Autoreducibility, Mitoticity, and Immunity*

Christian Glaßer†, Mitsunori Ogihara‡, A. Pavan§, Alan L. Selman¶, Liyu Zhang‖

October 24, 2006

Abstract

We show the following results regarding complete sets.

- NP-complete sets and PSPACE-complete sets are polynomial-time many-one autoreducible.
- Complete sets of any level of PH, MODPH, or the Boolean hierarchy over NP are polynomial-time many-one autoreducible.
- EXP-complete sets are polynomial-time many-one mitotic.
- If there is a tally language in $\mathrm{NP} \cap \mathrm{coNP} - \mathrm{P}$, then, for every $\epsilon > 0$, NP-complete sets are not $2^{n(1+\epsilon)}$-immune.

These results solve several of the open questions raised by Buhrman and Torenvliet in their 1994 survey paper on the structure of complete sets.

1 Introduction

We solve several open questions identified by Buhrman and Torenvliet in their 1994 survey paper on the structure of complete sets [BT94]. It is important to study the computational structure of complete sets, because they, by reductions of all the sets in the class to the complete sets, represent all of the structure that a class might have. Here we focus attention primarily on autoreducibility, mitoticity, and immunity.

Trakhtenbrot [Tra70] introduced the notion of autoreducibility in a recursion theoretic setting. A set A is *autoreducible* if there is an oracle Turing machine M such that $A = L(M^A)$ and M on input x never queries x. Ladner [Lad73a] showed that there exist Turing complete recursively enumerable sets that are not autoreducible. Ambos-Spies [AS84] introduced the polynomial-time variant of autoreducibility, where we require the oracle Turing machine to run in polynomial time. Yao [Yao90] introduced the notion of coherence, which coincides with probabilistic polynomial-time autoreducibility.

*A preliminary version of this paper appeared in MFCS 2005 [GOP+05]

†Lehrstuhl für Theoretische Informatik, Universität Würzburg. Email: glasser@informatik.uni-wuerzburg.de.

‡Department of Computer Science, University of Rochester. Email:ogihara@cs.rochester.edu. Research supported in part by NSF grants EIA-0080124, EIA-0205061, and NIH grant P30-AG18254

§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, University at Buffalo. Email:lzhang7@cse.buffalo.edu

The question of whether complete sets for various classes are polynomial-time autoreducible has been studied extensively [Yao90, BF92, BFvMT00], and is currently an area of active research [BT05]. Beigel and Feigenbaum [BF92] showed that polynomial-time Turing complete sets for the classes that form the polynomial-time hierarchy, $\Sigma_i^{\mathrm{P}}, \Pi_i^{\mathrm{P}}$, and Δ_i^{P}, are polynomial-time Turing autoreducible. Thus, all polynomial-time Turing complete sets for NP are polynomial-time Turing autoreducible. Buhrman et al. [BFvMT00] showed that polynomial-time Turing complete sets for EXP and Δ_i^{EXP} are polynomial-time autoreducible, whereas there exists a polynomial-time Turing complete set for EESPACE that is not polynomial-time Turing autoreducible. They showed that answering questions about polynomial-time autoreducibility of intermediate classes results in interesting separation results. Regarding NP, Buhrman et al. [BFvMT00] showed that all polynomial-time truth-table complete sets for NP are probabilistic polynomial-time truth-table autoreducible. Thus, all NP-complete sets are probabilistic polynomial-time truth-table autoreducible.

Buhrman and Torenvliet [BT94] asked whether all NP-complete sets are polynomial-time many-one autoreducible and whether all PSPACE-complete sets are polynomial-time many-one autoreducible. We resolve these questions positively: all nontrivial NP-complete sets and PSPACE-complete sets are (unconditionally) polynomial-time many-one autoreducible. Also we show that for each class in MODPH [HLS$^+$93] (the hierarchy constructed by applying to P a constant number of operators chosen from $\{\exists\cdot, \forall\cdot\} \cup \{\mathrm{MOD}_k\cdot, \mathrm{coMOD}_k\cdot \mid k \geq 2\}$), all of its nontrivial polynomial-time many-one complete sets are polynomial-time many-one autoreducible. ¿From this we obtain that no polynomial-time many-one complete sets for NP, PSPACE, or for the classes of MODPH are $2n$-generic.

Now we briefly discuss mitoticity. Informally, a set is *weakly mitotic* if it can be partitioned into two equivalent parts each of which is equivalent to the original set. Thus, both parts contain the same information as the original set. If additionally the partition is easy to compute (i.e., decidable in the recursive setting and in P in the polynomial-time setting), then the set is *mitotic*. Lachlan [Lac67] introduced and studied mitoticity of recursively enumerable sets. This notion was studied comprehensively by Ladner [Lad73a, Lad73b] who showed that with respect to r.e. sets, autoreducibility and mitoticity coincide. Ambos-Spies [AS84] formulated two notions in the polynomial time setting, polynomial-time mitoticity and weak polynomial-time mitoticity. Also, he showed that every polynomial-time mitotic set is polynomial-time autoreducible. Buhrman, Hoene, and Torenvliet [BHT98] showed that every polynomial-time many-one complete set for EXP is weakly polynomial-time many-one mitotic. We improve this by showing that every polynomial-time many-one complete set for EXP is indeed polynomial-time many-one mitotic. Buhrman and Torenvliet [BT05] mention that this result was obtained by Kurtz independently. We remark that a forthcoming paper [GPSZ06] shows that a set is polynomial-time many-one autoreducible if and only if it is polynomial-time many-one mitotic.

Autoreducible sets can be thought of as sets having local redundant information. For example, if A is polynomial-time many-one autoreducible by the reduction f, then x and $f(x)$ both contain the same information concerning whether x belongs to A. However, a mitotic set contains global redundant information. To repeat, if L is mitotic it can be partitioned into two equivalent parts and both parts contain the same information as L.

In Section 5, we study the question of whether NP-complete sets have easy subsets. The most natural notion of having an easy subset would mean having an infinite subset in P. An infinite set is P-immune if no infinite subset belongs to P. Berman [Ber77] showed that EXP-complete sets are not P-immune, and Tran [Tra95] showed that NEXP-complete sets are not P-immune.

However, we do not have such unconditional results for NP-complete sets, since these would imply $\mathrm{P} \neq \mathrm{NP}$. Glaßer et al. [GPSS06] showed that if polynomial-time one-way permutations exist, then NP-complete sets are not 2^{n^ϵ}-immune. Here we provide another partial result in this direction. In Section 5 we show that if there exists a tally language in $\mathrm{NP} \cap \mathrm{coNP} - \mathrm{P}$, then every NP-complete set includes an infinite subset that is recognizable in time $2^{n(1+\epsilon)}$.

Section 6 concludes this paper with a result on the robustness of polynomial-time Turing complete sets for NP. Schöning [Sch86] raised the following question: If a small amount of information is removed from a complete set, does the set remain hard? Tang, Fu, and Liu [TFL93] proved the existence of a sparse set S such that for every polynomial-time many-one complete set A for EXP, $A-S$ is not polynomial-time many-one hard for EXP. Buhrman and Torenvliet [BT04] proved that if A is a polynomial-time Turing complete sets for EXP and S is a log-dense sets in P, then $A-S$ remains polynomial-time Turing complete sets for EXP. We prove that if a set A is polynomial-time Turing complete for NP and S is a log-dense set in P, then $A-S$ remains polynomial-time Turing complete for NP. This result is easier to prove than Buhrman and Torenvliet's similar result.

2 Preliminaries

We use standard notation and assume familiarity with standard resource-bounded reductions. We consider words in lexicographic order. All reductions in the paper are polynomial-time computable. In particular, we write "autoreducible" to mean "polynomial-time autoreducible" and we write "mitotic" to mean "polynomial-time mitotic." Moreover, m-reduction (resp., tt-reduction, T-reduction) is an abbreviation for polynomial-time many-one reduction (resp., polynomial-time truth-table reduction, polynomial-time Turing reduction). From this we derive the abbreviations m-complete, tt-complete, and T-complete. For a class $\mathcal{C}$, we write $\mathcal{C}$-complete to mean polynomial-time many-one complete for $\mathcal{C}$.

Definition 2.1 ([AS84]). A set A is *polynomial-time Turing 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 many-one autoreducible* (m-autoreducible, for short) if $A \leq_m^p A$ via a reduction function f such that for all x, $f(x) \neq x$.

In the following definition we consider reductions $\leq_r^p$ where $r \in \{m, T\}$.

Definition 2.2 ([AS84]). A set A is *polynomial-time r-mitotic* (r-mitotic for short) if there exists a set $B \in \mathrm{P}$ such that

$$A \equiv_r^p A \cap B \equiv_r^p A \cap \overline{B}.$$

A 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

$$A \equiv_r^p A_0 \equiv_r^p A_1.$$

In general, r-autoreducible sets are sets that are autoreducible with respect to $\leq_r^p$-reductions. The same convention is used for mitotic sets.

A language is $\mathrm{DTIME}(T(n))$-*complex* if L does not belong to $\mathrm{DTIME}(T(n))$ 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 [BS85] 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 Autoreducibility

Since EXP-complete sets are complete with respect to length-increasing reductions [Ber77], they are m-autoreducible. Ganesan and Homer [GH92] showed that NEXP-complete sets are complete under 1-1 reductions. This implies that all NEXP-complete sets are also m-autoreducible. To see this, consider a 1-1 reduction from $0L \cup 1L$ to L, where L is the given NEXP-complete set. These techniques cannot be applied to NP-complete sets, as we do not know any unconditional result on the degree structure of NP-complete sets. Some partial results are known for NP-complete sets. Beigel and Feigenbaum [BF92] showed that T-complete sets for NP are T-autoreducible. Buhrman et al. [BFvMT00] showed that all tt-complete sets for NP are probabilistic tt-autoreducible. It has not been known whether NP-complete sets are m-autoreducible. Buhrman and Torenvliet raised this question in their survey papers [BT94, BT98]. Below, we resolve this question.

Note that singletons and complements of singletons cannot be m-autoreducible. Therefore, in connection with m-autoreducibility, a set L is called *nontrivial* if $|L| > 1$ and $|\overline{L}| > 1$.

Theorem 3.1. *All nontrivial* NP*-complete sets are m-autoreducible.*

Proof. Let L be NP-complete and let M be a nondeterministic machine that accepts L. For a suitable polynomial p we can assume that on input x, all computation paths of M have length $p(|x|)$. Since L is nontrivial, there exist different words $y_1, y_2 \in L$ and $\overline{y}_1, \overline{y}_2 \in \overline{L}$. We use the left set technique of Ogiwara and Watanabe [OW91]. Let

$$\begin{aligned}\mathrm{Left}(L) \stackrel{df}{=} \{\langle x, u\rangle \,\big|\, & |u| = p(|x|) \text{ and } \exists v, |v| = |u|, \text{such that} \\ & u \le v \text{ and } M(x) \text{ accepts along path } v\}.\end{aligned}$$

Since Left(L) is in NP, there is a function $f \in$ PF that reduces Left(L) to L. The algorithm below defines function g which is an m-autoreduction for L. Let x be an input. Define $n \stackrel{df}{=} |x|$ and $m \stackrel{df}{=} p(|x|)$.

```
1  if f(⟨x, 0^m⟩) ≠ x then output f(⟨x, 0^m⟩)
2  if f(⟨x, 1^m⟩) = x then
3     if M(x) accepts along 1^m then
4        output a string from {y1, y2} − {x}
5     else
6        output a string from {ȳ1, ȳ2} − {x}
7     endif
8  endif
9  // here f(⟨x, 0^m⟩) = x ≠ f(⟨x, 1^m⟩)
10 determine z of length m such that f(⟨x, z⟩) = x ≠ f(⟨x, z + 1⟩)
11 if M(x) accepts along z then output a string from {y1, y2} − {x}
12 else output f(⟨x, z + 1⟩)
```

Note that step 10 is an easy binary search: Start with $z_1 := 0^m$ and $z_2 := 1^m$. Let z' be the middle element between z_1 and z_2. If $f(z') = x$ then $z_1 := z'$ else $z_2 := z'$. Again, choose the middle element between z_1 and z_2, and so on. This shows $g \in \mathrm{PF}$. Clearly, $g(x) \neq x$, so it remains to show $L \leq_m^p L$ via g.

If the algorithm stops in step 1, then

$$x \in L \Leftrightarrow \langle x, 0^m \rangle \in \mathrm{Left}(L) \Leftrightarrow g(x) = f(\langle x, 0^m \rangle) \in L.$$

If the algorithm stops in step 4 or step 6, then $f(\langle x, 0^m \rangle) = f(\langle x, 1^m \rangle)$. Hence

$$x \in L \Leftrightarrow \langle x, 1^m \rangle \in \mathrm{Left}(L) \Leftrightarrow M(x) \text{ accepts along } 1^m \Leftrightarrow g(x) \in L.$$

Assume we reach step 9. There it holds that $f(\langle x, 0^m \rangle) = x \neq f(\langle x, 1^m \rangle)$. If the algorithm stops in step 11, then $x \in L$ and $g(x) \in L$. Assume we stop in step 12. So $M(x)$ does not accept along z.

$$x \in L \Leftrightarrow f(\langle x, 0^m \rangle) = f(\langle x, z \rangle) \in L \Leftrightarrow g(x) = f(\langle x, z+1 \rangle) \in L.$$

□

One may wonder whether the analogous result holds for other polynomial-time reductions, i.e., for which polynomial-time reducibility r every nontrivial r-complete set for NP is $\leq_r$-autoreducible. An important issue in considering the question is what relation between x and the queries produced by f on input $\langle x, y \rangle$ should be tested in the places where we test whether $f(\langle x, y \rangle) = x$ in the above algorithm. It is not difficult to see that, in the case when f is a polynomial-time disjunctive-truth-table reduction, we can use the test of whether x is among the query strings produced by f on input $\langle x, y \rangle$. Thus, we have the following two corollaries.

Corollary 3.1. *For every $k \geq 1$, each nontrivial $\leq_{k\text{-}dtt}^p$-complete set for* NP *is $\leq_{k\text{-}dtt}^p$-autoreducible.*

Corollary 3.2. *All nontrivial $\leq_{dtt}^p$-complete sets for* NP *are $\leq_{dtt}^p$-autoreducible.*

Also, in the case when f is a polynomial-time 1-truth-table reduction, we can use the test of whether the query string of f on input $\langle x, y \rangle$ is x. This observation yields the following corollary.

Corollary 3.3. *All nontrivial sets that are $\leq_{1\text{-}tt}^p$-complete for* NP *are $\leq_{1\text{-}tt}^p$-autoreducible.*

Extending these results to more generalized reducibilities seems quite difficult. It is unknown whether the analog holds for $\leq_{2\text{-}tt}^p$-reductions, even for $\leq_{2\text{-}ctt}^p$-reductions. To illustrate why it is difficult, consider the $\leq_{2\text{-}ctt}^p$-reducibility. Suppose we use, as in the case for the disjunctive reducibility, the test of whether one of the two query strings of f is x. Suppose, by binary search, we arrive at a point y such that both y and $y+1$ are rejecting paths, one query string of $f(\langle x, y \rangle)$ is x, and neither of the two queries of $f(\langle x, y+1 \rangle)$ are x. Let u be the other query of $f(\langle x, y \rangle)$ and let v and w be the queries of $f(\langle x, y+1 \rangle)$. The information gain by discovering this y is that: if $x \in L$ then $(u \in L \iff v, w \in L)$ and that if $x \notin L$ then either v or w is a nonmember of L. This information is not enough for us to reduce the membership of x in L to membership questions about u, v, and w.

The idea in the proof of Theorem 3.1 can be generalized to prove similar results for other classes. To illustrate how the generalization can be done, consider PSPACE. Cai and Furst [CF91] based

on Barrington [Bar89] show that for each language L in PSPACE there exist a polynomial p and a polynomial-time computable function f from $\Sigma^* \times \Sigma^*$ to S_5 such that for all x

$$x \in L \iff f(x, 0^{p(|x|)}) \circ \cdots \circ f(x, 1^{p(|x|)}) = I_5,$$

where $\circ$ is the product of permutations calculated from left to right and I_5 is the identity permutation. Let L be an arbitrary nontrivial PSPACE-complete set. Choose p and f that give the above characterization for L. Consider the set A of triples $\langle x, y, \gamma \rangle$ such that $|y| = p(|x|)$, $\gamma \in S_5$, and

$$\gamma \circ f(x, y) \circ \cdots \circ f(x, 1^{p(|x|)}) = I_5.$$

This set A is in PSPACE and has the following properties similar to left sets: (i) For each legitimate triple $\langle x, y, \gamma \rangle$ such that $|y| \neq 1^{p(|x|)}$, the membership of $\langle x, y, \gamma \rangle$ in A can be related to the membership of $\langle x, y+1, \gamma' \rangle$ in A for some $\gamma' \in S_5$. (ii) The membership of $\langle x, 1^{p(|x|)}, \gamma \rangle$ in A can be determined in polynomial time. (iii) If the membership of $\langle x, 1^{p(|x|)}, \gamma \rangle$ in A is reduced to the membership of a string $y \neq x$ in L, then y can be the output of the autoreduction; if this y happens to be x and $\gamma \neq I_5$ then it must be the case that $x \notin L$. Using these three properties, we can design a binary search algorithm quite similar to that in the proof of Theorem 3.1 to find y, γ, γ' such that $\langle x, y, \gamma \rangle \in A \iff \langle x, y+1, \gamma' \rangle \in A$, $x \in L \iff \langle x, y, \gamma \rangle \in A$, and $\langle x, y+1, \gamma' \rangle \in A \iff w \in L$, where $w \neq x$. Then this w can be the output of the autoreduction.

We formalize this idea using the concept of polynomial-time bit-reductions [HLS$^+$93]. Below we show that every class that is polynomial-time bit-reducible to a regular language has the property that all of its nontrivial r-complete sets are r-autoreducible, where r is any one of $\leq^p_m$, $\leq^p_{1\text{-}tt}$, $\leq^p_{dtt}$, and $\leq^p_{k\text{-}dtt}$ for $k \geq 2$. As a corollary to this, we show that many known classes, e.g., every class in the MODPH hierarchy [HLS$^+$93], has the property that all of its nontrivial r-complete sets are r-autoreducible, where r is any one of the above reduction types.

Definition 3.1. [HLS$^+$93] A language A is polynomial-time bit-reducible to a language B if there exists a pair of polynomial-time computable functions, (f, g), $f : \Sigma^* \times \mathbf{N}^+ \to \Sigma$, $g : \Sigma^* \to \mathbf{N}$, such that for all x,

$$x \in A \Leftrightarrow f(x, 1) \cdots f(x, g(x)) \in B.$$

Next we prove a technical lemma that is needed in the proof of Theorem 3.2.

Lemma 3.1. *Let $B \notin \{\emptyset, \Sigma^*\}$ be regular. Let $M = (Q, \{0,1\}, \delta, q_0, F)$ be a finite automaton recognizing B, where Q is its set of states, δ is its transition function, q_0 is its initial state, and F is the set of its accepting states. Assume that every state in Q is reachable from q_0. Let f and g be polynomial-time computable functions such that $f : \Sigma^* \times \mathbf{N}^+ \to \Sigma$ and $g : \Sigma^* \to \mathbf{N}$. Define*

$$A = \{\langle x, i, q \rangle \mid 1 \leq i \leq g(x) \wedge q \in Q \wedge \delta(q, f(x, i) \cdots f(x, g(x))) = q_f\}.$$

Then, A is polynomial-time bit-reducible to B.

Proof. Let B, $M = (Q, \{0,1\}, \delta, q_0, F)$, f, g, and A be as in the hypothesis of the lemma. Let $r_1, \cdots, r_t$ be an enumeration of the states in Q. Since $B \neq \emptyset$, F is a proper subset of Q. Assume, without loss of generality, that $r_1 \in Q - F$. For each j, $1 \leq j \leq t$, let $e(j)$ be a fixed word such that $\delta(q_0, e(j)) = r_j$. For each j, $1 \leq j \leq t$, let $m_j = |e(j)|$ and let $e(j,1), \ldots, e(j, m_j)$ be the bits of $e(j)$. Since $r_1 \notin F$, we have $e(1) \notin B$.

We define a bit-reduction (f', g') from A to B as follows: Let u be an input to the reduction. If u is not of the form $\langle x, i, q\rangle$ such that $1 \leq i \leq g(x)$ and $q \in Q$, u is clearly a nonmember of A. We set $g'(u) = m_1$ and for each k, $1 \leq k \leq m_1$, $f'(u, k) = e(1, k)$. We have $f'(u, 1) \cdots f(u, g'(u)) = e(1)$. Since $e(1) \notin B$, the reduction works correctly on u.

If u is of the form $\langle x, i, r_j\rangle$ such that $1 \leq i \leq g(x)$ and $1 \leq j \leq t$, then we set $g'(u) = m_j + g(x) - i + 1$ and for each k, $1 \leq k \leq g'(u)$,

$$f'(u,k) = \begin{cases} e(j,k) & \text{if } k \leq m_j, \\ f(x, k - m_j + i - 1) & \text{otherwise.} \end{cases}$$

It holds that $f'(u, 1) \cdots f'(u, g'(u)) = e(j)f(x, i) \cdots f(x, g(x))$. Since $\delta(q_0, e(j)) = r_j$, we have

$$\delta(q_0, e(j)f(x, i) \cdots f(x, g(x))) = \delta(r_j, f(x, i) \cdots f(x, g(x))).$$

So,

$$\delta(q_0, e(j)f(x, i) \cdots f(x, g(x))) \in F \Leftrightarrow \delta(r_j, f(x, i) \cdots f(x, g(x))) =\in F.$$

Thus, $f'(u, 1) \cdots f(u, g'(u)) \in B \Leftrightarrow u \in A$. The reduction thus works correctly on u.

It is easy to see that both f' and g' are polynomial-time computable. Thus, A is polynomial-time bit-reducible to B. □

Theorem 3.2. *Let r be one of the following reducibilities: $\leq^p_m$, $\leq^p_{1\text{-}tt}$, $\leq^p_{dtt}$, and $\leq^p_{k\text{-}dtt}$ for $k \geq 2$. Let $B \notin \{\emptyset, \Sigma^*\}$ be a regular language recognized by a finite automaton $M = (Q, \{0, 1\}, \delta, q_0, F)$. Let $\mathcal{C}$ be the polynomial-time bit-reduction closure of B. Then each nontrivial r-complete set for $\mathcal{C}$ is r-autoreducible.*[1]

Proof. We will prove only the case when $r =\leq^p_m$. The other cases can be proven similarly. Let B, M, and $\mathcal{C}$ be as in the hypothesis of the theorem and let L be an arbitrary m-complete language for $\mathcal{C}$. Let (f, g) be a polynomial-time bit-reduction from L to B. Define A as in Lemma 3.1; that is, A is the set of all $\langle x, i, q\rangle$ such that $1 \leq i \leq g(x)$, $q \in Q$, and $\delta(q, f(x, i) \cdots f(x, g(x))) \in F$. By Lemma 3.1, $A \in \mathcal{C}$, so A is m-reducible to L. Let h be an m-reduction from A to L. We will design an m-autoreduction, s, of L. Let a_1 and a_2 be fixed members of L and let b_1 and b_2 be fixed nonmembers of L. Let x be an input to s.

Let $y_0 = h(\langle x, 1, q_0\rangle)$. Since h is an m-reduction from A to L, and since $x \in L$ if and only if $\langle x, 1, q_0\rangle \in A$, we have $y_0 \in L \Leftrightarrow x \in L$. So, in the case where $y_0 \neq x$, we can set $s(x) = y_0$.

So, suppose that $x = y_0$. Suppose that there is some $q \in Q$ such that $h(\langle x, g(x), q\rangle) = x$. By definition $\langle x, g(x), q\rangle \in A \Leftrightarrow \delta(q, f(x, g(x))) \in F$. Since both f and g are polynomial-time computable, we can test in polynomial time whether $\langle x, g(x), q\rangle \in A$. Since h is an m-reduction from A to L, $\langle x, g(x), q\rangle \in A \Leftrightarrow x \in L$. So, in polynomial time we can test whether $x \in L$. If $x \in L$, then we set $s(x)$ to a string from $\{a_1, a_2\} - \{x\}$, otherwise we set $s(x)$ to a string from $\{b_1, b_2\} - \{x\}$.

Now, suppose that $x \notin \{h(\langle x, g(x), q\rangle) \mid q \in Q\}$. Since $x = y_0 = h(\langle x, 1, q_0\rangle)$, we have that $x \in \{h(\langle x, 1, q\rangle) \mid q \in Q\}$. By binary search over the interval $[1, g(x)]$ we can find some i,

[1] In terms of leaf languages, this theorem reads as follows: If B is a regular language different from $\emptyset$ and Σ^*, then nontrivial m-complete sets for $\mathrm{Leaf}^p_b(B)$ are m-autoreducible. (The latter denotes the balanced leaf-language class defined by B [HLS+93, Wag04].)

$1 \leq i \leq g(x) - 1$, such that $x \in \{h(\langle x, i, q\rangle) \mid q \in Q\}$ and $x \notin \{h(\langle x, i+1, q\rangle) \mid q \in Q\}$. Let $q \in Q$ be such that $x = h(\langle x, i, q\rangle)$. We have $x \in L \Leftrightarrow \langle x, i, q\rangle \in A$. By definition,

$$\langle x, i, q\rangle \in A \Leftrightarrow \delta(q, f(x,i) \cdots f(x, g(x))) = q_f.$$

Let $q' = \delta(q, f(x,i))$ and let $y_1 = h(\langle x, i+1, q'\rangle)$. By the choice of i, $x \neq y_1$. Also, $y_1 \in L \Leftrightarrow \langle x, i+1, q'\rangle \in A$. We have

$$\begin{aligned} \delta(q, f(x,i) \cdots f(x, g(x))) &= \delta(\delta(q, f(x,i)), f(x, i+1) \cdots f(x, g(x))) \\ &= \delta(q', f(x, i+1) \cdots f(x, g(x))). \end{aligned}$$

So,

$$\delta(q, f(x,i) \cdots f(x, g(x))) = q_f \Leftrightarrow \delta(q', f(x, i+1) \cdots f(x, g(x))) \in F.$$

This implies that

$$\langle x, i, q\rangle \in A \Leftrightarrow \langle x, i+1, q'\rangle \in A.$$

Since $x \in L \Leftrightarrow \langle x, i, q\rangle \in A$ and $y_1 \in L \Leftrightarrow \langle x, i+1, q'\rangle \in A$, we have $x \in L \Leftrightarrow y_1 \in L$. Since $x \neq y_1$, we set $s(x) = y_1$.

Since f is polynomial-time computable, the value of $s(x)$ can be computed in polynomial-time. This proves the theorem. □

We note here that Theorem 3.1 can be derived from Theorem 3.2 because for each $L \in$ NP we can construct a bit-reduction as follows: Let M be a polynomial-time nondeterministic Turing machine that accepts L. Let p be a polynomial bounding the running time of M. Define $g(x) = 2^{p(|x|)}$ and $f(x,i) = 1$ if the i-th word in $\Sigma^{p(|x|)}$ is an accepting path of M on x and 0 otherwise. Define $B = 0^*1\{0,1\}^*$. Then, for all x, $x \in L \iff f(x,1) \cdots f(x, g(x)) \in B$. So, the hypothesis of Theorem 3.2 holds.

Now we define the hierarchy MODPH.

Definition 3.2. [Sto77, Wra77] Let $\mathcal{C}$ be a language class. Define $\exists \cdot \mathcal{C}$ to be the set of all languages L for which there exist a polynomial p and a language $A \in \mathcal{C}$ such that for all x,

$$x \in L \Leftrightarrow (\exists y, |y| = p(|x|))[\langle x, y\rangle \in A]$$

and define $\forall \cdot \mathcal{C}$ to be the set of all languages L for which there exist a polynomial p and a language $A \in \mathcal{C}$ such that for all x,

$$x \in L \Leftrightarrow (\forall y, |y| = p(|x|))[\langle x, y\rangle \in A].$$

Definition 3.3. [BG92, Sch89] Let $k \geq 2$ be an integer. Define $\mathrm{MOD}_k \cdot \mathcal{C}$ to be the set of all languages L for which there exist a polynomial p and a language $A \in \mathcal{C}$ such that for all x,

$$x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \wedge \langle x, y\rangle \in A\}\| \not\equiv 0 \pmod{k},$$

and define $\mathrm{coMOD}_k \cdot \mathcal{C}$ to be the set of all languages L for which there exist a polynomial p and a language $A \in \mathcal{C}$ such that for all x,

$$x \in L \Leftrightarrow \|\{y \mid |y| = p(|x|) \wedge \langle x, y\rangle \in A\}\| \equiv 0 \pmod{k}.$$

Definition 3.4. [HLS$^+$93] MODPH is the hierarchy consisting of the classes inductively defined as follows:

- P belongs to MODPH.
- If $\mathcal{C}$ is a class belonging to MODPH then $\exists \cdot \mathcal{C}$ and $\forall \cdot \mathcal{C}$ belong to MODPH.
- For each integer $k \geq 2$, if $\mathcal{C}$ is a class belonging to MODPH then $\mathrm{MOD}_k \cdot \mathcal{C}$ and $\mathrm{coMOD}_k \cdot \mathcal{C}$ belong to MODPH.

Corollary 3.4. *Let $\mathcal{C}$ be one of the following classes*

- PSPACE.
- *The levels* Σ_k^P, Π_k^P, *and* Δ_k^P *of the polynomial-time hierarchy.*
- 1NP. [2]
- *The levels of the Boolean hierarchy over* NP.
- *The levels of the* MODPH *hierarchy.*

Let r be one of the following reductions: $\leq_m^p$, $\leq_{1\text{-}tt}^p$, $\leq_{dtt}^p$, and $\leq_{k\text{-}dtt}^p$ for $k \geq 2$. Then every nontrivial set that is r-complete for $\mathcal{C}$ is r-autoreducible.

Proof. Since we have Theorem 3.2, we have only to show that each of the above classes is the polynomial-time bit-reduction closure of some regular language. Hertrampf et al. [HLS$^+$93] show that this is the case for PSPACE (based on the result of Cai and Furst [CF91]) and for the MODPH classes. It is easy to observe that 1NP is the polynomial-time bit-reduction closure of the regular language 0^*10^*. Also, for all $k \geq 1$, Δ_k^P is the polynomial-time bit-reduction closure of a suitable regular language [BSS99, Tra02, BLS$^+$04], and every level of the Boolean hierarchy over NP is the polynomial-time bit-reduction closure of a suitable regular language [SW98, BKS99]. This proves the corollary. □

Now we obtain a result about genericity of complete sets. The notion of resource-bounded genericity was defined by Ambos-Spies, Fleischhack, and Huwig [ASFH87]. We use the following equivalent definition [BM95, PS02].

Definition 3.5. For a set L and a string x let $L|x = \{y \in L \,|\, y < x\}$. 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 3.6. A set L is $t(n)$-*generic* if it is a.e. unpredictable in time $t(2^n)$.

[2] 1NP [GW86] which is also called US [BG82] is the class of languages L for which there exists a nondeterministic polynomial-time-bounded machine M such that an input x belongs to L if and only if M on input x has exactly one accepting path. In contrast, UP is the class of languages L for which there exists a nondeterministic polynomial-time-bounded machine M such that $L = L(M)$ and on every input x, the machine M on input x has at most one accepting path [Val76].

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.

We obtain the following theorem regarding genericity of NP-complete sets. Earlier, it was known that there must exist a $k > 0$ such that NP-complete sets are not $O(n^k)$-generic. The result is implicit in the work on small span theorems [JL95, ASNT96]. Note that it is an existence result on k, i.e., the proof does not give an explicit value for k. Below we improve this.

Theorem 3.3. *If L is a decidable and m-autoreducible set, then L is not $2n$-generic.*

Proof. Let L be an m-autoreducible set, and let f be an m-autoreduction for L. Consider the following predictor for L. Recall that on input x, the predictor has access to the partial characteristic sequence $L|x$. Given x, the predictor computes $f(x)$. If $f(x) < x$, then the predictor can decide the membership of x in L by making one query to $L|x$. This takes polynomial time in $|x|$. If $f(x) > x$, then for every $y < x$, the predictor checks if $f(y) = x$. Once such y is found, the predictor can decide membership of x in L. If no such y is found, then the predictor runs the deterministic algorithm for L on x.

We claim that for infinitely many inputs, the predictor runs in time 2^{2n}. If for infinitely many x, $f(x) < x$, then, on these strings, the predictor decides x in polynomial time. Assume for all but finitely many x, $f(x) \geq x$. Note that since f is an autoreduction, $f(x) \neq x$. Thus for all but finitely many x, $f(x) > x$. Thus for infinitely many x, there exists $y < x$ such that $f(y) = x$. The predictor decides all such x by finding a y such that $f(y) = x$. The time for finding such y is bounded by 2^{2n}. Thus the predictor decides infinitely many x in time 2^{2n} time. Thus L is not $2n$-generic. □

Corollary 3.5. *All m-complete sets for the classes* NP*,* PSPACE*, and levels of* MODPH *are not $2n$-generic.*

3.1 Relativization

We obtain an oracle satisfying the following properties:

Theorem 3.4. *For any $k \geq 2$, there is an oracle A such that relative to A there is a set B that is $\leq^p_{k\text{-}dtt}$ complete for* NP *but not $\leq^p_{(k-1)\text{-}T}$ autoreducible.*

In light of this oracle it is probably not possible to improve Corollary 3.1 in the sense that $\leq^p_{k\text{-}dtt}$-complete sets for NP are $\leq^p_{(k-1)\text{-}dtt}$-autoreducible. Also, this theorem is interesting because of the following corollary:

Corollary 3.6. *There is an oracle A such that relative to A there exists a $\leq^p_{2\text{-}dtt}$-complete set for* NP *that is not m-mitotic.*

Proof. Taking $k = 2$ in Theorem 3.4, we obtain an oracle A such that there exists a $\leq^p_{2\text{-}dtt}$-complete set for NP that is not $\leq^p_{1\text{-}T}$ autoreducible. The corollary follows from the facts that if a set is m-mitotic then it is m-autoreducible [AS84] and every m-reduction is a 1-T reduction. □

Corollary 3.6 gives a relativized negative answer to an open question of Buhrman and Torenvliet [BT94] as to whether all T-complete sets for NP are m-mitotic. Another related question

raised by Buhrman and Torenvliet is whether all tt-complete sets for NP are tt-autoreducible. The following theorem gives a partial answer to this question in a relativized world.

Theorem 3.5. *For any $k \geq 2$, there is an oracle A such that relative to A, there is a set B that is $\leq^p_{dtt}$ complete for* NP *but not $\leq^p_{btt}$-autoreducible.*

The proof of Theorem 3.5 is similar to that of Theorem 3.4. Here we present the proof of Theorem 3.4. The construction is similar to previous successful oracle constructions that require both diagonalization and encoding and need to avoid conflicts. Diagonalization is needed to show that the set B^A is not $\leq^p_{(k-1)\text{-}T}$ autoreducible. At the same time one needs to encode a complete set for NP^A into B^A in such a way that B^A is $\leq^p_{k\text{-}dtt}$ complete for NP^A. These two requirements are seemingly contradictory and make the proof intricate.

Proof. Let $k' = \lceil \log k \rceil$. Without loss of generality, we assume k' is odd (the proof for the case where k' is even is essentially the same). Define G to be the set consisting of the first k strings of length k'. We assume a polynomial-time computable one-to-one pairing function that can take any finite number of inputs such that its range does not intersect with 0^* and such that for all i, x, and l, $|\langle i, x, 0^l\rangle| > l$. Let $\{M_j\}_{j\geq 1}$ be an enumeration of polynomial-time $(k-1)$-Turing reductions. Let $\{N_i\}_{i\geq 1}$ be an enumeration of all nondeterministic polynomial-time oracle Turing machines. For each $j \geq 1$, let p_j be a polynomial that bounds the running time of both M_j and N_j.

For any set A, let $K^A = \{\langle i, x, 0^l\rangle \mid N_i^A \text{ accepts } x \text{ within } l \text{ steps }\}$ be the canonical complete set for NP^A. Now define

$$B_e^A = \{s\langle i, x, 0^l\rangle \mid s \in G \wedge [\exists y[|y| = |\langle i, x, 0^l\rangle| \wedge s\langle i, x, 0^l\rangle y \in A]]\},$$

$$B_d^A = \{0^{2n} \mid \exists y[|y| = 2n \wedge y \in A]\},$$

and

$$B^A = B_e^A \cup B_d^A.$$

Clearly B_e^A, B_d^A and B^A all belong to NP^A. We will construct A such that for all but finitely many combinations of i, x, and l,

$$\langle i, x, 0^l\rangle \in K^A \Leftrightarrow \exists s[s \in G \wedge s\langle i, x, 0^l\rangle \in B^A],$$

which implies $K^A \leq^p_{k\text{-}dtt} B^A$. Also, we will construct A such that for every j, there exists an even number n_j such that

$$M_j^A \text{ accepts } 0^{n_j} \text{ with oracle } B^A \Leftrightarrow 0^{n_j} \notin B^A$$

or

$$M_j^A \text{ on input } 0^{n_j} \text{ queries } 0^{n_j} \text{ to oracle } B^A,$$

which will ensure that M_j is not an $\leq^p_{(k-1)\text{-}T}$ autoreduction of B.

We construct A in stages. We initialize $n_1 = 2$, so that $p_1(n_1) < 2^{n_1}$, and define $A^{<2} = \emptyset$. At Stage j, we assume the construction has been done up to strings of length $n_j - 1$, where n_j is some even number chosen in Stage $j-1$. Let $A_{j-1} = A^{\leq n_j - 1}$. Now the construction for Stage j proceeds in two steps. In the first step, we will *diagonalize* against M_j. If $M_j^{A_{j-1}}$ on input 0^{n_j} does not query the string 0^{n_j} to $B_{A_{j-1}}$, then we will find a string y of length n_j such that $M_j^{A_{j-1}\cup\{y\}}$, making $k-1$ queries to $B^{A_{j-1}\cup\{y\}}$ accepts 0^{n_j} if and only if $0^{n_j} \notin B^{A_{j-1}\cup\{y\}}$. In the second step, we

will do the *encoding*. We will choose some sufficiently large even number n_{j+1} and make sure, for every $\langle i,x,0^l\rangle$ such that $n_j < |s\langle i,x,0^l\rangle y| < n_{j+1}$, $s \in G$, and $|y| = |\langle i,x,0^l\rangle|$, that $\langle i,x,0^l\rangle \in K^A$ if and only if $\exists s \exists y[s \in G \wedge |y| = |\langle i,x,0^l\rangle| \wedge s\langle i,x,0^l\rangle y \in A$. This will ensure for those $\langle i,x,0^l\rangle$ that $\langle i,x,0^l\rangle \in K^A$ if and only if $\exists s[s \in G \wedge s\langle i,x,0^l\rangle \in B^A]$.

We first show that the string y of length n_j needed for the diagonalization step exists. We use a set N_A for reserving strings of length greater than or equal to n_j for $\overline{A}$. N_A is set to $\emptyset$ at the beginning of each stage. We then simulate M_j on input 0^{n_j} and resolve each query in the following way. For each query q made to A, we answer YES if and only if $q \in A_{j-1}$. If $|q| \geq n_j$, set N_A to $N_A \cup \{q\}$. For each query q made to B^A, if $q = 0^{n_j}$, we proceed to the encoding step. (In this case, M_j is not an autoreduction.) Now $q \neq 0^{n_j}$. We answer q with YES if and only if $q \in B^{A_{j-1}}$. For those q's such that q is of the form 0^{2n}, where $2n > n_j$, we set N_A to $N_A \cup \{y||y| = |q|\}$, which will ensure that $q = 0^{2n} \notin B^A$. For those q's such that $q = s\langle i,x,0^l\rangle$, $s \in G$, and $k' + 2|\langle i,x,0^l\rangle| > n_j$, we set N_A to $N_A \cup \{s\langle i,x,0^l\rangle y \mid |y| = |\langle i,x,0^l\rangle|\}$, which will ensure that $q = s\langle i,x,0^l\rangle \notin B^A$. Observe that strings of length n_j (an even length) can only be put into N_A if they are queried by M_j to A. So $\|N_A \cap \Sigma^{=n_j}\| \leq p_j(n_j)$ since M_j is p_j-time bounded. We will see below that in Stage $j-1$, n_j was chosen such that $p_j(n_j) < 2^{n_j} = \|\Sigma^{n_j}\|$. So we choose a string y of length n_j such that $y \notin N_A$. For this y, $M_j^{A_{j-1}}(0^{n_j})$, which makes $k-1$ queries to $B^{A_{j-1}}$, accepts if and only if $M_j^{A_{j-1}\cup\{y\}}(0^{n_j})$, making $k-1$ queries to $B^{A_{j-1}\cup\{y\}}$, accepts. Thus we add y to A_{j-1} if and only if $M_j^{A_{j-1}}(0^{n_j})$ rejects with oracle $B^{A_{j-1}}$. This completes the diagonalization step.

Now we need to do the encoding step. At the same time we want to maintain the diagonalization properties of the oracle from the diagonalization step. This can be ensured by not putting any string in N_A into A. However, by doing so we have to make sure encoding can be done up to length $\max_{y\in N_A}|y|$. Also we need $p_j(n_j) < 2^{n_j}$ for the diagonalization step at any Stage j. To fulfill all these requirements, we choose n_{j+1} to be the minimum even number such that the following hold:

- $p_{j+1}(n_{j+1}) < 2^{(n_{j+1}-k')/2}$
- $n_{j+1} > n_j$
- $n_{j+1} > \max_{y\in N_A}|y|$

Now we encode each triple $\langle i,x,0^l\rangle$ such that $n_j < |s\langle i,x,0^l\rangle y| < n_{j+1}$, $s \in G$, and $|y| = |\langle i,x,0^l\rangle|$. We have to decide whether to put some string $s\langle i,x,0^l\rangle y$ into A according to whether $\langle i,x,0^l\rangle \in K^A$, so as to make sure $\langle i,x,0^l\rangle \in K^A$ if and only if $\exists s[s \in G \wedge s\langle i,x,0^l\rangle \in B^A]$. As we said earlier, this task is nontrivial since some of these strings might have already been put into N_A. We need to show that there exist strings that are unreserved for each triple $\langle i,x,0^l\rangle$, to be used for encoding. We will also need a set Y_A that contains strings reserved for A during the encoding process. Now we use the following procedure in Fig. 1 to do the encoding from length n_j+1 to $n_{j+1}-1$.

The correctness of the above encoding procedure follows from the following claims where $Q_j = \{\langle i,x,0^l\rangle \mid n_j < k' + 2|\langle i,x,0^l\rangle| < n_{j+1}\}$.

Claim 3.1. *For every $\langle i,x,0^l\rangle \in Q_j$ in the above procedure, there exists a string $s\langle i,x,0^l\rangle y \notin N_A$ such that $s \in G$ and $|y| = |\langle i,x,0^l\rangle|$, to select when step 6 is executed.*

Proof. We look back at the diagonalization step. For each query q made to A by M_j, we put q into N_A only if $|q| \geq n_j$. Since there are at most $p_j(n_j)$ many such queries due to the fact

```
1 Y_A := ∅;
  //N_A is initially the set constructed in the diagonalization step.
2 Q_j := {⟨i,x,0^l⟩ | n_j < k' + 2|⟨i,x,0^l⟩| < n_{j+1}};
  // set of triples ⟨i,x,0^l⟩ that we need to consider for encoding.
3 For all triples ⟨i,x,0^l⟩ in Q_j in lexicographical order do
4    Simulate N_i on x for l steps with oracle A_{j-1} ∪ Y_A;
5    If N_i accepts then
6       Choose s⟨i,x,0^l⟩y ∉ N_A, such that s ∈ G and |y| = |⟨i,x,0^l⟩|;
7       Y_A := Y_A ∪ {s⟨i,x,0^l⟩y};
8    Else
9       N_A := N_A ∪ {s⟨i,x,0^l⟩y | s ∈ G ∧ |y| = |⟨i,x,0^l⟩|};
10 EndFor;
11 A_j := A_{j-1} ∪ Y_A;
```

Figure 1: Procedure for the encoding step at Stage j in Theorem 3.4 (comments start with //).

that M_j is p_j-time bounded, we put at most $p_j(n_j)$ many strings into N_A because of queries to A. Now for each $\langle i,x,0^l\rangle \in Q_j$, for each $s \in G$ consider $S_s = \{s\langle i,x,0^l\rangle y || y| = |\langle i,x,0^l\rangle|\}$. For each query q made to B^A by M_j, at most one such S_s might be put into N_A. Since M_j only makes $k-1$ queries to B^A, there must exist one S_s such that no string in S_s is put into N_A due to queries made to B^A by M_j. Let S_s be such a set. Note that $\|S_s\| = 2^{|\langle i,x,0^l\rangle|} \geq 2^{(n_j-k')/2} > p_j(n_j)$, where the last inequality follows from the choice of n_j at Stage $j-1$. So there must exist $z \in S_s$ that is not put into N_A because of queries made to A by M_j. Hence, z is a string of the form $s\langle i,x,0^l\rangle y$ such that $s \in G$ and $|y| = |\langle i,x,0^l\rangle|$, and $z \notin N_A$ by the end of the diagonalization step. Now during the encoding step, we will only put into N_A strings of the form $s\langle i',x',0^{l'}\rangle y$, where $\langle i',x',0^{l'}\rangle \neq \langle i,x,0^l\rangle$, upon processing triples $\langle i',x',0^{l'}\rangle$ that are lexicographically less than $\langle i,x,0^l\rangle$. So z cannot have been put into N_A during the encoding step by the time the triple $\langle i,x,0^l\rangle$ is processed. Hence, the claim follows. □

Claim 3.2. *For any $\langle i,x,0^l\rangle \in Q_j$ in the above procedure,*

$$\langle i,x,0^l\rangle \in K^{A_j} \Leftrightarrow \exists s[s \in G \wedge s\langle i,x,0^l\rangle \in B^{A_j}].$$

Proof. First note that the following equivalence holds after the triple $\langle i,x,0^l\rangle \in Q_j$ is processed:

$$\langle i,x,0^l\rangle \in K^{A_{j-1}\cup Y_A} \Leftrightarrow \exists s[s \in G \wedge s\langle i,x,0^l\rangle \in B^{A_{j-1}\cup Y_A}]. \quad (*)$$

For every triple $\langle i',x',0^{l'}\rangle \in Q_j$ that is processed after $\langle i,x,0^l\rangle$, either a string in the form $s'\langle i',x',0^{l'}\rangle y'$ such that $s' \in G$ and $|y'| = |\langle i',x',0^{l'}\rangle|$ is put into Y_A or all the members of the set $\{s'\langle i',x',0^{l'}\rangle y' \mid s' \in G \wedge |y'| = |\langle i',x',0^{l'}|\}$ are put into N_A. Neither of these will change the truth value of the right-hand side of $(*)$. Also note that all the strings added to Y_A or N_A after $\langle i,x,0^l\rangle$ is processed are of length at least $k' + 2|\langle i,x,0^l\rangle| > l$. So the truth value of the left side of $(*)$ will not change either after $\langle i,x,0^l\rangle$ is processed since the first l steps of N_i's computation on x

makes queries of length at most l. Therefore, $(*)$ becomes a loop invariant once the triple $\langle i, x, 0^l \rangle$ is processed. Since by the end of the loop we have $A_j = A_{j-1} \cup Y_A$, it follows that

$$\langle i, x, 0^l \rangle \in K^{A_j} \Leftrightarrow \exists s[s \in G \wedge s\langle i, x, 0^l \rangle \in B^{A_j}].$$

□

Claim 3.3. *For every triple $\langle i, x, 0^l \rangle$ such that $|\langle i, x, 0^l \rangle| > 2$,*

$$\langle i, x, 0^l \rangle \in K^{A} \Leftrightarrow \exists s[s \in G \wedge s\langle i, x, 0^l \rangle \in B^{A}].$$

Proof. Let $|\langle i, x, 0^l \rangle| > 2$. Then $\langle i, x, 0^l \rangle$ belongs to Q_j, for some $j \geq 1$. Then, by Claim 3.2, it holds that

$$\langle i, x, 0^l \rangle \in K^{A_j} \Leftrightarrow \exists s[s \in G \wedge s\langle i, x, 0^l \rangle \in B^{A_j}].$$

We know from the above construction that only strings of length greater than or equal to n_j will be added into A after Stage j and $n_j > k' + 2|\langle i, x, 0^l \rangle| > l$ for any $\langle i, x, 0^l \rangle \in Q_j$. It follows that

$$\begin{aligned} &\langle i, x, 0^l \rangle \in K^A \\ \Leftrightarrow\ &\langle i, x, 0^l \rangle \in K^{A_j} \\ \Leftrightarrow\ &\exists s[s \in G \wedge s\langle i, x, 0^l \rangle \in B^{A_j}] \\ \Leftrightarrow\ &\exists s[s \in G \wedge s\langle i, x, 0^l \rangle \in B^{A}]. \end{aligned}$$

□

This finishes the proof of Theorem 3.4. □

4 Mitoticity

Buhrman, Hoene, and Torenvliet [BHT98] show that all EXP-complete sets are weakly m-mitotic. We improve this to show that all EXP-complete sets are m-mitotic. We remark that Buhrman and Torenvliet [BT05] cite a private communication with Kurtz for an independent proof of this result.

Theorem 4.1. *Let $\mathcal{C}$ be a complexity class closed under many-one reductions. If L is many-one complete for $\mathcal{C}$ with respect to honest reductions, then L is m-mitotic.*

Proof. Note that $L \times \Sigma^* \in \mathcal{C}$ and hence there exists an honest many-one reduction f from $L \times \Sigma^*$ to L. So there exists an $l > 0$ such that

$$\forall x, |x|^{1/l} < |f(x)| < |x|^l.$$

Define $t_1 = 2$ and $t_{i+1} = (t_i)^{2l^2}$. Consider the following set.

$$S = \{x \mid \text{ for some odd } i,\ t_i \leq |x| < t_{i+1}\}$$

Clearly S is in P. We claim that $L \cap S$ and $L \cap \overline{S}$ are complete for $\mathcal{C}$. This shows that L is m-mitotic.

Claim 4.1. *$L \cap S$ is complete for $\mathcal{C}$.*

Proof. It is clear that $L \cap S \in \mathcal{C}$. We construct a reduction h from L to $L \cap S$. Given input x, the reduction computes i such that $t_i \leq |x| < t_{i+1}$. We consider two cases.

Case 1: i is odd, i.e., $x \in S$. Choose y sufficiently large such that

$$(t_{i+2})^l < |\langle x, y \rangle| < (t_{i+2})^{2l}.$$

This is possible in polynomial time in $|x|$, since

$$(t_{i+2})^l = (((t_i)^{2l^2})^{2l^2})^l = (t_i)^{4l^5} \leq |x|^{4l^5}.$$

Let $z = \langle x, y \rangle$ and set $h(x) = f(z)$. Observe that $|f(z)| > |z|^{1/l} > t_{i+2}$ and $|f(z)| < |z|^l < (t_{i+2})^{2l^2} = t_{i+3}$. So $t_{i+2} < |f(z)| < t_{i+3}$ and hence $f(z) \in S$, since $i+2$ is odd. Now,

$$\begin{aligned} x \in L &\Leftrightarrow z \in L \times \Sigma^* \\ &\Leftrightarrow f(z) \in L \qquad (f \text{ is a reduction from } L \times \Sigma^* \text{ to } L) \\ &\Leftrightarrow f(z) \in L \cap S \qquad (f(z) \in S). \end{aligned}$$

Case 2: i is even, i.e., $x \notin S$. Choose y sufficiently large such that

$$(t_{i+1})^l < |\langle x, y \rangle| < (t_{i+1})^{2l}.$$

Again this is possible in polynomial time in $|x|$. Let $z = \langle x, y \rangle$ and set $h(x) = f(z)$. As in Case 1, we obtain $t_{i+1} < |f(z)| < t_{i+2}$ and hence $f(z) \in S$, since $i+1$ is odd. Therefore, as above,

$$x \in L \Leftrightarrow f(z) \in L \cap S.$$

The function h is computable in polynomial time. Thus $L \cap S$ is complete for $\mathcal{C}$. □

By symmetry we obtain:

Claim 4.2. *$L \cap \overline{S}$ is complete for $\mathcal{C}$.*

The theorem follows from Claims 4.1 and 4.2. □

Corollary 4.1. *All* EXP*-complete sets are m-mitotic.*

Proof. EXP is closed under many-one reductions and all EXP-complete sets are complete with respect to length increasing reductions [Ber77]. □

A preliminary version of this paper [GOP+05] stated the following results regarding mitoticity of complete sets for NEXP, and PSPACE.

Theorem 4.2. *All m-complete sets for* NEXP *are weakly m-mitotic.*

Theorem 4.3. *All m-complete sets for* PSPACE *are weakly T-mitotic.*

Since then, these results were improved [GPSZ06] to show that m-complete sets for NEXP and PSPACE are m-mitotic. This is shown by proving that every m-autoreducible set is m-mitotic. So the Theorems 4.2 and 4.3 are consequences of the m-autoreducibility of m-complete sets for NEXP (cf. beginning of section 3) and m-complete sets for PSPACE (Corollary 3.4). Here we present a proof of Theorem 4.2, because it is much easier than the proof of the stronger result.

For this we use a result of Ganesan and Homer [GH92] who showed that every NEXP-complete set is complete via 1-1 reductions. We first describe the simplified idea of our proof: If K is the standard NEXP-complete set and if L is an arbitrary NEXP-complete set, then $0K \cup 1K$ m-reduces to L by a 1-1 function f. It follows that $L_1 = f(0K)$ and $L_2 = L - f(0K)$ are m-hard for NEXP. Moreover, L_1 and L_2 are disjoint subsets of L and it holds that $L = L_1 \cup L_2$. In order to obtain weak m-mitoticity it remains to argue that both sets belong to NEXP. In our proof we argue that appropriate modified sets L_1 and L_2 have all the properties we described so far and additionally belong to NEXP.

Proof of Theorem 4.2. Let K be the standard NEXP-complete set. Let L be any given NEXP-complete set. We show that L is weakly m-mitotic.

$$K' \stackrel{df}{=} 0K \cup 1K$$

is NEXP-complete. K' reduces to L via some $f \in$ PF. L reduces to K (really K, not K') via some $g \in$ PF. Choose k such that f and g can be computed in time $O(n^k)$. Since every NEXP-complete sets are complete via 1-1 reductions [GH92], we can assume f and g to be 1-1.

$$\begin{aligned} L_0 &\stackrel{df}{=} \{y \mid \exists x, |x| \leq |y|^k, f(0x) = y\} \\ L_1 &\stackrel{df}{=} \{y \mid \exists x, |x| \leq |y|^k, f(1x) = y\} \end{aligned}$$

Observe that $L_0, L_1 \in$ EXP. Define the following function:

$$f_0(x) \stackrel{df}{=} \begin{cases} f(0x) & : \text{ if } |f(0x)|^k \geq |x|, \\ f_0(g(f(0x))) & : \text{ otherwise.} \end{cases}$$

Note that if $|f(0x)|^k < |x|$ then $|g(f(0x))| \leq |f(0x)|^k < |x|$. So, for some y, $|y| < |x|$, $f_0(x) = f_0(y)$. So, the recursion terminates. Thus, $f_0 \in$ PF. Note that for every x there exists some y such that $f_0(x) = f(0y)$ and $|f(0y)|^k \geq |y|$. This implies that for all x, $f_0(x) \in L_0$.

Similarly define f_1 as

$$f_1(x) \stackrel{df}{=} \begin{cases} f(1x) & : \text{ if } |f(1x)|^k \geq |x|, \\ f_1(g(f(1x))) & : \text{ otherwise.} \end{cases}$$

By following an argument similar to the one above we can show that $f_1 \in$ PF and that $f_1(\Sigma^*) \subseteq L_1$.

We first show $K \leq_m^p L$ via reduction f_0. This is done by induction on the number of recursion steps r in the definition of $f_0(x)$. If $r = 0$, then $f_0(x) = f(0x)$ and therefore,

$$x \in K \Leftrightarrow 0x \in 0K \Leftrightarrow 0x \in K' \Leftrightarrow f_0(x) = f(0x) \in L.$$

If $r \geq 1$ then $f_0(x) = f_0(g(f(0x)))$. Let $y = g(f(0x))$. By our induction hypothesis, $y \in K \Leftrightarrow f_0(y) \in L$. So we obtain

$$\begin{aligned} x \in K &\Leftrightarrow 0x \in 0K \Leftrightarrow 0x \in K' \Leftrightarrow f(0x) \in L \\ &\Leftrightarrow y = g(f(0x)) \in K \Leftrightarrow f_0(y) = f_0(g(f(0x))) = f_0(x) \in L. \end{aligned}$$

Thus, $K \leq_m^p L$ via f_0. Analogously we show $K \leq_m^p L$ via f_1.

Since $K \leq_m^p L$ via f_0 and $f_0(\Sigma^*) \subseteq L_0$ we have

- $K \leq_m^p (L \cap L_0)$ via f_0.

Since f is 1-1, L_0 and L_1 are disjoint. Since $f_1(\Sigma^*) \subseteq L_1$, we have $f_1(\Sigma^*) \subseteq \overline{L_0}$. Combining this with $K \leq_m^p L$ via f_1, we have:

- $K \leq_m^p (L \cap \overline{L_0})$ via f_1.

Therefore $(L \cap L_0)$ and $(L \cap \overline{L_0})$ are each NEXP-hard. Both sets are in NEXP since $L_0 \in$ EXP and $L \in$ NEXP. $\square$

5 Immunity

In Glaßer et al. [GPSS06], the authors proved immunity results for NP-complete sets under the assumption that certain average-case hardness conditions are true. For example, they show that if one-way permutations exist, then NP-complete sets are not 2^{n^ϵ}-immune. Here we obtain a non-immunity result for NP-complete sets under the assumption that the following worst-case hardness hypothesis holds.

Hypothesis H: There is an NP machine M that accepts 0^* and no P-machine can compute its accepting computations. This means that for every polynomial-time machine N there exist infinitely many n such that $N(0^n)$ is not an accepting computation of $M(0^n)$.

Though the hypothesis looks verbose, we note that it is implied by a simply stated and believable hypothesis.

Observation 5.1. *If there is a tally language in* $\mathrm{NP} \cap \mathrm{coNP} - \mathrm{P}$*, then Hypothesis H is true.*

We show that if Hypothesis H holds, then NP-complete languages are not $2^{n(1+\epsilon)}$-immune. Ganesan and Homer [GH92] defined a function f to be exponentially honest if $\forall x, 2^{|f(x)|} \geq |x|$.

Theorem 5.1. *If Hypothesis H holds, then, for every* $\epsilon > 0$*,* NP*-complete languages are not* $2^{n(1+\epsilon)}$*-immune.*

Proof. Let L be any NP-complete language. We first show that there is a reduction from 0^* to L that is infinitely often exponentially honest. We then use this fact to show that L is not $2^{n(1+\epsilon)}$-immune.

Lemma 5.1. *Assume that the hypothesis holds. For every* NP *complete language* L*, there exists a constant* $c > 0$*, such that for every* $k' > 0$*, there exists a reduction* f *from* 0^* *to* L *such that for infinitely many* n*,* $|f(0^n)| > k' \log n$*. The reduction* f *can be computed in time* $O(n^{k'+c})$*.*

Before giving a formal proof, we mention the intuition behind the proof. Consider the left-set [OW91] of an NP-machine M. Suppose there is a reduction from this set to L that shrinks inputs to logarithmically small strings. This can be interpreted as "there is a reduction from the left-set to a sparse language". Now we can apply Ogihara-Watanabe type arguments to show that accepting computations of M can be computed in polynomial time. If M is the machine from the Hypothesis H, then this gives a contradiction. Thus, any reduction from the left-set of M to L must be exponentially honest infinitely often. However, this does not quite give an exponentially-honest

reduction from 0^* to L. We modify Ogihara-Watanabe construction to exhibit a reduction from 0^* to L that is exponentially honest infinitely often.

Proof. Let M be the NP machine from the hypothesis. Let a_n be the lexicographically maximum accepting computation of M on 0^n. Consider the left-set

$$S = \{\langle 0^n, y\rangle \mid y \leq a_n\},$$

where $\leq$ is the dictionary order on strings with $0 < 1$. It is obvious that S is in NP. Let g be an m-reduction from S to L. We now describe a reduction from 0^* to L with the desired properties. Let T be the computation tree of M on 0^n. Without loss of generality, assume T is a complete binary tree, and let d denote the depth of T. The reduction traverses T in stages. At stage k it maintains a list of nodes at level k. The reduction also maintains a variable called *next*. Stage 1, places the root of the tree into $list_1$, and sets *next* to 1^{m+1}, where m is the length of an accepting computation of M on 0^n. We now describe Stage $k > 1$.

1. Let $list_{k-1} = \langle u_1, u_2, \ldots, u_l\rangle$. Let $v_1, v_2, \ldots, v_{2l}$ be the children of nodes in $list_{k-1}$. Assume $v_1 < v_2 < \cdots < v_{2l}$. Set $list_k = \langle v_1, \ldots, v_{2l}\rangle$.
2. Consider the first j such that $|g(\langle 0^n, v_j\rangle)| > k' \log n$.
3. If such j exists, then let $list_k = \langle v_1, v_2, \ldots, v_{j-1}\rangle$ and $next = v_j$.
4. Prune $list_k$, i.e., if there exist $i < r$ such that $g(\langle 0^n, v_i\rangle) = g(\langle 0^n, v_r\rangle)$, then remove $v_i, v_{i+1}, \ldots, v_{r-1}$ from $list_k$.

The following is the desired reduction f from 0^* to L. Let x be the input and let $n = |x|$.

- If $x \neq 0^n$, then output a fixed string not in L.
- Run Stages $1, \ldots, d$.
- All nodes in $list_d$ are leaf nodes. If $list_d$ contains an accepting node, then output a fixed string in L, else output $g(\langle 0^n, next\rangle)$.

We claim that the above reduction has the desired properties. It is obvious that the reduction is correct on non-tally strings. So we focus only on tally strings. We show a series of claims that help us in showing the correctness of the reduction. Let $max(list_k)$ denote the maximum node of $list_k$.

Observation 5.2. *Consider Step 4 of Stage k, if a node v_m is removed from $list_k$, then the rightmost accepting computation of M on 0^n does not pass through v_m.*

Proof. Step 4 removes nodes $v_i, \ldots, v_{r-1}$ from $list_k$, if there exist $i < r$ such that $g(\langle 0^n, v_i\rangle) = g(\langle 0^n, v_r\rangle)$. Assume that the right most accepting computation passes through a node v_m such that $i \leq m \leq r-1$. By definition of S, $\langle 0^n, v_i\rangle \in S$ and $\langle 0^n, v_r\rangle \notin S$. Since, g is an m-reduction from S to L, and $g(\langle 0^n, v_i\rangle) = g(\langle 0^n, v_r\rangle)$, this is a contradiction. □

Observation 5.3. *For every k, at the end of stage k, $max(list_k) < next$.*

Proof. We prove the claim by induction. At the end of Stage 1, $list_1$ contains the root and $next$ is 1^{m+1}. Thus the claim holds after Stage 1. Assume that the claim holds at the end of Stage $k-1$. We consider two cases: First assume that the value of $next$ does not change during Stage k. Observe that every node in $list_k$ is a children of some node in $list_{k-1}$. Since every node in $list_{k-1}$ is smaller than $next$, the claim follows in this case. Suppose the value of $next$ changes during Stage k. This happens in Step 3, where $next$ is set to v_j and $list_k$ becomes $\langle v_1, \cdots, v_{j-1}\rangle$. In the later stages of Stage k, $list_k$ will be a subset of $\langle v_1, \cdots, v_{j-1}\rangle$. Thus the claim holds in this case also. □

Claim 5.1. *For every $k \geq 1$, the following statement holds at the end of Stage k: There is no node v such that rightmost accepting computation passes through v and $max(list_k) < v < next$.*

Proof. We prove the claim by induction. After Stage 1, $list_1$ contains the root node and $next = 1^{m+1}$. Since the rightmost accepting computation passes through the root the claim holds after Stage 1. Assume that the claim holds after Stage $k-1$. We again consider two cases.

First, assume that the value of $next$ does not change during Stage k. After Step 1, $list_k$ contains all children of all nodes of $list_{k-1}$. Thus the claim holds after Step 1. During Step 4, the algorithm removes some nodes from $list_k$. However, by Observation 5.2, the rightmost accepting computation can not pass through any of the removed nodes. Thus the claim holds after Stage k.

Now assume that the value of $next$ changes during Stage k. This happens in Step 3, and when it happens $list_k = \langle v_1, \cdots, v_{j-1}\rangle$ and $next = v_j$. Since there is no node with $v_{j-1} < v < v_j$ the claim holds at this time. During Step 4, the algorithm may remove some nodes from $list_k$. Again, by Observation 4, the rightmost accepting computation can not pass through any of these removed nodes. Thus the claim holds in this case also. □

Claim 5.2. *The above reduction f is correct, i.e., for every n, $f(0^n)$ is a string in L.*

Proof. Note that $f(0^n)$ is a fixed string in L, if $list_d$ contains an accepting leaf node. In this case the reduction is obviously correct. Suppose none of the leaf nodes in $list_d$ is an accepting node. Then the right most accepting computation does not pass through any node in $list_d$. By the previous claim, at the end of Stage d, the right most accepting computation either passes through $next$ or lies to the right of $next$. Thus $\langle 0^n, next\rangle \in S$. Since the reduction outputs $g(\langle 0^n, next\rangle)$, and g is an m-reduction from S to L, the claim follows. □

Now we show that the reduction runs in polynomial time. Assume that, for any node $u \in T$, the computation of $g(\langle 0^n, u\rangle)$ takes n^r steps. Let the depth of the tree T be n^l. We define c to be $r + l$. Note that the constant c depends only on L, and is independent of k'.

Claim 5.3. *The running time of the reduction is $O(n^{k'+c})$.*

Proof. We first bound the size of $list_k$. Consider Stage $k-1$. Note that after Step 3, for every node u in $list_k$, $|g(\langle 0^n, u\rangle)| \leq k' \log n$. After Step 4, for every pair of nodes u and v in $list_r$, we have $g(\langle 0^n, u\rangle) \neq g(\langle 0^n, v\rangle)$. Thus, after Stage $k-1$, $|list_{k-1}| \leq n^{k'}$. Thus after Step 1 of Stage k, $|list_k| \leq 2n^{k'}$.

Now we calculate the running time for Stage k. Observe that Step 2 is the most expensive step, where the reduction g is applied on every tuple $\langle 0^n, v_j\rangle$ with $v_j \in list_k$. At this time $|list_k| \leq 2n^{k'}$.

Thus time required for this computation is $O(n^{k'+r})$. So, Stage k requires $O(n^{k'+r})$ steps. Since there are $d = n^l$ stages, the time taken for the reduction is $O(n^{k'+r+l}) = O(n^{k'+c})$. □

Claim 5.4. *For infinitely many n, the reduction outputs a string whose length is bigger than $k' \log n$.*

Proof. Let 0^n be the input to the reduction. The reduction outputs a fixed string from L, only when it discovers an accepting computation of $M(0^n)$, else it outputs $g(\langle 0^n, next \rangle)$. Since the reduction runs in polynomial time, the reduction cannot discover an accepting computation of $M(0^n)$ for all but finitely many n. Otherwise, it will contradict Hypothesis H. Thus for infinitely many n, the reduction outputs $g(\langle 0^n, next \rangle)$. Since the reduction is correct, the value of next can not be 1^{m+1}. Thus *next* is set to a node v during some stage. However, the reduction sets *next* to v only if $|g(\langle 0^n, v \rangle)| > k' \log n$. Thus for infinitely many n, the reduction outputs a string whose length is bigger than $k' \log n$. □

This concludes the proof of Lemma 5.1. □

Combing the following lemma with Lemma 5.1 completes the proof of the theorem.

Lemma 5.2. *Let L be any* NP*-complete language. If there is a constant $c > 0$ such that for every $k > 0$ there is a reduction f from 0^* to L such that, for infinitely many n, $|f(0^n)| > k \log n$, and f can be computed in time $O(n^{k+c})$, then for every $\epsilon > 0$, L is not $2^{n(1+\epsilon)}$-immune.*

Proof. Given $\epsilon > 0$, pick k such that $k > (c+1)/\epsilon$. There is a reduction f from 0^* to L such that for infinitely many n, $|f(0^n)| > k \log n$. From this it follows that there exist infinitely many $x, |x| = m$, for which there exist $n < 2^{m/k}$ such that $f(0^n) = x$. Consider the algorithm that, on input x, $|x| = m$, behaves as follows:

1. For $i = 1$ to $2^{m/k}$, if $f(0^i) = x$ then accept x and halt.
2. Reject x.

The above algorithm accepts a string x only when it finds that for some i, $f(0^i) = x$. Since f is an m-reduction from 0^* to L, the above algorithm accepts a subset of L. There exist infinitely many x for which $f(0^i) = x$ for some i, $1 \leq i \leq 2^{m/k}$. So, the above algorithm accepts an infinite set. The most time-consuming step of the algorithm is computation of $f(0^i)$ and this requires i^{k+c} steps. Since i ranges between 1 and $2^{m/k}$, the running time of the algorithm is $2^{m(1+\frac{c+1}{k})}$. Since $\frac{c+1}{k} < \epsilon$, the time taken by the algorithm is $2^{m(1+\epsilon)}$. Thus L is not $2^{n(1+\epsilon)}$-immune. □

Theorem 5.1 follows from the Lemmas 5.1 and 5.2. □

Corollary 5.1. *If there is a tally language in* $\mathrm{NP} \cap \mathrm{coNP} - \mathrm{P}$*, then, for every $\epsilon > 0$,* NP*-complete languages are not $2^{n(1+\epsilon)}$-immune.*

Next we consider the possibility of obtaining an unconditional result regarding non-immunity of NP-complete languages. We show that for certain type of NP-complete languages we get an unconditional result.

Definition 5.1. A language L does not have superpolynomial gaps, if there exists $k > 0$ such that for all but finitely many n, there exists a string x in L such that $n \leq |x| \leq n^k$.

We show that NP-complete languages that have no superpolynomial gaps are not immune.

Theorem 5.2. *If L is an* NP*-complete language that has no superpolynomial gaps, then for every $\epsilon > 0$, L is not $2^{n(1+\epsilon)}$-immune.*

Proof. Suppose there exists a $2^{n(1+\epsilon)}$-immune NP-complete language L that does not have superpolynomial gaps. Thus there exists $k > 0$ such that for all but finitely many n, there exists a string x in L such that $n \leq |x| \leq n^k$. Consider a machine N that on input 0^n, guesses strings x and w and accepts if $n \leq |x| \leq n^k$ and w witnesses that $x \in L$. Clearly N accepts 0^*, and if there is a machine that computes infinitely many accepting computations of N, then L is not $2^{n(1+\epsilon)}$-immune. Thus the Hypothesis H holds. By Theorem 5.1, we have a contradiction. □

The above theorem prompts the following question: Are there NP-complete languages with superpolynomial gaps? We have the following result. Given two complexity classes $\mathcal{A}$ and $\mathcal{B}$, we say $\mathcal{A} \subseteq io\text{-}\mathcal{B}$, if for every language $A \in \mathcal{A}$, there exists a language $B \in \mathcal{B}$ such that for infinitely many n, $A^n = B^n$.

Theorem 5.3. *If* NP *has a complete language with superpolynomial gaps, then for every $\epsilon > 0$,* NP $\subseteq io$-DTIME(2^{n^ϵ}).

Proof. Assume L is an NP-complete language with superpolynomial gaps, let $L \in$ DTIME(2^{n^k}). Let S be any language in NP. Since L is NP-complete, S is m-reducible to L via a reduction f whose running time is n^r. Note that for every x, $|f(x)| \leq |x|^r$. Without loss of generality, assume $r > k$, and let $l = r/\epsilon$. Since L has superpolynomial gaps, for infinitely many n, L is empty between lengths $n^{1/l}$ and n^l. Consider the following algorithm M.

1. Input x, $|x| = n$.
2. Compute $f(x)$.
3. If $n^{1/l} \leq |f(x)|$, then reject x.
4. Else accepts x if and only if $f(x)$ belongs to L.

The algorithm checks for membership of $f(x)$ in L, only when $|f(x)| < n^{1/l}$. The time taken for this step is $2^{|f(x)|^k} \leq 2^{|f(x)|^r} \leq 2^{n^{r/l}} = 2^{n^\epsilon}$. Thus the language accepted by the above machine M belongs to DTIME(2^{n^ϵ}).

We now claim that $L(M)$ equals S at infinitely many lengths. Since L has superpolynomial gaps, for infinitely many n, L is empty between lengths $n^{1/l}$ and n^l. Consider one such length n. Observe that for any string x of length n, $|f(x)| \leq n^r \leq n^l$. If $n^{1/l} \leq |f(x)| \leq n^l$, then $x \notin S$. In this case the algorithm decides x correctly. If $|f(x)| < n^{1/l}$, the algorithm accepts x if and only if $f(x) \in L$. Thus the algorithm is correct on x. Thus at length n, $S^n = L(M)^n$. Thus $S \in io$-DTIME(2^{n^ϵ}). □

Combining Theorems 5.2 and 5.3 we have the following corollary.

Corollary 5.2. *If for some $\delta > 0$,* NP $\not\subseteq io$-DTIME(2^{n^δ})*, then for every $\epsilon > 0$, no* NP*-complete language is $2^{n(1+\epsilon)}$-immune.*

6 Robustness

Recently Buhrman and Torenvliet [BT04] proved that T-complete sets for EXP are robust against log-dense sets in P. Using similar ideas, we can prove the same for NP. The proof is easier though, due to the fact that search reduces to decision for all T-complete sets for NP [HS01]. A set S is *log-dense* if there is a constant $c > 0$ such that for all n, $\|S^{\leq n}\| \leq c \log n$.

Theorem 6.1. *If a set A is T-complete for* NP *and S is a* log*-dense set in* P*, then $A - S$ is T-complete for* NP*.*

Proof. Let A be T-complete for NP and let S be a log-dense set in P. Then $A - S$ belongs to NP, because S belongs to P. Now we show that A is T-reducible to $A - S$. Consider the machine T that reduces search to decision for A in time $p(n)$, for some polynomial p. We may assume that T never asks the same query twice, since T can hold a list that stores queries that have been asked together with the corresponding answers. Let $c > 0$ be a constant such that for all n, $\|S^{\leq n}\| \leq c \log n$. Then the following machine M is a T-reduction from A to $A - S$:

```
1  Input x
2  For all bit strings s of length c log p(|x|) do
3     k ⟵ 0;
4     Repeat
5        Simulate T on x until T makes the next query q;
6        If q ∉ S then continue simulation with YES iff q ∈ A − S;
7        Else
8           Continue the simulation with YES iff s_k = 1;
            //s_k is the k-th bit of s
9           k ⟵ k + 1;
10    Until T outputs y_s or rejects;
11    If T outputs y_s and y_s is a witness of x then ACCEPT;
12 Endfor;
13 REJECT.
```

If $q \notin S$, then a query to $A - S$ has the same answer as a query to A. So in this case M's simulation of T behaves correctly. At most $c \log p(|x|)$ queries q belong to S. For each of these queries, the bit string s determines the response. If $x \notin A$, then no witness y_s for x exists. So in this case M does not accept x. If $x \in A$, then the bit string s that represents the correct answers to the queries that are in S will make T output a correct witness y_s. Hence, M accepts x. Therefore, the algorithm is correct.

For the running time, there are at most $p(|x|)^c$ iterations of the outer loop, each of which is a simulation of T on x, which costs $p(|x|)$ time. Hence, the total running time is bounded by a polynomial in $|x|$. □

Acknowledgments. We thank the anonymous referees for their helpful suggestions and comments.

References

[AS84] 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.

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

[ASNT96] K. Ambos-Spies, H. Neis, and A. Terwijn. Genericity and measure for exponential time. *Theoretical Computer Science*, 168(1):3–19, 1996.

[Bar89] D. A. Mix Barrington. Bounded-width polynomial size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38:150–164, 1989.

[Ber77] L. Berman. *Polynomial Reducibilities and Complete Sets*. PhD thesis, Cornell University, Ithaca, NY, 1977.

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

[BFvMT00] 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.

[BG82] A. Blass and Y. Gurevich. On the unique satisfiability problem. *Information and Control*, 82:80–88, 1982.

[BG92] R. Beigel and J. Gill. Counting classes: Thresholds, parity, mods, and fewness. *Theoretical Computer Science*, 103:3–23, 1992.

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

[BKS99] B. Borchert, D. Kuske, and F. Stephan. On existentially first-order definable languages and their relation to NP. *Theoretical Informatics and Applications*, 33:259–269, 1999.

[BLS+04] B. Borchert, K. Lange, F. Stephan, P. Tesson, and D. Thérien. The dot-depth and the polynomial hierarchy correspond on the delta levels. In *Developments in Language Theory*, pages 89–101, 2004.

[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.

[BS85] J. Balcázar and U. Schöning. Bi-immune sets for complexity classes. *Mathematical Systems Theory*, 18(1):1–18, 1985.

[BSS99] B. Borchert, H. Schmitz, and F. Stephan. Unpublished manuscript, 1999.

[BT94] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proceedings 9th Structure in Complexity Theory*, pages 118–133, 1994.

[BT98] H. Buhrman and L. Torenvliet. Complete sets and structure in subrecursive classes. In *Proceedings of Logic Colloquium '96*, pages 45–78. Springer-Verlag, 1998.

[BT04] 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.

[BT05] H. Buhrman and L. Torenvliet. A Post's program for complexity theory. *Bulleting of the EATCS*, 85:41–51, 2005.

[CF91] J.-Y. Cai and M. Furst. PSPACE survives constant-width bottlenecks. *International Journal of Foundations of Computer Science*, 2:67–76, 1991.

[GH92] K. Ganesan and S. Homer. Complete problems and strong polynomial reducibilities. *SIAM Journal on Computing*, 21:733–742, 1992.

[GOP+05] C. Glaßer, M. Ogihara, A. Pavan, A. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. In *30th International Symposium on Mathematical Foundations of Computer Science*, LNCS 3618, pages 387–398, 2005.

[GPSS06] C. Glaßer, A. Pavan, A. L. Selman, and S. Sengupta. Properties of NP-complete sets. *SIAM Journal on Computing*, 36(2):516–542, 2006.

[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.

[GW86] T. Gundermann and G. Wechsung. Nondeterministic Turing machines with modified acceptance. In *Proceedings 12th Symposium on Mathematical Foundations of Computer Science*, volume 233 of *Lecture Notes in Computer Science*, pages 396–404. Springer Verlag, 1986.

[HLS+93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. W. Wagner. On the power of polynomial time bit-reductions. In *Proceedings 8th Structure in Complexity Theory*, pages 200–207, 1993.

[HS01] S. Homer and A. Selman. *Computability and Complexity Theory*. Texts in Computer Science. Springer, New York, 2001.

[JL95] D. W. Juedes and J. H. Lutz. The complexity and distribution of hard problems. *SIAM Joutnal on Computing*, 24:279–295, 1995.

[Lac67] A. H. Lachlan. The priority method. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 13:1–10, 1967.

[Lad73a] R. Ladner. Mitotic recursively enumerable sets. *Journal of Symbolic Logic*, 38(2):199–211, 1973.

[Lad73b] R. E. Ladner. A completely mitotic nonrecursive r.e. degree. *Trans. American Mathematical Society*, 184:479–507, 1973.

[OW91] 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.

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

[Sch86] U. Schöning. *Complexity and Structure*, volume 211 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin Heidelberg, 1986.

[Sch89] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences*, 39:84–100, 1989.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[SW98] H. Schmitz and K. W. Wagner. The Boolean hierarchy over level 1/2 of the Straubing-Thèrien hierarchy. Technical Report 201, Inst. für Informatik, Univ. Würzburg, 1998.

[TFL93] S. Tang, B. Fu, and T. Liu. Exponential-time and subexponential-time sets. *Theoretical Computer Science*, 115(2):371–381, 1993.

[Tra70] B. Trahtenbrot. On autoreducibility. *Dokl. Akad. Nauk SSSR*, 192, 1970. Translation in Soviet Math. Dokl. 11: 814– 817, 1970.

[Tra95] N. Tran. On P-immunity of nondeterministic complete sets. In *Proceedings of the 10th Annual Conference on Structure in Complexity Theory*, pages 262–263. IEEE Computer Society Press, 1995.

[Tra02] S. Travers. Blattsprachen Komplexitätsklassen: Über Turing-Abschluss und Counting-Operatoren, Studienarbeit, 2002.

[Val76] L. G. Valiant. Relative complexity of checking and evaluation. *Information Processing Letters*, 5:20–23, 1976.

[Wag04] K. W. Wagner. Leaf language classes. In *Proceedings International Conference on Machines, Computations, and Universality*, volume 3354 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.

[Wra77] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1977.

[Yao90] A. Yao. Coherent functions and program checkers. In *Proceedings of the 22nd Annual Symposium on Theory of Computing*, pages 89–94, 1990.