Electronic Colloquium on Computational Complexity, Report No. 68 (2005)

Redundancy in Complete Sets

Christian Glaßer[*], A. Pavan[†], Alan L. Selman[‡], Liyu Zhang[§]

July 6, 2005

Abstract

We show that a set is m-autoreducible if and only if it is m-mitotic. This solves a long standing open question in a surprising way. As a consequence of this unconditional result and recent work by Glaßer et al. [11], complete sets for all of the following complexity classes are m-mitotic: NP, coNP, $\oplus$P, PSPACE, and NEXP, as well as all levels of PH, MODPH, and the Boolean hierarchy over NP. In the cases of NP, PSPACE, NEXP, and PH, this at once answers several well-studied open questions. These results tell us that complete sets share a redundancy that was not known before.

We disprove the equivalence between autoreducibility and mitoticity for all polynomial-time-bounded reducibilities between 3-tt-reducibility and Turing-reducibility: There exists a sparse set in EXP that is polynomial-time 3-tt-autoreducible, but not weakly polynomial-time T-mitotic. In particular, polynomial-time T-autoreducibility does not imply polynomial-time weak T-mitoticity, which solves an open question by Buhrman and Torenvliet.

We generalize autoreducibility to define poly-autoreducibility and give evidence that NP-complete sets are poly-autoreducible.

1 Introduction

It is a well known observation that for many interesting complexity classes, all *known* complete sets contain "redundant" information. For example, consider SAT. Given a boolean formula ϕ one can produce two different formulas ϕ_1 and ϕ_2 such that the question of whether ϕ is satisfiable or not is equivalent to the question of whether ϕ_1 or ϕ_2 are satisfiable. Thus ϕ_1 and ϕ_2 contain information about ϕ. Another example is the Permanent. Given a matrix M, we can reduce the computation of the permanent of M to computing the permanent of $M+R$, $M+2R, \ldots, M+nR$, where R is a randomly chosen matrix. Thus information about the permanent of M is contained

[*]Lehrstuhl für Informatik IV,Universität Würzburg. Email: glasser@informatik.uni-wuerzburg.de.

[†]Department of Computer Science, Iowa State University. Research supported in part by NSF grants CCR-0344817 and CCF-0430807. Email: pavan@cs.iastate.edu

[‡]Department of Computer Science and Engineering, University at Buffalo. Research supported in part by NSF grant CCR-0307077. Email: selman@cse.buffalo.edu

[§]Department of Computer Science, University at Buffalo. Email:lzhang7@cse.buffalo.edu

ISSN 1433-8092

in a few random looking matrices. We interpret this as "SAT and Permanent contain redundant information".

In this paper we study the question of how much redundancy is contained in complete sets of complexity classes. There are several ways to measure "redundancy". We focus on the two notions *autoreducibility* and *mitoticity.*

Trakhtenbrot [16] defined a set A to be *autoreducible* if there is an oracle Turing machine M such that $A = L(M^A)$ and M on input x never queries x. For complexity classes like NP and PSPACE refined measures are needed. In this spirit, Ambos-Spies [2] defined the notion of polynomial-time autoreducibility and the more restricted form m-autoreducibility. A set A is *polynomial-time autoreducible* if it is autoreducible via a oracle Turing machine that runs in polynomial-time. A is *m-autoreducible* if A is polynomial-time many-one reducible to A via a function f such that $f(x) \neq x$ for every x. Both notions demand information contained in $A(x)$ to be present among strings different from x. In the case of m-autoreducibility, the redundancy in A is even more apparent—if a set A is m-autoreducible, then x and $f(x)$ have the same information about A.

A stronger form of redundancy is described by the notion of *mitoticity* which was introduced by Ladner [13] for the recursive setting and by Ambos-Spies [2] for the polynomial-time setting. A set A is *m-mitotic* if there is a set $S \in \mathrm{P}$ such that A, $A \cap S$, and $A \cap \overline{S}$ are polynomial-time many-one equivalent. Thus if a set is m-mitotic, then A can be split into two parts such that both parts have exactly the same information as the original set has.

Ambos-Spies [2] showed that if a set is m-mitotic, then it is m-autoreducible and he raised the question of whether the converse holds. In this paper we resolve this question and show that every m-autoreducible set is m-mitotic. This is our main result. Since its proof is very involved, we present our main combinatorial idea with help of a simplified graph problem which will be described in Section 3. This simplification drops many of the important details from our formal proof, but still contains the combinatorial core of the problem. Our main result is all the more surprising, because it is known [2] that polynomial-time T-autoreducibility does not imply polynomial-time T-mitoticity. We improve this and disprove the equivalence between autoreducibility and mitoticity for all polynomial-time-bounded reducibilities between 3-tt-reducibility and Turing-reducibility: There exists a sparse set in EXP that is polynomial-time 3-tt-autoreducible, but not weakly polynomial-time T-mitotic. In particular, polynomial-time T-autoreducible does not imply polynomial-time weakly T-mitotic. This result settles another open question raised by Buhrman and Torenvliet [8].

Our main result relates local redundancy to global redundancy in the following sense. If a set A is m-autoreducible, then x and $f(x)$ contain the same information about A. This can be viewed as local redundancy. Whereas if A is m-mitotic, then A can be split into two sets B and C such that A, B, and C are polynomial-time many-one equivalent. Thus the sets B and C have exactly the same information as the original set A. This can be viewed as global redundancy in A. Our main result states that local redundancy is the same as global redundancy.

As a consequence of this result and recent work of Glaßer et al. [11], we can show that all complete sets for many interesting classes such as NP, PSPACE, NEXP, and levels of PH are m-mitotic. Thus they all contain redundant information in a strong sense. This resolves several long standing open questions raised by Ambos-Spies [2], Buhrman, Hoene, and Torenvliet [7], and Buhrman and Torenvliet [8].

Our result can also be viewed as a step towards understanding the isomorphism conjecture [5]. This conjecture states that all NP-complete sets are isomorphic to each other. In spite of several years of research, we do not have any concrete evidence either in support or against the isomorphism conjecture[1]. It is easy to see that if the isomorphism conjecture holds for classes such as NP, PSPACE, and EXP, then complete sets for these classes are m-autoreducible as well as m-mitotic. Given our current inability to make progress about the isomorphism conjecture, the next best thing we can hope for is to make progress on statements that the isomorphism conjecture implies. We note that this is not an entirely new approach. For example, if the isomorphism conjecture is true, then NP-complete sets cannot be sparse. This motivated researchers to consider the question of whether complete sets for NP can be sparse. This line of research led to the beautiful results of Mahaney [14] and Ogiwara and Watanabe [15] who showed that complete sets for NP cannot be sparse unless P = NP. Our results show that another consequence of isomorphism, namely "NP-complete sets are m-mitotic" holds. Note that this is an unconditional result.

Buhrman et al. [6] and Buhrman and Torenvliet [9, 10] argue that it is critical to study the notions of autoreducibility and mitoticity. They showed that resolving questions regarding autoreducibility of complete sets leads to unconditional separation results. For example, consider the question of whether truth-table complete sets for PSPACE are non-adaptive autoreducible. An affirmative answer separates NP from NL, while a negative answer separates the polynomial-time hierarchy from PSPACE. They argue that this approach does not have the curse of *relativization* and is worth pursuing. We refer the reader to the recent survey by Buhrman and Torenvliet [10] for more details.

In Section 5, we extend the notion of autoreducibility and define *poly-autoreducibility.* A motivation for this is to understand the isomorphism conjecture and the notion of paddability. Recall that the isomorphism conjecture is true if and only if all NP-complete sets are paddable. Paddability implies the following: If L is paddable, then given x and a polynomial p, we can produce $p(|x|)$ distinct strings such that if x is in L, then all these strings are in L and if x is not in L, then none of these strings are in L. Autoreducibility implies that given x we can produce a *single* string y different from x such that $L(x) = L(y)$. A natural question that arises is whether we can produce more strings whose membership in L is the same as the membership of x in L. This leads us to the notion of $f(n)$-autoreducibility: A set L is $f(n)$-autoreducible, if there is a polynomial-time algorithm that on input x outputs $f(|x|)$ distinct strings (different from x) whose membership in L is the same as the membership of x in L. It is obvious that paddability implies poly-autoreducibility. The question of whether "NP complete sets are poly-autoreducible" is weaker than the question of whether "NP-complete sets are paddable."

We provide evidence for poly-autoreducibility of NP-complete sets. We show that if one-way permutations exist, then NP-complete sets are log-autoreducible. Moreover, if one-way permutations and quick pseudo-random generators exist, then NP-complete sets are poly-autoreducible. We also show that if NP-complete sets are poly-autoreducible, then they have infinite subsets that can be decided in linear-exponential time.

[1]It is currently believed that if one-way functions exist, then the isomorphism conjecture is false. However, we do not have a proof of this.

1.1 Previous Work

The question of whether complete sets for various classes are autoreducible has been studied extensively [17, 4, 6]. Beigel and Feigenbaum [4] showed that Turing complete sets for the classes that form the polynomial hierarchy, $\Sigma_i^{\mathrm{P}}, \Pi_i^{\mathrm{P}}$, and Δ_i^{P}, are Turing autoreducible. Thus, all Turing complete sets for NP are Turing autoreducible. Buhrman et al. [6] showed that Turing complete sets for EXP and Δ_i^{EXP} are autoreducible, whereas there exists a Turing complete set for EESPACE that is not Turing auto-reducible. Regarding NP, Buhrman et al. [6] showed that truth-table complete sets for NP are probabilistic truth-table autoreducible. Recently, Glaßer et al. [11] showed that complete sets for classes such as NP, PSPACE, Σ_i^{P} are m-autoreducible.

Buhrman, Hoene, and Torenvliet [7] showed that EXP complete sets are weakly many-one mitotic. This result was recently improved independently by Kurtz [10] and Glaßer et al. [11]. Glaßer et al. also showed that NEXP complete sets are weakly m-mitotic and PSPACE-complete sets are weak Turing-mitotic.

2 Preliminaries

We use standard notation and assume familiarity with standard resource-bounded reductions. We consider words in lexicographic order. All used reductions are polynomial-time computable.

Definition 2.1 ([2]) *A set A is* polynomially T-autoreducible *(T-autoreducible, for short) if there exists a polynomial-time-bounded oracle Turing machine M such that $A = L(M^A)$ and for all x, M on input x never queries x. A set A is* polynomially m-autoreducible *(m-autoreducible, for short) if $A \leq_m^p A$ via a reduction function f such that for all x, $f(x) \neq x$.*

Definition 2.2 ([2]) *A recursive set A is* polynomial-time T-mitotic *(T-mitotic, for short) if there exists a set $B \in \mathrm{P}$ such that $A \equiv_T^p A \cap B \equiv_T^p A \cap \overline{B}$. A is* polynomial-time m-mitotic *(m-mitotic, for short) if there exists a set $B \in \mathrm{P}$ such that $A \equiv_m^p A \cap B \equiv_m^p A \cap \overline{B}$.*

Definition 2.3 ([2]) *A recursive set A is* polynomial-time weakly T-mitotic *(weakly T-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and $A \equiv_T^p A_0 \equiv_T^p A_1$. A is* polynomial-time weakly m-mitotic *(weakly m-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and $A \equiv_m^p A_0 \equiv_m^p A_1$.*

Definition 2.4 *Let f be a function from $\mathbb{N}$ to $\mathbb{N}$. A set L is $f(n)$-*autoreducible*, if there is a polynomial-time algorithm $\mathcal{A}$ that on input x outputs $y_1, y_2, \cdots, y_m$ such that $f(|x|) = m$, if $x \in L$, then $\{y_1, y_2, \cdots, y_m\} \subseteq L$, and if $x \notin L$, then $\{y_1, y_2, \cdots, y_m\} \cap L = \emptyset$. A set is* poly-autoreducible*, if it is n^k-autoreducible for every $k \geq 1$.*

A language is DTIME($T(n)$)-*complex* if L does not belong to DTIME($T(n)$) almost everywhere; that is, every Turing machine M that accepts L runs in time greater than $T(|x|)$, for all but finitely many words x. A language L is *immune* to a complexity class $\mathcal{C}$, or $\mathcal{C}$-*immune*, if L is infinite and no infinite subset of L belongs to $\mathcal{C}$. A language L is *bi-immune* to a complexity class $\mathcal{C}$, or $\mathcal{C}$-*bi-immune*, if both L and $\overline{L}$ are $\mathcal{C}$-*immune*. Balcázar and Schöning [3] proved that for every time-constructible function T, L is DTIME($T(n)$)-complex if and only if L is bi-immune to DTIME($T(n)$).

3 m-Autoreducibility equals m-Mitoticity

It is easy to see that if a nontrivial language L is m-mitotic, then it is m-autoreducible. If L is m-mitotic, then there is a set $S \in \mathrm{P}$ such that $L \cap S \leq_m^p L \cap \overline{S}$ via some f and $L \cap \overline{S} \leq_m^p L \cap S$ via some g. On input x, the m-autoreduction for L works as follows: If $x \in S$ and $f(x) \notin S$, then output $f(x)$. If $x \notin S$ and $g(x) \in S$, then output $g(x)$. Otherwise, output a fixed element from $\overline{L} - \{x\}$.

So m-mitoticity implies m-autoreducibility. The main result of this paper shows that surprisingly the converse holds true as well, i.e., m-mitoticity and m-autoreducibility are equivalent notions.

Theorem 3.1 *Let L be any set such that $|\overline{L}| \geq 2$. L is m-autoreducible if and only if L is m-mitotic.*

Before proceeding to the proof we first mention the main ideas and the intuition behind the proof and describe the combinatorial core of the problem.

Assume that L is m-autoreducible via reduction function f. Given x, the repeated application of f yields a sequence of words $x, f(x), f(f(x)), \ldots$, which we call the trajectory of x. These trajectories either are infinite or end in a cycle of length at least 2. Note that as f is an autoreduction, $x \neq f(x)$.

At first glance it seems that m-mitoticity can be easily established by the following idea: In every trajectory, label the words at even positions with $+$ and all other words with $-$. Define S to be the set of strings whose label is $+$. With this 'definition' of S it seems that f reduces $L \cap S$ to $L \cap \overline{S}$ and $L \cap \overline{S}$ to $L \cap S$.

However, this labeling strategy has at least two problems. First, it is not clear that $S \in \mathrm{P}$; because given a string y, we have to compute the parity of the position of y in a trajectory. As trajectories can be of exponential length, this might take exponential time. The second and more fundamental problem is the following: The labeling generated above is *inconsistent* and not well defined. For example, let $f(x) = y$. To label y which trajectory should we use? The trajectory of x or the trajectory of y? If we use trajectory of x, y gets a label of $+$, whereas if we use the trajectory of y, then it gets a label of $-$. Thus S is not well defined and so this idea does not work. It fails because the labeling strategy is a global strategy. To label a string we have to consider all the trajectories in which x occurs. Every single x gives rise to a labeling of possibly infinitely many words, and these labelings may overlap in an inconsistent way.

We resolve this by using a *local labeling strategy*. More precisely, we compute a label for a given x just by looking at the neighboring values x, $f(x)$, and $f(f(x))$. It is immediately clear that such a strategy is well-defined and therefore defines a consistent labeling. We also should guarantee that this local strategy strictly alternates labels, i.e., x gets $+$ if and only if $f(x)$ gets $-$. Such an alternation of labels would help us to establish the m-mitoticity of L.

Thus our goal will be to find a local labeling strategy that has a nice alternation behavior. However, we settle for something less. Instead of requiring that the labels strictly alternate, we only require that given x, at least one of $f(x), f(f(x)), \cdots, f^m(x)$ gets a label that is different from the label of x, where m is polynomially bounded in the length of x. This suffices to show m-mitoticity.

The most difficult part in our proof is to show that there exists a local labeling strategy that has this weaker alternation property.

We now formulate the core underlying problem. To keep this proof sketch simpler, we make several assumptions and ignore several technical but important details. If we assume (for simplicity) that on strings $x \notin 1^*$ the autoreduction is length preserving such that $f(x) > x$, then we arrive at the following graph labeling problem.

Core Problem: Let G_n be a directed graph with 2^n vertices such that every string of length n is a vertex of G_n. Assume that 1^n is a sink, that nodes $u \neq 1^n$ have outdegree 1, and that $u < v$ for edges (u, v). For $u \neq 1^n$ let $s(u)$ denote u's unique successor, i.e., $s(u) = v$ if (u, v) is an edge. Find a strategy that labels each node with either $+$ or $-$ such that:

(i) Given a node u, its label can be computed in polynomial time in n.

(ii) There exists a polynomial p such that for every node u, at least one of the nodes $s(u), s(s(u)), \ldots, s^{p(n)}(u)$ gets a label that is different from the label of u.

We exhibit a labeling strategy with these properties. To define this labeling, we use the following *distance function*: $d(x, y) \overset{df}{=} \lfloor \log |y - x| \rfloor$ (our formal proof uses a variant of this function). The core problem is solved by the following local strategy.

```
0   // Strategy for labeling node x
1   let $y = s(x)$ and $z = s(y)$.
2   if $d(x,y) > d(y,z)$ then output $-$
3   if $d(x,y) < d(y,z)$ then output $+$
4   $r := d(x,y)$
5   output $+$ iff $\lfloor x/2^{r+1} \rfloor$ is even
```

Clearly, this labeling strategy satisfies condition (i). We give a sketch of the proof that it also satisfies condition (ii). Define $m = 5n$ and let $u_1, u_2, \ldots, u_m$ be a path in the graph. It suffices to show that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. Assume that this does not hold, say all these nodes get label $+$. So no output is made in line 2 and therefore, the distances $d(u_i, u_{i+1})$ do not decrease. Note that the distance function maps to natural numbers. If we have more than n increases, then the distance between u_{m-1} and u_m is bigger than n. Therefore, $u_m - u_{m-1} > 2^{n+1}$, which is impossible for words of length n. So along the path $u_1, u_2, \ldots, u_m$ there exist at least $m - n = 4n$ positions where the distance stays the same. By a pigeon hole argument there exist four consecutive such positions, i.e., nodes $v = u_i$, $w = u_{i+1}$, $x = u_{i+2}$, $y = u_{i+3}$, $z = u_{i+4}$ such that $d(v, w) = d(w, x) = d(x, y) = d(y, z)$. So for the inputs v, w, and x, we reach line 4 where the algorithm will assign $r = d(v, w)$. Observe that for all words w_1 and w_2, the value $d(w_1, w_2)$ allows an approximation of $w_2 - w_1$ up to a factor of 2. More precisely, $w - v$, $x - w$, and $y - x$ belong to the interval $[2^r, 2^{r+1})$. It is an easy observation that this implies that not all of the following values can have the same parity: $\lfloor v/2^{r+1} \rfloor$, $\lfloor w/2^{r+1} \rfloor$, and $\lfloor x/2^{r+1} \rfloor$. According to line 5, not all words v, w, and x obtain the same label. This is a contradiction which shows that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. This proves (ii) and solves the core of the labeling problem.

The labeling strategy allows the definition of a set $S \in \mathrm{P}$ such that whenever we follow the trajectory of x for more than $5|x|$ steps, then we find at least one alternation between S and $\overline{S}$. This establishes m-mitoticity for L.

Now we give a formal proof of Theorem 3.1.

The dyadic representation of natural numbers provides a one-one correspondence between words over $\Sigma = \{0, 1\}$ and natural numbers. This correspondence translates operations and relations over natural numbers to operation and relations over words. We denote the absolute value of an integer by $\mathrm{abs}(x)$. This avoids a conflict between the notation of the length of a word w and the notation of the absolute value of the integer represented by w. Moreover, $\log(x)$ denotes x's logarithm to base 2. We use the following proposition.

Proposition 3.2 *Let L be any set such that $|\overline{L}| \geq 2$. L is m-mitotic if and only if there exist a total $g \in \mathrm{PF}$ and a set $S \in \mathrm{P}$ such that for all x,*

1. *$x \in L \Leftrightarrow g(x) \in L$, and*
2. *$x \in S \Leftrightarrow g(x) \notin S$.*

Proof. Choose distinct words $w_1, w_2 \in \overline{L}$. If L is m-mitotic, then there exists $S \in \mathrm{P}$ such that $L \cap S \leq_m^p L \cap \overline{S}$ via some $g_1 \in \mathrm{PF}$ and $L \cap \overline{S} \leq_m^p L \cap S$ via some $g_2 \in \mathrm{PF}$. We may assume that $w_1 \in S$ and $w_2 \in \overline{S}$; otherwise the set $S \cup \{w_1\} - \{w_2\}$ can be used instead of S. Observe that the following function g satisfies the statements 1 and 2 from the proposition.

$$g(x) \stackrel{df}{=} \begin{cases} g_1(x) & : \text{ if } x \in S \text{ and } g_1(x) \in \overline{S} \\ w_2 & : \text{ if } x \in S \text{ and } g_1(x) \in S \\ g_2(x) & : \text{ if } x \in \overline{S} \text{ and } g_2(x) \in S \\ w_1 & : \text{ if } x \in \overline{S} \text{ and } g_2(x) \in \overline{S} \end{cases}$$

Now assume there exist a total $g \in \mathrm{PF}$ and an $S \in \mathrm{P}$ that satisfy the statements 1 and 2. It follows that $L \cap S \leq_m^p L \cap \overline{S}$ and $L \cap \overline{S} \leq_m^p L \cap S$, both via g. The following function reduces L to $L \cap S$.

$$g'(x) \stackrel{df}{=} \begin{cases} x & : \text{ if } x \in S \\ g(x) & : \text{ if } x \in \overline{S} \end{cases}$$

The following function reduces $L \cap S$ to L.

$$g''(x) \stackrel{df}{=} \begin{cases} x & : \text{ if } x \in S \\ w_1 & : \text{ if } x \in \overline{S} \end{cases}$$

This shows $L \equiv_m^p L \cap S \equiv_m^p L \cap \overline{S}$ and hence L is m-mitotic. $\square$

Proof. (Theorem 3.1) If L is m-mitotic, then there exist $S \in \mathrm{P}$ and $f_1, f_2 \in \mathrm{PF}$ such that $L \cap S \leq_m^p L \cap \overline{S}$ via f_1 and $L \cap \overline{S} \leq_m^p L \cap S$ via f_2. By assumption, there exist different words $v, w \in \overline{L}$. The following function is an m-autoreduction for L.

$$f'(x) \stackrel{df}{=} \begin{cases} f_1(x) & : \text{ if } x \in S \text{ and } f_1(x) \notin S \\ f_2(x) & : \text{ if } x \notin S \text{ and } f_2(x) \in S \\ \min(\{v, w\} - \{x\}) & : \text{ otherwise} \end{cases}$$

For the other direction, let us assume that L is m-autoreducible and let $f \in \mathrm{PF}$ be an m-autoreduction for L. Choose $k \geq 1$ such that f is computable in time $n^k + k$. Using Proposition 3.2, we show L's m-mitoticity as follows: We construct a total $g \in \mathrm{PF}$ and an $S \in \mathrm{P}$ such that $(x \in L \Leftrightarrow g(x) \in L)$ and $(x \in S \Leftrightarrow g(x) \notin S)$.

Let t be a tower function defined by: $t(0) = 0$ and $t(i+1) = t(i)^k + k$ for $i \geq 0$. Define the inverse tower function as $t^{-1}(n) = \min\{i \mid t(i) \geq n\}$. Note that $t^{-1} \in \mathrm{PF}$. We partition the set of all words according to the parity of the inverse tower function of their lengths.

$$\begin{aligned} S_0 &\stackrel{df}{=} \{x \mid t^{-1}(|x|) \equiv 0 (\mathrm{mod}\ 2)\} \\ S_1 &\stackrel{df}{=} \{x \mid t^{-1}(|x|) \equiv 1 (\mathrm{mod}\ 2)\} \end{aligned}$$

Note that $S_0, S_1 \in \mathrm{P}$.

The following *distance* function for natural numbers x and y plays a crucial role in our proof.

$$d(x, y) \stackrel{df}{=} \mathrm{sgn}(y - x) \cdot \lfloor \log(\mathrm{abs}(y - x)) \rfloor.$$

This function is computable in polynomial time. We define a set S (which will be used as separator for L) by the following algorithm which works on input x.

```
0  // Algorithm for set S
1  y := f(x), z := f(f(x))
2  if |y| > |x| then (accept iff x ∈ S_0)
3  if |z| > |y| then (accept iff y ∈ S_1)
4  if x = z then (accept iff x > f(x))
5  // here x, y, and z are pairwise different
6  if d(x,y) > d(y,z) then reject
7  if d(x,y) < d(y,z) then accept
8  r := d(x,y)
9  accept iff ⌊y/2^(abs(r)+1)⌋ is even
```

Observe that $S \in \mathrm{P}$. We will show $L \equiv_m^p L \cap S \equiv_m^p L \cap \overline{S}$ which implies that L is m-mitotic.

Claim 3.3 *Let y be any word and let $m = |y|$. If $\forall i \in [0, 6m+3], |f^i(y)| \geq |f^{i+1}(y)|$, then there exists $j \in [0, 6m+3]$ such that*

$$f^j(y) \in S \Leftrightarrow f^{j+1}(y) \notin S.$$

Proof. Assume the claim does not hold. Moreover, assume that for all $j \in [0, 6m+4]$, $f^j(y) \in S$. For the other case (i.e., for all $j \in [0, 6m+4]$, $f^j(y) \notin S$) one can argue analogously. Consider the algorithm for S.

Fact 1: For $j \in [0, 6m+2]$, the algorithm on input $f^j(y)$ stops either in line 7 or in line 9.

Assume there exists $j \in [0, 6m+2]$ such that the algorithm on input $f^j(y)$ stops in lines 2 or 3. In this case, $|f^j(y)| < |f^{j+1}(y)|$ or $|f^{j+1}(y)| < |f^{j+2}(y)|$ which contradicts our assumption.

Assume there exists $j \in [0, 6m+2]$ such that the algorithm on input $f^j(y)$ stops in lines 4. By assumption of the claim, $|f^j(y)| \geq |f^{j+1}(y)| \geq |f^{j+2}(y)|$. Moreover, $f^j(y) = f^{j+2}(y)$, since we stop in line 4. So $|f^j(y)| = |f^{j+1}(y)|$. Therefore, on both inputs, $f^j(y)$ and $f^{j+1}(y)$, the algorithm stops in line 4. Note that $f^j(y) \neq f^{j+1}(y)$, since f is an m-autoreduction. Hence by line 4, $f^j(y) \in S \Leftrightarrow f^{j+1}(y) \notin S$, which contradicts our assumption.

Assume there exists $j \in [0, 6m+2]$ such that the algorithm on input $f^j(y)$ stops in lines 6. So $f^j(y) \notin S$ which contradicts the assumption. This proves Fact 1.

$$\begin{aligned} J &\overset{df}{=} \{j \in [0, 6m+2] \,|\, \text{on input } f^j(y) \text{ the algorithms for } S \text{ stops in line 7}\} \\ K &\overset{df}{=} \{j \in [0, 6m+2] \,|\, \text{on input } f^j(y) \text{ the algorithms for } S \text{ stops in line 9}\} \end{aligned}$$

By Fact 1, $J \cup K = \{0, \ldots, 6m+2\}$. From the algorithm we see the following.

$$\forall j \in J, \quad d(f^j(y), f^{j+1}(y)) < d(f^{j+1}(y), f^{j+2}(y)) \tag{1}$$

$$\forall j \in K, \quad d(f^j(y), f^{j+1}(y)) = d(f^{j+1}(y), f^{j+2}(y)) \tag{2}$$

Case 1: $\|J\| > 2m$. Together with (1) and (2) this shows

$$d(f^{6m+3}(y), f^{6m+4}(y)) \; - \; d(f^0(y), f^1(y)) > 2m. \tag{3}$$

It follows that

$$d(f^{6m+3}(y), f^{6m+4}(y)) > m \tag{4}$$

or

$$d(f^0(y), f^1(y)) < -m. \tag{5}$$

Assume that (4) holds. By the assumption of the claim, $f^{6m+3}(y)$ and $f^{6m+4}(y)$ are words of length $\leq m$. So the length of $\mathrm{abs}(f^{6m+4}(y) - f^{6m+3}(y))$ is $\leq m$. From the dyadic representation of numbers it follows that $\log(\mathrm{abs}(f^{6m+4}(y) - f^{6m+3}(y))) < m+1$ and therefore, $d(f^{6m+3}(y), f^{6m+4}(y)) \leq m$. This is a contradiction, since we assumed that (4) holds.

Assume now that (5) holds. Again, $f^0(y)$ and $f^1(y)$ are words of length $\leq m$. So the length of $\mathrm{abs}(f^0(y) - f^1(y))$ is $\leq m$. It follows that $\log(\mathrm{abs}(f^1(y) - f^0(y))) < m+1$ and therefore, $d(f^0(y), f^1(y)) \geq -m$. This is a contradiction, since we assumed that (5) holds.

Case 2: $\|J\| \leq 2m$. Note that $[0, 6m+2]$ contains $6m+3$ elements while J contains at most $2m$ elements. So there exists $j \in [0, 6m]$ such that $j, j+1, j+2 \in K$. A look at the algorithm tells us the following.

$$d(f^j(y), f^{j+1}(y)) = d(f^{j+1}(y), f^{j+2}(y)) = d(f^{j+2}(y), f^{j+3}(y)) = d(f^{j+3}(y), f^{j+4}(y)) \qquad (6)$$

Define r as the number shown in (6), and let $z_1 \overset{\mathit{df}}{=} f^j(y)$, $z_2 \overset{\mathit{df}}{=} f^{j+1}(y)$, $z_3 \overset{\mathit{df}}{=} f^{j+2}(y)$, and $z_4 \overset{\mathit{df}}{=} f^{j+3}(y)$. Recall that $z_1, z_2, z_3 \in S$ and that on input of these words, the algorithm stops in line 9. Therefore, the following must hold.

$$\begin{aligned} a_1 &\overset{\mathit{df}}{=} \lfloor z_2 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even} && (7)\\ a_2 &\overset{\mathit{df}}{=} \lfloor z_3 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even} && (8)\\ a_3 &\overset{\mathit{df}}{=} \lfloor z_4 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even} && (9) \end{aligned}$$

Case 2a: $r = 0$. Here $z_2 \neq z_4$, since otherwise on input z_2 the algorithm stops in line 4 which contradicts Fact 1. Also, $z_2 \neq z_3$ and $z_3 \neq z_4$, since f is an m-autoreduction. From (6) and from the definition of the distance function d we obtain, either $z_2 = z_3 - 1 = z_4 - 2$, or $z_4 = z_3 - 1 = z_2 - 2$. So $z_4 - z_2$ equals 2 or -2, and hence $a_3 - a_1$ equals 1 or -1. The latter contradicts the observations (7) and (9).

Case 2b: $r > 0$. Here we have $z_1 < z_2 < z_3 < z_4$ and therefore, $a_1 \leq a_2 \leq a_3$.

Assume $a_1 = a_3$. Since $d(z_2, z_3) = r$, it holds that $\log(\mathrm{abs}(z_3 - z_2)) \geq r$ and hence, $z_3 - z_2 \geq 2^r$. The same argument shows $z_4 - z_3 \geq 2^r$. So $z_4 \geq z_2 + 2^{r+1} = z_2 + 2^{\mathrm{abs}(r)+1}$ and hence, $a_3 \geq a_1 + 1$. The latter contradicts the assumption $a_1 = a_3$.

So assume $a_1 < a_3$ which implies $a_3 - a_1 \geq 2$, since both values are even. Since a_2 is even as well, we obtain $a_2 - a_1 \geq 2$ or $a_3 - a_2 \geq 2$. If $a_2 - a_1 \geq 2$, then $z_3 - z_2 > 2^{r+1}$ and so $d(z_2, z_3) > r$. If $a_3 - a_2 \geq 2$, then $z_4 - z_3 > 2^{r+1}$ and so $d(z_3, z_4) > r$. Both conclusions contradict (6).

Case 2c: $r < 0$. Here we have $z_1 > z_2 > z_3 > z_4$ and therefore, $a_1 \geq a_2 \geq a_3$.

Assume $a_1 = a_3$. Since $d(z_2, z_3) = r$, it holds that $\log(\mathrm{abs}(z_3 - z_2)) \geq \mathrm{abs}(r)$ and hence, $z_2 - z_3 \geq 2^{\mathrm{abs}(r)}$. The same argument shows $z_3 - z_4 \geq 2^{\mathrm{abs}(r)}$. So $z_2 \geq z_4 + 2^{\mathrm{abs}(r)+1}$ and hence, $a_1 \geq a_3 + 1$. The latter contradicts the assumption $a_1 = a_3$.

So assume $a_1 > a_3$ which implies $a_1 - a_3 \geq 2$, since both values are even. Since a_2 is even as well, we obtain $a_1 - a_2 \geq 2$ or $a_2 - a_3 \geq 2$. If $a_1 - a_2 \geq 2$, then $z_2 - z_3 > 2^{\mathrm{abs}(r)+1}$ and so $d(z_2, z_3) < -\mathrm{abs}(r) = r$. If $a_2 - a_3 \geq 2$, then $z_3 - z_4 > 2^{\mathrm{abs}(r)+1}$ and so $d(z_3, z_4) < -\mathrm{abs}(r) = r$. Both conclusions contradict (6).

This proves Claim 3.3. □

Claim 3.4 *There exists a total $r \in \mathrm{PF}$ such that $L \leq^p_m L$ via r and for every x,*

1. $|f(r(x))| \leq |r(x)|$ *or*
2. $x \in S \Leftrightarrow r(x) \notin S$.

Proof. For every x, let

$$r(x) \stackrel{df}{=} f^i(x)$$

where i is the smallest number such that $|f^{i+1}(x)| \leq |f^i(x)|$ or $(x \in S \Leftrightarrow f^i(x) \notin S)$. We will prove that such i exists. Consider the following algorithm which works on input x.

```
0   // Algorithm for function r
1   z := x
2   while (|f(z)| > |z| and (x ∈ S ⇔ z ∈ S))
3       // here |z| < |x|^k + k
4       z := f(z)
5   end
6   output z
```

Observe that this algorithm computes the function r.

We prove the invariant in line 3, which will guarantee that the loop in the algorithm halts within polynomial steps in $|x|$. Assume that at some point this invariant does not hold. We consider the first time when this happens. In this case, we must have reached line 3 before, since otherwise $|x| \geq |x|^k + k$ which is not possible. Let z' denote the value of variable z when line 3 was reached last time. So $z = f(z')$. Note that the following inequalities hold, since otherwise the algorithm stops earlier.

$$|x| < |f(x)| \tag{10}$$
$$|z'| < |f(z')| \tag{11}$$
$$|z| < |f(z)| \tag{12}$$
$$|x| < |z'| \tag{13}$$

Moreover,

$$|z'| < |x|^k + k, \tag{14}$$

since otherwise already z' violates the invariant, which contradicts the fact that with z we chose the earliest violation of the invariant. From (13) and (14) we obtain

$$t^{-1}(|x|) \leq t^{-1}(|z'|) \leq t^{-1}(|x|^k + k) = t^{-1}(|x|) + 1. \tag{15}$$

From (10) it follows that on input x, the algorithm for S stops in line 2. We see the same for z' and z using (11) and (12). This implies the following.

$$x \in S \Leftrightarrow x \in S_0 \tag{16}$$
$$z' \in S \Leftrightarrow z' \in S_0 \tag{17}$$
$$z \in S \Leftrightarrow z \in S_0 \tag{18}$$

Note that

$$x \in S \Leftrightarrow z' \in S \Leftrightarrow z \in S, \tag{19}$$

since otherwise the algorithm for r stops earlier. Together with (16), (17), and (18) this shows

$$x \in S_0 \Leftrightarrow z' \in S_0 \Leftrightarrow z \in S_0 \tag{20}$$

and therefore,

$$t^{-1}(|x|) \equiv t^{-1}(|z'|) \equiv t^{-1}(|z|) \pmod 2. \tag{21}$$

Now (15) implies $t^{-1}(|x|) = t^{-1}(|z'|)$ and we obtain

$$t^{-1}(|z'|) = t^{-1}(|x|) < t^{-1}(|x|^k + k) \leq t^{-1}(|z|). \tag{22}$$

From (21) and (22) it follows that $t^{-1}(|z|) - t^{-1}(|z'|) \geq 2$. Therefore, $|f(z')| > |z'|^k + k$. This contradicts f's computation time and proves the invariant in line 3.

From the invariant we immediately obtain that every single step of the algorithm can be carried out in time polynomial in $|x|$. Each execution of line 4 increases the length of z. By our invariant, the algorithm must terminate within $|x|^k + k$ iterations of the loop. This shows that r is total and polynomial-time computable. Since r is defined by repeated applications of f, and since f is an autoreduction of L, we obtain $L \leq_m^p L$ via r. The statements 1 and 2 of the claim follow immediately from line 2 of the algorithm. This proves Claim 3.4. $\square$

Choose a function r according to Claim 3.4. Define a function g by the following algorithm which works on input x. Below we will show that g satisfies the conditions in Proposition 3.2.

```
0   // Algorithm for function g
1   $y := r(x)$, $m := |y|$
2   if $|y| < |f(y)|$ then return y
3   // here $|y| \geq |f(y)|$
4   $z := y$
5   for $i := 0$ to $6m + 3$
6      // here $z = f^i(y)$, $|z| \leq m$, and for all $0 \leq j \leq i$, $|f^j(y)| \geq |f^{j+1}(y)|$
7      if $|f(z)| < |f(f(z))|$ then
8         if $(f(z) \in S \Leftrightarrow x \in S)$ then return z else return f(z)
9      endif
10     $z := f(z)$
11  next i
12  // here for all $0 \leq j \leq 6m + 3$, $|f^j(y)| \geq |f^{j+1}(y)|$
13  $z := y$
14  for $i := 0$ to $6m + 3$
15     // here $z = f^i(y)$ and $|z| \leq m$
16     if $z \in S \Leftrightarrow f(z) \notin S$ then
17        if $(z \in S \Leftrightarrow x \in S)$ then return f(z) else return z
18     endif
19     $z := f(z)$
20  next i
21  // this line is never reached
```

Claim 3.5 *The statements claimed in the comments of the algorithm for g hold true.*

Proof. Clearly, the condition in line 3 holds. Observe that whenever we reach line 6, then $z = f^i(y)$ and $|z| \geq |f(z)|$. Therefore, the condition in line 6 holds. It follows that if we reach line 12, then we must have passed line 6 for $i = 6m+3$. This shows the condition in line 12. Whenever we reach line 15 it holds that $z = f^i(y)$. From the condition in line 12 it follows that $|z| \leq m$ in line 15.

Finally we argue that we do not reach line 21. Assume that we reach line 12. By the condition in line 12, we satisfy the assumption of Claim 3.3. Therefore, there exists $j \in [0, 6m+3]$ such that $f^j(y) \in S \Leftrightarrow f^{j+1}(y) \notin S$. So for $i = j$, the condition in line 16 is true and therefore, the algorithm stops before reaching line 21. □

Claim 3.6 *g is a total function in* PF *and $L \leq_m^p L$ via g.*

Proof. We immediately see that g is total, since line 21 is never reached.

We argue that $g \in \mathrm{PF}$. Recall that f and r are total functions in PF, and recall that $S \in \mathrm{P}$. So steps 1–4 are computable in polynomial time in $|x|$. Note that m is polynomially bounded in $|x|$. By the remark in line 6, the loop 5–11 needs only polynomial time in $|x|$. The remark in line 15 implies the same for the loop 14–20. This shows $g \in \mathrm{PF}$.

We show $L \leq_m^p L$ via g. Observe that in any case the algorithm returns $f^j(y)$ for a suitable $j \geq 0$. By Claim 3.4, $x \in L \Leftrightarrow y = r(x) \in L$. Since f is an autoreduction of L, we obtain $x \in L \Leftrightarrow g(x) = f^j(y) \in L$. □

Claim 3.7 *For every x, $x \in S \Leftrightarrow g(x) \notin S$.*

Proof. Consider the computation of the algorithm for g on input x.

Case 1: The output is made in line 2. So we have $|f(r(x))| > |r(x)|$. From Claim 3.4 it follows $x \in S \Leftrightarrow g(x) = r(x) \notin S$.

Case 2: The output is made in line 8. By lines 6 and 7,

$$|f^i(y)| \geq |f^{i+1}(y)| \quad \text{and} \quad |f^{i+1}(y)| < |f^{i+2}(y)|.$$

Therefore, if we look at the algorithm for S, then we see that on input $f^i(y)$ the algorithm stops in step 3, while on input $f^{i+1}(y)$ the algorithm stops in step 2. It follows that

$$\begin{aligned} f^i(y) \in S \quad &\Leftrightarrow \quad f^{i+1}(y) \in S_1 \quad \text{and} \\ f^{i+1}(y) \in S \quad &\Leftrightarrow \quad f^{i+1}(y) \in S_0. \end{aligned}$$

So $z = f^i(y) \in S \Leftrightarrow f(z) \notin S$ and therefore, by line 8 of the algorithm for g,

$$x \in S \Leftrightarrow g(x) \notin S.$$

Case 3: The output is made in line 17. From line 16 it follows that $x \in S \Leftrightarrow g(x) \notin S$. $\square$

The Claims 3.6 and 3.7 allow the application of Proposition 3.2. Hence L is m-mitotic. $\square$

Call a set L *nontrivial* if $\|L\| \geq 2$ and $\|\overline{L}\| \geq 2$.

Corollary 3.8 *Every nontrivial set that is many-one complete for one of the following complexity classes is m-mitotic.*

- NP*,* coNP*,* $\oplus$P*,* PSPACE*,* EXP*,* NEXP
- *any level of* PH*,* MODPH*, or the Boolean hierarchy over* NP

Proof. Glaßer et al. [11] showed that all many-one complete sets of the above classes are m-autoreducible. By Theorem 3.1, these sets are m-mitotic. $\square$

Corollary 3.9 *A nontrivial set L is* NP*-complete if and only if L is the union of two disjoint* P*-separable* NP*-complete sets.*

So unions of disjoint P-separable NP-complete sets form exactly the class of NP-complete sets. What class is obtained when we drop P-separability? Does this class contain a set that is not NP-complete? In other words, is the union of disjoint NP-complete sets always NP-complete? We leave this as an open question.

Ambos-Spies [2] defined a set A to be *ω-m-mitotic* if for every n there exists a partition $(Q_1, \ldots, Q_n)$ of Σ^* such that the following sets are polynomial-time many-one equivalent: $A, A \cap Q_1, \ldots, A \cap Q_n$.

Corollary 3.10 *Every nontrivial infinite set that is many-one complete for a class mentioned in Corollary 3.8 is ω-m-mitotic.*

4 3-tt-Autoreducibility does not imply Weak T-Mitoticity

In this section we prove a theorem that shows in a strong way that T-autoreducible does not imply weakly T-mitotic. Hence, our main theorem cannot be generalized.

Lemma 4.1 *Let $l, m \geq 0$ and let $k \geq (l+2)^{2^m}$. If $Q_1, \ldots, Q_k$ are sets of cardinality $\leq l$ and if $n_1, \ldots, n_k$ are pairwise different numbers, then there exist pairwise different indices $i_1, \ldots, i_m$ such that for all $s, t \in [1, m]$,*

$$s \neq t \;\Rightarrow\; n_{i_s} \notin Q_{i_t}.$$

Proof. The proof is by induction on $n = l + m$ such that the induction base covers all cases where $l = 0$ or $m = 0$. For these cases the lemma holds trivially. In particular, this covers the case $n = 1$.

Assume there exists $n \geq 1$ such that the lemma holds for all l and m such that $l = 0$ or $m = 0$ or $l + m \leq n$. Now we prove it for l and m such that $l \geq 1$, $m \geq 1$, and $l + m = n + 1$.

Case 1: There exist at least $k - \sqrt{k} - l - 1$ indices $j > 1$ such that $n_1 \in Q_j$. Let $k' = \lceil k - \sqrt{k} - l - 1 \rceil$ and choose pairwise different indices $j_1, \ldots, j_{k'}$ such that for all i, $j_i \neq 1$ and $n_1 \in Q_{j_i}$. Let $l' = l - 1$ and let $R_i = Q_{j_i} - \{n_1\}$ and $r_i = n_{j_i}$ for $1 \leq i \leq k'$. Observe $l' \geq 0$ and $m \geq 1$. We estimate k' as follows.

$$\begin{aligned}
k' &\geq k - \sqrt{k} - l - 1 \\
&\geq (l+2)^{2^m} - \sqrt{(l+2)^{2^m}} - l - 1 \quad &&\text{(since } (a \geq b \Rightarrow a - \sqrt{a} \geq b - \sqrt{b}) \text{ for } a, b \geq 1) \\
&\geq (l'+2)^{2^m} &&\text{(follows from (24) in the estimation below)} \qquad (23)
\end{aligned}$$

For (23) the following estimation is needed.

$$\begin{aligned}
l + 1 &\geq l + 1 \\
(l+2)^{2^{m-1}-1} \cdot (l + 2 - 1) &\geq (l+1)^{2^{m-1}-1} \cdot (l+1) \\
(l+2)^{2^{m-1}} - 1 &\geq (l+1)^{2^{m-1}} \qquad \text{(since } (l+2)^{2^{m-1}-1} \geq 1) \\
(l+2)^{2^{m-1}} \cdot \left[(l+2)^{2^{m-1}} - 1\right] &\geq (l+2)^{2^{m-1}} \cdot \left[(l+1)^{2^{m-1}}\right] \\
(l+2)^{2^m} - (l+2)^{2^{m-1}} &\geq (l+1+1) \cdot (l+1)^{2^{m-1}-1} \cdot \left[(l+1)^{2^{m-1}}\right] \\
(l+2)^{2^m} - \sqrt{(l+2)^{2^m}} &\geq (l+1)^{2^m} + (l+1)^{2^m-1} \\
(l+2)^{2^m} - \sqrt{(l+2)^{2^m}} - l - 1 &\geq (l'+2)^{2^m} \qquad \text{(since } (l+1)^{2^m-1} \geq l+1) \qquad (24)
\end{aligned}$$

Note that $l' + m = n$. Also, $R_1, \ldots, R_{k'}$ are sets of cardinality $\leq l'$ and $r_1, \ldots, r_{k'}$ are pairwise different numbers. By induction hypothesis there exist pairwise different indices $i_1, \ldots, i_m$ such that for all $s, t \in [1, m]$, $(s \neq t \Rightarrow r_{i_s} \notin R_{i_t})$. For all $s \in [1, m]$, $r_{i_s} \neq n_1$. Therefore, for all $s, t \in [1, m]$,

$$(s \neq t \Rightarrow r_{i_s} \notin R_{i_t} \cup \{n_1\})$$

and hence

$$(s \neq t \Rightarrow n_{j_{i_s}} \notin Q_{j_{i_t}}).$$

So the lemma is satisfied by the indices $j_{i_1}, j_{i_2}, \ldots, j_{i_m}$.

Case 2: There exist less than $k - \sqrt{k} - l - 1$ indices $j > 1$ such that $n_1 \in Q_j$. So there exist more than $\sqrt{k} + l$ indices $j > 1$ such that $n_1 \notin Q_j$. Since $\|Q_1\| \leq l$, there exist more than $\sqrt{k}$ indices $j > 1$ such that $n_1 \notin Q_j$ and $n_j \notin Q_1$. Hence there exist at least $k' \overset{df}{=} \lceil \sqrt{k} \rceil$ such indices. So we can choose pairwise different indices $j_1, \ldots, j_{k'}$ such that for all i,

$$j_i \neq 1 \wedge n_1 \notin Q_{j_i} \wedge n_{j_i} \notin Q_1. \qquad (25)$$

Let $m' = m - 1$ and let $R_i = Q_{j_i}$ and $r_i = n_{j_i}$ for $1 \leq i \leq k'$. Note that $l \geq 1$ and $m' \geq 0$. Observe that

$$k' \geq \sqrt{k} \geq \sqrt{(l+2)^{2^m}} = (l+2)^{2^{m'}}$$

and $l + m' = n$. Also, $R_1, \dots, R_{k'}$ are sets of cardinality $\leq l$ and $r_1, \dots, r_{k'}$ are pairwise different numbers. So by induction hypothesis there exist pairwise different indices $i_1, \dots, i_{m'}$ such that for all $s, t \in [1, m']$,

$$s \neq t \Rightarrow r_{i_s} \notin R_{i_t}$$

and hence

$$s \neq t \Rightarrow n_{j_{i_s}} \notin Q_{j_{i_t}}.$$

From (25) it follows that the lemma is satisfied by the indices $1, j_{i_1}, j_{i_2}, \dots, j_{i_{m'}}$. □

Theorem 4.2 *There exists* $L \in \mathrm{SPARSE} \cap \mathrm{EXP}$ *such that*

- *L is 3-tt-autoreducible, but*
- *L is not weakly T-mitotic.*

Proof. Define a tower function by $t(0) = 4$ and

$$t(n+1) = 2^{2^{2^{2^{t(n)}}}}.$$

For any word s, let $W(s) = \{s00, s01, s10, s11\}$. We will define L such that it satisfies the following:

(i) If $w \in L$, then there exists n such that $|w| = t(n)$.

(ii) For all n and all $s \in \Sigma^{t(n)-2}$, the set $W(s) \cap L$ either is empty or contains exactly two elements.

It is easy to see that such an L is 3-tt-autoreducible: On input w, determine n such that $|w| = t(n)$. If such n does not exist, then reject. Otherwise, let s be w's prefix of length $|w| - 2$. Accept if and only if the set $L \cap (W(s) - \{w\})$ contains an odd number of elements. This is a 3-tt-autoreduction.

We turn to the construction of L. Let $M_1, M_2, \dots$ be an enumeration of deterministic, polynomial-time-bounded Turing machines such that the running time of M_i is $n^i + i$. Let $\langle \cdot, \cdot \rangle$ be a pairing function such that $\langle x, y \rangle > x + y$. We construct L stagewise such that in stage n we determine which of the words of length $t(n)$ belong to L. In other words, at stage n we define a set $W_n \subseteq \Sigma^{t(n)}$, and finally we define L to be the union of all W_n.

We start by defining $W_0 = \emptyset$. Suppose we are at stage $n > 0$. Let $m = t(n)$ and determine i and j such that $n = \langle i, j \rangle$. If such i and j do not exist, then let $W_n = \emptyset$ and go to stage $n+1$. Otherwise, i and j exist. In particular, $i + j < \log\log m$. Let $O \overset{df}{=} W_0 \cup \cdots \cup W_{n-1}$ be the part of L that has been constructed so far. Let $O_1, O_2, \dots, O_l$ be the list of all subsets of O (lexicographically ordered according to their characteristic sequences). Since $O \subseteq \Sigma^{\leq t(n-1)}$ we obtain $\|O\| \leq 2^{t(n-1)+1}$. Therefore,

$$l \leq 2^{2^{t(n-1)+1}} \leq 2^{2^{2^{t(n-1)}}} = \log\log t(n) = \log\log m. \tag{26}$$

We give some intuition for the claim below. If L is weakly T-mitotic, then in particular, there exists a partition $L = L_1 \cup L_2$ such that $L_2 \leq_T^p L_1$ via some machine M_i. Hence $O \cap L_1$ must appear (say as O_k) in our list of subsets of O. The following claim makes sure that we can find a list of words $s_1, \dots, s_l$ of length $m-2$ such that for all $k \in [1,l]$ it holds that if the partition of L is such that $O \cap L_1 = O_k$, then M_i on input of a string from $\{s_k00, s_k01, s_k10, s_k11\}$ does not query the oracle for words from $W(s_r)$ if $r \neq k$. Hence, if M_i queries a word of length m that does not belong to $\{s_k00, s_k01, s_k10, s_k11\}$, then it always gets a no answer. So the following is the only information about the partition of L that can be exploited by M_i:

- the partition of $O = \Sigma^{<t(n)} \cap L$
- the partition of $W(s_k) \cap L$

In particular, M_i cannot exploit information about the partition of $W(s_r) \cap L$ for $r \neq k$. This independence of M_i makes our diagonalization possible.

Claim 4.3 *There exist pairwise different words $s_1, \dots, s_l \in \Sigma^{m-2}$ such that for all $k, r \in [1,l]$, $k \neq r$, and all $y \in W(s_k)$, neither $M_i^{O-O_k}(y)$ nor $M_j^{O_k}(y)$ query the oracle for words in $W(s_r)$.*

Proof. For $s \in \Sigma^{m-2}$, let

$$Q_s \overset{df}{=} \{s' \in \Sigma^{m-2} \,|\, \exists k \in [1,l],\ \exists y \in W(s),\ \exists q \in W(s') \text{ such that } q \text{ is queried by } M_i^{O-O_k}(y) \text{ or } M_j^{O_k}(y)\}.$$

Observe that for every $s \in \Sigma^{m-2}$,

$$\begin{aligned} \|Q_s\| &\leq 4l[(m^i + i) + (m^j + j)] \\ &\leq 4(\log\log m)[m^{\log\log m} + \log\log m] \\ &\leq 8(\log\log m)m^{\log\log m} \\ &\leq m^{2\log\log m} \\ &\leq 2^{\log^2 m} - 2. \end{aligned} \tag{27}$$

We identify numbers in $[1, 2^{m-2}]$ with strings in Σ^{m-2}. Considered in this way, each Q_s is a subset of $[1, 2^{m-2}]$. By (27), $Q_1, Q_2, \dots, Q_{2^{m-2}}$ are sets of cardinality $\leq 2^{\log^2 m} - 2$. Clearly, $1, 2, \dots, 2^{m-2}$ are pairwise different numbers. By (26),

$$2^{m-2} \geq (2^{\log^2 m})^{\log m} \geq (2^{\log^2 m})^{2^l}.$$

Therefore, we can apply Lemma 4.1. We obtain indices $s_1, \dots, s_l$ such that for all $k, r \in [1,l]$,

$$r \neq k \ \Rightarrow\ s_r \notin Q_{s_k}. \tag{28}$$

Assume there exist $k, r \in [1,l]$, $k \neq r$, and $y \in W(s_k)$ such that some $q \in W(s_r)$ is queried by $M_i^{O-O_k}(y)$ or $M_j^{O_k}(y)$. Hence $s_r \in Q_{s_k}$. This contradicts (28) and finishes the proof of Claim 4.3.

$\square$

Let $s_1, \ldots, s_l \in \Sigma^{m-2}$ be the words assured by Claim 4.3. We define W_n such that for every $k \in [1, l]$ we define a set $V_k \subseteq W(s_k)$, and finally we define W_n to be the union of all V_k. The cardinality of each V_k is either 0 or 2.

Fix some $k \in [1, l]$ and let $Q_k \overset{df}{=} O - O_k$.

Case 1: $M_i^{Q_k}(s_k00)$ accepts or $M_j^{O_k}(s_k00)$ accepts. Define $V_k \overset{df}{=} \emptyset$.

Case 2: $M_i^{Q_k}(s_k00)$ and $M_j^{O_k}(s_k00)$ reject.

Case 2a: For all $y \in \{s_k01, s_k10, s_k11\}$, $M_i^{Q_k \cup \{s_k00\}}(y)$ rejects. Define V_k as a subset of $W(s_k)$ such that $|V_k| = 2$, $s_k00 \in V_k$, and

$$s_k01 \in V_k \Leftrightarrow M_j^{O_k \cup \{s_k00\}}(s_k01) \text{ rejects.}$$

Case 2b: For all $y \in \{s_k01, s_k10, s_k11\}$, $M_j^{O_k \cup \{s_k00\}}(y)$ rejects. Define V_k as a subset of $W(s_k)$ such that $|V_k| = 2$, $s_k00 \in V_k$, and

$$s_k01 \in V_k \Leftrightarrow M_i^{Q_k \cup \{s_k00\}}(s_k01) \text{ rejects.}$$

Case 2c: $\exists y \in \{s_k01, s_k10, s_k11\}$ and $\exists z \in \{s_k01, s_k10, s_k11\}$ such that $M_i^{Q_k \cup \{s_k00\}}(y)$ accepts and $M_j^{O_k \cup \{s_k00\}}(z)$ accepts. Choose $v \in W(s_k) - \{s_k00, y, z\}$ and define $V_k \overset{df}{=} \{s_k00, v\}$.

This finishes the construction of V_k. We define $W_n \overset{df}{=} \bigcup_{k \in [1,l]} V_k$. Finally, L is defined as the union of all W_n.

Note that by the construction, $W_n \subseteq \Sigma^{t(n)}$ which shows (i). Observe that the construction also ensures (ii). We argue for $L \in \mathrm{EXP}$: Since $l \leq \log\log m$, there are not more than $2^{m \log\log m}$ possibilities to choose the strings $s_1, \ldots, s_l$. For each such possibility we have to simulate $O(l^2)$ computations $M_i(y)$ and $M_j(y)$. This can be done in exponential time in m. For the definition of each V_k we have to simulate a constant number of computations $M_i(y)$ and $M_j(y)$. This shows that L is printable in exponential time. Hence $L \in \mathrm{EXP}$. From the construction it follows that $L \cap \Sigma^m \leq 2l \leq 2 \log\log m$. In particular, $L \in \mathrm{SPARSE}$. It remains to show that L is not weakly T-mitotic.

Assume L is weakly T-mitotic. So L can be partitioned into $L = L_1 \cup L_2$ (a disjoint union) such that

(iii) $L_1 \leq_T^p L_2$ via machine M_i and

(iv) $L_2 \leq_T^p L_1$ via machine M_j.

Let $n = \langle i, j\rangle$, $m = t(n)$, and $O = W_0 \cup \cdots \cup W_{n-1}$, i.e., $O = L \cap \Sigma^{<t(n)}$. Let $O_1, O_2, \ldots, O_l$ be the list of all subsets of O (again lexicographically ordered according to their characteristic sequences). Let $s_1, \ldots, s_l$ and $V_1, \ldots, V_l$ be as in the definition of W_n. Choose $k \in [1, l]$ such that $L_1 \cap \Sigma^{<t(n)} = O_k$. Let $Q_k = O - O_k$. So $L_2 \cap \Sigma^{<t(n)} = Q_k$. Clearly, V_k must be defined according to one of the cases above.

Assume V_k was defined according to Case 1: So $V_k = \emptyset$ and in particular, $s_k00 \notin L_1$. Without loss of generality assume that $M_i^{Q_k}(s_k00)$ accepts. $M_i^{L_2}(s_k00)$ has running time $m^i{+}i < m^m{+}m < t(n{+}1)$. Hence $M_i^{L_2}(s_k00)$ behaves like $M_i^{L_2 \cap \Sigma^{\leq t(n)}}(s_k00)$. Since s_k was chosen according to Claim 4.3, for all $r \in [1, l] - \{k\}$, $M_i^{Q_k}(s_k00)$ does not query the oracle for words in $W(s_r)$. Note that $W(s_k) \cap L = V_k = \emptyset$. Therefore, $M_i^{L_2}(s_k00)$ behaves like $M_i^{L_2 \cap \Sigma^{<t(n)}}(s_k00)$ which is the same as $M_i^{Q_k}(s_k00)$. The latter accepts, but $s_k00 \notin L_1$. This contradicts (iii).

Assume V_k was defined according to Case 2: So $V_k = \{s_k00, u\}$ where $u \in \{s_k01, s_k10, s_k11\}$. Assume $V_k \subseteq L_1$. Then as above, $M_i(s_k00)$ with oracle L_2 behaves the same way as $M_i(s_k00)$ with oracle Q_k. The latter rejects, because we are in Case 2. So $s_k00 \notin L_1$ which contradicts our assumption. Analogously the assumption $V_k \subseteq L_2$ implies a contradiction. Therefore,

$$\text{either } (s_k00 \in L_1 \wedge u \in L_2) \text{ or } (u \in L_1 \wedge s_k00 \in L_2). \tag{29}$$

Assume V_k was defined according to Case 2a: So for all $y \in \{s_k01, s_k10, s_k11\}$, $M_i^{Q_k \cup \{s_k00\}}(y)$ rejects. In particular, $M_i^{Q_k \cup \{s_k00\}}(u)$ rejects. Assume $u \in L_1$ and $s_k00 \in L_2$. So $M_i^{L_2}(u)$ rejects, since it behaves the same way as $M_i^{Q_k \cup \{s_k00\}}(u)$. By (iii) this contradicts $u \in L_1$. Therefore, by (29) we must have $s_k00 \in L_1$ and $u \in L_2$. In Case 2a, V_k is defined such that

$$s_k01 \in V_k \Leftrightarrow M_j^{O_k \cup \{s_k00\}}(s_k01) \text{ rejects.}$$

Note that $M_j^{O_k \cup \{s_k00\}}(s_k01)$ and $M_j^{L_1}(s_k01)$ behave the same way. Hence,

$$s_k01 \in V_k \Leftrightarrow M_j^{L_1}(s_k01) \text{ rejects.}$$

If $s_k01 \in V_k$, then $u = s_k01$ and hence $M_j^{L_1}(u)$ rejects. This contradicts (iv). Otherwise, if $s_k01 \notin V_k$, then $M_j^{L_1}(s_k01)$ accepts and hence $u = s_k01 \notin V_k$. This contradicts the assumption $u \in V_k$.

Assume V_k was defined according to Case 2b: Here we obtain contradictions analogously to Case 2a.

Assume V_k was defined according to Case 2c: Choose y and z such that $M_i^{Q_k \cup \{s_k00\}}(y)$ accepts and $M_j^{O_k \cup \{s_k00\}}(z)$ accepts. So $u \in \{s_k01, s_k10, s_k11\} - \{y, z\}$. Assume $s_k00 \in L_2$. Hence $M_i^{L_2}(y)$ and $M_i^{Q_k \cup \{s_k00\}}(y)$ behave the same way showing that $M_i^{L_2}(y)$ accepts. So $y \in L_1$ which contradicts the definition of V_k. Assume $s_k00 \in L_1$. Hence $M_j^{L_1}(z)$ and $M_j^{O_k \cup \{s_k00\}}(z)$ behave the same way showing that $M_j^{L_1}(z)$ accepts. So $z \in L_2$ which contradicts the definition of V_k.

This finishes Case 2. From the fact that all possible cases led to contradictions, we obtain that the initial assumption was false. Hence, L is not weakly T-mitotic. $\square$

5 Poly-Autoreducibility

In this section we consider the question of whether NP-complete sets are $f(n)$-autoreducible, for some growing function f. We first start with the following lemma.

Lemma 5.1 *Let L be an* NP*-complete language. For every polynomial $q(.)$ there is a polynomial-time algorithm $\mathcal{A}$ such that $\mathcal{A}$ on input x, $|x| = n$,*

- *either decides the membership of x in L*
- *or outputs strings $y_1, \cdots, y_m$ such that*
  - $x \in L \Rightarrow \{y_1, y_2, \cdots, y_m\} \subseteq L$,
  - $x \notin L \Rightarrow \{y_1, y_2, \cdots, y_m\} \cap L = \emptyset$,
  - $m = q(n)$, *and* $x \neq y_1, \neq y_2 \neq \cdots \neq y_m$.

Proof.

Let $R(.,.)$ be a polynomial-time search predicates associated with L. Given x, let w_x be the lexicographically maximum witness of x. Without loss of generality, assume that for every $x \in L$, every witness of x is of length $p(|x|)$, for some polynomial p. Consider the following set in NP.

$$L' = \{\langle x, y\rangle \mid x \in L, |y| = p(|x|), y \leq w_x\}.$$

Since L is NP complete, there is a polynomial-time reduction f from L' to L. We now describe the algorithm $\mathcal{A}$.

```
Input $x$, $|x| = n$. Let $m = p(n)$.
**if** $1^m$ is a witness of $x$, *Accept.*
$\ell = 0^m$
If $f(\langle x, \ell\rangle) = f(\langle x, 1^m\rangle)$, *Reject.*
$Q = \{f(\langle x, \ell\rangle)\}$.
**While** $|Q| \leq q(n) + 1$ **do**
    By doing a binary search find a string $a$ such that $\ell \leq a \leq 1^m$
    and, $f(\langle x, a\rangle) \in Q$ and $f(\langle x, a+1\rangle) \notin Q$. Set $\ell = a + 1$.
    **if** $a$ is a witness of $x$, then *Accept.*
    **if** $f(\langle x, \ell\rangle) = f(\langle x, 1^m\rangle)$, *Reject.*
    $Q = Q \cup \{f(\langle x, \ell\rangle)\}$.
Output the first $q(n)$ elements of $Q - \{x\}$.
```

Claim 5.2 *When the algorithm halts, for every string $y \in Q$, $x \in L \Leftrightarrow y \in L$.*

Proof. If $x \notin L$, then none of $\langle x, c\rangle, 0^m \leq c \leq 1^m$, belong to L'. Observe that the algorithm places a string y in Q only if $y = f(\langle x, a\rangle)$ where $0^m \leq a \leq 1^m$. Since f is a many-one reduction from L' to L no string from Q belongs to L. So from now we assume $x \in L$. We prove the claim by induction. Initially, $Q = \{f(\langle x, 0^m\rangle\}$. Clearly, $x \in L \Leftrightarrow \langle x, 0^m\rangle \in L'$. Since f is a many-one reduction L' to L, the claim holds initially. Assume that the claim holds before an iteration of while loop. The while loop finds a node a such that $f(\langle x, a\rangle) \in Q$, but $f(\langle x, a+1\rangle) \notin Q$. Since $f(\langle x, a\rangle) \in Q$, $x \in L$, by the induction hypothesis $f(\langle x, a\rangle) \in L$. Thus $\langle x, a\rangle \in L'$ which implies $a \leq w_x$. At this point the algorithm checks if a is a witness of x. If a is a witness of x, then it accepts and halts. If a is not a witness, then we have $a+1 \leq w_x$. Thus $\langle x, a+1\rangle \in L'$. Thus $f(\langle x, a+1\rangle) \in L$. Since the algorithm places $f(\langle x, a+1\rangle)$ in Q at this step, after the iteration of the while loop the claim holds. □

Claim 5.3 *If the algorithm Accepts or Rejects x, then the algorithm is correct.*

Proof. The algorithm accepts x only when it finds a witness of x. Thus if the algorithm accepts x, then $x \in L$. The algorithm rejects when $f(\langle x, \ell\rangle) = f(\langle x, 1^m\rangle)$. Note that $f(\langle x, \ell\rangle) \in Q$, thus by previous claim $x \in L$ if and only if $f(\langle x, \ell\rangle) \in L$. Observe that since 1^m is not a witness of x, and 1^m is the largest string at length m, $\langle x, 1^m\rangle \notin L'$. Thus $f(\langle x, 1^m\rangle) \notin L$. Since $f(\langle x, \ell\rangle) = f(\langle x, 1^m\rangle)$, the claim follows. □

Observe that algorithm places a string $f(\langle x, \ell\rangle)$ in Q only if $f(\langle x, \ell\rangle) \neq f(\langle x, 1^m\rangle)$. Thus $f(\langle x, 1^m\rangle)$ is not placed in Q during any iteration. So the binary search step always finds a with desired properties. Every iteration of the while loop adds a new string to Q or decides the membership of x in L. Thus the algorithm halts in polynomial time and when it halts all elements in Q are distinct. This finishes the proof of the lemma. □

The above lemma comes close to showing that NP-complete sets are poly-autoreducible, except for a small caveat. Let L be any NP-complete language. If the algorithm from Lemma 5.1 neither accepts x or rejects x, then it produces polynomially many equivalent strings. However, to show L is poly-autoreducible, we must produce polynomially-many equivalent strings even when the algorithm accepts or rejects.

This boils down to the following problem: Let L be an NP-complete language. Given 0^n as input, in polynomial time output polynomially many distinct strings such that all of them are in L. Similarly, output polynomially many distinct strings such that none of them are in L.

Below, we show that if one-way permutations exist, then we can achieve this task. We start with a result by Agrawal [1].

Definition 5.4 *Let f be a many-one reduction from A to B. We say f is $g(n)$-sparse, if for every n, no more than $g(n)$ strings of length n are mapped to a single string via f.*

Lemma 5.5 *([1]) If one-way permutations exist, then* NP*-complete sets are complete with respect reductions that are $2^n/2^{n^\gamma}$ sparse. Here γ is a fixed constant less than 1.*

Lemma 5.6 *Let L be* NP*-complete. If one-way permutations exist, then there exists a polynomial-time algorithm that on input 0^n outputs $\log n$ distinct strings in L and $\log n$ strings out of L.*

Proof. By Lemma 5.5, there is a $2^n/2^{n^\gamma}$ sparse reduction f from Σ^* to L. Thus

$$|f(\Sigma^{\log n})| \geq 2^{(\log n)^\gamma} \geq 2^{\log\log n} = \log n.$$

Thus by applying f to every string in $\Sigma^{\log n}$, we obtain at least $\log n$ distinct strings. Since f is a reduction from Σ^* to L, these strings are in L. It is obvious that this can be done in polynomial-time. To generate strings out of L, consider a $2^n/2^{n^\gamma}$-sparse reduction from $\emptyset$ to L. □

If we consider probabilistic algorithms, then we obtain a stronger consequence.

Lemma 5.7 *Let L be* NP*-complete. Assume one-way permutations exist. For every polynomial q, there exists a polynomial-time probabilistic algorithm $\mathcal{B}$ that on input 0^n outputs $q(n)$ distinct strings from L and $q(n)$ distinct strings from $\overline{L}$.*

Proof. Let L be any NP-complete language. Consider a $2^n/2^{n^\gamma}$-sparse reduction from Σ^* to L. Consider the following probabilistic procedure.

```
1. Input $0^n$, $S = \emptyset$
2. **Repeat** Steps 3 and 4 $q(n)$ times
3.  Randomly pick a string $y \in \Sigma^n$.
4.  If $f(y) \notin f(S)$, $S = S \cup \{y\}$.
5. Output $S$.
```

We claim that the above procedure outputs a set S of size $q(n)$, with high probability. Consider Steps 3 and 4 during an iteration, let $|S| = k$ at this time. Since at most $2^n/2^{n^\gamma}$ strings from Σ^n are mapped to same string, there exist at most $k2^n/2^{n^\gamma}$ strings y from Σ^n such that $f(y) \in f(S)$. Thus the probability that we do not add a new string S during Steps 3 and 4 is at most $k/2^{n^\gamma}$. Thus the probability that we fail to add a string during one of the $q(n)$ iterations is at most $q(n)k/2^{n^\gamma} < 1/4$. Thus the above algorithm outputs a set of size $q(n)$ with high probability. By the construction of S no two strings from S are mapped to same string via f, thus $|f(S)| = q(n)$. Since f is a reduction from Σ^* to L, $f(S) \subseteq L$. To generate strings from $\overline{L}$, we consider a reduction from $\emptyset$ to L. □

If we assume quick pseudo-random generators exist, then we can derandomize the above procedure.

Lemma 5.8 *Let L be any* NP*-complete language. If one-way permutations and quick pseudo-random generators exist, then for every polynomial $q(n)$, there is a polynomial-time algorithm that on input 0^n outputs $q(n)$ many distinct strings from L and $q(n)$ many distinct strings out of L.*

Combining Lemmas 5.1 and 5.6, we obtain the following result.

Theorem 5.9 *If one-way permutations exist, then every* NP*-complete language is* $\log n$*-autoreducible.*

Combining Lemmas 5.1 and 5.8, we obtain the following result.

Theorem 5.10 *If one-way permutations and quick pseudo-random generators exist,* NP*-complete sets are poly-autoreducible.*

Finally, we consider another hypothesis from which poly-autoreducibility of NP-complete sets follows.

Theorem 5.11 *If there exists a* UP *machine* M *that accepts* 0^* *such that no* P*-machine can compute infinitely many accepting computations of* $M(0^n)$*, then* NP*-complete sets are poly-autoreducible.*

Proof.

Let L be any NP-complete language. We show that if the hypothesis is true, then there is a polynomial-time algorithm that on input 0^n outputs polynomially many strings from L and polynomially string from $\overline{L}$. This combined with Lemma 5.1 shows that NP-complete sets are poly-autoreducible.

We first show how to produce polynomially many strings from L. The hypothesis implies that there is a polynomial-time decidable predicate $R(.,.)$ and a polynomial p such that for every n, there exists a unique string w_n and $R(0^n, w_n)$ holds. Moreover, no polynomial-time algorithm, on input 0^n, can output w_n for infinitely many n. Consider the following language

$$L' = \{\langle 0^n, y\rangle \mid |y| = p(n), y \leq w_n\}.$$

Since L' is in UP, there is a many-one reduction f from L' to L. Now consider the behavior algorithm $\mathcal{A}$, described in Lemma 5.1, on input 0^n. Since 0^n belongs to 0^*, the algorithm never rejects. The algorithm accepts only when it finds a witness w_n for 0^n. However, our hypothesis says that no polynomial-time algorithm can output infinitely witnesses w_n. Thus this algorithm must output a set Q of cardinality $q(n)$.

Using similar arguments as in Lemma 5.1, we can show that for every $y \in Q$, $0^n \in 0^* \Leftrightarrow y \in L$. Thus the algorithm outputs $q(n)$ strings from L.

To output strings from $\overline{L}$, consider the many-one reduction g from L' to $\overline{L}$ and proceed as before. Since L' is in $UP \cap CoUP$ such a reduction exists.

Thus there exists a polynomial-time algorithm that outputs polynomially many strings from L and polynomially many strings from $\overline{L}$. □

Next we consider the possibility of an unconditional proof that NP-complete sets are poly-autoreducible. We relate this with the notion of immunity. We show that if NP-complete sets are poly-autoreducible, then they are not E-immune. It is known that NP-complete sets are not generic [11]. This proof is based on the fact that NP-complete sets are autoreducible. Genericity is stronger notion than immunity, i.e., if a language L is not immune, then it can not be generic. Our result says that improving the autoreducibility result for NP-complete sets gives a stronger consequence—namely they are not immune.

Theorem 5.12 *If every* NP*-complete set is poly-autoreducible, then no* NP*-complete set is* E*-immune.*

Proof. Suppose every NP-complete set is poly-autoreducible. Let L be an NP-complete set. We need to show that L is not E-immune. Let $T = \{0^{t(i)} | i \geq 0\}$, where $t(i)$ is defined as $t(0) = 2$ and $t(i) = 2^{2t(i-1)}$ for $i \geq 1$. Clearly $T \in \mathrm{P} \cap \mathrm{SPARSE}$. If $L \subseteq T$, then L is an NP-complete sparse set, from which it follows NP = P [14]. Therefore, $L \in \mathrm{P}$ and so L is trivially not E-immune. Hence, we assume that $L - T \neq \emptyset$. Then $L - T$ is NP-complete by Theorem 3.1 of a paper by Glaßer et al. [12]. Hence, $L \cup T$ is NP-complete since $L - T \leq_m^p L \cup T$. By assumption, $L \cup T$ is poly-autoreducible. So there exists a polynomial-time computable function f such that for every input x,

1. $f(x)$ is a set of $|x|$ words different from x, and
2. for all $y \in f(x)$ it holds that $x \in L \cup T \iff y \in L \cup T$.

Suppose f is computable in time n^l for some $l \geq 1$. For each $i \geq 1$, define

$$S_i = \{x \mid x \in f(0^{t(i)}) \cup f(0^{t(i-1)}) - \{0^{t(i-1)}\}, \text{ where } t(i-1) \leq |x| < t(i)\}$$

and let $S = \cup_{i \geq 1} S_i$.

Then the theorem follows from the following claims:

Claim 5.13 $S \in \mathrm{DTIME}(2^{2(l+1)n})$

Proof. For any input x, to decide whether $x \in S_i$ for some $i \geq 1$, we just need to first determine i such that $t(i-1) \leq |x| < t(i)$ and then check whether $x \in f(0^{t(i)}) \cup f(0^{t(i-1)}) - \{0^{t(i-1)}\}$. The major cost is for computing $f(0^{t(i)})$, which takes $2^{2(l+1)n}$ time, since $|0^{t(i)}| = t(i) = 2^{2t(i-1)} \leq 2^{2|x|}$. □

Claim 5.14 $S \subseteq L$

Proof. First observe, for each $i \geq 1$, that $f(0^{t(i)}) \subseteq L \cup T$ because f is an autoreduction of $L \cup T$ and $0^{t(i)} \in T \subseteq L \cup T$. So $S \subseteq L \cup T$. Now for each $i \geq 1$, we have $S_i \cap T = \emptyset$, because for every $x \in S_i$, $t(i-1) \leq |x| < t(i)$ and $x \notin \{0^{t(i-1)}\}$. So $S \cap T = \emptyset$ and hence, $S \subseteq L$. □

Claim 5.15 $\|S\| = \infty$.

Proof. We first observe that all S_i's ($i \geq 1$) are pairwise disjoint. So it suffices to show for each sufficiently large i that $S_i \cup S_{i+1} \neq \emptyset$. Now suppose for some i, where $t(i)^l < 2^{2t(i)}$, that $S_i = \emptyset$. Then for every $x \in f(0^{t(i)})$, $|x| \geq t(i)$ or $|x| < t(i-1)$ or $x = 0^{t(i-1)}$. Note that there are fewer than $2^{t(i-1)}$ strings of length less than $t(i-1)$ and $\|f(0^{t(i)})\| = t(i) = 2^{2t(i-1)} > 2^{t(i-1)} + 1$. So there must exist $x \in f(0^{t(i)})$ with $|x| \geq t(i)$. For this x, it holds that $x \neq 0^{t(i)}$ since f is an autoreduction. Also, $|x| \leq t(i)^l < 2^{2t(i)} = t(i+1)$. So $x \in S_{i+1}$. Therefore, $S_{i+1} \neq \emptyset$. □

This finishes the proof of Theorem 5.12. □

References

[1] M. Agrawal. Pseudo-random generators and structure of complete degrees. In *17th Annual IEEE Conference on Computational Complexity*, pages 139–145, 2002.

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

[3] J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Mathematical Systems Theory*, 18(1):1–10, June 1985.

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

[5] L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, 1977.

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

[7] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness, and structure of complete sets. *SIAM Journal on Computing*, 27:637–653, 1998.

[8] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proceedings 9th Structure in Complexity Theory*, pages 118–133, 1994.

[9] H. Buhrman and L. Torenvliet. Separating complexity classes using structural properties. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 130–138, 2004.

[10] H. Buhrman and L. Torenvliet. A post's program for complexity theory. *BEATCS Complexity Column*, pages 41–51, 2005.

[11] C. Glaßer, M. Ogihara, A. Pavan, A. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. Technical Report TR05-11, ECCC, 2005.

[12] C. Glaßer, A. Pavan, A. Selman, and S. Sengupta. Properties of NP-complete sets. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 184–197, 2004.

[13] R. Ladner. Mitotic recursively enumerable sets. *Journal of Symbolic Logic*, 38(2):199–211, 1973.

[14] S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and Systems Sciences*, 25(2):130–143, 1982.

[15] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal of Computing*, 20(3):471–483, 1991.

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

[17] A. Yao. Coherent functions and program checkers. In *Proceedings of the 22n Annual Symposium on Theory of Computing*, pages 89–94, 1990.