-wd self-reducible, for short) if there exists a polynomial-time Turing transducer M and a polynomial p such that for every x, M witnesses that A is 1-wd self-reducible and for every $x \in \Sigma^*$,*

(3) if $M(x)$ is of the form (α, y), then there exist z, z', w, and w' in Σ^* such that

(i) $x = \langle z, z' \rangle$ and $y = \langle w, w' \rangle$,

(ii) $|z| = |w|$ and $|z'| = |w'|$, and

(iii) $w = \mathrm{pred}(z)$ and $\mathrm{ord}(z'), \mathrm{ord}(w') < p(|x|)$.

Next we define bounded truth-table reductions. For a natural number $k > 0$, *a k-truth-table* is a mapping from $\{true, false\}^k$ to $\{true, false\}$. For a k-truth-table α and k strings $y_1, \cdots, y_k$, $(\alpha, y_1, \cdots, y_k)$ is called *a k-tt condition.* For a k-tt condition $(\alpha, y_1, \cdots, y_k)$ and a set B, $(\alpha, y_1, \cdots, y_k)$ is *satisfied by B* if $\alpha(\chi_B(y_1), \cdots, \chi_B(y_k)) = true$. By convention, a 0-truth-table is a constant boolean function of 0-arguments; that is, a 0-truth-table is either *true* or *false*. Furthermore, a 0-tt condition is either $(true)$ or $(false)$, and a 0-tt condition σ is satisfied by a set B if $\sigma = (true)$.

Definition 2.4 *For a natural number $k \geq 0$, a set A is polynomial-time k-truth-table reducible to a set B (polynomial-time k-tt reducible, for short, and write $A \leq^{\mathrm{P}}_{k-\mathrm{tt}} B$) if there exists a polynomial-time deterministic Turing transducer M such that for every $x \in \Sigma^*$, the following conditions are satisfied:*

(1) $M(x)$ is a k-tt condition and

(2) $x \in A$ if and only if $M(x)$ is satisfied by B.

Moreover, a set A is polynomial-time bounded truth-table reducible to a set B (polynomial-time btt reducible, for short, and write $A \leq^{\mathrm{P}}_{\mathrm{btt}} B$) if there exists some $k \in \mathbf{N}$ such that $A \leq^{\mathrm{P}}_{k-\mathrm{tt}} B$.

Definition 2.5 *For a natural number $k \geq 0$, a set A is strongly nondeterministic polynomial-time k-truth-table reducible to a set B (SN k-tt reducible, for short, and write $A \leq^{\mathrm{SN}}_{k-\mathrm{tt}} B$) if there exists a polynomial-time nondeterministic Turing transducer N such that for every $x \in \Sigma^*$, the following conditions are satisfied:*

(1) $N(x)$ is not the empty set and

(2) for every string $z \in N(x)$,

(a) z is a k-tt condition and

(b) $x \in A$ if and only if z is satisfied by B.

Moreover, a set A is strongly nondeterministic polynomial-time bounded truth-table reducible to a set B (SN btt reducible, for short and write $A \leq^{\mathrm{SN}}_{\mathrm{btt}} B$) if there exists some $k \in \mathbf{N}$ such that $A \leq^{\mathrm{SN}}_{k-\mathrm{tt}} B$.

The following observation is immediate from the definition.

Observation 2.6 *For every two sets A and B,*

1. *if A is $\leq^{\mathrm{P}}_{0-\mathrm{tt}}$-reducible to B, then A is in P and*
2. *if A is $\leq^{\mathrm{SN}}_{0-\mathrm{tt}}$-reducible to B, then A is in NP $\cap$ co-NP.*

Now we define the counting classes in a general setting, using the notation in [13], and the main complexity classes that we will use, in terms of the functions $\#\mathrm{acc}_N(\cdot)$ and $\#\mathrm{rej}_N(\cdot)$.

Definition 2.7 *For a polynomial-time decidable two-place predicate[1] Q on $\mathbf{N} \times \mathbf{N}$, a set L is in $\{Q\}$P if there exists a polynomial-time nondeterministic Turing machine N such that for every $x \in \Sigma^*$,*

$$x \in L \iff Q(\#\mathrm{acc}_N(x), \#\mathrm{rej}_N(x)).$$

Let $k \geq 2$ and let Q_{plus}, Q_{maj}, Q_{half}, and $Q^{(k)}_{\not\equiv 0}$ be two-place predicates such that for every $a, b \in \mathbf{N}$, $Q_{\mathrm{plus}}(a,b) = [a > 0]$, $Q_{\mathrm{maj}}(a,b) = [a \geq b]$, $Q_{\mathrm{half}}(a,b) = [a = b]$, and $Q^{(k)}_{\not\equiv 0}(a,b) = [a \not\equiv 0 \pmod k]$, respectively. Then, all of these predicates are polynomial-time decidable and NP $= \{Q_{\mathrm{plus}}\}$P, PP $= \{Q_{\mathrm{maj}}\}$P, $\mathrm{C_{=}P} = \{Q_{\mathrm{half}}\}$P, and $\mathrm{MOD}_k\mathrm{P} = \{Q^{(k)}_{\not\equiv 0}\}$P for every $k \geq 2$. Especially, MOD_2P is denoted by $\oplus$P.

For a polynomial-time decidable two-place predicate Q on $\mathbf{N} \times \mathbf{N}$, define K_Q to be the set of all strings of the form $\langle M, x, 0^t \rangle$ such that

(i) M is an encoding of a nondeterministic Turing machine and

(ii) by letting a and b denote the number of accepting and rejecting computation paths of M on x within time t respectively, $Q(a,b) = \mathit{true}$ holds.

Then, obviously, K_Q is $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for $\{Q\}$P. Thus, for any predicate Q, $\{Q\}$P has a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set.

Concerning these classes, the following relationships are well-known.

Proposition 2.8 *1. [12, 35]* NP $\cup$ co-NP $\subseteq$ PP.

2. [26, 33] co-NP $\subseteq \mathrm{C_{=}P} \subseteq$ PP.

3. [30, 38] For any $k \geq 1$, if $\Sigma^{\mathrm{P}}_k = \Pi^{\mathrm{P}}_k$, then PH $= \Sigma^{\mathrm{P}}_k = \Pi^{\mathrm{P}}_k$.

4. [33] PP $\subseteq \mathrm{NP}^{\mathrm{C_{=}P}}$.

3 Self-Reducible Complete Sets for Counting Classes

In this section, we will show the existence of one word-decreasing self-reducible sets that are complete for the classes PP, NP, $\mathrm{C_{=}P}$, MOD_2P, MOD_3P, ..., and strict word-decreasing self-reducible sets that are complete for NP, MOD_2P, MOD_3P, ... In fact, for the non-strict version of these self-reducibilities, the result holds for any possible counting class. Here there are the mentioned results.

[1] Throughout the present paper, we assume that each two-place predicate is not trivial; that is, Q is neither constantly *true* nor constantly *false*.

Lemma 3.1 *Let Q be a polynomial-time decidable two-place predicate on $\mathbf{N} \times \mathbf{N}$. Then, there exists a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for the class $\{Q\}\mathrm{P}$ that is 1-wd self-reducible.*

Proof Let Q be as in the hypothesis and A be a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for $\{Q\}\mathrm{P}$. Take a nondeterministic Turing machine N whose time is bounded by polynomial p, and witnesses that A is in $\{Q\}\mathrm{P}$. Since we can compute the total number of paths of machine N on inputs of a given length, we will use the predicate $Q_n(i) = Q(i, 2^{p(n)} - i)$. So, we have that $Q_{|x|}(\#\mathrm{acc}_N(x)) = Q(\#\mathrm{acc}_N(x), \#\mathrm{rej}_N(x))$. Now define

$$\begin{aligned} PropQ \;=\; \{\langle x, w_1, w_2\rangle : {} & |w_1| = |w_2| = p(|x|), \\ & \mathrm{ord}(w_1) + \mathrm{ord}(w_2) \leq 2^{p(|x|)}, \text{and} \\ & Q_{|x|}(\#\mathrm{acc}_{N,\leq w_1}(x) + \mathrm{ord}(w_2))\}. \end{aligned}$$

This set is complete for the class $\{Q\}\mathrm{P}$. It belongs to the class because we can define a polynomial time nondeterministic machine which, on input $\langle x, w_1, w_2\rangle$, has $\#\mathrm{acc}_{N,\leq w_1}(x) + \mathrm{ord}(w_2)$ accepting paths. To see that it is complete, notice that for every $x \in \Sigma^*$, it holds that $x \in A$ if and only if $\langle x, 1^{p(|x|)}, 0^{p(|x|)}\rangle \in PropQ$.

So, in order to establish the lemma, we only have to show that $PropQ$ is 1-wd self-reducible. Suppose we have the string $\langle x, w_1, w_2\rangle$ with $|w_1| = |w_2| = p(|x|)$ and $\mathrm{ord}(w_1) + \mathrm{ord}(w_2) \leq 2^{p(|x|)}$, and we would like to find some smaller string to establish the self-reducibility relation. There are four different cases:

(1) If $w_1 \neq 0^{p(|x|)}$ and $w_1 \notin \mathrm{acc}_N(x)$ then

$\langle x, w_1, w_2\rangle \in PropQ \iff$
$\langle x, \mathrm{pred}(w_1), w_2\rangle \in PropQ$.

(2) If $w_1 \neq 0^{p(|x|)}$ and $w_1 \in \mathrm{acc}_N(x)$ then

$\langle x, w_1, w_2\rangle \in PropQ \iff$
$\langle x, \mathrm{pred}(w_1), \mathrm{suc}(w_2)\rangle \in PropQ$.

(3) If $w_1 = 0^{p(|x|)}$ and $w_1 \notin \mathrm{acc}_N(x)$ then

$\langle x, w_1, w_2\rangle \in PropQ \iff Q_{|x|}(\mathrm{ord}(w_2))$.

(4) If $w_1 = 0^{p(|x|)}$ and $w_1 \in \mathrm{acc}_N(x)$ then

$\langle x, w_1, w_2\rangle \in PropQ \iff Q_{|x|}(\mathrm{ord}(w_2) + 1)$.

Note that a polynomial-time deterministic Turing machine can check which of the four conditions is true and then write a string related with the input (in cases 1 and 2) or decide the input (in cases 3 and 4), always following the behaviour of a machine that witnesses the 1-wd self-reducibility of $PropQ$. □

Lemma 3.2 *For any $k \geq 2$, there exists a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for the class $\mathrm{MOD}_k\mathrm{P}$ that is strictly 1-wd self-reducible.*

Proof Let A be a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for $\mathrm{MOD}_k\mathrm{P}$. Take a nondeterministic Turing machine N whose time is bounded by polynomial p, and witnesses that A is in $\mathrm{MOD}_k\mathrm{P}$. For each $i, 0 \leq i \leq k-1$, define n_i to be the string in $\Sigma^{=p(|x|)}$ such that $\mathrm{ord}(n_i) = i$, and define

$$\begin{aligned} Mod \;=\; \{\langle x, w, n_i\rangle : {} & |w| = p(|x|) \text{ and } 0 \leq i \leq k-1 \\ & \text{and } \#\mathrm{acc}_{N,\leq w}(x) \not\equiv i \pmod{k}\}. \end{aligned}$$

This set is complete for the class $\mathrm{MOD}_k\mathrm{P}$. To see that it belongs to the class, note that $\#\mathrm{acc}_{N,\leq w}(x) \not\equiv i \pmod{k}$ if and only if $\#\mathrm{acc}_{N,\leq w}(x) + k - i \not\equiv 0 \pmod{k}$, and then we can define a polynomial time nondeterministic machine which, on input $\langle x, w, n_i\rangle$, has $\#\mathrm{acc}_{N,\leq w}(x) + k - i$ accepting paths. To see that it is complete, notice that for every $x \in \Sigma^*$, it holds that $x \in A$ if and only if $\langle x, 1^{p(|x|)}, n_0\rangle \in Mod$.

So, in order to prove the lemma, it is only left to show that Mod is 1-wd self-reducible. Suppose we have the string $\langle x, w, n_i\rangle$, with $|w| = p(|x|)$ and $0 \leq i \leq k-1$, and we would like to find some smaller string to establish the self-reducibility relation. There are four cases:

(1) If $w \neq 0^{p(|x|)}$ and $w \notin \mathrm{acc}_N(x)$ then

$\langle x, w, n_i\rangle \in Mod \iff \langle x, \mathrm{pred}(w), n_i\rangle \in Mod$.

(2) If $w \neq 0^{p(|x|)}$ and $w \in \mathrm{acc}_N(x)$ then

$\langle x, w, n_i\rangle \in Mod \iff \langle x, \mathrm{pred}(w), n_j\rangle \in Mod$, where j is the number such that $0 \leq j \leq k-1$ and $j \equiv i - 1 \pmod{k}$.

(3) If $w = 0^{p(|x|)}$ and $w \notin \mathrm{acc}_N(x)$ then

$\langle x, w, n_i\rangle \in Mod \iff n_i \not\equiv 0 \pmod{k}$.

(4) If $w = 0^{p(|x|)}$ and $w \in \mathrm{acc}_N(x)$ then

$\langle x, w, n_i\rangle \in Mod \iff n_i - 1 \not\equiv 0 \pmod{k}$.

In the same way as in the previous theorem, these four cases show how to define a machine that witnesses that Mod is 1-wd self-reducible. But note that Mod is also strictly 1-wd self-reducible, because in any self-reducibility chain, every string $\langle x, w, n_i\rangle$ can be decomposed in a first component, $\langle x, w\rangle$, that decreases (in lexicographical order) along the chain, and a second component, n_i, that is bounded by $k-1$, thus fulfilling the conditions in the definition of strict 1-wd self-reducibility. □

Lemma 3.3 *There exists a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for the class NP that is strictly 1-wd self-reducible.*

Proof Let A be a $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set for NP. Take a nondeterministic Turing machine N whose time is bounded by polynomial p, and witnesses that A is in NP. Define

$$\mathit{Right} = \{\langle x, w\rangle : |w| = p(|x|) \text{ and } \#\mathrm{acc}_{N,\leq w}(x) \geq 1\}$$

This set is complete for NP: it is in NP and for every $x \in \Sigma^*$, it holds that $x \in A$ if and only if $\langle x, 1^{p(|x|)}\rangle \in \mathit{Right}$.

Now, suppose we have the string $\langle x, w\rangle$, with $|w| = p(|x|)$, and we want to find some smaller string to establish the self-reducibility relation. There are three cases:

(1) If $w \neq 0^{p(|x|)}$ and $w \notin \mathrm{acc}_N(x)$ then
$\langle x, w\rangle \in \mathit{Right} \iff \langle x, \mathrm{pred}(w)\rangle \in \mathit{Right}$.

(2) If $w \neq 0^{p(|x|)}$ and $w \in \mathrm{acc}_N(x)$ then $\langle x, w\rangle \in \mathit{Right}$.

(3) If $w = 0^{p(|x|)}$ then
$\langle x, w\rangle \in \mathit{Right} \iff w \in \mathrm{acc}_N(x)$.

These relations show us that the set *Right* is 1-wd self-reducible. A simple encoding of the set, as for example $\mathit{Right} \times \{\lambda\}$, is strictly 1-wd self-reducible and $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete for NP, and then the theorem is proved. □

Note that if a set is strictly 1-wd self-reducible, then its complement has the same self-reducibility structure. So, from this and the above lemma it follows that co-NP has, as NP, some $\leq^{\mathrm{P}}_{\mathrm{m}}$-complete set that is strictly 1-wd self-reducible.

4 The Main Technical Theorems

In this section, we prove that every one word-decreasing self-reducible set (respectively, strict one word-decreasing self-reducible set) which is $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-reducible (respectively, $\leq^{\mathrm{P}}_{\mathrm{btt}}$-reducible) to some sparse set is already in NP ∩ co-NP (respectively, P). The proof is inductive; that is, we will show that if an 1-wd self-reducible set is $\leq^{\mathrm{SN}}_{k+1-\mathrm{tt}}$-reducible to a sparse set S for some $k \in \mathbf{N}$, then the set is $\leq^{\mathrm{SN}}_{k-\mathrm{tt}}$-reducible to S. The same scheme applies in the case of strict 1-wd self-reducibility. Then, by using the above argument repeatedly and thereby reducing the number of queries to 0, we obtain the results.

We first study strong nondeterministic polynomial-time bounded truth-table reducibility of 1-wd self-reducible sets to sparse sets. We prove the following theorem.

Theorem 4.1 *Let A be an 1-wd self-reducible set. If for some $k \in \mathbf{N}$, A is $\leq^{\mathrm{SN}}_{k+1-\mathrm{tt}}$-reducible to a sparse set S, then A is $\leq^{\mathrm{SN}}_{k-\mathrm{tt}}$-reducible to S.*

Proof Let A be an 1-wd self-reducible set and S be a sparse set to which A is $\leq^{\mathrm{SN}}_{k+1-\mathrm{tt}}$-reducible. Then, there exist a polynomial-time deterministic Turing transducer M and a polynomial-time nondeterministic Turing transducer N which witnesses that A is 1-wd self-reducible and A is $\leq^{\mathrm{SN}}_{k+1-\mathrm{tt}}$-reducible to S, respectively. We will construct a polynomial-time Turing machine N_0 which $\leq^{\mathrm{SN}}_{k-\mathrm{tt}}$ reduces A to S. Let $x \in \Sigma^*$ and $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ be a list of pairs of a string and a $k+1$-tt condition. Λ is called *an (M, N)-chain w.r.t. x* if the following conditions are satisfied:

(c1) $x = y_1$ and for every $i, 1 \leq i < m$, $y_{i+1} < y_i$,

(c2) for every $i, 1 \leq i \leq m$, $\sigma_i \in N(y_i)$,

(c3) for every $i, 1 < i \leq m$, either

- $M(y_{i-1}) = (\alpha, y_i)$ for some $\alpha \in \{\mathrm{id}, \neg\}$ or
- $\sigma_i \in \{\sigma_1, \cdots, \sigma_{i-1}\}$, and

(c4) for every $i, 1 \leq i < m$, if $\sigma_i \in \{\sigma_1, \cdots, \sigma_{i-1}\}$, then $\sigma_{i+1} \notin \{\sigma_1, \cdots, \sigma_i\}$.

Moreover, a list $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ is called *a full (M, N)-chain w.r.t. x* if Λ is an (M, N)-chain w.r.t. x and

(c5) $M(y_m) \in \{\mathit{true}, \mathit{false}\}$.

The following facts are easy to prove.

Fact 1 *The sets*

$$C = \{\langle x, \Lambda\rangle : \Lambda \text{ is an } (M, N)\text{-chain w.r.t. } x\} \text{ and}$$
$$\widehat{C} = \{\langle x, \Lambda\rangle : \Lambda \text{ is a full } (M, N)\text{-chain w.r.t. } x\}$$

are in NP.

Fact 2 *Let $x \in \Sigma^*$ and $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ be an (M, N)-chain w.r.t. x. Then, $|\langle x, \Lambda\rangle|$ is bounded above by some polynomial in $|x| + m$.*

The machine N_0 performs in the following way:

{ The Description of N_0 }

(1) For a given input $x \in \Sigma^*$, N_0 nondeterministically guesses $m, 1 \leq m \leq r(|x|)$ and a list $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$, and N_0 nondeterministically checks that Λ is an (M, N)-chain w.r.t. x, and that

(♣) if $m < r(|x|)$, then Λ is full,

where r is a polynomial defined later. If the check fails, then N_0 halts immediately. Otherwise, N_0 proceeds to the next step.

(2) If Λ is full, N_0 deterministically computes $\chi_A(x)$ from Λ and outputs *true* if $\chi_A(x) = true$ and *false* otherwise. Otherwise, N_0 proceeds to the next step.

(3) If Λ is not full (therefore, $m = r(|x|)$), then N_0 deterministically computes an h-tt condition ζ with $h \leq k$ such that $x \in A$ if and only if ζ is satisfied by S, outputs ζ, and halts.

{ End of the Description of N_0 }

It is not hard to show that there exists some full (M,N)-chain w.r.t. x. So, assume that $\Lambda = [(z_1,\tau_1),\cdots,(z_m,\tau_m)]$ is one of such chains and take now the subchain $\Lambda' = [(z_1,\tau_1),\cdots,(z_{m_0},\tau_{m_0})]$, where $m_1 = \min\{m_0, r(|x|)\}$, for some polynomial r. Then, obviously, Λ' satisfies (♣). and, therefore, there exists at least one computation path that leads to step (2). Furthermore, from Facts 1 and 2, step (1) can be executed within time polynomial in $|x|$. Therefore, in order to establish the theorem, we only have to show that there exist two polynomial-time algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ that implement steps (2) and (3), satisfying the following conditions:

1. for a given x and a full (M,N)-chain Λ w.r.t. x, $\mathcal{A}_1$ computes $\chi_A(x)$ and
2. for a given x and an (M,N)-chain Λ of length $r(|x|)$, $\mathcal{A}_2$ computes an h-tt condition ζ with $h \leq k$ such that $x \in A$ if and only if ζ is satisfied by S.

In the following we will develop the above two algorithms.

For two strings x and y, $\oplus_A[x,y]$ denotes the relationship between $\chi_A(x)$ and $\chi_A(y)$. More precisely, for every x and $y \in \Sigma^*$,

$$\oplus_A[x,y] = \begin{cases} \mathrm{id} & \text{if } \chi_A(x) = \chi_A(y), \\ \neg & \text{if } \chi_A(x) \neq \chi_A(y). \end{cases}$$

It is not hard to see that the following facts holds.

Fact 3 *For every x, y, and z, $\oplus_A[x,z] = \oplus_A[x,y] \circ \oplus_A[y,z]$, where $\gamma \circ \delta$ denotes the function $\gamma(\delta(\cdot))$.*

Fact 4 *Let $x \in \Sigma^*$ and $\Lambda = [(y_1,\sigma_1),\cdots,(y_m,\sigma_m)]$ be an (M,N)-chain w.r.t. x. Then, for every $i, 1 \leq i \leq m$, $\oplus_A[x,y_i]$ is computable within time polynomial in $|x|+m$.*

Proof of Fact 4 Let $x \in \Sigma^*$ and $\Lambda = [(y_1,\sigma_1),\cdots,(y_m,\sigma_m)]$ be an (M,N)-chain w.r.t. x. Since M witnesses the 1-wd self-reducibility of A and N witnesses the $\leq^{\mathrm{SN}}_{k+1-\mathrm{tt}}$-reducibility of A to S, the following properties hold:

- for every i and $j, 1 \leq i < j \leq m$, if $\sigma_i = \sigma_j$, then $\oplus_A[y_i,y_j] = \mathrm{id}$ because $y_i \in A$ iff σ_i is satisfied by S, $y_j \in A$ iff σ_j is satisfied by S, and $\sigma_i = \sigma_j$, and
- for every $i, 1 \leq i < m$, if $M(y_i) = (\alpha, y_{i+1})$, then $\oplus_A[y_i,y_{i+1}] = \alpha$ because $y_i \in A$ iff $\alpha(\chi_A(y_{i+1})) = true$.

Recall that, from the definition of (M,N)-chain, for each $i, 1 < i \leq m$, either

(a) there exists some index $j, 1 \leq j < i$, such that $\sigma_i = \sigma_j$ or

(b) $M(y_{i-1}) = (\alpha, y_i)$ for some $\alpha \in F_1$.

Note that $\sigma_i = \sigma_j$ implies $\oplus_A[y_i,y_j] = \mathrm{id}$ and $M(y_{i-1}) = (\alpha, y_i)$ implies $\oplus_A[y_{i-1},y_i] = \alpha$. So, for every $i, 1 < i \leq m$, $\oplus_A[x,y_i]$ is computable from $\{\oplus_A[x,y_j] : 1 \leq j < i\}$ by testing conditions (a) and (b). Since $y_1 = x$, it holds that $\oplus_A[x,y_1] = \mathrm{id}$. Thus, starting from $\oplus_A[x,y_1]$, we can inductively compute all $\oplus_A[x,y_i]$, $1 \leq i \leq m$, within time polynomial in $|\langle x,\Lambda\rangle|$, and thus, from Fact 2, they are computable within time polynomial in $|x|+m$.

□ **Proof of Fact 4**

From the above fact, we can prove immediately the existence of algorithm $\mathcal{A}_1$.

Lemma 4.2 *There exists a deterministic polynomial-time algorithm $\mathcal{A}_1$ which, for a given x and a full (M,N)-chain Λ, computes $\chi_A(x)$.*

Proof of Lemma 4.2 For every $x \in \Sigma^*$ and for every full (M,N)-chain $\Lambda = [(y_1,\sigma_1),\cdots,(y_m,\sigma_m)]$ w.r.t. x, it holds that $M(y_m) \in \{true, false\}$. So, $\chi_A(x) = \oplus_A[x,y_m](M(y_m))$. Then, from Fact 4 and from the fact that M runs in polynomial time, $\chi_A(x)$ is computable within time polynomial in $|x|+m$. This proves the lemma.

□ **Proof of Lemma 4.2**

Next we define the polynomial r. Since N runs in polynomial time and S is sparse, then there must exist a polynomial q such that for every $x \in \Sigma^*$ and for every (M,N)-chain $\Lambda = [(y_1,\sigma_1),\cdots,(y_m,\sigma_m)]$ w.r.t. x,

$$\#\{w \in S \ : \ w \text{ appears as an argument in } \sigma_i \text{ for some } i, 1 \leq i \leq m\} \leq q(|x|).$$

Define $r(n) = 4 \cdot 2^{2^{k+1}} \cdot q(n)^{k+1} + 1$. Now, our goal is to show the existence of algorithm $\mathcal{A}_2$.

Lemma 4.3 *There exists a deterministic polynomial-time algorithm $\mathcal{A}_2$ which, given $x \in \Sigma^*$ and an (M,N)-chain $\Lambda = [(y_1,\sigma_1),\cdots,(y_m,\sigma_m)]$ w.r.t. x, with $m = r(|x|)$, computes an h-tt condition ζ, with $h \leq k$, such that $x \in A$ if and only if ζ is satisfied by S.*

Proof of Lemma 4.3 Let K denote the set of indices $\{1, \cdots, k+1\}$. For a $k+1$-tt condition $\sigma = (\beta, w_1, \cdots, w_{k+1})$, let $\beta(\sigma)$ denote β; for $\ell \in K$, let $\sigma[\ell]$ denote w_ℓ, and for a set $Q \subseteq K$, let $\sigma[Q]$ denote $\{\sigma[\ell] : \ell \in Q\}$. Note that, in consequence, $\sigma[K]$ denotes the set of all the arguments of the $k+1$-tt condition σ. $\mathcal{A}_2$ performs in the following way:

{ The Description of $\mathcal{A}_2$ }

For a given $x \in \Sigma^*$ and a given (M, N)-chain $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ w.r.t. x with $m = r(|x|)$, do the following:

(1) Find $I \subseteq \{1, \cdots, m\}$ with $\#I \geq q(|x|) + 1$, $Q \subset K$, $\alpha_0 \in \{\mathrm{id}, \neg\}$, and a $k+1$-truth-table β_0 such that

(a) for every $i \in I$, $\oplus_A[x, y_i] = \alpha_0$,

(b) for every $i \in I$, $\beta(\sigma_i) = \beta_0$,

(c) for every i and $j \in I$ and for every $\ell \in Q$, $\sigma_i[\ell] = \sigma_j[\ell]$, and

(d) for every distinct i and $j \in I$, $\sigma_i[K \backslash Q] \cap \sigma_j[K \backslash Q] = \emptyset$.

(2) Let $h = \#Q$. Compute an h-truth-table $\widehat{\beta}$ from β_0 by substituting every argument at the position $\ell \in K \backslash Q$ with *false* at the same time.

(3) Compute an h-truth-table $\widetilde{\beta}$ such that for every $b_1, \cdots, b_h \in \{true, false\}$ it holds that

$$\widetilde{\beta}(b_1, \cdots, b_h) = \alpha_0(\widehat{\beta}(b_1, \cdots, b_h)).$$

Set ζ to $(\widetilde{\beta}, w_{\ell_1}, \cdots, w_{\ell_h})$, where $\ell_1, \cdots, \ell_h$ is an enumeration of all indices in Q in increasing order and for every t, $1 \leq t \leq h$, $w_{\ell_t} = \sigma_i[\ell_t]$ for every $i \in I$.

{ End of the Description of $\mathcal{A}_2$ }

Notice that in order to find I, Q, α_0, and β_0 in step (1), we only have to execute a brute-force search method over Q and $i_0 = \min\{i \in I\}$. That is, we only have to move Q over elements in $2^K - \{K\}$ and i_0 from 1 to m and enumerate all indices $i \neq i_0$ satisfying

(a') $\oplus_A[x, y_i] = \oplus_A[x, y_{i_0}]$,

(b') $\beta(\sigma_i) = \beta(\sigma_{i_0})$,

(c') for every $\ell \in Q$, $\sigma_i[\ell] = \sigma_{i_0}[\ell]$, and

(d') $\sigma_i[K \backslash Q] \cap \sigma_{i_0}[K \backslash Q] = \emptyset$,

and test whether the number of such indices is $\geq q(|x|)$ or not. Obviously, this search can be executed within time polynomial in $|x|$. Since steps (2) and (3) also have this time bound, algorithm $\mathcal{A}_2$ runs in time polynomial in $|x|$.

Therefore, in order to establish the lemma, we only have to show that $\mathcal{A}_2$ works correctly, that is to say, that there always exist I, Q, α_0, and β_0 that satisfy conditions (a), (b), (c), and (d). This is seen as follows. First define I_{init} to be the set of all $i \in \{1, \cdots, m\}$ such that $\sigma_i \notin \{\sigma_1, \cdots, \sigma_{i-1}\}$. Then, it is not hard to see that

- $\#I_{\mathrm{init}} \geq 2 \cdot 2^{2^{k+1}} \cdot q(|x|)^{k+1} + 1$, and
- for every distinct i and j in I_{init}, $\sigma_i \neq \sigma_j$.

Next, for each $\alpha \in \{\mathrm{id}, \neg\}$, define

$$I_\alpha = \{i \in I_{\mathrm{init}} : \oplus_A[x, y_i] = \alpha\}.$$

Then, obviously, exactly one of $\#I_{\mathrm{id}}$ and $\#I_\neg$ is $\geq 2^{2^{k+1}} \cdot q(|x|)^{k+1} + 1$. So, let α_0 be such a truth-table.

Furthermore, for each $k+1$-truth-table β, define

$$I(\beta) = \{i \in I_{\alpha_0} : \beta(\sigma_i) = \beta\}.$$

Since there are $2^{2^{k+1}}$ possible β, there is at least one β such that $\#I(\beta) \geq q(|x|)^{k+1} + 1$. Let β_0 be one of such $k+1$-truth-tables. Then, by construction, the set $I(\beta_0)$ satisfies the conditions (a) and (b).

Finally, consider the set of arguments $\sigma_i[K]$ of a $k+1$-tt condition σ_i, with $i \in I(\beta_0)$, as an ordered set of cardinality $k+1$. By a simple modification of the theorem in [10], we obtain the following proposition, in which, for an ordered set W, we denote with $W(\ell)$ its ℓ-th element.

Proposition 4.4 *Let $\mathcal{F}$ be a family of ordered sets, each one with cardinality t. If $\#\mathcal{F} \geq d^t + 1$, then there exist $\mathcal{G} \subset \mathcal{F}$ with $\#\mathcal{G} \geq d + 1$ and a set $Q \subset \{1, \cdots, t\}$ such that*

(i) for every $\ell \in Q$ and for every U and V in $\mathcal{G}$, $U(\ell) = V(\ell)$ and

(ii) for every distinct U and V in $\mathcal{G}$, $\{U(\ell) : \ell \in \{1, \cdots, t\} \setminus Q\} \cap \{V(\ell) : \ell \in \{1, \cdots, t\} \setminus Q\} = \emptyset$,

Then, from the above proposition, there exists a set $I \subseteq I(\beta_0)$ with $\#I \geq q(|x|) + 1$, $h \in \{0, \cdots, k\}$ and a set $Q \subset K$ with $\#Q = h$ satisfying the conditions (c) and (d). Since $I \subseteq I(\beta_0)$, conditions (a) and (b) are satisfied. Therefore, the search procedure is always successful.

On the other hand, $\{\sigma_i[K \backslash Q] : i \in I\}$ is a family of at least $\geq q(|x|) + 1$ disjoint sets in Σ^*. Since $q(|x|)$ is an upper bound for the number of strings in S appearing as an argument in σ_i, $1 \leq i \leq m$, there must exist an i such that $\sigma_i[K \backslash Q] \subseteq \overline{S}$. Thus, we know that there is some

$k+1$-tt condition σ_i, with $i \in I$, such that the arguments corresponding to the indices in $K \setminus Q$ are not in S, and so, we can omit these queries. As, by Proposition 4.4, the rest of the queries (with indices in Q) are the same for all the $k+1$-tt conditions σ_i with $i \in I$, and the truth table is also the same, we know that for some $i \in I$, and for the $\widehat{\beta}$, $\widetilde{\beta}$ and ζ defined in algorithm $\mathcal{A}_2$, it holds that

$$\begin{aligned} y_i \in A &\iff \beta_0(\chi_S(\sigma_i[1]), \cdots, \chi_S(\sigma_i[k+1])) = \mathit{true} \\ &\iff \widehat{\beta}(\chi_S(\sigma_i[\ell_1]), \cdots, \chi_S(\sigma_i[\ell_h])) = \mathit{true}. \end{aligned}$$

Since $\oplus_A[x, y_i] = \alpha_0$ from (a), the above property implies

$$\begin{aligned} x \in A &\iff \alpha_0(\widehat{\beta}(\chi_S(\sigma_i[\ell_1]), \cdots, \chi_S(\sigma_i[\ell_h]))) = \mathit{true} \\ &\iff \widetilde{\beta}(\chi_S(\sigma_i[\ell_1]), \cdots, \chi_S(\sigma_i[\ell_h])) = \mathit{true} \\ &\iff \zeta \text{ is satisfied by } S. \end{aligned}$$

Therefore, ζ is an h-tt condition such that $x \in A$ if and only if ζ is satisfied by S. Hence, $\mathcal{A}_2$ works correctly. This proves the lemma, and consequently, this proves the theorem.

□ **Proof of Lemma 4.3**
□ **Proof of Theorem 4.1**

From Theorem 4.1, we obtain the following theorem.

Theorem 4.5 *If a set A is 1-wd self-reducible and $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-reducible to some sparse set, then A is in* NP ∩ co-NP.

Proof Let A be a set which is 1-wd self-reducible and $\leq^{\mathrm{SN}}_{k-\mathrm{tt}}$-reducible to a sparse set S for some $k \in \mathbf{N}$. Then, by using Theorem 4.1 repeatedly, we have that A is $\leq^{\mathrm{SN}}_{0-\mathrm{tt}}$-reducible to S. This implies $A \in$ NP ∩ co-NP. □

Next we consider $\leq^{\mathrm{P}}_{\mathrm{btt}}$-reducibility of strict 1-wd self-reducible sets to sparse sets.

Theorem 4.6 *Let A be a strict 1-wd self-reducible set. If for some $k \in \mathbf{N}$, A is $\leq^{\mathrm{P}}_{k+1-\mathrm{tt}}$-reducible to a sparse set S, then A is $\leq^{\mathrm{P}}_{k-\mathrm{tt}}$-reducible to S.*

Proof Let A be a strict 1-wd self-reducible set, M be a machine witnessing this property and p be the polynomial from the definition of strict 1-wd self-reducibility. Furthermore, let S be a sparse set to which A is $\leq^{\mathrm{P}}_{k+1-\mathrm{tt}}$-reducible via a polynomial-time deterministic machine N. The proof is similar to that of Theorem 4.1. For a string $x \in \Sigma^*$, call x a *bottom* if $M(x) \in \{\mathit{true}, \mathit{false}\}$ and call x a *median* otherwise. For a median $x \in \Sigma^*$, let $\mathrm{Q}(x)$ denote the unique string y such that $M(x) = (\alpha, y)$ for some $\alpha \in \{\mathrm{id}, \neg\}$. Furthermore, define other notions, notations, and polynomials as in the proof of Theorem 4.1. Then, all Facts and Lemmas in the proof of Theorem 4.1 are established for this situation, too.

Moreover, we claim the following:

Lemma 4.7 *There exists a polynomial-time algorithm which, for a given $x \in \Sigma^*$ and a natural number $n \geq 1$, computes an (M, N)-chain $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$, with $1 \leq m \leq n$, satisfying that*

($*$) *if $m < n$ then Λ is full.*

Proof of Lemma 4.7 Let $x \in \Sigma^*$ and $n \geq 1$. Suppose that x is a bottom. Then, obviously, $\Lambda = [(x, N(x))]$ is the desired chain. So, assume that x is a median. For an (M, N)-chain $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$, define ($*$) to be the following property:

($*$) y_m is a median and $\sigma_m \notin \{\sigma_1, \cdots, \sigma_{m-1}\}$.

Note that $[(x, N(x))]$ satisfies this property.

Moreover, suppose that there exists a polynomial-time algorithm SEARCH which, for a given (M, N)-chain $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$, finds an element $z < y_m$ such that one of the following conditions hold: either

(a) $N(z) \in \{\sigma_1, \cdots, \sigma_m\}$, z is a median and $N(\mathrm{Q}(z)) \notin \{\sigma_1, \cdots, \sigma_m\}$,

(b) $N(z) \in \{\sigma_1, \cdots, \sigma_m\}$ and z is a bottom, or

(c) $N(z) \notin \{\sigma_1, \cdots, \sigma_m\}$ and $z = \mathrm{Q}(y_m)$.

Let $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ be the input to SEARCH satisfying ($*$) and z be the output of SEARCH. Define Λ' to be $[(y_1, \sigma_1), \cdots, (y_m, \sigma_m), (z, N(z)), (\mathrm{Q}(z), N(\mathrm{Q}(z)))]$ if (a) is satisfied and $[(y_1, \sigma_1), \cdots, (y_m, \sigma_m), (z, N(z))]$ otherwise. Then, Λ' is an (M, N)-chain of length $> m$ and either is full or satisfies ($*$). So, for given x and n, if we start from $\Lambda = [(x, N(x))]$ and repeatedly execute SEARCH on Λ and set Λ to the resulting (M, N)-chain until Λ is full or its length is $\geq n$, we obtain an (M, N)-chain $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ w.r.t. x such that either (i) Λ is full, (ii) $m = n$, or (iii) $m = n + 1$. Then, by deleting the $(n+1)$-th pair from Λ if it exists, we obtain the desired (M, N)-chain w.r.t. x. Furthermore, since SEARCH runs in polynomial time, the total running time of the above algorithm is bounded by a polynomial in $|x|$ and n. So, in the following, we will develop the method SEARCH.

Let $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ be the input to SEARCH satisfying ($*$). Let $s, t \in \mathbf{N}$ and $u, v, w \in \Sigma^*$ be such that $y_m = \langle u, v \rangle$, $\mathrm{Q}(y_m) = \langle \mathrm{pred}(u), w \rangle$, $u \in \Sigma^{=s}$, and $v, w \in \Sigma^{=t}$. Let p be the polynomial from the definition of strict 1-wd self-reducibility such that $\mathrm{ord}(v), \mathrm{ord}(w) < p(|y_m|)$. By convention, let W denote $\{\sigma_1, \cdots, \sigma_m\}$ and D denote $\{d \in \Sigma^{=t} : \mathrm{ord}(d) < p(|y_m|)\}$. For input Λ, SEARCH first excludes some trivial cases below:

Case 1 $N(\mathrm{Q}(y_m)) \notin W$. If this is the case, $z = \mathrm{Q}(y_m)$ satisfies (c).

Case 2 $N(Q(y_m)) \in W$ and $Q(y_m)$ is a bottom. If this is the case, $z = Q(y_m)$ satisfies (b).

Case 3 $N(Q(y_m)) \in W$, $Q(y_m)$ is a median and $N(Q(Q(y_m))) \notin W$. If this is the case, $z = Q(y_m)$ satisfies (a).

Case 4 $Q(y_m)$ is a median, $Q(Q(y_m))$ is a bottom and $N(Q(y_m)), N(Q(Q(y_m))) \in W$. If this is the case, $z = Q(Q(y_m))$ satisfies (b).

Case 5 Both $Q(y_m)$ and $Q(Q(y_m))$ are medians, $N(Q(y_m)), N(Q(Q(y_m))) \in W$ and for some $d \in D$, $N(\langle 0^s, d\rangle) \in W$. If this is the case, by letting d_0 be the smallest $d \in D$ such that $N(\langle 0^s, d\rangle) \in W$, $z = \langle 0^s, d_0\rangle$ satisfies (b).

Now assume that Λ passed all checks in the above; that is, both $Q(y_m)$ and $Q(Q(y_m))$ are medians, $N(Q(y_m)), N(Q(Q(y_m))) \in W$ and for every $d \in D$, $N(\langle 0^s, d\rangle) \notin W$. Then, we execute the following binary-search-like procedure:

(1) Set a and b to 0^s and $\mathrm{pred}(u)$, respectively.

(2) Repeat the following operations until $a = \mathrm{pred}(b)$:

(i) Set c to the string in $\Sigma^{=s}$ such that $\lfloor(\mathrm{ord}(a) + \mathrm{ord}(b))/2\rfloor = \mathrm{ord}(c)$.

(ii) If for some $d \in D$, $N(\langle c, d\rangle) \in W$, then set b to c. Otherwise set a to c.

(3) Set d_0 to the smallest $d \in D$ such that $N(\langle b, d\rangle) \in W$ and z to $\langle b, d_0\rangle$. Output z and halt.

Note that the following conditions hold:

(c1) From our assumption, when entering the above step (2), for some $d \in D$, $N(\langle b, d\rangle) \in W$ and for every $d \in D$, $N(\langle a, d\rangle) \notin W$.

(c2) In step (2), if b is set to c, then for some $d \in D$, $N(\langle b, d\rangle) \in W$, and if a is set to c, then for every $d \in D$, $N(\langle a, d\rangle) \notin W$.

Therefore, the conditions (for some $d \in D$, $N(\langle b_0, d\rangle) \in W$) and (for every $d \in D$, $N(\langle a_0, d\rangle) \notin W$) are always preserved during the procedure. Let a_0 and b_0 denote a and b after SEARCH quitted (2), respectively. Then, for some $d \in D$, $N(\langle b_0, d\rangle) \in W$ and for every $d \in D$, $N(\langle a_0, d\rangle) \notin W$. Suppose that z is a bottom. Obviously, z satisfies the condition (b). On the other hand, suppose that z is a median. Since $|\langle b_0, d\rangle| = |\langle u, v\rangle| = |y_m|$ for every $d \in D$ and $a_0 = \mathrm{pred}(b_0)$, $Q(\langle b_0, d_0\rangle) = \langle a_0, d\rangle$ for some $d \in D$. Thus, $N(z) \in W$ and $N(Q(z)) \notin W$; that is, z satisfies (a). Therefore, z satisfies either (a) or (b).

Moreover, the number of repetitions of the loop in (2) is bounded by $\mathcal{O}(t)$, thus by $\mathcal{O}(|\Lambda|)$. Therefore, SEARCH runs in time polynomial in $|\Lambda|$ and this proves the lemma.

□ **Proof of Lemma 4.7**

Now consider the following machine N_0.

{ The Description of N_0 }

(1) For a given input $x \in \Sigma^*$, N_0 computes an (M, N)-chain $\Lambda = [(y_1, \sigma_1), \cdots, (y_m, \sigma_m)]$ w.r.t. x with $1 \leq m \leq r(|x|)$ by using the above described method.

(2) If Λ is full, N_0 computes $\chi_A(x)$ from Λ with algorithm $\mathcal{A}_1$, outputs $(\chi_A(x))$, and halts.

(3) If Λ is not full, as $m = r(|x|)$, N_0 computes ζ with algorithm $\mathcal{A}_2$, outputs ζ and halts.

{End of the Description of N_0 }

From Lemma 4.7, Fact 4, and Lemma 4.3, it is not hard to see that for every $x \in \Sigma^*$, N_0 runs in time polynomial in $|x|$ and $x \in A$ if and only if $N_0(x)$ is satisfied by S. Therefore, $A \leq^{\mathrm{P}}_{k-\mathrm{tt}} S$. This proves the theorem.

□ **Proof of Theorem 4.6**

From this theorem, we can obtain the following one.

Theorem 4.8 *If a set A is strictly 1-wd self-reducible and $\leq^{\mathrm{P}}_{\mathrm{btt}}$ reducible to some sparse set, then A is in* P.

Proof Let A be a set which is strictly 1-wd self-reducible and $\leq^{\mathrm{P}}_{k-\mathrm{tt}}$-reducible to a sparse set S for some $k \in \mathbf{N}$. Then, by using Theorem 4.6 repeatedly, we have that A is $\leq^{\mathrm{P}}_{0-\mathrm{tt}}$-reducible to S. This implies that $A \in \mathrm{P}$. □

5 Sparse Bounded Truth-Table Hard Sets

In this section, we will consider the possibility of the existence of sparse bounded truth-table hard sets for some complexity classes by using the theorems we showed in the previous sections.

For $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-reducibility to sparse sets, we obtain the following theorem.

Theorem 5.1 *Let $K = \{Q\}\mathrm{P}$ for some polynomial-time decidable two-place predicate on $\mathbf{N} \times \mathbf{N}$. If there is a sparse $\leq^{\mathrm{SN}}_{\mathrm{btt}}$-hard set for K, then $K \subseteq \mathrm{NP} \cap \text{co-NP}$.*

Proof Let K be as in the hypothesis of the theorem and let S be a sparse $\leq^{SN}_{btt}$-hard set for K. Furthermore, let A be the 1-wd self-reducible set that is $\leq^{P}_{m}$-complete for K, as shown in Lemma 3.1. So, A is $\leq^{SN}_{btt}$-reducible to S and then from Theorem 4.5, A is in NP $\cap$ co-NP, and consequently $K \subseteq$ NP $\cap$ co-NP. □

For $\leq^{P}_{btt}$-reducibility to sparse sets, from Theorem 4.8 and the existence of strictly 1-wd self-reducible sets that have been shown to be complete for different classes in section 3, we obtain the following theorems.

Theorem 5.2 *Let $K = \{Q\}$P for some polynomial-time decidable two-place predicate Q on $\mathbf{N}\times\mathbf{N}$ such that* NP $\subseteq K \cup$ co$K$. *If there is a sparse $\leq^{P}_{btt}$-hard set for K, then $K =$* P.

Proof Let K be as in the hypothesis of the theorem and S be a sparse $\leq^{P}_{btt}$-hard set for K. Since NP $\subseteq K \cup$ co$K$, S is $\leq^{P}_{btt}$-hard for NP. So, by Lemma 3.3 and Theorem 4.8 we have P = NP.

Furthermore, since $\leq^{P}_{btt}$-reducibility implies $\leq^{SN}_{btt}$-reducibility, from Theorem 5.1, $K \subseteq$ NP $\cap$ co-NP. So, NP $\cap$ co-NP $= K$, and this implies $K =$ P. □

Theorem 5.3 *For any $k \geq 2$, if* MOD$_k$P *has a sparse $\leq^{P}_{btt}$-hard set then* MOD$_k$P = P.

Proof Let $k \geq 2$. Suppose that there is a sparse set S that is $\leq^{P}_{btt}$-hard for MOD$_k$P. Then, by Lemma 3.2 and Theorem 4.8 we have MOD$_k$P = P. □

Next we consider the consequences of theorems 5.1, 5.2 and 5.3. From Theorem 5.1, we obtain the following corollaries first stated in [23].

Corollary 5.4 *[23] If there is a sparse $\leq^{SN}_{btt}$-hard set for* NP, *then* PH = NP.

Proof Suppose that there is a sparse $\leq^{SN}_{btt}$-hard set for NP. Then, from Theorem 5.1, NP $\subseteq$ NP $\cap$ co-NP, and this implies NP = co-NP. Therefore, from Proposition 2.8, we have PH = NP. □

Corollary 5.5 *[23] If there is a sparse $\leq^{SN}_{btt}$-hard set for* PP, *then* PH = NP = PP.

Proof Suppose that there is a sparse $\leq^{SN}_{btt}$-hard set for PP. Then, from Theorem 5.1, PP $\subseteq$ NP $\cap$ co-NP. Since NP $\cup$ co-NP $\subseteq$ PP from Proposition 2.8, we have NP = co-NP = PP, and this implies PH = NP = PP. □

Moreover, we have a similar result for C$_=$P.

Corollary 5.6 *If there is a sparse $\leq^{SN}_{btt}$-hard set for* C$_=$P, *then* PH = NP = PP = C$_=$P.

Proof Suppose that there is a sparse $\leq^{SN}_{btt}$-hard set for C$_=$P. Then, from Theorem 5.1, C$_=$P $\subseteq$ NP $\cap$ co-NP. Since co-NP $\subseteq$ C$_=$P from Proposition 2.8, we have NP = co-NP = C$_=$P, and this implies PH = NP = C$_=$P. Furthermore, since PP $\subseteq$ NP$^{C_=P}$ from Proposition 2.8, we have PP = NP. □

On the other hand, from Theorem 5.2, we obtain the following corollaries.

Corollary 5.7 *[24] If there is a sparse $\leq^{P}_{btt}$-hard set for* NP, *then* NP = P.

Corollary 5.8 *[23] If there is a sparse $\leq^{P}_{btt}$-hard set for* PP, *then* PP = P.

Corollary 5.9 *If there is a sparse $\leq^{P}_{btt}$-hard set for* C$_=$P, *then* PP = C$_=$P = P.

Finally, we consider the class MOD$_k$P. For a class K, a set L is in BP $\cdot$ K if there exists a set A in K and a polynomial p such that for every $x \in \Sigma^*$,

$$\#\{y \in \Sigma^{=p(|x|)} : \chi_A(\langle x, y\rangle) = \chi_L(x)\} \geq \frac{1}{3}\cdot 2^{p(|x|)}.$$

The BP-operator is first introduced in [27]. Also, BPP is the class of sets for which there exists a probabilistic polynomial-time acceptor with error probability $\leq 1/3$ [12]. The following relationships between the BP-operator and the polynomial-time hierarchy are widely known.

Theorem 5.10
1. *[27] For every $k \geq 1$,* BP $\cdot \Sigma^{P}_{k} \subseteq \Pi^{P}_{k+1}$.
2. *[32]* PH $\subseteq$ BP $\cdot$ C$_=$P, PH $\subseteq$ BP $\cdot$ PP, *and* PH $\subseteq$ BP $\cdot$ MOD$_k$P, *where $k \geq 2$.*
3. *[27]* BPP = BP $\cdot$ P.
4. *[18, 29]* BPP $\subseteq \Sigma^{P}_{2} \cap \Pi^{P}_{2}$.

From Proposition 2.8, Theorem 5.1 and Theorem 5.10, we obtain the following corollaries.

Corollary 5.11 *Let K be any class chosen from* $\{$MOD$_2$P, MOD$_3$P, $\cdots\}$. *If K has sparse $\leq^{SN}_{btt}$-hard sets, then* PH $\subseteq \Sigma^{P}_{2} \cap \Pi^{P}_{2}$.

Corollary 5.12 *Let K be any class chosen from* $\{$MOD$_2$P, MOD$_3$P, $\cdots\}$. *If K has sparse $\leq^{P}_{btt}$-hard sets, then* PH = BPP.

Since there are complete sets for PSPACE that are 1-wd self-reducible [21], we can also derive the following result from Theorem 4.5 and Corollary 5.7.

Corollary 5.13 *1. If there is a sparse $\leq^{P}_{btt}$-hard set for* PSPACE, *then* PSPACE = P.

2. If there is a sparse $\leq^{SN}_{btt}$-hard set for PSPACE, *then* PSPACE = NP.

Note that this is the biggest class to which we can directly apply our technique, since all 1-wd self-reducible sets can be decided in PSPACE. However, although the class E (i.e., $\bigcup_c$ DTIME[2^{cn}]) probably does not have 1-wd self-reducible sets, we can apply our results to it in an indirect way and show that E has not sparse $\leq^{P}_{btt}$-hard sets. This is a particular case of the result of Watanabe saying that all $\leq^{P}_{btt}$-hard sets for E have bi-exponential density [36].

Corollary 5.14 *[36] There are not sparse $\leq^{P}_{btt}$-hard sets for* E.

Proof This is proved by contradiction. Assume that E has sparse $\leq^{P}_{btt}$-hard sets. Then, E has polynomial size circuits and, by [15], E = PSPACE. Also, by part 1. in last corollary, we know that PSPACE = P. This leads to the equality E = P, which is false. □

Acknowledgments The authors are very grateful to Richard Beigel, Ricard Gavaldà and Jacobo Torán for their careful reading of the manuscript and many valuable comments, which made the manuscript more readable and understandable. Also, they would like to thank Masao Ikekawa, who let them know the paper [10]. The first author would like to thank Professor Kojiro Kobayashi of Tokyo Institute of Tecnology for his valuable suggestions.

References

[1] L. Adleman and K. Manders, Reducibility, randomness, and intractability, *Proceedings of the 9th Annual Symposium on Theory of Computing* (ACM, 1977) 151-163.

[2] J. L. Balcázar, Self-reducibility, *Proceedings of the 4th Symposium on Theoretical Aspects of Computer Science* (Lecture Notes in Computer Science **247**, Springer-Verlag, 1987) 136-147; *Journal of Computer and System Sciences* **41** (1990) 367-388.

[3] J. L. Balcázar, J. Díaz, and J. Gabarró, *Structural Complexity I* (EATCS monographs on Theoretical Computer Science 11, Springer-Verlag, 1988).

[4] R. Beigel, J. Gill, and U. Hertrampf, Counting classes: Thresholds, parity, mods, and fewness, *Proceedings of the 7th Symposium on Theoretical Aspects of Computer Science* (Lecture Notes in Computer Science **415**, Springer-Verlag, 1990) 49-57.

[5] R. Beigel, L. A. Hemachandra, and G. Wechsung, On the power of probabilistic polynomial time: $P^{NP}[\log] \subseteq PP$, *Proceedings of the 4th Symposium on Structure in Complexity Theory* (IEEE, 1989) 225-227.

[6] R. Beigel, N. Reingold, and D. Spielman, PP is closed under intersection, Technical Report 803, Yale University, June 1990.

[7] L. Berman and J. Hartmanis, On isomorphism and density of NP and other sets, *SIAM Journal of Computing* **6** (1977) 305-322.

[8] R. V. Book and K. Ko, On sets truth-table reducible to sparse sets, *SIAM Journal of Computing* **17** (1988) 903-919.

[9] J. Cai and L. A. Hemachandra, On the power of parity polynomial time, *Proceedings of the 6th Symposium on Theoretical Aspects of Computer Science* (Lecture Notes in Computer Science **349**, Springer-Verlag, 1989) 229-240.

[10] P. Erdös and R. Rado, Intersection theorems for systems of sets, *Journal of London Mathematical Society* **35** (1960) 85-90.

[11] S. Fortune, A note on sparse complete sets, *SIAM Journal of Computing* **8** (1979) 431-433.

[12] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM Journal of Computing* **6** (1975) 675-695.

[13] T. Gundermann, N. A. Nasser, and G. Wechsung, A survey on counting classes, *Proceedings of the 5th Annual Symposium on Structure in Complexity Theory* (IEEE, 1990) 140-153.

[14] J. Kadin, $P^{NP[\log n]}$ and sparse Turing-complete sets for NP, *Proceedings of the 2nd Annual Symposium on Structure in Complexity Theory* (IEEE, 1987) 33-40.

[15] R. M. Karp and R. J. Lipton, Some connections between nonuniform and uniform complexity classes, *Proceedings of the 12th Annual Symposium on Theory of Computing* (ACM, 1980) 302-309.

[16] K. Ko, Distinguishing bounded reducibilities by sparse sets, *Proceedings of the 3rd Annual Symposium on Structure in Complexity Theory* (IEEE, 1988) 181-191.

[17] K. Ko, Distinguishing conjunctive and disjunctive reducibilities by sparse sets, *Information and Computation* **81** (1989) 62-87.

[18] C. Lautemann, BPP and the polynomial hierarchy, *Information Processing Letters* **17** (1983) 215-217.

[19] T. J. Long, On γ-reducibility versus polynomial time reducibility, *Theoretical Computer Science* **14** (1981) 91-101.

[20] T. J. Long, Strong nondeterministic polynomial time reducibilities, *Theoretical Computer Science* **21** (1982) 1-25.

[21] A. Lozano and J. Torán, Self-reducible sets of small density, accepted for publication in *Mathematical Systems Theory.*

[22] S. R. Mahaney, Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis, *Journal of Computer and System Sciences* **25** (1982) 130-143.

[23] M. Ogiwara, On sparse bounded truth-table hard sets for PP and NP, Manuscript, June 1990.

[24] M. Ogiwara and O. Watanabe, On polynomial time bounded truth-table reducibility of NP sets to sparse sets, *Proceedings of the 22nd Annual Symposium on Theory of Computing* (ACM, 1990) 457-467; also, accepted for publication in *SIAM Journal of Computing.*

[25] C. Papadimitriou and S. Zachos, Two remarks on the power of counting, *Proceedings of the 6th GI Conference on Theoretical Computer Science* (Lecture Notes in Computer Science **145**, Springer-Verlag, 1983) 269-276.

[26] D. A. Russo, Structural properties of complexity classes, Ph. D. dissertation, Department of Mathematics, University of California at Santa Barbara, 1985.

[27] U. Schöning, Probabilistic complexity classes and lowness, *Proceedings of the 2nd Annual Symposium on Structure in Complexity Theory* (IEEE, 1988) 2-8; *Journal of Computer and System Sciences* **39** (1988) 84-100.

[28] J. Simon, On the difference between one and many, *Proceedings of the 4th Colloquium on Automata, Languages and Programming* (Lecture Notes in Computer Science **52**, Springer-Verlag, 1977) 480-491.

[29] M. Sipser, A complexity theoretic approach to randomness, *Proceedings of the 15th Annual Symposium on Theory of Computing* (ACM, 1983) 330-335.

[30] L. J. Stockmeyer, The polynomial-time hierarchy, *Theoretical Computer Science* **3** (1977) 1-22.

[31] S. Toda, On the computational power of PP and $\oplus$P, *Proceedings of the 30th Symposium on Foundation of Computer Science* (IEEE, 1989) 514-519.

[32] S. Toda and M. Ogiwara, Counting classes are as hard as the polynomial-time hierarchy, Technical Report CSIM 90-09, University of Electro-Communications, July 1990.

[33] J. Torán, An oracle characterization of the counting hierarchy, *Proceedings of the 3rd Annual Symposium on Structure in Complexity Theory* (IEEE, 1988) 213-223.

[34] E. Ukkonen, Two results on polynomial time truth-table reductions to sparse sets, *SIAM Journal of Computing* **12** (1983) 580-587.

[35] K. W. Wagner, The complexity of combinatorial problems with succinct input representation, *Acta Informatica* **23** (1986) 325-356.

[36] O. Watanabe, On the structure of intractable complexity classes. Ph. D. dissertation, Department of Computer Science, Tokyo Institute of Technology, 1987.

[37] O. Watanabe, On $\leq^{\mathrm{P}}_{1\text{-tt}}$ sparseness and nondeterministic complexity classes, *15th International Colloquium on Algorithms and Linear Programming* (Lecture Notes in Computer Science **317**, Springer-Verlag, 1988) 697-709.

[38] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoretical Computer Science* **3** (1977) 23-33.

[39] C. K. Yap, Some consequences of non-uniform conditions on uniform classes, *Theoretical Computer Science* **26** (1983) 287-300.

[40] Y. Yesha, On certain polynomial-time truth-table reducibilities of complete sets to sparse sets, *SIAM Journal of Computing* **12** (1983) 411-425.