Redundancy in Complete Sets*

Christian Glaßer[1], A. Pavan[2**], Alan L. Selman[3***], and Liyu Zhang[3]

[1] Universität Würzburg, glasser@informatik.uni-wuerzburg.de
[2] Iowa State University, pavan@cs.iastate.edu
[3] University at Buffalo, {selman,lzhang7}@cse.buffalo.edu

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. [12], 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.

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

1 Introduction

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

* A full version of this paper is available as ECCC Technical Report TR05-068.
** Research supported in part by NSF grants CCCF-0430807.
*** Research supported in part by NSF grant CCR-0307077.

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

Trakhtenbrot [18] defined a set A to be *autoreducible* if there is an oracle Turing machine M such that $A = L(M^A)$ and M on input x never queries x. For complexity classes like NP and PSPACE refined measures are needed. In this spirit, Ambos-Spies [2] defined the notion of polynomial-time autoreducibility and the more restricted form m-autoreducibility. A set A is *polynomial-time autoreducible* if it is autoreducible via a oracle Turing machine that runs in polynomial-time. A is *m-autoreducible* if A is polynomial-time many-one reducible to A via a function f such that $f(x) \neq x$ for every x. Both notions demand information contained in $A(x)$ to be present among strings different from x. In the case of m-autoreducibility, the redundancy in A is even more apparent—if a set A is m-autoreducible, then x and $f(x)$ have the same information about A.

A stronger form of redundancy is described by the notion of *mitoticity* which was introduced by Ladner [15] for the recursive setting and by Ambos-Spies [2] for the polynomial-time setting. A set A is *m-mitotic* if there is a set $S \in \mathrm{P}$ such that A, $A \cap S$, and $A \cap \overline{S}$ are polynomial-time many-one equivalent. Thus if a set is m-mitotic, then A can be split into two parts such that both parts have exactly the same information as the original set has.

Ambos-Spies [2] showed that if a set is m-mitotic, then it is m-autoreducible and he raised the question of whether the converse holds. In this paper we resolve this question and show that every m-autoreducible set is m-mitotic. This is our main result. Since its proof is very involved, we present our main idea with help of a simplified graph problem which will be described in Section 3. This simplification drops many of the important details from our formal proof, but still captures the spirt of the core problem. Our main result is all the more surprising, because it is known [2] that polynomial-time T-autoreducibility does not imply polynomial-time T-mitoticity. We improve this and disprove the equivalence between autoreducibility and mitoticity for all polynomial-time-bounded reducibilities between 3-tt-reducibility and Turing-reducibility: There exists a sparse set in EXP that is polynomial-time 3-tt-autoreducible, but not weakly polynomial-time T-mitotic. In particular, polynomial-time T-autoreducible does not imply polynomial-time weakly T-mitotic. This result settles another open question raised by Buhrman and Torenvliet [9].

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

As a consequence of this result and recent work of Glaßer et al. [13, 12], we can show that all complete sets for many interesting classes such as NP, PSPACE, NEXP, and levels of PH are m-mitotic. Thus they all contain redundant information in a strong sense. This resolves several long standing open questions raised by Ambos-Spies [2], Buhrman, Hoene, and Torenvliet [8], and Buhrman and Torenvliet [9].

Our result can also be viewed as a step towards understanding the isomorphism conjecture [5]. This conjecture states that all NP-complete sets are isomorphic to each other. In spite of several years of research, we do not have any concrete evidence either in support or against the isomorphism conjecture[4]. It is easy to see that if the isomorphism conjecture holds for classes such as NP, PSPACE, and EXP, then complete sets for these classes are m-autoreducible as well as m-mitotic. Given our current inability to make progress about the isomorphism conjecture, the next best thing we can hope for is to make progress on statements that the isomorphism conjecture implies. We note that this is not an entirely new approach. For example, if the isomorphism conjecture is true, then NP-complete sets cannot be sparse. This motivated researchers to consider the question of whether complete sets for NP can be sparse. This line of research led to the beautiful results of Mahaney [16] and Ogiwara and Watanabe [17] who showed that complete sets for NP cannot be sparse unless P = NP. Our results show that another consequence of isomorphism, namely "NP-complete sets are m-mitotic" holds. Note that this is an unconditional result.

Buhrman et al. [7] and Buhrman and Torenvliet [10, 11] 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 [11] for more details.

In Section 4, we extend the notion of autoreducibility and define *poly-autoreducibility*. A motivation for this is to understand the isomorphism conjecture and the notion of paddability. Recall that the isomorphism conjecture is true if and only if all NP-complete sets are paddable. Paddability implies the following: If L is paddable, then given x and a polynomial p, we can produce $p(|x|)$ distinct strings such that if x is in L, then all these strings are in L and if x is not in L, then none of these strings are in L. Autoreducibility implies that given x we can produce a *single* string y different from x such that $L(x) = L(y)$. A natural question that arises is whether we can produce more strings whose membership in L is the same as the membership of x in L. This leads us to the notion of f-autoreducibility: A set L is f-autoreducible, if there is a polynomial-time algorithm that on input x outputs $f(|x|)$ distinct strings (different from x)

[4] 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.

whose membership in L is the same as the membership of x in L. It is obvious that paddability implies poly-autoreducibility. The question of whether "NP complete sets are poly-autoreducible" is weaker than the question of whether "NP-complete sets are paddable."

We provide evidence for poly-autoreducibility of NP-complete sets. We show that if one-way permutations exist, then NP-complete sets are log-autoreducible. Moreover, if one-way permutations and quick pseudo-random generators exist, then NP-complete sets are poly-autoreducible. We also show that if NP-complete sets are poly-autoreducible, then they have infinite subsets that can be decided in linear-exponential time. A complete version of this paper can be found at ECCC [14].

1.1 Previous Work

The question of whether complete sets for various classes are autoreducible has been studied extensively [19, 4, 7]. Beigel and Feigenbaum [4] showed that Turing complete sets for the classes that form the polynomial hierarchy, $\Sigma_i^{\mathrm{P}}, \Pi_i^{\mathrm{P}}$, and Δ_i^{P}, are Turing autoreducible. Thus, all Turing complete sets for NP are Turing autoreducible. Buhrman et al. [7] 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. [7] showed that truth-table complete sets for NP are probabilistic truth-table autoreducible. Recently, Glaßer et al. [13, 12] showed that complete sets for classes such as NP, PSPACE, Σ_i^{P} are m-autoreducible.

Buhrman, Hoene, and Torenvliet [8] showed that EXP complete sets are weakly many-one mitotic. This result was recently improved independently by Kurtz [11] and Glaßer et al. [13, 12]. Buhrman and Torenvliet [11] 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 [6]. Glaßer et al. also showed that NEXP complete sets are weakly m-mitotic and PSPACE-complete sets are weak Turing-mitotic.

2 Preliminaries

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

Definition 1 ([2]). *A set A is* polynomially T-autoreducible *(T-autoreducible, for short) if there exists a polynomial-time-bounded oracle Turing machine M such that $A = L(M^A)$ and for all x, M on input x never queries x. A set A is* polynomially m-autoreducible *(m-autoreducible, for short) if $A \leq_m^p A$ via a reduction function f such that for all x, $f(x) \neq x$.*

Definition 2 ([2]). *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 3 ([2]). *A recursive set A is* polynomial-time weakly T-mitotic *(weakly T-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and $A \equiv_T^p A_0 \equiv_T^p A_1$. A is* polynomial-time weakly m-mitotic *(weakly m-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and $A \equiv_m^p A_0 \equiv_m^p A_1$.*

Definition 4. *Let f be a function from $\mathbb{N}$ to $\mathbb{N}$. A set L is* f-autoreducible*, if there is a polynomial-time algorithm $\mathcal{A}$ that on input x outputs $y_1, y_2, \cdots, y_m$ such that $f(|x|) = m$, if $x \in L$, then $\{y_1, y_2, \cdots, y_m\} \subseteq L$, and if $x \notin L$, then $\{y_1, y_2, \cdots, y_m\} \cap L = \emptyset$. A set is* poly-autoreducible*, if it is n^k-autoreducible for every $k \geq 1$.*

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

3 m-Autoreducibility equals m-Mitoticity

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

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

Theorem 1. *Let L be any set such that $|\overline{L}| \geq 2$. L is m-autoreducible if and only if L is m-mitotic.*

We mention the main ideas and the intuition behind the proof and describe the combinatorial core of the problem.

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

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

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

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

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

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

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

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

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

We exhibit a labeling strategy with these properties. To define this labeling, we use the following *distance function*: $d(x, y) \stackrel{df}{=} \lfloor \log |y - x| \rfloor$ (our formal proof

uses a variant of this function). The core problem is solved by the following local strategy.

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

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

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

Call a set L *nontrivial* if $\|L\| \geq 2$ and $\|\overline{L}\| \geq 2$. We have the following corollaries of the main theorem.

Corollary 1. *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. [12] showed that all many-one complete sets of the above classes are m-autoreducible. By Theorem 1, these sets are m-mitotic. □

Corollary 2. *A nontrivial set L is* NP*-complete if and only if L is the union of two disjoint* P*-separable* NP*-complete sets.*

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

Ambos-Spies [2] defined a set A to be *ω-m-mitotic* if for every $n \geq 2$ 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. *Every nontrivial infinite set that is many-one complete for a class mentioned in Corollary 1 is ω-m-mitotic.*

We note that the proof of the main theorem actually yields the following theorem.

Theorem 2. *Every 1-tt-autoreducible set is 1-tt-mitotic.*

The following theorem shows in a strong way that T-autoreducible does not imply weakly T-mitotic. Hence, our main theorem cannot be generalized.

Theorem 3. *There exists $L \in \mathrm{SPARSE} \cap \mathrm{EXP}$ such that*

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

Thus there exist 3-tt-autoreducible sets that are not even T-mitotic, whereas every 1-tt-autoreducible set is 1-tt mitotic. We do not know what happens when we consider 2-tt reductions. Is every 2-tt-autoreducible set 2-tt-mitotic or does there exist a 2-tt-autoreducible set that is not 2-tt-mitotic? We leave this as an open question.

4 Poly-Autoreducibility

In this section we consider the question of whether NP-complete sets are f-autoreducible, for some growing function f.

Lemma 1. *Let L be an* NP*-complete language. For every polynomial $q(.)$ there is a polynomial-time algorithm $\mathcal{A}$ such that $\mathcal{A}$ on input x, $|x| = n$,*

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

This above lemma comes close to showing that NP-complete sets are poly-autoreducible, except for a small caveat. Let L be any NP-complete language. If the algorithm from Lemma 1 neither accepts x or rejects x, then it produces polynomially many equivalent strings. However, to show L is poly-autoreducible, we must produce polynomially-many equivalent strings even when the algorithm accepts or rejects.

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

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

Definition 5. *Let f be a many-one reduction from A to B. We say f is $g(n)$-sparse, if for every n, no more than $g(n)$ strings of length n are mapped to a single string via f.*

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

Lemma 3. *Let L be* NP*-complete. If one-way permutations exist, then there exists a polynomial-time algorithm that on input 0^n outputs $\log n$ distinct strings in L and $\log n$ strings out of L.*

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

Lemma 4. *Let L be* NP*-complete. Assume one-way permutations exist. For every polynomial q, there exists a polynomial-time probabilistic algorithm $\mathcal{B}$ that on input 0^n outputs $q(n)$ distinct strings from L and $q(n)$ distinct strings from $\overline{L}$ with high probability.*

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

Lemma 5. *Let L be any* NP*-complete language. If one-way permutations and quick pseudo-random generators exist, then for every polynomial $q(n)$, there is a polynomial-time algorithm that on input 0^n outputs $q(n)$ many distinct strings from L and $q(n)$ many distinct strings out of L.*

Combining Lemmas 1 and 3, we obtain the following result.

Theorem 4. *If one-way permutations exist, then every* NP*-complete language is $\log n$-autoreducible.*

Combining Lemmas 1 and 5, we obtain the following result.

Theorem 5. *If one-way permutations and quick pseudo-random generators exist,* NP*-complete sets are poly-autoreducible.*

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

Theorem 6. *If there exists a* UP *machine M that accepts 0^* such that no* P*-machine can compute infinitely many accepting computations of $M(0^n)$, then* NP*-complete sets are poly-autoreducible.*

Next we consider the possibility of an unconditional proof that NP-complete sets are poly-autoreducible. We relate this with the notion of immunity. We show that if NP-complete sets are poly-autoreducible, then they are not E-immune. It is known that NP-complete sets are not generic [12]. This proof is based on the fact that NP-complete sets are autoreducible. Genericity is stronger notion than immunity, i.e., if a language L is not immune, then it can not be generic. Our result says that improving the autoreducibility result for NP-complete sets gives a stronger consequence—namely they are not immune.

Theorem 7. *If every* NP*-complete set is poly-autoreducible, then no* NP*-complete set is* E*-immune.*

References

1. M. Agrawal. Pseudo-random generators and structure of complete degrees. In *17th Annual IEEE Conference on Computational Complexity*, pages 139–145, 2002.
2. K. Ambos-Spies. P-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines, Lecture Notes in Computer Science 177*, pages 1–23. Springer-Verlag, 1984.
3. J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Mathematical Systems Theory*, 18(1):1–10, June 1985.
4. R. Beigel and J. Feigenbaum. On being incoherent without being very hard. *Computational Complexity*, 2:1–17, 1992.
5. L. Berman and J. Hartmanis. On isomorphism and density of NP and other complete sets. *SIAM Journal on Computing*, 6:305–322, 1977.
6. H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet. Separating complexity classes using autoreducibility. *SIAM Journal on Computing*, 29(5):1497–1520, 2000.
7. H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet. Using autoreducibility to separate complexity classes. *SIAM Journal on Computing*, 29(5):1497–1520, 2000.
8. H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness, and structure of complete sets. *SIAM Journal on Computing*, 27:637–653, 1998.
9. H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proceedings 9th Structure in Complexity Theory*, pages 118–133, 1994.
10. H. Buhrman and L. Torenvliet. Separating complexity classes using structural properties. In *Proceedings of the 19th IEEE Conference on Computational Complexity*, pages 130–138, 2004.
11. H. Buhrman and L. Torenvliet. A Post's program for complexity theory. *Bulleting of the EATCS*, 85:41–51, 2005.

12. 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*, volume 3618 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 2005.
13. C. Glaßer, M. Ogihara, A. Pavan, A. L. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. Technical Report TR05-11, ECCC, 2005.
14. C. Glaßer, A. Pavan, A. L. Selman, and L. Zhang. Redundancy in complete sets. Technical Report 05-068, Electronic Colloquium on Computational Complexity (ECCC), 2005.
15. R. Ladner. Mitotic recursively enumerable sets. *Journal of Symbolic Logic*, 38(2):199–211, 1973.
16. S. Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and Systems Sciences*, 25(2):130–143, 1982.
17. M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal of Computing*, 20(3):471–483, 1991.
18. B. Trakhtenbrot. On autoreducibility. *Dokl. Akad. Nauk SSSR*, 192, 1970. Translation in Soviet Math. Dokl. 11: 814– 817, 1970.
19. A. Yao. Coherent functions and program checkers. In *Proceedings of the 22n Annual Symposium on Theory of Computing*, pages 89–94, 1990.