SPLITTING NP-COMPLETE SETS[*]

CHRISTIAN GLAßER[†], A. PAVAN[‡], ALAN L. SELMAN[§], AND LIYU ZHANG[¶]

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., 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. In particular, every NP-complete set A splits into two NP-complete sets A_1 and A_2.

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.

Key words. computational and structural complexity, NP-complete sets, autoreducibility, mitoticity

AMS subject classifications. 68Q17,68Q15

1. Introduction.

We show in this paper that NP-complete sets split into two equivalent parts. Let L be an NP-complete set containing an infinite number of strings. Then there is a set $S \in$ P such that the sets $L_1 = S \cap L$ and $L_2 = \overline{S} \cap L$ are both NP-complete, and $L = L_1 \cup L_2$. Since L_1 and L_2 are both many-one-equivalent to L, we may say that they contain the same information as L. For this reason, sets L with this property are called *mitotic*.[1] Briefly, we prove that all NP-complete sets are mitotic.

Our story begins with the notion of autoreducibility. Trakhtenbrot [19] 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 [1] 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 an 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. Autoreducible sets contain redundant information just as do mitotic sets. For example, if a set A is m-autoreducible, then x and $f(x)$ contain the same information about membership in A.

A recent paper of Glaßer et al. [11] showed that the complete sets for many interesting classes such as NP, PSPACE, NEXP, and levels of PH are m-autoreducible.

[*] A preliminary version of this paper appeared at the conference STACS 2006.

[†] 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 and by the Alexander von Humboldt-Stiftung. Email: selman@cse.buffalo.edu

[¶] Department of Computer Science and Information Systems, University of Texas at Brownsville. Email: Liyu.Zhang@utb.edu

[1] Mitosis in biology is the process by which a cell separates its duplicated genome into two identical halves.

The main technical result of the present paper is that m-autoreducible implies m-mitotic. As a consequence, complete sets for interesting complexity classes such as NP, PSPACE, NEXP, and levels of PH are m-mitotic. This result resolves several long-standing open questions raised by Ambos-Spies [1], Buhrman, Hoene, and Torenvliet [5], and Buhrman and Torenvliet [6].

The notion of *mitoticity* was originally introduced by Lachlan [14] for recursively enumerable sets. Mitoticity was studied comprehensively by Ladner [16, 15]. Ambos-Spies [1] formulated two related notions in the polynomial time setting, mitoticity and weak mitoticity. 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. If the latter holds without the assumption $S \in \mathrm{P}$, then A is *weakly m-mitotic.*

Ambos-Spies [1] showed that if a set is m-mitotic, then it is m-autoreducible and he raised the question of whether the converse holds. As stated above, we resolve this question and show that every m-autoreducible set is m-mitotic. Since its proof is technically involved, we illustrate parts of the main combinatorial idea with the help of a simplified graph problem that will be described in Section 3. We remark that this example is a strong simplification of the general case that we have to solve. Our main result is all the more surprising, because it is known [1] 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 [6].

Our main result relates local redundancy of information to global redundancy of information 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.

Our result can also be viewed as a step towards understanding the isomorphism conjecture [3]. 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[2]. 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 beautiful results: Mahaney [17] showed that many-one-hard sets for NP are not sparse unless P = NP. Karp and Lipton [13] proved that if sparse Turing-hard sets for NP exist, then PH collapses to the second level. Ogiwara and Watanabe [18] showed that bounded-truth-table-hard sets for

[2]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.

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. [4] and Buhrman and Torenvliet [7, 8] 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 [8] for more details.

1.1. Previous Work. The question of whether complete sets for various classes are autoreducible has been studied extensively [20, 2, 4]. Beigel and Feigenbaum [2] 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. [4] 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. [4] 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 [5] showed that EXP complete sets are weakly many-one mitotic. This result was recently improved independently by Kurtz [8] and Glaßer et al. [12, 11]. Buhrman and Torenvliet [8] observed that Kurtz' proof can be extended to show that 2-tt complete sets for EXP are 2-tt mitotic. This cannot be extended to 3-tt reductions: There exist 3-tt complete sets for EXP that are not btt-autoreducible and hence not btt-mitotic [4]. Glaßer et al. also showed that NEXP complete sets are weakly m-mitotic and PSPACE-complete sets are weakly 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. PF denotes the class of polynomial-time computable functions.

DEFINITION 2.1 ([1]). *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 ([1]). *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 ([1]). *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 $A \equiv_m^p A_0 \equiv_m^p A_1$.*

3. m-Autoreducibility equals m-Mitoticity. It is easy to see that if a non-trivial 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 discuss parts of the main ideas and the intuition behind the proof. To give an example of the difficulties in the proof, we strongly simplify the general case so that we end at a much easier problem for finite graphs. Later, we have to solve this problem for *infinite* graphs.

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 $-$, i.e., $f(x), f(f(f(x))), \ldots$ obtain $+$ and $x, f(f(x)), \ldots$ obtain $-$. 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 $f(f(x))$. The 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. Speaking in terms of graphs, we will solve the following problem.

Infinite Graph Labeling Problem: Let G be an infinite, loop-free, directed graph whose set of nodes is $\mathbb{N}$ such that all nodes have outdegree 1. Moreover, let

s be a polynomial-time computable function that on input of a node u computes its successor, i.e., $s(u)$ is the uniquely determined node such that $(u, s(u))$ is an edge. Find a strategy that labels each node of G with either $+$ or $-$ such that:

(i) The label of a given node can be computed in polynomial time.

(ii) There is a polynomial p such that for a given node u we can compute in polynomial time a node $v \in \{s(u), s(s(u)), \ldots, s^{p(n)}(u)\}$ that has a different label than u. In particular, at least one of the nodes $s(u), s(s(u)), \ldots, s^{p(n)}(u)$ has a different label than u.

If we can solve this problem, then this shows that m-autoreducibility implies m-mitoticity. This is seen as follows: Let L be m-autoreducible via autoreduction s and let G be the graph whose set of nodes is $\mathbb{N}$ and whose set of edges is $\{(u, s(u)) \mid u \in \mathbb{N}\}$. Observe that G and s have the properties mentioned in the infinite graph labeling problem. By solving this problem we obtain a labeling strategy $S = \{u \in \mathbb{N} \mid u \text{ has label } +\}$ that satisfies (i) and (ii). By (i), $S \in \mathrm{P}$. By (ii), there exists a polynomial-time computable function g that for a given node u computes the node v described in (ii). In particular,

$$u \in S \Leftrightarrow g(u) \notin S. \tag{3.1}$$

Moreover, from $g(u) \in \{s(u), s(s(u)), \ldots, s^{p(n)}(u)\}$ and from the fact that s is an autoreduction for L it follows that

$$u \in L \Leftrightarrow g(u) \in L. \tag{3.2}$$

The equivalences (3.1) and (3.2) yield the m-mitoticity of L (formally proved in Proposition 3.2).

To keep this proof sketch simpler, we now restrict to finite graphs, 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 labeling problem for finite graphs.

Simplified 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 $s(u), s(s(u)), \ldots, s^{p(n)}(u)$ has a different label than u.

Cole and Vishkin [9] solve the r-ruling set problem which corresponds to the following restriction of the above simplified problem for finite graphs: First, the graph must be connected and it must have indegree 1. This means that the graph has to be a simple ring. Second, the predecessor of a node must be computable efficiently (i.e., in polynomial time in n). In contrast, the general problem we have to solve here comprises graphs that are infinite, that are not necessarily connected, and that have an unbounded indegree.

We now exhibit a labeling strategy for the simplified problem. To define this labeling, we use the following *distance function*: $d(x, y) \stackrel{df}{=} \lfloor \log |y - x| \rfloor$ (our formal proof uses a variant of this function).

```
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   if $\lfloor x/2^{r+1} \rfloor$ is even then output + else output −
```

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 $u_1, u_2, \dots, u_m$ be a path in the graph. It suffices to show that not all the nodes $u_1, u_2, \dots, u_m$ 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 the distance increases more than n times, then $d(u_{m-1}, u_m) > n$. Therefore, $u_m - u_{m-1} > 2^{n+1}$, which is impossible for words of length n. Hence we have seen that the distances do not decrease and that they increase at most n times. So along the path $u_1, u_2, \dots, u_m$ 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 $u_1, u_2, \dots, u_m$ obtain the same label. This proves (ii) and solves the simplified problem.

A generalization of this strategy allows us to solve the infinite graph labeling problem which in turn establishes m-mitoticity for the m-autoreducible set 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. A simple proof shows that the following function g satisfies the statements 1 and 2 from the proposition.

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

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. ☐

Proof. (Theorem 3.1) If L is m-mitotic, then there exist $S\in\text{P}$ and $f_1,f_2\in\text{PF}$ 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 $f\in\text{PF}$ 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 $g\in\text{PF}$ and an $S\in\text{P}$ 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 $t^{-1}\in\text{PF}$. We partition the set of all words according to the parity of the inverse tower function of their lengths.

$$S_0 \stackrel{df}{=} \{x \mid t^{-1}(|x|)\equiv 0 (\text{mod } 2)\}$$
$$S_1 \stackrel{df}{=} \{x \mid t^{-1}(|x|)\equiv 1 (\text{mod } 2)\}$$

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

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

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

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 (if $x \in S_0$ then accept else reject)
3   if $|z| > |y|$ then (if $y \in S_1$ then accept else reject)
4   if $x = z$ then (if $x > f(x)$ then accept else reject)
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   if $\lfloor y/2^{abs(r)+1} \rfloor$ is even then accept else reject
```

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.

This fact is proved as follows. 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 line 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 line 6. So $f^j(y) \notin S$ which contradicts the assumption. This proves Fact 1.

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

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

$$\forall j \in J, \quad d(f^j(y), f^{j+1}(y)) < d(f^{j+1}(y), f^{j+2}(y)) \tag{3.3}$$
$$\forall j \in K, \quad d(f^j(y), f^{j+1}(y)) = d(f^{j+1}(y), f^{j+2}(y)) \tag{3.4}$$

Case 1: $\|J\| > 2m$. Together with (3.3) and (3.4) this shows

$$d(f^{6m+3}(y), f^{6m+4}(y)) \; - \; d(f^0(y), f^1(y)) > 2m. \tag{3.5}$$

It follows that

$$d(f^{6m+3}(y), f^{6m+4}(y)) > m \tag{3.6}$$

or

$$d(f^0(y), f^1(y)) < -m. \tag{3.7}$$

Assume that (3.6) 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 (3.6) holds.

Assume now that (3.7) holds. Again, $f^0(y)$ and $f^1(y)$ are words of length $\le m$. So $1 \le \mathrm{abs}(f^0(y) - f^1(y)) \le 2^{m+1} - 2$. It follows that $0 \le \log(\mathrm{abs}(f^1(y) - f^0(y))) < m+1$ and therefore, $d(f^0(y), f^1(y)) \ge -m$. This is a contradiction, since we assumed that (3.7) holds.

Case 2: $\|J\| \le 2m$. 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.

$$d(f^j(y), f^{j+1}(y)) = d(f^{j+1}(y), f^{j+2}(y)) = d(f^{j+2}(y), f^{j+3}(y)) = d(f^{j+3}(y), f^{j+4}(y)) \tag{3.8}$$

Define r as the number shown in (3.8), and let $z_1 \overset{df}{=} f^j(y)$, $z_2 \overset{df}{=} f^{j+1}(y)$, $z_3 \overset{df}{=} f^{j+2}(y)$, and $z_4 \overset{df}{=} f^{j+3}(y)$. 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.

$$a_1 \overset{df}{=} \lfloor z_2 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even} \tag{3.9}$$

$$a_2 \overset{df}{=} \lfloor z_3 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even} \tag{3.10}$$

$$a_3 \overset{df}{=} \lfloor z_4 / 2^{\mathrm{abs}(r)+1} \rfloor \text{ is even} \tag{3.11}$$

Case 2a: $r = 0$. Here $z_2 \ne z_4$, since otherwise on input z_2 the algorithm stops in line 4 which contradicts Fact 1. Also, $z_2 \ne z_3$ and $z_3 \ne z_4$, since f is an m-autoreduction. From (3.8) 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. This contradicts the observations (3.9) and (3.11).

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

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

So assume $a_1 < a_3$ which implies $a_3 - a_1 \ge 2$, since both values are even. Since a_2 is even as well, we obtain $a_2 - a_1 \ge 2$ or $a_3 - a_2 \ge 2$. If $a_2 - a_1 \ge 2$, then $z_3 - z_2 > 2^{r+1}$ and so $d(z_2, z_3) > r$. If $a_3 - a_2 \ge 2$, then $z_4 - z_3 > 2^{r+1}$ and so $d(z_3, z_4) > r$. Both conclusions contradict (3.8).

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

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

So assume $a_1 > a_3$ which implies $a_1 - a_3 \ge 2$, since both values are even. Since a_2 is even as well, we obtain $a_1 - a_2 \ge 2$ or $a_2 - a_3 \ge 2$. If $a_1 - a_2 \ge 2$, then $z_2 - z_3 > 2^{\mathrm{abs}(r)+1}$ and so $d(z_2, z_3) < -\mathrm{abs}(r) = r$. If $a_2 - a_3 \ge 2$, then $z_3 - z_4 > 2^{\mathrm{abs}(r)+1}$ and so $d(z_3, z_4) < -\mathrm{abs}(r) = r$. Both conclusions contradict (3.8). (End of proof of Claim 3.3) □

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

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

Proof. For every x, let

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

where i is the smallest number such that $|f^{i+1}(x)| \le |f^i(x)|$ 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.

$$|x| < |f(x)| \tag{3.12}$$
$$|z'| < |f(z')| \tag{3.13}$$
$$|z| < |f(z)| \tag{3.14}$$
$$|x| < |z'| \tag{3.15}$$

Moreover,

$$|z'| < |x|^k + k, \tag{3.16}$$

since otherwise already z' violates the invariant, which contradicts the fact that with z we chose the earliest violation of the invariant. From (3.15) and (3.16) we obtain

$$t^{-1}(|x|) \leq t^{-1}(|z'|) \leq t^{-1}(|x|^k + k) = t^{-1}(|x|) + 1. \tag{3.17}$$

From (3.12) it follows that on input x, the algorithm for S stops in line 2. We see the same for z' and z using (3.13) and (3.14). This implies the following.

$$x \in S \Leftrightarrow x \in S_0 \tag{3.18}$$
$$z' \in S \Leftrightarrow z' \in S_0 \tag{3.19}$$
$$z \in S \Leftrightarrow z \in S_0 \tag{3.20}$$

Note that

$$x \in S \Leftrightarrow z' \in S \Leftrightarrow z \in S, \tag{3.21}$$

since otherwise the algorithm for r stops earlier. Together with (3.18), (3.19), and (3.20) this shows

$$x \in S_0 \Leftrightarrow z' \in S_0 \Leftrightarrow z \in S_0 \tag{3.22}$$

and therefore,

$$t^{-1}(|x|) \equiv t^{-1}(|z'|) \equiv t^{-1}(|z|) \pmod 2. \tag{3.23}$$

Now (3.17) implies $t^{-1}(|x|) = t^{-1}(|z'|)$ and we obtain

$$t^{-1}(|z'|) = t^{-1}(|x|) < t^{-1}(|x|^k + k) \leq t^{-1}(|z|). \tag{3.24}$$

From (3.23) and (3.24) 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. (End of proof of Claim 3.4) □

We continue the proof of Theorem 3.1. 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. (End of proof of Claim 3.5) □

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 $g \in$ 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 $x \in L \Leftrightarrow g(x) = f^j(y) \in L$. (End of proof of Claim 3.6) □

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,

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

(Here i refers to the value of the variable i in the algorithm for g at the time when the algorithm stops in line 8.) Therefore, if we look at the algorithm for S (page 7), 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

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

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 $x \in S \Leftrightarrow g(x) \notin S$. (End of proof of Claim 3.7) □

The Claims 3.6 and 3.7 allow the application of Proposition 3.2. Hence L is m-mitotic. (End of proof of Theorem 3.1) □

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. □

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 [1] defined a set A to be *ω-m-mitotic* if for every $n \geq 2$ there exists a partition $(Q_1, \ldots, Q_n)$ of Σ^* such that each Q_i is polynomial-time decidable and 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.*

Proof. Fix a class mentioned in Corollary 3.8 and let A be a nontrivial, infinite, many-one complete set. We need to prove that for every $n \geq 2$, there exists a partition

$(Q_1, \ldots, Q_n)$ of Σ^* such that each Q_i belongs to P and the sets $A, A \cap Q_1, \ldots, A \cap Q_n$ are all polynomial-time many-one equivalent. We prove this by induction on n. The base $n = 2$ is an immediate consequence of Corollary 3.8. Now assume $n \geq 3$. By induction hypothesis, there exists a partition $(Q_1, \ldots, Q_{n-1})$ of Σ^* such that each Q_i belongs to P and the sets $A, A \cap Q_1, \ldots, A \cap Q_{n-1}$ are all polynomial-time many-one equivalent. Since A is infinite and $(Q_1, \ldots, Q_{n-1})$ is a partition of Σ^*, there exists some j such that $A \cap Q_j$ is infinite as well. Moreover, $A \cap Q_j$ is nontrivial and polynomial-time many-one complete. By Corollary 3.8, $A \cap Q_j$ is m-mitotic. So there exists $S \in \mathrm{P}$ such that $A \cap Q_j$, $A \cap Q_j \cap S$, and $A \cap Q_j \cap \overline{S}$ are polynomial-time many-one equivalent. Define $Q'_j = Q_j \cap S$ and $Q''_j = Q_j \cap \overline{S}$. The following polynomial-time decidable partition of Σ^* completes the proof: $(Q_1, \ldots, Q_{j-1}, Q'_j, Q''_j, Q_{j+1}, \ldots, Q_{n-1})$. $\square$

Next, we note that the proof the main theorem also yields the following result.

THEOREM 3.11. *Every 1-tt-autoreducible set is 1-tt-mitotic.*

The proof of Theorem 3.1 provides a strategy that solves the infinite graph labeling problem defined at the beginning of this section. In particular, for every graph that satisfies certain prerequisites there exists a polynomial-time algorithm that labels given nodes with either $+$ or $-$ such that each node has a polynomial-bounded path that leads to a node with a different label. This is made precise as follows.

COROLLARY 3.12. *Let $G = (\mathbb{N}, E)$ be an infinite, loop-free, directed graph with outdegree 1 such that there exists an $f \in \mathrm{PF}$ that computes the successor of a given node u, i.e., $(u, f(u)) \in E$. Then there exist a polynomial p, $S \in \mathrm{P}$, and $g \in \mathrm{PF}$ such that for all x,*

$$(\exists i \in [1, p(|x|)], g(x) = f^i(x)) \quad \textit{and} \quad (x \in S \Leftrightarrow g(x) \notin S).$$

Proof. Note that f is an m-autoreduction for $L \overset{df}{=} \Sigma^*$. Choose $k \geq 1$ such that f is computable in time $n^k + k$. Now consider the implication from left to right in the proof of Theorem 3.1. There the assumption $|\overline{L}| \geq 2$ is only needed at the end of the proof when Proposition 3.2 is applied. So for $L = \Sigma^*$ the proof goes through until Proposition 3.2 is applied. In particular, we obtain a total $g \in \mathrm{PF}$ and an $S \in \mathrm{P}$ such that for all x,

$$x \in S \Leftrightarrow g(x) \notin S. \tag{3.25}$$

Consider the function r which is defined in the proof of Claim 3.4. From the claim it follows that r is a total, polynomial-time computable function such that $r(x) = f^i(x)$ for some $i \geq 0$. Moreover, in the proof of the claim we show that the algorithm for r on input x terminates within $|x|^k + k$ iterations of the loop. From the algorithm it follows that for all x,

$$\exists i \in [0, |x|^k + k], r(x) = f^i(x). \tag{3.26}$$

Let q be a polynomial bounding the computation time for r. Now consider the algorithm for g in the proof of Theorem 3.1. By Claim 3.5, the statements in the comments of this algorithm hold true. In particular, at any time it holds that $z = f^i(y)$ for some $i \in [0, 6m + 4]$ where $y = r(x)$ and $m = |y| \leq q(|x|)$. If the algorithm for g stops, then it returns either z or $f(z)$. So $g(x) = f^i(y)$ for some $i \in [0, 6m + 5] \subseteq [0, 6q(|x|) + 5]$. Together with (3.26) this shows

$$\exists i \in [0, p(|x|)], g(x) = f^i(x),$$

where $p(n) \overset{df}{=} 6q(n) + 5 + n^k + k$. By (3.25), $g(x) \neq x$ which implies

$$\exists i \in [1, p(|x|)], g(x) = f^i(x). \tag{3.27}$$

The corollary follows from (3.25) and (3.27). □

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.

$$\begin{aligned}
k' &\geq k - \sqrt{k} - l - 1 \\
&\geq (l+2)^{2^m} - \sqrt{(l+2)^{2^m}} - l - 1 \quad && \text{(since } (a \geq b \Rightarrow a - \sqrt{a} \geq b - \sqrt{b}) \text{ for } a, b \geq 1) \\
&\geq (l'+2)^{2^m} && \text{(follows from (4.2) in the estimation below)}
\end{aligned} \tag{4.1}$$

For (4.1) the following estimation is needed.

$$\begin{aligned}
l+1 &\geq l+1 \\
(l+2)^{2^{m-1}-1} \cdot (l+2-1) &\geq (l+1)^{2^{m-1}-1} \cdot (l+1) \\
(l+2)^{2^{m-1}} - 1 &\geq (l+1)^{2^{m-1}} && \text{(since } (l+2)^{2^{m-1}-1} \geq 1) \\
(l+2)^{2^{m-1}} \cdot \left[(l+2)^{2^{m-1}} - 1\right] &\geq (l+2)^{2^{m-1}} \cdot \left[(l+1)^{2^{m-1}}\right] \\
(l+2)^{2^m} - (l+2)^{2^{m-1}} &\geq (l+1+1) \cdot (l+1)^{2^{m-1}-1} \cdot \left[(l+1)^{2^{m-1}}\right] \\
(l+2)^{2^m} - \sqrt{(l+2)^{2^m}} &\geq (l+1)^{2^m} + (l+1)^{2^m-1} \\
(l+2)^{2^m} - \sqrt{(l+2)^{2^m}} - l - 1 &\geq (l'+2)^{2^m} && \text{(since } (l+1)^{2^m-1} \geq l+1)
\end{aligned} \tag{4.2}$$

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,

$$j_i \neq 1 \wedge n_1 \notin Q_{j_i} \wedge n_{j_i} \notin Q_1. \tag{4.3}$$

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

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

and hence

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

From (4.3) it follows that the lemma is satisfied by the indices $1, j_{i_1}, j_{i_2}, \ldots, j_{i_{m'}}$. ☐

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 $M_1, M_2, \ldots$ 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

$O_1, O_2, \ldots, O_l$ 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,

$$l \leq 2^{2^{t(n-1)+1}} \leq 2^{2^{2^{t(n-1)}}} = \log\log t(n) = \log\log m. \tag{4.4}$$

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 $s_1, \ldots, s_l$ 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 $s_1, \ldots, s_l \in \Sigma^{m-2}$ 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

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

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

$$\begin{aligned}\|Q_s\| &\leq 4l[(m^i + i) + (m^j + j)] \\ &\leq 4(\log\log m)[m^{\log\log m} + \log\log m] \\ &\leq 8(\log\log m)m^{\log\log m} \\ &\leq m^{2\log\log m} \\ &\leq 2^{\log^2 m} - 2. \end{aligned} \tag{4.5}$$

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 (4.5), $Q_1, Q_2, \ldots, Q_{2^{m-2}}$ are sets of cardinality $\leq 2^{\log^2 m} - 2$. Clearly, $1, 2, \ldots, 2^{m-2}$ are pairwise different numbers. By (4.4),

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

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

$$r \neq k \ \Rightarrow\ s_r \notin Q_{s_k}. \tag{4.6}$$

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 (4.6) and finishes the proof of Claim 4.3. $\square$

Let $s_1, \dots, s_l \in \Sigma^{m-2}$ 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 $Q_k \stackrel{df}{=} O - O_k$.

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

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 both, $M_i^{Q_k \cup \{s_k00\}}(y)$ and $M_j^{O_k \cup \{s_k00\}}(z)$ accept. Choose $v \in W(s_k) - \{s_k00, y, z\}$ and define $V_k \stackrel{df}{=} \{s_k00, v\}$.

This finishes the construction of V_k. We define $W_n \stackrel{df}{=} \bigcup_{k \in [1,l]} V_k$. 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 $L \in$ EXP: Since $l \leq \log\log m$, there are not more than $2^{m \log\log m}$ possibilities to choose the strings $s_1, \dots, s_l$. 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 $L \in$ EXP. From the construction it follows that $L \cap \Sigma^m \leq 2l \leq 2 \log\log m$. In particular, $L \in$ 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 $O_1, O_2, \dots, O_l$ be the list of all subsets of O (again lexicographically ordered according to their characteristic sequences). Let $s_1, \dots, s_l$ and $V_1, \dots, V_l$ 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,

$$\text{either } (s_k00 \in L_1 \wedge u \in L_2) \text{ or } (u \in L_1 \wedge s_k00 \in L_2). \tag{4.7}$$

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 (4.7) 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 both, $M_i^{Q_k \cup \{s_k00\}}(y)$ and $M_j^{O_k \cup \{s_k00\}}(z)$ accept. 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. ☐

Thus there exist 3-tt-autoreducible sets that are not even weakly T-mitotic.[3] By Theorem 3.11, every 1-tt-autoreducible set is 1-tt-mitotic.

5. Remarks. One might wonder whether Theorem 3.1 still holds if one replaces polynomial-time many-one reductions by logarithmic-space many-one reductions. However, in this logspace setting, the proof of Theorem 3.1 does not go through, since the logspace analog of Claim 3.6 fails. More precisely, we cannot argue that the function g is computable in logspace. Roughly speaking, g is defined as a polynomial superposition of f. This means that in order to compute g, we have to iterate f a polynomial number of times. If f is polynomial-time computable and not length increasing (recall that the length-increasing case is already treated by the function r), then g is computable in polynomial time. In contrast, for a logspace computable f, we cannot iterate f for more than a constant number of times. So in this case, g is not logspace computable. The logspace analog of Theorem 3.1 is indeed false in a relativized world [10].

REFERENCES

[3]In a forthcoming paper we improve this result to 2-tt-autoreducible sets.

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

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

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

[4] H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet, *Separating complexity classes using autoreducibility*, SIAM Journal on Computing, 29 (2000), pp. 1497–1520.

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

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

[7] ———, *Separating complexity classes using structural properties*, in Proceedings of the 19th IEEE Conference on Computational Complexity, 2004, pp. 130–138.

[8] ———, *A Post's program for complexity theory*, Bulletin of the EATCS, 85 (2005), pp. 41–51.

[9] R. Cole and U. Vishkin, *Deterministic coin tossing with applications to optimal parallel list ranking*, Information and Control, 70 (1986), pp. 32–53.

[10] C. Glasser, *Logspace mitoticity*, Tech. Report 400, Inst. für Informatik, Univ. Würzburg, 2007.

[11] C. Glasser, M. Ogihara, A. Pavan, A. L. Selman, and L. Zhang, *Autoreducibility, mitoticity, and immunity*, in Proceedings 30th International Symposium on Mathematical Foundations of Computer Science, vol. 3618 of Lecture Notes in Computer Science, Springer-Verlag, 2005, pp. 387–398.

[12] ———, *Autoreducibility, mitoticity, and immunity*, Tech. Report TR05-11, ECCC, 2005.

[13] R. Karp and R. Lipton, *Turing machines that take advice*, L'enseignement mathématique, 28 (1982), pp. 191–209.

[14] A. H. Lachlan, *The priority method I*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik, 13 (1967), pp. 1–10.

[15] R. E. Ladner, *A completely mitotic nonrecursive r.e. degree*, Trans. American Mathematical Society, 184 (1973), pp. 479–507.

[16] ———, *Mitotic recursively enumerable sets*, Journal of Symbolic Logic, 38 (1973), pp. 199–211.

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

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

[19] B. Trakhtenbrot, *On autoreducibility*, Dokl. Akad. Nauk SSSR, 192 (1970). Translation in Soviet Math. Dokl. 11(3): 814– 817, 1970.

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