Non-Mitotic Sets

Christian Glaßer[1], Alan L. Selman[2*], Stephen Travers[1**], and Liyu Zhang[2]

[1] Universität Würzburg, Germany, {glasser,travers}@informatik.uni-wuerzburg.de
[2] University at Buffalo, USA, {selman,lzhang7}@cse.buffalo.edu

Abstract. We study the question of the existence of non-mitotic sets in NP. We show under various hypotheses that

– 1-tt-mitoticity and m-mitoticity differ on NP.
– 1-tt-reducibility and m-reducibility differ on NP.
– There exist non-T-autoreducible sets in NP (by a result from Ambos-Spies, these sets are neither T-mitotic nor m-mitotic).
– T-autoreducibility and T-mitoticity differ on NP (this contrasts the situation in the recursion theoretic setting, where Ladner showed that autoreducibility and mitoticity coincide).
– 2-tt autoreducibility does not imply weak 2-tt-mitoticity.
– 1-tt-complete sets for NP are nonuniformly m-complete.

Classification: Computational and structural complexity

1 Introduction

A recursive set A is *T-mitotic* if there is a set $B \in \mathrm{P}$ such that $A \equiv_{\mathrm{T}}^{\mathrm{p}} A \cap B \equiv_{\mathrm{T}}^{\mathrm{p}} A \cap \overline{B}$. Ambos-Spies [AS84] introduced this notion of mitoticity into complexity theory and he also showed how to construct recursive non-mitotic sets. Buhrman, Hoene, and Torenvliet [BHT98] showed that EXP contains non-mitotic sets. Here we investigate the question of the existence of non-mitotic sets in NP. This is a difficult question because there are no natural examples of non-mitotic sets. Natural NP-complete sets are all paddable, and for this reason are T-mitotic. Moreover, Glasser et al. [GPSZ06] proved that all NP-complete sets are m-mitotic (and therefore T-mitotic). Also, nontrivial sets belonging to the class P are T-mitotic. So any unconditional proof of the existence of non-mitotic sets in NP would prove at the same time that $\mathrm{P} \neq \mathrm{NP}$.

Our first result was prompted by the question of whether NP contains sets that are not m-mitotic. We prove that if $\mathrm{EEE} \neq \mathrm{NEEE} \cap \mathrm{coNEEE}$, then there exists an $L \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$ that is 1-tt-mitotic but not m-mitotic. From this, it follows that under the same hypothesis, 1-tt-reducibility and m-reducibility differ on sets in NP. On the one hand, this consequence explains the need for a reasonably strong hypothesis. On the other hand, with essentially known techniques using P-selective sets, we show that 1-tt-reducibility and m-reducibility separate within NP under the weaker hypothesis that $\mathrm{E} \neq \mathrm{NE} \cap \mathrm{coNE}$.

* This work was done while the author was visiting the Department of Computer Science at the University of Würzburg, Germany. Research supported in part by NSF grant CCR-0307077 and by the Alexander von Humboldt-Stiftung.
** Supported by the Konrad-Adenauer-Stiftung.

This foray into questions about 1-tt-reducibility and m-reducibility provides a segue into our next result: We would like to know whether 1-tt-complete sets for NP are m-complete as well. We prove under a reasonable hypothesis that every 1-tt-complete sets for NP is complete under nonuniform m-reductions. The hypothesis states that the NP-complete set SAT does not infinitely-often belong to the class coNP.

In Glasser et al. [GPSZ06] the authors proved that every m-autoreducible set is m-mitotic. The same result follows for 1-tt-autoreducibility. In contrast, Ambos-Spies [AS84] proved that T-autoreducible does not imply T-mitotic. Also, Glasser et al. [GPSZ06] constructed a 3-tt-autoreducible set that is not weakly-T-mitotic. Hence, it is known that autoreducibility and mitoticity are not equivalent for all polynomial-time-bounded reductions between 3-tt-reducibility and Turing-reducibility. However, the question for 2-tt-reducibility has been open. Here we settle this question by showing the existence of a set in EXP that is 2-tt-autoreducible, but not weakly 2-tt-mitotic.

The last two results to be proved both give evidence of non-mitotic sets in NP. The first of these states that if $\mathrm{EEE} \neq \mathrm{NEEE}$, then there exists a set $C \in \mathrm{NP} - \mathrm{P}$ such that C is not T-autoreducible. Hence, C is not T-mitotic. The second such result shows that if $\mathrm{NP} \cap \mathrm{coNP}$ contains n-generic sets, then there exists a set $L \in \mathrm{NP} \cap \mathrm{coNP}$ such that L is 2-tt-autoreducible and L is not T-mitotic. Roughly speaking, a set L is n-generic [ASFH87] if membership of x in L cannot be predicted from the initial segment $L|x$ in time 2^n, for almost all x, where $|x| = n$. This result is interesting, since under the mentioned hypothesis it shows that within NP the notions of T-autoreducibility and T-mitoticity differ. In contrast, Ladner [Lad73] showed that in the recursion theoretic setting, autoreducibility and mitoticity coincide.

Assumption	Conclusion	Remark
$\mathrm{EEE} \neq \mathrm{NEEE}$	$\exists A \in \mathrm{NP}$ that is not T-autoreducible	$A \in \mathrm{NP} - \mathrm{P}$
$\mathrm{NP} \cap \mathrm{coNP}$ contains n-generic sets	$\exists A \in \mathrm{NP}$ that is 2-tt-autoreducible but not T-mitotic	$A \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$
$\mathrm{EEE} \neq \mathrm{NEEE} \cap \mathrm{coNEEE}$	$\exists A \in \mathrm{NP}$ that is 1-tt-mitotic but not m-mitotic	$A \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$
$\mathrm{E} \neq \mathrm{NE} \cap \mathrm{coNE}$	$\exists A, B \in \mathrm{NP}$ such that $A \leq^{\mathrm{p}}_{1-\mathrm{tt}} B$ but $A \not\leq^{\mathrm{p}}_{\mathrm{m}} B$	$A, B \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$
$\mathrm{NP} \overset{\text{i.o.}}{\not\subseteq} \mathrm{coNP}$	1-tt-complete sets for NP are nonuniformly m-complete	

Table 1. Summary of our results related to NP

2 Preliminaries

We recall basic notions. Σ denotes a finite alphabet with at least two letters, Σ^* denotes the set of all words, and $|w|$ denotes the length of a word w. A tally set is a subset of 0^*. The language accepted by a machine M is denoted by $L(M)$. $\overline{L}$ denotes the complement of a language L and co$\mathcal{C}$ denotes the class of complements of languages in $\mathcal{C}$. FP denotes the class of functions computable in deterministic polynomial time.

We recall standard polynomial-time reducibilities [LLS75]. A set B *many-one-reduces* to a set C (*m-reduces* for short; in notation $B \leq_{\mathrm{m}}^{\mathrm{p}} C$) if there exists a total, polynomial-time-computable function f such that for all strings x, $x \in B \Leftrightarrow f(x) \in C$.

A set B *Turing-reduces* to a set C (*T-reduces* for short; in notation $B \leq_{\mathrm{T}}^{\mathrm{p}} C$) if there exists a deterministic polynomial-time-bounded oracle Turing machine M such that for all strings x, $x \in B \Leftrightarrow M$ with C as oracle accepts the input x.

Let $Q(M, x)$ denote the set of all queries to the oracle made by the oracle Turing machine M on input x.

A set B *truth-table-reduces* to a set C (*tt-reduces* for short; in notation $B \leq_{\mathrm{tt}}^{\mathrm{p}} C$) if there exists a deterministic polynomial-time-bounded oracle Turing machine M that behaves *non-adaptively* such that for all strings x,

$$x \in B \Leftrightarrow M \text{ with } C \text{ as oracle accepts the input } x.$$

This means there exists a polynomial-time function g such that on input x, $g(x) = cq_1c \ldots cq_n$ where $c \notin \Sigma$ and for all $1 \leq i \leq n$, $q_i \in \Sigma^*$, and $Q(M, x) = \{q_1, \ldots, q_n\}$.

Furthermore, B 1-tt reduces to C (in notation $B \leq_{1-\mathrm{tt}}^{\mathrm{p}} C$) if for some M, $B \leq_{\mathrm{tt}}^{\mathrm{p}} C$ via M and for all x, $|Q(M, x)| = 1$. Similarly, we define 2-tt, and so on.

If $B \leq_{\mathrm{m}}^{\mathrm{p}} C$ and $C \leq_{\mathrm{m}}^{\mathrm{p}} B$, then we say that B and C are *many-one-equivalent* (*m-equivalent* for short, in notation $B \equiv_{\mathrm{m}}^{\mathrm{p}} C$). Similarly, we define equivalence for other reducibilities. A set B is *many-one-hard* (*m-hard* for short) for a complexity class $\mathcal{C}$ if every $B \in \mathcal{C}$ m-reduces to B. If additionally $B \in \mathcal{C}$, then we say that B is *many-one-complete* (*m-complete* for short) for $\mathcal{C}$. Similarly, we define hardness and completeness for other reducibilities. We use "$\mathcal{C}$-complete" as an abbreviation for m-complete for $\mathcal{C}$.

A set B is *p-selective* [Sel79] if there exists a total function $f \in$ FP (the selector function) such that for all x and y, $f(x, y) \in \{x, y\}$ and if either of x and y belongs to B, then $f(x, y) \in B$.

Definition 1 ([AS84]) *A set A is* polynomial-time 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* polynomial-time m-autoreducible *(m-autoreducible, for short) if $A \leq_{\mathrm{m}}^{\mathrm{p}} A$ via a reduction function f such that for all x, $f(x) \neq x$.*

Let $\leq_{\mathrm{r}}^{\mathrm{p}}$ be a polynomial time reducibility.

Definition 2 ([AS84]) *A recursive set A is* polynomial-time r-mitotic *(r-mitotic, for short) if there exists a set $B \in$ P such that:*

$$A \equiv_{\mathrm{r}}^{\mathrm{p}} A \cap B \equiv_{\mathrm{r}}^{\mathrm{p}} A \cap \overline{B}.$$

A recursive set A is polynomial-time weakly r-mitotic *(weakly r-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and*

$$A \equiv_{\mathrm{r}}^{\mathrm{p}} A_0 \equiv_{\mathrm{r}}^{\mathrm{p}} A_1.$$

Let EEE $= \mathrm{DTIME}(2^{2^{2^{O(n)}}})$ and let NEEE $= \mathrm{NTIME}(2^{2^{2^{O(n)}}})$.

3 Separation of Mitoticity Notions

Ladner, Lynch, and Selman [LLS75] and Homer [Hom90,Hom97] ask for reasonable assumptions that imply separations of polynomial-time reducibilities within NP. In this section we demonstrate that a reasonable assumption on exponential-time classes allows a separation of mitoticity notions within NP. This implies a separation of the reducibilities $\leq_{\mathrm{m}}^{\mathrm{p}}$ and $\leq_{1-\mathrm{tt}}^{\mathrm{p}}$ within NP. Then we show the same separation under an even weaker hypothesis. On the technical side, a key ingredient to our proof is the observation by Beigel and Feigenbaum [BF92] that very sparse sets lack certain redundancy properties.

Theorem 3 *If* $\mathrm{EEE} \neq \mathrm{NEEE} \cap \mathrm{coNEEE}$*, then there exists an* $L \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$ *that is 1-tt-mitotic but not m-mitotic.*

The proof of this theorem can be found in the appendix.

Selman [Sel82] showed under the hypothesis $\mathrm{E} \neq \mathrm{NE} \cap \mathrm{coNE}$ that there exist $A, B \in \mathrm{NP} - \mathrm{P}$ such that A tt-reduces to B but A does not positive-tt-reduce to B. The separation of mitoticity notions given in the last theorem allows us to prove a similar statement:

Corollary 4 *If* $\mathrm{EEE} \neq \mathrm{NEEE} \cap \mathrm{coNEEE}$*, then there exist* $A, B \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$ *such that* $A \leq_{1-\mathrm{tt}}^{\mathrm{p}} B$*, but* $A \not\leq_{\mathrm{m}}^{\mathrm{p}} B$*.*

Proof. Take the set L from Theorem 3 and let $S \in \mathrm{P}$ be a separator that witnesses L's 1-tt-mitoticity, i.e., L, $L \cap S$, and $L \cap \overline{S}$ are pairwise 1-tt-equivalent. These sets cannot be pairwise m-equivalent, since otherwise L is m-mitotic. This gives us A and B. □

However, an even weaker assumption separates 1-tt-reducibility from m-reducibility within NP.

Theorem 5 *If* $\mathrm{E} \neq \mathrm{NE} \cap \mathrm{coNE}$*, then there exist* $A, B \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$ *such that* $A \leq_{1-\mathrm{tt}}^{\mathrm{p}} B$*, but* $A \not\leq_{\mathrm{m}}^{\mathrm{p}} B$*.*

Proof. If $\mathrm{E} \neq \mathrm{NE} \cap \mathrm{coNE}$, then there exists a tally set $T \in \mathrm{NP} \cap \mathrm{coNP} - \mathrm{P}$ and there exists a p-selective set A such that $A \equiv_{\mathrm{T}}^{\mathrm{p}} T$ [Sel79]. Trivially, $A \leq_{1-\mathrm{tt}}^{\mathrm{p}} \overline{A}$, and since A is p-selective, and not in P, A is not m-reducible to $\overline{A}$. □

We now discuss that autoreducibility and weak mitoticity do not coincide for 2-tt reducibility. This completes a result by Glaßer et al. [GPSZ06] which shows that for all reducibilities between 3-tt and T, autoreducibility does not imply weak mitoticity. We present a counterexample in EXP, i.e., we construct a set $L \in \mathrm{EXP}$ such that L is 2-tt-autoreducible but not weakly 2-tt-mitotic.

Theorem 6 *There exists* $L \in \mathrm{SPARSE} \cap \mathrm{EXP}$ *such that*

- *L is 2-tt-autoreducible, but*
- *L is not weakly 2-tt-mitotic.*

The proof is based on the diagonalization proof of Theorem 4.2 in Glasser et al. [GPSZ06]. However, a straightforward adaptation does not work. The reason is that if one considers groups of three strings at certain super-exponential lengths for diagonalization, the set constructed as in the previous proof will have to be 2-tt-mitotic if we were to make it 2-tt-autoreducible. The new idea is to consider two groups of three strings at super-exponential lengths that overlap at one string. This way we can make the set 2-tt-autoreducible while not 2-tt-mitotic. The detailed construction is omitted due to space restrictions.

The full paper demonstrates that the proof technique cannot be generalized to show that there exists a set in EXP that is 2-tt-autoreducible, but not weakly T-mitotic. So this question remains open.

4 Non-Mitotic Sets of Low Complexity

Buhrman, Hoene, and Torenvliet [BHT98] show that EXP contains non-m-mitotic sets. We are interested in constructing non-T-mitotic sets in NP. Recall that the existence of such sets implies that $\mathrm{P} \neq \mathrm{NP}$ and hence we cannot expect to prove this without a sufficiently strong hypothesis. Moreover, the same holds for the non-existence of non-m-mitotic sets in NP, since this implies $\mathrm{NP} \neq \mathrm{EXP}$ [BHT98].

It is well known that mitoticity implies autoreducibility [AS84], hence it suffices to construct non-T-autoreducible sets in NP. Beigel and Feigenbaum [BF92] construct incoherent sets in NP under the assumption that $\mathrm{NEEEXP} \not\subseteq \mathrm{BPEEEXP}$. In particular, these sets are non-T-autoreducible. With the next theorem, we show that there are non-T-autoreducible sets in NP under the weaker assumption that $\mathrm{NEEE} \not\subseteq \mathrm{EEE}$. Observe that these sets are not necessarily incoherent.

Also, under a strong assumption, we prove that 2-tt-autoreducibility and T-mitoticity (and hence r-autoreducibility and r-mitoticity for every reduction r between 2-tt and T) do not coincide for NP.

Theorem 7 *If* $\mathrm{EEE} \neq \mathrm{NEEE}$*, then there exists* $C \in \mathrm{NP} - \mathrm{P}$ *such that* C *is not T-autoreducible.*

The proof of this theorem can be found in the appendix.

Corollary 8 *If* $\mathrm{EEE} \neq \mathrm{NEEE}$*, then there exists* $C \in \mathrm{NP} - \mathrm{P}$ *such that* C *is not T-mitotic.*

Proof. T-mitoticity implies T-autoreducibility [AS84]. Consequently, the set C in Theorem 7 cannot be T-mitotic since it is not T-autoreducible. $\square$

Under a stronger assumption we construct non-T-autoreducible sets in $(\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$.

Corollary 9 *If* $\mathrm{EEE} \neq \mathrm{NEEE} \cap \mathrm{coNEEE}$*, then there exists* $C \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$ *such that*

- *C is not T-autoreducible and*
- *C is not T-mitotic.*

Proof. This can easily be seen by using the set C from the proof of Theorem 3 in the proof of Theorem 7 instead of the one constructed in the latter. $\square$

Ladner [Lad73] showed that autoreducibility and mitoticity coincide for computably enumerable sets. Under the strong assumption that NP $\cap$ coNP contains n-generic sets, we can show that the similar question in complexity theory has a negative answer.

The notion of resource-bounded genericity was defined by Ambos-Spies, Fleischhack, and Huwig [ASFH87]. We use the following equivalent definition [BM95,PS02], where $L(x)$ denotes L's characteristic function on x.

Definition 10 *For a set L and a string x let $L|x = \{y \in L \mid y < x\}$. A deterministic oracle Turing machine M is a* predictor *for a set L, if for all x, $M^{L|x}(x) = L(x)$. L is* a.e. unpredictable in time $t(n)$, *if every predictor for L requires more than $t(n)$ time for all but finitely many x.*

Definition 11 *A set L is* $t(n)$-generic *if it is a.e. unpredictable in time $t(2^n)$.*

This is equivalent to say that for every oracle Turing machine M, if $M^{L|x}(x) = L(x)$ for all x, then the running time of M is at least $t(2^{|x|})$ for all but finitely many x.

For a given set L and two strings x and y, there are 4 possibilities for the string $L(x)L(y)$. For 1-cheatable sets L, a polynomial-time-computable function can reduce the number of possibilities to 2.

Definition 12 ([Bei87,Bei91]) *A set L is* 1-cheatable *if there exists a polynomial-time-computable function f such that $f : \Sigma^* \times \Sigma^* \longrightarrow \{0,1\}^2 \times \{0,1\}^2$ and for all x and y, the string $L(x)L(y)$ belongs to $f(x,y)$.*

Note that in this definition and in the following text we identify the pair $f(x,y) = (w_1, w_2)$ with the set $\{w_1, w_2\}$. Moreover, if $f(x,y) = (w_1, w_2)$, then $f(x,y)^R$ denotes the pair (w_1^R, w_2^R) where w^R denotes the reverse of the word w.

Theorem 13 *If* NP $\cap$ coNP *contains n-generic sets, then there exists a tally set $S \in$* NP $\cap$ coNP *such that*

- *S is 2-tt-autoreducible and*
- *S is not T-mitotic.*

Proof. Let $t(0) = 2$ and $t(n+1) = 2^{2^{t(n)}}$ be a tower function. Let $A' = \{0^{t(n)} \mid n \geq 0\}$, $A'' = A' \cup 0A'$, and $A''' = A' \cup 0A' \cup 00A'$. In this way, the number of primes indicates the number of words in the set with length around $t(n)$ for each n. By assumption, there exists an n-generic set $L \in$ NP $\cap$ coNP. Define $L'' = L \cap A''$ and observe that $L'' \in$ NP $\cap$ coNP.

Claim 14 *L'' is not 1-cheatable.*

Assuming that L'' is 1-cheatable we will show that L is not n-generic. Let f be a function that witnesses the 1-cheatability of L''. Without loss of generality we may assume that if $f(x,y) = (v,w)$, then $v \neq w$.

$$g(x,y) =_{\mathrm{def}} \begin{cases} f(x,y) & : \text{ if } x < y \\ f(y,x)^R & : \text{ if } x > y \\ (00,11) & : \text{ if } x = y \end{cases}$$

Observe that also g witnesses the 1-cheatability of L'' such that if $g(x,y) = (v,w)$, then $v \neq w$. In addition, for all x and y,

$$g(x,y) = g(y,x)^R. \tag{1}$$

We describe a predictor M for L on input x.

```
1. if $x \notin A''$ then accept if and only if $x \in L$
2. // here either $x = 0^{t(n)}$ or $x = 0^{t(n)+1}$ for some n
3. if $x = 0^{t(n)}$ then let $y = 0^{t(n)+1}$ else let $y = 0^{t(n)}$
   (i.e., with y we compute the neighbor of x in $A''$)
4. compute $g(x,y) = (ab, cd)$ where a, b, c, and d are suitable bits
5. if $a = c$ then return a
6. if $b = d$ then accept if and only if $x \in L$
7. // here $ab = \overline{cd}$ and hence $g(x,y) = \{00,11\}$ or $g(x,y) = \{01,10\}$
8. if $a = b$ and $|x| > |y|$ then accept if and only if y belongs to the
   oracle $L|x$
9. if $a = b$ and $|x| \leq |y|$ then accept if and only if $x \in L$
10. // here $g(x,y) = \{01,10\}$
11. if $|x| > |y|$ then accept if and only if y does not belong to the
    oracle $L|x$
12. accept if and only if $x \in L$
```

In the algorithm, the term `accept if and only if` $x \in L$ means that first, in deterministic time $2^{n^{O(1)}}$, we find out whether x belongs to L, and then we accept accordingly.

We observe that M is a predictor for L: In line 5, M predicts correctly, since $g(x,y) = (ab, ad)$ and therefore, $L(x) = a$. M predicts correctly in line 8, since $g(x,y) = \{00,11\}$ implies $x \in L \Leftrightarrow y \in L$ and $|y| < |x|$ implies $y \in L|x \Leftrightarrow y \in L$. M predicts correctly in line 11, since $g(x,y) = \{01,10\}$ implies $x \in L \Leftrightarrow y \notin L$ and again $|y| < |x|$ implies $y \in L|x \Leftrightarrow y \in L$. Hence M is a predictor for L.

If we do not take the lines 1, 6, 9, and 12 into account, then the running time of M is polynomially bounded, say by the polynomial p. Now we are going to show the following.

For all n, at least one of the following holds: $M^{L|x}(x)$ stops within $p(|x|)$ steps or $M^{L|y}(y)$ stops within $p(|y|)$ steps, where $x = 0^{t(n)}$ and $y = 0^{t(n)+1}$. $(*)$

Assume $(*)$ does not hold for a particular n, and let $x = 0^{t(n)}$ and $y = 0^{t(n)+1}$. Hence, both computations, $M^{L|x}(x)$ and $M^{L|y}(y)$ must stop in one of the lines 1, 6, 9, and 12. Since, $x, y \in A''$, these computations do not stop in line 1.

Assume $M^{L|x}(x)$ stops in line 6. In this case, $g(x,y) = (ab, cb)$. By (1), the computation $M^{L|y}(y)$ computes the value $g(y,x) = (ba, bc)$ in line 4. So $M^{L|y}(y)$ stops in line 5, which contradicts our observation that we must stop in the lines 6, 9, or 12. This shows that $M^{L|x}(x)$ does not stop in line 6. Analogously we obtain that $M^{L|y}(y)$ does not stop in line 6. So both computations must stop in line 9 or line 12.

$M^{L|y}(y)$ does not stop in line 9, since in this computation, the second condition in line 9 evaluates to false. So $M^{L|y}(y)$ stops in line 12. However, this is not possible, since $M^{L|y}(y)$ would have stopped already in line 11. This proves $(*)$.

From $(*)$ it follows that for infinitely many x, $M^{L|x}(x)$ stops within $p(|x|)$ steps. Hence L is not $(\log p(n))$-generic and in particular, not n-generic. This contradicts our assumption on L. (Note that we obtain also a contradiction if we assume L to be $t(n)$-generic such that $t(n) > c \log n$ for all $c > 0$.) This finishes the proof of Claim 14.

So far we constructed an $L'' \in \mathrm{NP} \cap \mathrm{coNP}$ such that $L'' \subseteq A''$ and L'' is not 1-cheatable. Now we define a set $L''' \subseteq A'''$ (this will be the set asserted in the theorem). For $n \geq 0$ let $x_n = 0^{t(n)}$, $y_n = 0^{t(n)+1}$, $z_n = 0^{t(n)+2}$, and $c_n = L''(x_n)L''(y_n)$. Define L''' to be the unique subset of A''' that satisfies the following conditions where $d_n = L'''(x_n)L'''(y_n)L'''(z_n)$:

1. if $c_n = 00$ then $d_n = 000$
2. if $c_n = 01$ then $d_n = 110$
3. if $c_n = 10$ then $d_n = 101$
4. if $c_n = 11$ then $d_n = 011$

Observe that L''' is a tally set in $\mathrm{NP} \cap \mathrm{coNP}$. Moreover, note that for all n, either 0 or 2 words from $\{x_n, y_n, z_n\}$ belong to L'''. This implies that L''' is 2-tt-autoreducible: If the input x is not in A''', then reject. Otherwise, determine the n such that $x \in \{x_n, y_n, z_n\}$. Ask the oracle for the two words in $\{x_n, y_n, z_n\} - \{x\}$ and output the parity of the answers.

Claim 15 *L''' is not T-mitotic.*

Assume L''' is T-mitotic, and let $S \in \mathrm{P}$ be a witnessing separator. Let $L''' \leq^{\mathrm{p}}_{\mathrm{T}} L''' \cap \overline{S}$ via machine M_1 and let $L''' \leq^{\mathrm{p}}_{\mathrm{T}} L''' \cap S$ via machine M_2. We will obtain a contradiction by showing that L'' is 1-cheatable. We define the witnessing function $h(x,y)$ as follows.

1. If $x = y$ then output $(00, 11)$.
2. If $|x| > |y|$ then output $h(y,x)^R$.
3. If $x \notin A''$ then output $(00, 01)$.
4. If $y \notin A''$ then output $(00, 10)$.
5. // Here $|x| < |y|$ and $x, y \in A''$.
6. If $|y| - |x| > 1$ then let $a = L''(x)$ and output $(a0, a1)$.
7. Determine n such that $x = x_n$ and $y = y_n$.
8. Distinguish the following cases.
   (a) $S \cap \{x_n, y_n, z_n\} = \emptyset$: Simulate $M_2(x_n)$, $M_2(y_n)$, and $M_2(z_n)$ where oracle queries q of length $\leq t(n-1)+2$ are answered according to $q \in L''' \cap S$ and all other oracle queries are answered negatively. Let d_n be the concatenation of the outputs of these simulations. Let c_n be the value corresponding to d_n according to the definition of L'''. Output $(c_n, 00)$.

(b) $\overline{S} \cap \{x_n, y_n, z_n\} = \emptyset$: Do the same as in step 8a, but use M_1 instead of M_2 and answer short queries q according to $q \in L''' \cap \overline{S}$.

(c) $|S \cap \{x_n, y_n, z_n\}| = 1$: Without loss of generality we assume $x_n \in S$ and $y_n, z_n \notin S$. For $r \in \{\text{yes}, \text{no}\}$ we simulate $M_2(x_n)$, $M_2(y_n)$, and $M_2(z_n)$ where oracle queries q of length $\leq t(n-1)+2$ are answered according to $q \in L''' \cap S$, the oracle query x_n is answered with r, and all other oracle queries q are answered negatively. Let d_r be the concatenation of the outputs of these simulations. Let c_r be the value corresponding to d_r according to the definition of L''' (if such c_r does not exist, then let $c_r = 00$). Output $(c_{\text{yes}}, c_{\text{no}})$.

(d) $|\overline{S} \cap \{x_n, y_n, z_n\}| = 1$: Do the same as in step 8c, but use M_1 instead of M_2 and answer short queries q according to $q \in L''' \cap \overline{S}$.

We argue that h is computable in polynomial time. Note that if we recursively call $h(y, x)$ in step 2, then the computation of $h(y, x)$ will not call h again. So the recursion depth of the algorithm is ≤ 2. In step 6, $|x| < |y|$ and $x, y \in A''$, since $|x| = |y|$ implies that we stop in line 3 or 4. From the definition of A'' it follows that there exists an n such that $|x| \leq t(n-1)+1$ and $|y| \geq t(n)$. So the computation of a in step 6 takes time

$$\leq 2^{|x|^{O(1)}} \leq 2^{t(n-1)^{O(1)}} \leq 2^{2^{t(n-1)}} = t(n) \leq |y|. \tag{2}$$

The n in step 7 exists, since $x, y \in A''$ and $|y| - |x| = 1$. In step 8, queries q of length $\leq t(n-1)+2$ must be answered according to $q \in L''' \cap S$ or according to $q \in L''' \cap \overline{S}$. Similar to (2) these simulations can be done in polynomial time in $|x|$. This shows that h is computable in polynomial time.

We now argue that h witnesses that L'' is 1-cheatable, i.e., if $f(x, y) = (ab, cd)$, then $L''(x)L''(y) = ab$ or $L''(x)L''(y) = cd$. It suffices to show this for the case $|x| < |y|$. If we stop in step 3, then $x \notin L''$ and hence $L''(x)L''(y) = 00$ or $L''(x)L''(y) = 01$. Similarly, if we stop in step 4, then $y \notin L''$ and hence $L''(x)L''(y) = 00$ or $L''(x)L''(y) = 10$. If we stop in step 6, then $L''(x) = a$ and so $L''(x)L''(y) = a0$ or $L''(x)L''(y) = a1$. So it remains to argue for step 8.

Now assume the output is made in step 8a. Consider the computations $M_2^{L''' \cap S}(x_n)$, $M_2^{L''' \cap S}(y_n)$, and $M_2^{L''' \cap S}(z_n)$. Since these are polynomial-time computations, they cannot ask for words of length $\geq t(n+1) = 2^{2^{t(n)}}$. So x_n, y_n, and z_n are the only candidates for words that are of length $> t(n-1)+2$ and that can be queried by these computations. But by assumption of case 8a, these words are not in $L''' \cap S$. Therefore, the simulations of $M_2(x_n)$, $M_2(y_n)$, and $M_2(z_n)$ in step 8a behave the same way as the computations $M_2^{L''' \cap S}(x_n)$, $M_2^{L''' \cap S}(y_n)$, and $M_2^{L''' \cap S}(z_n)$. Hence we obtain $d_n = L'''(x_n)L'''(y_n)L'''(z_n)$ and $c_n = L''(x_n)L''(y_n)$. So the output contains the string $L'''(x)L'''(y)$. Step 8b is argued similar to step 8a.

Assume the output is made in step 8c. We can reuse the argument from step 8a. The only difference is the words x_n. It can be an element of $L''' \cap S$ and it can be queried by the computations $M_2^{L''' \cap S}(x_n)$, $M_2^{L''' \cap S}(y_n)$, and $M_2^{L''' \cap S}(z_n)$. So we simulate both possibilities, the one where $x_n \in L''' \cap S$ and the one where $x_n \notin L''' \cap S$. So at least one of the strings c_{yes} and c_{no} equals $L'''(x)L'''(y)$ and so the output contains the string $L'''(x)L'''(y)$. Step 8d is argued similar to step 8c.

This shows that L'' is 1-cheatable via function h. This contradicts Claim 14 and therefore, L''' is not T-mitotic. This finishes the proof of Claim 15 and of Theorem 13. □

Corollary 16 *If* NP ∩ coNP *contains n-generic sets, then T-autoreducibility and T-mitoticity differ on* NP.

Proof. Follows from the fact that every 2-tt-autoreducible set is T-autoreducible. □

Corollary 17 *Let $t(n)$ be a function such that for all $c > 0$, $t(n) > c \log n$. If* NP ∩ coNP *contains $t(n)$-generic sets, then there exists a tally set $L \in$* NP ∩ coNP *that is 2-tt-autoreducible, but not T-mitotic.*

Proof. Follows from the proof of Theorem 13. □

5 Uniformly Hard Languages in NP

In this section we assume that NP contains uniformly hard languages, i.e., languages that are uniformly not contained in coNP. After discussing this assumption we show that it implies that every $\leq^{\mathrm{p}}_{1-\mathrm{tt}}$-complete set for NP is nonuniformly NP-complete.

Recall that we have separated 1-tt-reducibility from m-reducibility within NP under a reasonable assumption in Section 3. Nevertheless the main result of this section indicates that these two reducibilities are pretty similar in terms of NP-complete problems: Every $\leq^{\mathrm{p}}_{1-\mathrm{tt}}$-complete set for NP is m-complete if we allow the reducing function to use an advice of polynomial length.

Definition 18 *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes, and let A and B be subsets of Σ^*.*

1. $A \overset{\text{i.o.}}{=} B \overset{df}{\iff}$ *for infinitely many n it holds that $A \cap \Sigma^n = B \cap \Sigma^n$.*
2. $A \overset{\text{i.o.}}{\in} \mathcal{C} \overset{df}{\iff}$ *there exists $C \in \mathcal{C}$ such that $A \overset{\text{i.o.}}{=} C$.*
3. $\mathcal{C} \overset{\text{i.o.}}{\subseteq} \mathcal{D} \overset{df}{\iff}$ $C \overset{\text{i.o.}}{\in} \mathcal{D}$ *for all $C \in \mathcal{C}$.*

The following proposition is easy to observe.

Proposition 19 *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes, and let A and B be subsets of Σ^*.*

1. $A \overset{\text{i.o.}}{=} B$ *if and only if* $\overline{A} \overset{\text{i.o.}}{=} \overline{B}$.
2. $A \overset{\text{i.o.}}{\in} \mathcal{C}$ *if and only if* $\overline{A} \overset{\text{i.o.}}{\in} \mathrm{co}\mathcal{C}$.
3. $\mathcal{C} \overset{\text{i.o.}}{\subseteq} \mathcal{D}$ *if and only if* $\mathrm{co}\mathcal{C} \overset{\text{i.o.}}{\subseteq} \mathrm{co}\mathcal{D}$.

Proposition 20 *The following are equivalent:*

(i) coNP $\overset{\text{i.o.}}{\not\subseteq}$ NP
(ii) NP $\overset{\text{i.o.}}{\not\subseteq}$ coNP
(iii) *There exists an $A \in$* NP *such that $A \overset{\text{i.o.}}{\notin}$* coNP.
(iv) *There exists a paddable* NP*-complete A such that $A \overset{\text{i.o.}}{\notin}$* coNP.

Proof. The equivalence of (i) and (ii) is by Proposition 19. Moreover, from the definition it immediately follows that $\neg$(ii) $\Rightarrow \neg$(iii) and $\neg$(iii) $\Rightarrow \neg$(iv). It remains to show $\neg$(iv) $\Rightarrow \neg$(ii). So we assume that for all paddable NP-complete A it holds that $A \overset{\text{i.o.}}{\in} \text{coNP}$. Choose any $C \in \text{NP}$ and let $B = 0C \cup 1\text{SAT}$. Hence B is NP-complete and paddable (for paddability, observe that elements from $0\Sigma^*$ are first mapped to $1\Sigma^*$ via the reduction $0C \leq_{\text{m}}^{\text{p}} 1\text{SAT}$). By our assumption $B \overset{\text{i.o.}}{\in} \text{coNP}$. So there exists a $D \in \text{coNP}$ such that $B \overset{\text{i.o.}}{=} D$. Let $D' = \{w \mid 0w \in D\}$ and note that $D' \in \text{coNP}$. Observe that for every n, if $B \cap \Sigma^{n+1} = D \cap \Sigma^{n+1}$, then $C \cap \Sigma^n = D' \cap \Sigma^n$. Hence $C \overset{\text{i.o.}}{=} D'$ which shows $C \overset{\text{i.o.}}{\in} \text{coNP}$. □

We define polynomial-time many-one reductions with advice. Non-uniform reductions are of interest in cryptography, where they model an adversary who is capable of long preprocessing [BV97]. They also have applications in structural complexity theory. Agrawal [Agr02] and Hitchcock and Pavan [HP06] investigate non-uniform reductions and show under reasonable hypotheses that every many-one complete set for NP is also hard for length-increasing, non-uniform reductions.

Definition 21 *$A \leq_{\text{m}}^{\text{p/poly}} B$ if there exists an $f \in \text{FP/poly}$ such that for all words x, $x \in A \Leftrightarrow f(x) \in B$.*

The following theorem assumes as hypothesis that $\text{NP} \overset{\text{i.o.}}{\not\subseteq} \text{coNP}$. This hypothesis says for sufficiently long formulas that not all tautologies of a given size have short proofs. We use this hypothesis to show that 1-tt-complete sets for NP are nonuniformly m-complete.

Theorem 22 *If $\text{NP} \overset{\text{i.o.}}{\not\subseteq} \text{coNP}$, then every $\leq_{1-\text{tt}}^{\text{p}}$-complete set for NP is $\leq_{\text{m}}^{\text{p/poly}}$-complete.*

Proof. By assumption, there exists an NP-complete K such that $K \overset{\text{i.o.}}{\notin} \text{coNP}$. Choose $g \in \text{FP}$ such that $\{(u,v) \mid u \in K \vee v \in K\} \leq_{\text{m}}^{\text{p}} K$ via g. Let A be $\leq_{1-\text{tt}}^{\text{p}}$-complete for NP. So $K \leq_{1-\text{tt}}^{\text{p}} A$, i.e., there exists a polynomial-time computable function $f : \Sigma^* \mapsto \Sigma^* \cup \{\overline{w} \mid w \in \Sigma^*\}$ such that for all words x:

1. If $f(x) = w$ for some $w \in \Sigma^*$, then $(x \in K \Leftrightarrow w \in A)$.
2. If $f(x) = \overline{w}$ for some $w \in \Sigma^*$, then $(x \in K \Leftrightarrow w \notin A)$.

Moreover, choose $r \in \text{FP}$ such that $A \leq_{\text{m}}^{\text{p}} K$ via r. Define

$$\text{EASY} =_{\text{def}} \{u \mid \exists v, |v| = |u|, f(g(u,v)) = \overline{w} \text{ for some } w \in \Sigma^*, \text{ and } r(w) \in K\}$$

EASY belongs to NP. We see $\text{EASY} \subseteq \overline{K}$ as follows: $r(w) \in K$ implies $w \in A$, hence $g(u,v) \notin K$, and hence $u \notin K$. From our assumption $\overline{K} \overset{\text{i.o.}}{\notin} \text{NP}$ it follows that there exists an $n_0 \geq 0$ such that

$$\forall n \geq n_0, \overline{K}^{=n} \not\subseteq \text{EASY}^{=n}.$$

So for every $n \geq n_0$ we can choose a word $w_n \in \overline{K}^{=n} - \text{EASY}$. For $n < n_0$, let $w_n = \varepsilon$. Choose fixed $z_1 \in A$ and $z_0 \notin A$. We define the reduction that witnesses $K \leq_{\text{m}}^{\text{p/poly}} A$.

$$h(v) =_{\text{def}} \begin{cases} f(g(w_{|v|}, v)) & : \text{ if } |v| \geq n_0 \text{ and } f(g(w_{|v|}, v)) \in \Sigma^* \\ z_1 & : \text{ if } |v| \geq n_0 \text{ and } f(g(w_{|v|}, v)) = \overline{w} \text{ for some } w \in \Sigma^* \\ z_1 & : \text{ if } |v| < n_0 \text{ and } v \in K \\ z_0 & : \text{ if } |v| < n_0 \text{ and } v \notin K \end{cases}$$

Observe that $h \in \mathrm{FP/poly}$ (even FP/lin) with the advice $n \mapsto w_n$. We claim for all v,

$$v \in K \Leftrightarrow h(v) \in A. \tag{3}$$

This equivalence clearly holds for all v such that $|v| < n_0$. So assume $|v| \geq n_0$ and let $n = |v|$.

If $f(g(w_n, v)) \in \Sigma^*$, then h is defined according to the first line of its definition and equivalence (3) is obtained as follows.

$$v \in K \Leftrightarrow g(w_n, v) \in K \Leftrightarrow f(g(w_n, v)) \in A$$

Otherwise, $f(g(w_n, v)) = \overline{w}$ for some $w \in \Sigma^*$. We claim that v must belong to K. If not, then $g(w_n, v) \notin K$ and hence $w \in A$ (since $K \leq^{\mathrm{p}}_{1-\mathrm{tt}} A$ via f). So $r(w) \in K$ which witnesses that $w_n \in \mathrm{EASY}$. This contradicts the choice of w_n and it follows that $v \in K$. This shows $v \in K \Leftrightarrow h(v) = z_1 \in A$ and proves equivalence (3). □

References

[Agr02] M. Agrawal. Pseudo-random generators and structure of complete degrees. In *IEEE Conference on Computational Complexity*, pages 139–147, 2002.

[AS84] K. Ambos-Spies. P-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines*, volume 171 of *Lecture Notes in Computer Science*, pages 1–23. Springer Verlag, 1984.

[ASFH87] K. Ambos-Spies, H. Fleischhack, and H. Huwig. Diagonalizations over polynomial time computable sets. *Theoretical Computer Science*, 51:177–204, 1987.

[Bei87] R. Beigel. *Query-Limited Reducibilities.* PhD thesis, Stanford University, 1987.

[Bei91] R. Beigel. Relativized counting classes: Relations among thresholds, parity, mods. *Journal of Computer and System Sciences*, 42:76–96, 1991.

[BF92] R. Beigel and J. Feigenbaum. On being incoherent without being very hard. *Computational Complexity*, 2:1–17, 1992.

[BHT98] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness, and structure of complete sets. *SIAM Journal on Computing*, 27:637–653, 1998.

[BM95] J. Balcazar and E. Mayordomo. A note on genericty and bi-immunity. In *Proceedings of the Tenth Annual IEEE Conference on Computational Complexity*, pages 193–196, 1995.

[BV97] D. Boneh and R. Venkatesan. Rounding in lattices and its cryptographic applications. In *SODA*, pages 675–681, 1997.

[GPSZ06] C. Glaßer, A. Pavan, A. L. Selman, and L. Zhang. Redundancy in complete sets. In *Proceedings 23nd Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*, pages 444–454. Springer, 2006.

[Hom90] S. Homer. Structural properties of nondeterministic complete sets. In *Structure in Complexity Theory Conference*, pages 3–10, 1990.

[Hom97] S. Homer. Structural properties of complete problems for exponential time. In A. L. Selman and L. A. Hemaspaandra, editors, *Complexity Theory Retrospective II*, pages 135–153. Springer Verlag, New York, 1997.

[HP06] J. Hitchcock and A. Pavan. Comparing reductions to NP-complete sets. Technical Report TR06-039, Electronic Colloquium on Computational Complexity, 2006.

[Lad73] R. E. Ladner. Mitotic recursively enumerable sets. *Journal of Symbolic Logic*, 38(2):199–211, 1973.

[LLS75] R. E. Ladner, N. A. Lynch, and A. L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1:103–123, 1975.

[PS02] A. Pavan and A. L. Selman. Separation of NP-completeness notions. *SIAM Journal on Computing*, 31(3):906–918, 2002.

[Sel79] A. L. Selman. P-selective sets, tally languages, and the behavior of polynomial-time reducibilities on NP. *Mathematical Systems Theory*, 13:55–65, 1979.

[Sel82] A. L. Selman. Reductions on NP and p-selective sets. *Theoretical Computer Science*, 19:287–304, 1982.

Appendix

Theorem 3 *If* EEE $\neq$ NEEE $\cap$ coNEEE*, then there exists an* $L \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$ *that is 1-tt-mitotic but not m-mitotic.*

Proof. Choose $B \in (\mathrm{NEEE} \cap \mathrm{coNEEE}) - \mathrm{EEE}$. So there exists a constant $c \geq 1$ such that B and $\overline{B}$ are decidable in nondeterministic time $2^{2^{2^{c \cdot n}}}$. Let $t(x) =_{\mathrm{def}} 2^{2^{x^{2c}}}$ be a tower function and let $A =_{\mathrm{def}} \{0^{t(n)} \mid n \geq 0\}$ and $C =_{\mathrm{def}} \{0^{t(x)} \mid x \in B\}$ (note that we identify Σ^* with the natural numbers). Note $A \in \mathrm{P}$.

Claim 23 $C \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$.

A membership test for C has to decide $x \in B$ on input $y = 0^{2^{2^{x^{2c}}}}$. The test $x \in B$ can be carried out in nondeterministic time

$$2^{2^{2^{c \cdot |x|}}} \leq 2^{2^{2^{c \cdot 2 \cdot \log x}}} = 2^{2^{x^{2c}}} = |y|.$$

Therefore, $C \in \mathrm{NP}$ and analogously $C \in \mathrm{coNP}$, since $B \in \mathrm{coNEEE}$.

Assume $C \in \mathrm{P}$. Then B can be decided as follows: On input x we construct the string $y = 0^{2^{2^{x^{2c}}}}$ and simulate the deterministic polynomial-time decision procedure for C. Clearly, this algorithm decides C.

$$|y| = 2^{2^{x^{2c}}} \leq 2^{2^{(2^{|x|})^{2c}}} = 2^{2^{2^{(2c|x|)}}}$$

So the described algorithm has a running time that is polynomial in $2^{2^{2^{(2c|x|)}}}$. This shows $B \in \mathrm{EEE}$ which contradicts the choice of B. So $C \notin \mathrm{P}$ which proves Claim 23.

We define the language that we show to be 1-tt-mitotic, but not m-mitotic.

$$L = C \cup 0(\overline{C} \cap A)$$

Note that the union above is disjoint, since C consists of strings of length $t(n)$ while $0(\overline{C} \cap A)$ consists of strings of length $t(n) + 1$. Observe that $L \in (\mathrm{NP} \cap \mathrm{coNP}) - \mathrm{P}$.

Claim 24 *L is 1-tt-mitotic.*

The separator is $S = A$. First, we describe the 1-tt-reduction from L to $L \cap S$ on input x: If $x \notin A \cup 0A$, then reject. If $x \in A$, then accept if and only if $x \in L \cap S$. Otherwise, accept if and only if $y \notin L \cap S$ where $x = 0y$. Second, we describe the 1-tt-reduction from $L \cap S$ to $L \cap \overline{S}$ on input x: If $x \notin S$, then reject. Otherwise, accept if and only if $0x \notin L \cap \overline{S}$. Finally, we describe the 1-tt-reduction from $L \cap \overline{S}$ to L on input x: If $x \in S$, then reject. Otherwise, accept if and only if $x \in L$. This shows that L is 1-tt-mitotic.

Claim 25 *L is not m-mitotic.*

Assume L is m-mitotic. Hence L is m-autoreducible [AS84], i.e., $L \leq_{\mathrm{m}}^{\mathrm{p}} L$ via a reduction such that $f(x) \neq x$. Let p be a polynomial bounding the computation time of f. Choose the smallest number k such that for all $n \geq k$ it holds that $p(t(n)+1) < t(n+1)$. This choice is possible because

$$p(t(n)+1) \leq t(n)^d = \left(2^{2^{n^{2c}}}\right)^d = 2^{d \cdot 2^{n^{2c}}} \leq 2^{2^{d+n^{2c}}} < 2^{2^{n+n^{2c}}} \leq 2^{2^{(n+1)^{2c}}}$$

for a suitable constant $d \geq 1$. Define the finite set

$$L' =_{\mathrm{def}} \{w \mid |w| \leq t(k)+1 \text{ and } w \in L\}.$$

The following algorithm decides in polynomial time whether the input z belongs to L.

```
1. $x := z$
2. if $|x| \leq t(k)+1$ then accept if and only if $x \in L'$
3. if $|f(x)| \geq |x|$ then reject
4. $x := f(x)$, goto 2
```

The algorithm runs in polynomial time, since each iteration decreases the length of x. Also, since f is an m-autoreduction, at any time it holds that

$$z \in L \Leftrightarrow x \in L. \tag{4}$$

So if we stop in line 2, then we accept if and only if $z \in L$. It remains to argue for a stop in line 3.

Assume $z \in L$ but we reject in line 3; we will derive a contradiction. By (4), at the moment we reject, it holds that

$$x \in L \text{ and } |x| \geq t(k)+1 \tag{5}$$

In particular, $x \in A \cup 0A$, i.e., $x = 0^{t(n)}$ or $x = 0^{t(n)+1}$ for a suitable n. By definition,

$$0^{t(n)} \in L \Leftrightarrow 0^{t(n)+1} \notin L.$$

It follows that $f(x) \neq 0^{t(n)}$ and $f(x) \neq 0^{t(n)+1}$, since otherwise either $f(x) = x$ or $(0^{t(n)} \in L \Leftrightarrow 0^{t(n)+1} \in L)$. Note that $n \geq k$, since otherwise $|x| \leq t(n)+1 < t(k)+1$ which contradicts (5). Therefore, by the choice of k,

$$|f(x)| \leq p(|x|) \leq p(t(n)+1) < t(n+1).$$

However, besides x there are no words in L that have a length in $[t(n), t(n+1)-1]$. It follows that $|f(x)| < |x|$, since $f(x)$ must belong to L. This contradicts our assumption that we reject in line 3. Therefore, if we stop in line 3, then $z \notin L$. So the algorithm above decides L in polynomial time. This is a contradiction. So L is not m-mitotic. □

Theorem 7 *If* EEE $\neq$ NEEE, *then there exists* $C \in$ NP $-$ P *such that* C *is not T-autoreducible.*

Proof. Choose $B \in \mathrm{NEEE}-\mathrm{EEE}$. So there exists a constant $c \geq 1$ such that B is decidable in nondeterministic time $2^{2^{2^{c \cdot n}}}$. Let $t(x) =_{\mathrm{def}} 2^{2^{x^{2c}}}$ be a tower function and let $A =_{\mathrm{def}} \{0^{t(n)} \,|\, n \geq 0\}$ and $C =_{\mathrm{def}} \{0^{t(x)} \,|\, x \in B\}$. Note that $A \in \mathrm{P}$. By the same argument as in the proof of Claim 23 we obtain $C \in \mathrm{NP} - \mathrm{P}$.

We will now show that the set C is not T-autoreducible. Let us assume that C is T-autoreducible. So there exists a deterministic polynomial time oracle Turing-machine M' such that $L(M'^{C}) = C$. Furthermore, it holds for all x that during its work on input x, M' never queries the oracle C for x.

Let $k \geq 0$ such that the running-time of M' on inputs of length $n \geq 1$ is bounded by the polynomial n^k. Observe that $t(n)^k <_{\mathrm{ae}} t(n+1)$. More precisely,

$$\big(n > \log(k)-1\big) \Longrightarrow t(n)^k = (2^{2^{n^{2c}}})^k < t(n+1) = 2^{2^{(n+1)^{2c}}}. \tag{6}$$

Let $\log(k) \leq m$, and assume that M' is running on input $0^{t(m)}$. Since M' is an oracle machine, it can query C for a string q. Observe that such a query q can have length at most $t(m)^k$. We can assume that M' queries C only for strings from A (i.e. strings of the form $0^{t(i)}$ for $i \geq 0$). As $C \subseteq A$, these are the only queries that have a chance of getting a positive answer from C. Notice that M' is not allowed to query C for $0^{t(m)}$ because M' proves that C is T-autoreducible. Furthermore, due to (6), M' on input $0^{t(m)}$ cannot query C for $0^{t(m+1)}$ or longer strings. So M' on input $0^{t(m)}$ can only query C for strings in $\{0^{t(i)} \,|\, 0 \leq i < m\}$.

We construct a deterministic polynomial-time Turing-machine M such that $L(M) = C$. On input x, M first checks whether $x \in A$, i.e., whether $x = 0^{t(n)}$ for some $n \geq 0$. If no such n exists, M rejects. Since this can easily be done in polynomial time, we assume that there exists an $n \geq 0$ such that M is running on input $0^{t(n)}$. We define

$$E[i] = \begin{cases} 1, \text{ if } 0^{t(i)} \in C \\ 0, \text{ if } 0^{t(i)} \notin C. \end{cases}$$

M will compute $E[0], E[1], \ldots, E[n]$ one after another and accept the input $0^{t(n)}$ if and only if $E[n] = 1$. Since k is a constant, we can encode $E[0], E[1], \ldots, E[\log(k) - 1]$ into the program of M.

During its work on input $0^{t(n)}$, M will simulate M'. Notice that while M' is equipped with oracle C, M is not an oracle machine and hence cannot query an oracle while simulating M'. Instead, M will make use of the values $E[0], E[1], \ldots$ it has computed so far to answer possible oracle queries of M'.

Let $\log(k) \leq m \leq n$. We now describe how M computes $E[m]$ if it has access to $E[0], E[1], \ldots, E[m-1]$.

```
Subroutine compute_E[m];
1. Compute $0^{t(m)}$.
2. Simulate $M'$ on input $0^{t(m)}$. For every oracle query q of $M'$ on input
   $0^{t(m)}$, proceed as follows:
  (a) Compute $j \geq 0$ such that $q = 0^{t(j)}$.          // Note that $j < m$.
  (b) If $E[j] = 0$, continue the simulation of $M'$ with a negative answer
      to query q. If $E[j] = 1$, continue the simulation of $M'$ with a
      positive answer to query q.
```

```
3. If M' accepts, set E[m] := 1, else set E[m] := 0.
```

From our above argumentation it follows that for $0 \leq i \leq n$, the algorithm computes $E[i]$ correctly if it has access to $E[0], \ldots, E[i-1]$. Since M is running on input $0^{t(n)}$ and computes $E[0], E[1], \ldots, E[n]$ one after another, M clearly is a polynomial time machine and it holds that $L(M) = C$.

This proves $C \in \mathrm{P}$, which contradicts our assumption. Hence, such machine M' cannot exist. So C is not T-autoreducible. □