Separating Complexity Classes Using Structural Properties

Harry Buhrman Leen Torenvliet

November 26, 2003

Abstract

We study the robustness of complete sets for various complexity classes. A complete set A is robust if for any $f(n)$-dense set $S \in \mathrm{P}$, $A - S$ is still complete, where $f(n)$ ranges from log(n), polynomial, to subexponential. We show that robustness can be used to separate complexity classes:

- for every $\leq_m^p$-complete set A for EXP and any subexponential dense sets $S \in \mathrm{P}$, $A - S$ is still Turing complete and under a reasonable hardness assumption even $\leq_m^p$-complete.
- For EXP and the delta levels of the exponential hierarchy we show that for every Turing complete set A and any log-dense set $S \in \mathrm{P}$, $A - S$ is still Turing complete.
- There exists a 3-truth-table complete set A for EEXPSPACE, and a log-dense set $S \in \mathrm{P}$ such that $A - S$ is not Turing complete. This implies that settling this issue for EEXP will either separate P from PSPACE or PH from EXP.
- We show that the robustness results for EXP and the delta levels of the exponential hierarchy do not relativize.

1 Introduction

Proving good lower bounds is *the* main problem in computational complexity theory. It is for example widely believed that $\mathrm{P} \neq \mathrm{NP}$ and that SAT, the set of satisfiable formulas, requires super polynomial time, and probably exponential time. It is even believed that SAT requires exponential size non-uniform circuits. Unfortunately all these problems remain elusive and no super *linear* lower bound for SAT is known. It is even not known whether EXP, the class of languages that can be decided in exponential time, requires super polynomial size non-uniform circuits. A large body of work has tried to tackle these question via combinatorial properties with moderate success. Only for very restricted computational models, like for example constant depth circuits, good lower bounds can be proven. See e.g., [All96, HO02, Vol98].

A big problem that seems to stand in the way of proving for example $\mathrm{P} \neq \mathrm{NP}$ is relativization. Baker, Gill, and Solovay [BGS75] proved in the 70's that there exist relativized worlds where $\mathrm{P} = \mathrm{NP}$ and others where $\mathrm{P} \neq \mathrm{NP}$. Hence a proof that would settle the P versus NP problem, should not relativize. However most proof techniques, including the combinatorial ones mentioned above, relativize and non relativizing techniques are few and far between.

In this paper we follow a different route for proving lower bounds. We use the following strategy. Suppose we want to separate complexity class A from B. We first define some structural property P. Next we demonstrate that class A has the property and that B does not have it.

We can then conclude that A $\neq$ B. It was Post [Pos44] who proposed this line of attack in computability theory and though the property that eventually came out was arguably not as simple as Post envisioned [Deg73], the program *was* successful. Similar proposals have been made before in computational complexity theory [Sel79, BvMFT00]. Their status: the jury is still out.

The property that we study in this paper is the *robustness* of complete sets. Take any complete set A, say for EXP, and let S be a set in P. We call A robust if $A - S$ is *still* complete for EXP. Note that if we don't restrict the density of S we could set $S = \Sigma^*$ and in that case $A - S = \emptyset$ is not complete for EXP. We will therefore only be interested in sets of sub-exponential density. It turns out that the density on one hand and the type of completeness, i.e., many-one, bounded-truth table or Turing, on the other gives a good handle to separate complexity classes. Thus our approach is tunable in three dimensions: density, complexity and reduction type, and leads to interesting results in all three directions. In addition we show that our approach does not relativize.

In [BHT98] we showed for any $\leq_m^p$-complete set A for EXP and for any polynomially dense set, also sometimes called sparse, $S \in$ P, that $A - S$ is still $\leq_m^p$-complete for EXP. We examine here what happens if the density of $S \in$ P is subexponential, $\|S\| \leq 2^{n^\epsilon}$ for any $\epsilon > 0$. We show that if there exist sets in E that require $2^{\Omega(n)}$-sized circuits, then $A - S$ is still $\leq_m^p$-complete. We show a relativized world where $A - S$ is not even $\leq_{tt}^p$-complete for EXP anymore. We show unconditionally that $A - S$ remains Turing complete. The same statement is true for 2-query truth-table complete sets for EXP. The situation changes dramatically for 3-truth-table complete sets. We exhibit a 3-truth-table complete set A and a $\log(n)$-dense set $S \in$ P such that $A - S$ is not n^c-truth-table-complete for EXP. For a relativized world we extend this so that $A - S$ is not even Turing-complete anymore. However we show, generalizing and extending the techniques from [BvMFT00], that for any Turing complete set B for EXP, in particular the set A from the previous sentence, and for any log-dense set $S \in P$, $B - S$ is still Turing complete for EXP. This shows that this technique does not relativize, very similar to the results in [BvMFT00]. As noted in [BHT98] extending this to sparse sets in P would demonstrate that EXP is not in P/*poly*. We are also able to show that a similar robustness statement is true for the Delta levels of the exponential hierarchy.

The ideas from the oracle construction can be used to unconditionally show that there exists a set $A \in$ EEXPSPACE, the class of sets that require double exponential space, that is 3-truth table complete, and a log-dense set $S \in$ P, such that $A - S$ is not Turing complete for EEXPSPACE. This indicates that settling this robustness issue for Turing complete sets for EEXP will separate complexity classes: if EEXP has a Turing complete set A and there is a log-dense set $S \in$ P such that $A - S$ is not Turing complete for EEXP, then EEXP $\neq \Delta_k^{EXP}$, and by padding, EXP is not equal to the polynomial time hierarchy. On the other hand if for all 3-truth table complete sets A for EEXP and all log-dense sets $S \in P$, $A - S$ is still Turing complete, then EEXP is not equal to EEXPSPACE, and by padding P $\neq$ PSPACE.

2 Definitions and Notation

Strings are elements of $\{0,1\}^*$. Let $|x|$ denote the length of string x and let $\|S\|$ denote the cardinality of the set S. $S^{=n}$ is notation for $\{x \in S \mid |x| = n\}$. Thus, $\|S^{=n}\|$ is a function that for argument n returns the cardinality of set S at length n. $\|S^{\leq n}\|$ is defined in the same way. We assume a pairing function $\langle .,. \rangle$ that is computable in $O(n^2)$ time and such that $|\langle x,y \rangle| \leq |x| + |y| + 2\log|x|$. Such functions are abundant, e.g., if s is a self delimiting code then $s(x)y$ fits this purpose. We need this bound later on to ensure that for any x and y, $|x| + |y| \leq |\langle x,y \rangle| \leq |x| + |y| + 2\log|x| + 1$.

For strings x and y let xy denote the concatenation of x and y, and let x_i denote the ith bit of x. Let $x_{i\dots j}$ denote the string obtained by concatenating the ith through jth bits of x. Let $f_x(k)$ be a function that returns the kth bit of x. We use the notation $\langle\!\langle f_x(k) \mid i \leq k \leq j \rangle\!\rangle$ for $x_{i\dots j}$. We adopt the convention that $x_i = \lambda$ for $i > |x|$ so that we can use this notation as $\langle\!\langle f_x(k) \mid k \rangle\!\rangle = x$. Turing machines can output strings or bits. By convention we denote a(n oracle) Turing machine M that accepts input x by $M(x) = 1$. Likewise we denote rejection by $M(x) = 0$.

A set S will be called $f(n)$-dense if for almost all x the number of elements of length less than or equal to n is bounded by $f(n)$. In particular, a set S is said to be *log-dense* iff $(\forall^\infty n)[\|S^{\leq n}\| \leq \log n]$, polynomially dense (or sparse) iff there is a polynomial p such that $(\forall^\infty n)[\|S^{\leq n}\| \leq\ p(n)]$, and of *sub-exponential density* iff $(\forall \epsilon > 0)(\forall^\infty n)[\|S^{\leq n}\| \leq 2^{n^\epsilon}]$.

We assume the reader familiar with most commonly used complexity classes like, e.g., P, NP, EXP and classes in the polynomial time hierarchy. In this paper we also study EEXP, the class of sets computable in DTIME($2^{2^{p(n)}}$), EEXPSPACE, the class of sets computable in DSPACE($2^{2^{p(n)}}$) and the delta-levels of the exponential time hierarchy defined as $\Delta_{k+1}^{\mathrm{exp}} = \mathrm{EXP}^{\Sigma_k^p}$

Define $K = \{\langle i, x, t\rangle \mid M_i \text{ accepts } x \text{ in } \leq t \text{ steps }\}$. Then K can easily be seen to be complete for EXP. Define $K' = \{\langle y, x\rangle \mid x \in K\}$. As K' is in EXP and K reduces to K', K' is also complete for EXP. We will assume a programming system in which all exponential time machines are *clocked*. That is, we take a recursive presentation of EXP $= \{e_i\}_i$ in which each e_i is time bounded by an exponential function described by e_i. For ease of notation we will assume that for almost all n, e_i is time bounded by $2^{n^{e_i}}$, where n is the length of the input.

3 Exploring the Limits

In [BHT98] it is shown that a many-one exponential time complete set A remains complete if a polynomial time computable sparse set S is removed. The proof is quite simple. Many-one complete sets are complete under 1-1 length increasing reductions, and are, by that same fact, also superpolynomially dense. Hence for any string x there must be a string y of length $|x|$ such that $f_{K'}(\langle y, x\rangle) \notin S$. Moreover, if $\|S^{=n}\|$ is bounded by n^s and $f_{K'}$ is n^k time bounded. such a string y can be found within the lexicographically first $|\langle y, x\rangle|^{ks}$ strings of length $|x|$.

This the density bound cannot be improved to anything approaching exponential density. In particular the following holds.

Observation 1 *For any fixed $\epsilon > 0$ there exists a $\leq_m^p$-complete set A for* EXP *and a polynomial time computable set S with $\|S^{=n}\| \leq 2^{n^\epsilon}$ such that $A - S$ is not many-one complete for* EXP.

Proof. Let $\epsilon > 0$. Let $S = \{\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle \mid x \in \Sigma^*\}$. Let $K_\epsilon = \{\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle \mid x \in K\}$. Then clearly K_ϵ is $\leq_m^p$-complete for EXP. Now $K_\epsilon - S = \emptyset$ and so $K_\epsilon - S$ is not many-one complete for EXP by the hierarchy theorem. All that is left to prove is that $\|S^{=n}\| \leq 2^{n^\epsilon}$. Now S only has strings of the form $\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle$. So if $|x| \neq |z|$ then $|\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle| \neq |\langle y, 0^{|y|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle|$. This means that S has *exactly* $2^{|x|}$ strings at length $|\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle|$. Since $|\langle x, 0^{|x|^{\lceil \frac{1}{\epsilon} \rceil}}\rangle| \geq |x|^{\frac{1}{\epsilon}+1}$ it follows that $\|S^{=|x|^{\frac{1}{\epsilon}+1}}\| \leq 2^{|x|}$ or that $\|S^{=|x|^{\frac{1}{\epsilon}}}\| \leq 2^{|x|}$ or that $\|S^{=n}\| \leq 2^{n^\epsilon}$ □

For Turing complete sets the situation is quite different, since it is not known whether Turing complete sets are superpolynomially dense. In fact, as remarked in [BHT98], if EXP has small circuits, then EXP also has a tally complete set A. Clearly then the set $A - \{0\}^*$ is no longer complete for EXP.

4 Many-one complete sets

In this section we study the many-one complete sets for EXP. We will see that they remain complete even if we take out a subexponential dense set $S \in$ P. Note that because of Observation 1 this is optimal.

4.1 Derandomization

We will first show that for a many-one complete set A for EXP, and subexponential dense set $S \in$ P, $A - S$ is complete under a randomized many-one reduction. We will then show how to derandomize this reduction under the hypothesis that there is a set in E that requires $2^{\Omega(n)}$-sized circuits.

We will need the following Lemma due to Berman [Ber77].

Lemma 1 ([Ber77]) *Every many-one complete set for* EXP *is complete via a reduction that is* 1-1.

Theorem 1 *Let A be a many-one complete set for* EXP *and let $S \in$* P *have sub-exponential density. If there is a set in* E *that requires $2^{\Omega(n)}$-sized circuits, then $A - S$ is many-one complete for* EXP.

Proof. Let A be a many-one complete set for EXP. Recall the definition of $K' = \{\langle y, x\rangle \mid x \in K\}$. Let $f_{K'}$ be the 1-1 reduction that witnesses that $K' \leq_m^p A$ by Lemma 1. It is easy to see that, because of the 1-1 reduction, for any x of length n the $\text{Prob}_{y\in\{0,1\}^n}[f_{K'}(\langle y, x\rangle) \notin S] \geq 1 - 1/2^{n^\delta}$ for $0 < \delta < 1$.

Now if we want to decide whether $x \in K$ and we would be able to find in polynomial time a y such that $f_{K'}(\langle y, x\rangle) \notin S$, we have that $x \in K$ iff $f_{K'}(\langle y, x\rangle) \in A - S$ and we are done.

Next we use the assumption that there is a set in E that requires $2^{\Omega(n)}$-sized circuits to find for every x in polynomial time a y such that $f_{K'}(\langle y, x\rangle) \notin S$. Impagliazzo and Wigderson [IW97] showed that this assumption implies the existence of a pseudo random generator $G : \{0,1\}^l \mapsto \{0,1\}^n$, with $l = O(\log n)$, G computable in time exponential in its input size, and for any circuit C of size n:

$$|\Pr_{z\in\{0,1\}^n}[C(z) = 1] - \Pr_{a\in\{0,1\}^l}[C(G(a)) = 1]| \leq \frac{1}{n}$$

Next construct a circuit that simulates the following polynomial time computation: inputs are of the form $z = uv$, with $|u| = n$, $|z| = m$. Ignore v and accept iff $f_{K'}(\langle u, x\rangle) \notin S$. Let the size of this circuit be m (which is some polynomial in n). The probability that this circuit accepts an input is much bigger than $\frac{1}{2}$, and hence by the property of the pseudo random generator G, the probability that $C(G(a)) = 1$ is at least $\frac{1}{2} - \frac{1}{m}$. This implies that for at least one of the $a \in \{0,1\}^l$, $G(a)$ produces a strings $z = uv$, with u such that $f_{K'}(\langle u, x\rangle) \notin S$. □

In the next subsection we will show that we can do without the assumption that there is a set in E that requires $2^{\Omega(n)}$-sized circuits. As it turns out, $A - S$ is still complete under Turing reductions.

4.2 Searching for a good y

In the proof of the previous theorem we used that for any x there is a y of length $|x|$ such that $f_{K'}(\langle y, x\rangle) \notin S$. In fact *most* y of length $|x|$ have the property that for *any* z of length $|x|$ it holds that $f_{K'}(\langle y, z\rangle) \notin S$. We will again make use of this fact. We design a query machine that finds such a string y by recursive search, querying an exponential time machine that searches for these strings by exhaustive search. At the bottom of the recursion, the strings are so small that a polynomial time machine can search them all and produce a string that can be used for the next higher level to query A outside S, using $f_{K'}$.

Theorem 2 *Let A be a many-one complete set in* EXP*. Let S be a polynomial time computable set of sub-exponential density. The set $A - S$ is Turing complete for* EXP

Proof. Let $f_{K'}$ be a 1-1 function reducing K' to A. Assume that $f_{K'}$ runs in time n^k for some k and almost all n. Assume furthermore that computing membership in S of a string x takes time $|x|^s$ for some constant s and almost all x. Fix some x large enough. Now, $x \in K$ if and only if $(\forall y)[|y| = |x| \implies \langle y, x\rangle \in K']$. Since S is of sub-exponential density and $f_{K'}$ is 1-1, $(\forall^\infty x \exists y)[|y| = |x| \ \&\ f_{K'}(\langle y, x\rangle) \notin S]$.
Consider the following machines:

M_E:
Input $\langle m, x, i\rangle$;
$\ell = 2^{(2|x|)^m}$;
$\ell' = 2^{(2|x|+1))^m}$;
$y = \min\{z \mid |z| = |x| \ \&$

$$(\forall v, j)[|x| \le |v| \le |x|+1, j \le |x|] \Rightarrow \begin{cases} f_{K'}(\langle z, \langle m, xv, \ell\rangle\rangle) \notin S \\ f_{K'}(\langle z, \langle m, xv, \ell'\rangle\rangle) \notin S \\ f_{K'}(\langle z, \langle m, \langle m, xv, j\rangle, \ell\rangle\rangle) \notin S \\ f_{K'}(\langle z, \langle m, \langle m, xv, j\rangle, \ell'\rangle\rangle) \notin S \end{cases}$$

};
return y_i
end.

M_E finds for given input strings x bits of a string y such that $f_{K'}(\langle y, z\rangle) \notin S$, where x is a prefix of z and $\lfloor |z|/2 \rfloor = |x|$. Moreover, it identifies a single string y that works for a whole slew of such strings z. The first set of strings that M_E searches for is the set of strings that work for the top level of the recursion that follows. In the second part of the program M_E searches for strings that fit the recursive calls. First we prove that M_E indeed finds such a string and next we prove that M_E finds such a string in limited exponential time.

CLAIM 1 *$M_E(\langle m, x, i\rangle)$ is the i-th bit of a string y_x of length $|x|$ such that* $(\forall v, j)[|x| \le |v| \le |x|+1, j \le |x|] \Rightarrow \begin{cases} f_{K'}(\langle y_x, \langle m, xv, \ell\rangle\rangle) \notin S \\ f_{K'}(\langle y_x, \langle m, xv, \ell'\rangle\rangle) \notin S \\ f_{K'}(\langle y_x, \langle m, \langle m, xv, j\rangle, \ell\rangle\rangle) \notin S \\ f_{K'}(\langle y_x, \langle m, \langle m, xv, j\rangle, \ell'\rangle\rangle) \notin S \end{cases}$, *where* $\ell = 2^{(2|x|)^m}$ *and* $\ell' = 2^{(2(|x|+1))^m}$

Proof. First note that since M_E searches for a lexicographically minimum string with the desired properties, and since this search is independent of input i, the bits returned by M_E, if any, indeed belong to a single string. Now we just need to prove, by counting, that such a string exists. The string y is of length $|x|$ and there are $2^{|x|}$ of these strings. Suppose that none of these strings satisfies all requirements. That is, for every y of length $|x|$ there is a v_y with $|x| \leq |v_y| \leq |x|+1$ and a $j \leq |x|$ such that at least one of the four non-membership statements is violated.

As both $f_{K'}$ and the pairing function are 1-1 by assumption, every violation of one of these membership statements requires a different string in S for different y. As $f_{K'}$ is computable in time n^k, and for all v and w, $|\langle v, w\rangle| \leq |v| + |w| + 2\log|v|$, by assumption on the pairing function *and* $|2^{|v|^m}| = |v|^m$, the length of the input to $f_{K'}$ is less than, roughly, $(4|x|)^m$ for almost all x in all four cases. This means that for almost all x the length of the *output* of $f_{K'}$, i.e., the length of the strings tested for membership in S is less than $(4|x|)^{mk}$. At these length S has less than $2^{(4|x|)^{mk\epsilon}}$ strings for any ϵ, which is less than $2^{|x|}$ for almost all x if we take $\epsilon < \frac{1}{mk}$. $\Box$

M_E has a program, let us call it m_e. We need to show that M_E is an exponential time computable algorithm since we want, later on in this proof, to decide the output of M_E on input x by reducing a string $\langle m_e, x, t\rangle$ to K, or equivalently a string $\langle y, \langle m_e, x, t\rangle\rangle$ to K'. We can show that in fact M_E runs in almost linear exponential time.

CLAIM 2 *M_E runs in time $O(2^{(1+o(1))n})$ where n is the length of the input.*

Proof. For almost all x, the input $\langle m, x, i\rangle$ is at least of length $|x|$. We show that for any $\epsilon > 0$ and for almost all x, M_E is time bounded by $2^{|x|(1+\epsilon)}$. M_E searches a set of strings of cardinality bounded by $2^{|x|}$. For each of these strings it tests whether a set of strings in cardinality bounded by $2(2^{|x|} + |x|2^{|x|})$ satisfies the property that the function $f_{K'}$ has an image outside S. $|2^{(2|x|+1)^m}| \leq (3|x|)^m$ for almost all x. Then, for almost all x, to perform this check, $f_{K'}$ has to be computed on inputs of length bounded by $(4|x|)^m$. The machine computing $f_{K'}$ itself is n^k bounded, so generating a string for testing membership in S takes at most $(4|x|)^{mk}$ time. One such computation produces a string of length at most $(4|x|)^{mk}$ bits, which can be checked for membership in S at a cost of $(4|x|)^{mks}$ steps. We note that since for any $\epsilon > 0$ and almost all x the set S has less than $2^{(4|x|)^{mk\epsilon}}$ strings, not all $2^{|x|}$ strings have to be searched, but for almost all x and any $\delta > 0$ the search will be successful after examining $2^{|x|^\delta}$ strings. Summing up we find that for almost all x and for any $\delta > 0$ the running time is bounded by $2^{|x|^\delta} \times (2(2^{|x|} + |x|2^{|x|})) \times ((4|x|)^{mks} + (4|x|)^{mks})$ steps. $\Box$

Next we consider a recursive oracle machine, that on input z by querying A polynomially often, produces a string y such that $f_{K'}(\langle y, z\rangle) \notin S$. This string y can then be used for computing membership of z in K by querying $A - S$. In fact, the following program can inductively be shown never to query strings in S, so that the whole computation can be carried out with queries to $A - S$. The program of M_E is used to compute queries to A.

M_Q^A
Input $\langle\langle m,x\rangle,i,b\rangle$;
If $|b|\geq 2^{2|x|}$ then
$\ell = 2^{(2|x|)^m}$;
$\ell' = 2^{(2|x|+1)^m}$;
$y=\min\{z \mid |z|=|x|$ &

$$(\forall v,j)[|x|\leq|v|\leq|x|+1, j\leq|x|] \Rightarrow \begin{cases} f_{K'}(\langle z,\langle m,xv,\ell\rangle\rangle)\notin S \\ f_{K'}(\langle z,\langle m,xv,\ell'\rangle\rangle)\notin S \\ f_{K'}(\langle z,\langle m,\langle m,xv,j\rangle,\ell\rangle\rangle)\notin S \\ f_{K'}(\langle z,\langle m,\langle m,xv,j\rangle,\ell'\rangle\rangle)\notin S \end{cases}$$

}
return y_i
else
$h=\lfloor\frac{|x|}{2}\rfloor$; $v=x_{1\ldots h}$; $w=\langle m,x,i\rangle$; $r=\lfloor\frac{|w|}{2}\rfloor$; $w'=w_{1\ldots r}$; $y=\lambda$;
for $j=1$ to $|v|$ do
If $M_Q(\langle m_e,w'\rangle,j,b)=1$ then $y=y1$ else $y=y0$;
if $f'_K(\langle y,\langle m_e,w,2^{|x|^{m_e}}\rangle\rangle)\in A$ then return 1
else return 0
end.

First, it is clear that M_Q runs in polynomial time relative to oracle A since it can never have more than a linear number of recursive calls and the base of the recursion performs a linear number of computations of $f_{K'}$ and checks on membership in S. Next, we claim that $M_Q^A(\langle z,i,b\rangle$ returns the ith bit of a string y with the desired properties, that is we prove the following.

CLAIM 3 *$M_Q^A(\langle\langle m,x\rangle,i,b\rangle)$ returns the ith bit of a string y of length $|x|$ such that for all v with $|x|\leq|v|\leq|x|+1$, and $j\leq|x|$*

1. $f_{K'}(\langle y,\langle m,xv,2^{(2|x|)^m}\rangle\rangle)\notin S$
2. $f_{K'}(\langle y,\langle m,xv,2^{(2|x|+1)^m}\rangle\rangle)\notin S$
3. $f_{K'}(\langle y,\langle m,\langle m,xv,j\rangle,2^{(2|x|)^m}\rangle\rangle)\notin S$
4. $f_{K'}(\langle y,\langle m,\langle m,xv,j\rangle,2^{(2|x|+1)^m}\rangle\rangle)\notin S$

Proof. The proof is by induction on the depth of the recursion. Clearly, if $2|x|\leq\log|b|$ then M_Q explicitly searches for a string fitting these conditions. Moreover, it does so in approximately linear time by a proof similar to the proof of Claim 2. Note that we explicitly take $|b|\geq 2^{\mathbf{2}|x|}$ to avoid another exponential factor. Let us assume as an induction hypothesis that the recursive calls $M_Q(\langle m_e,\langle v,j,b\rangle,2^{|v|^{m_e}}\rangle)$ build a string y of length $|v|$ such that: for all w with $|v|\leq|w|\leq|v|+1$, and $j\leq|v|$

1. $f_{K'}(\langle y,\langle m,vw,2^{(2|v|)^m}\rangle\rangle)\notin S$
2. $f_{K'}(\langle y,\langle m,vw,2^{(2|v|+1)^m}\rangle\rangle)\notin S$
3. $f_{K'}(\langle y,\langle m,\langle m,vw,j\rangle,2^{(2|v|)^m}\rangle\rangle)\notin S$

4. $f_{K'}(\langle y, \langle m, \langle m, vw, j\rangle, 2^{(2|v|+1)^m}\rangle\rangle) \notin S$

Then in particular, since w' consists of the first $\lfloor \frac{|w|}{2} \rfloor$ bits of w and $2^{|x|^{m_e}}$ is either $2^{(2|w'|)^{m_e}}$ or $2^{(2|w'|+1)^{m_e}}$, depending on whether $|x|$ is even or odd, the string y has the property that $f_{K'}(\langle y, \langle m_e, w, 2^{|x|^m}\rangle\rangle) \notin S$, or that $f_{K'}(\langle y, \langle m_e, w, 2^{|x|^m}\rangle\rangle) \in A - S$, which is the case if and only if $f_{K'}(\langle y, \langle m_e, w, 2^{|x|^m}\rangle\rangle) \in A - S$ if and only if M_E accepts w. The claim then follows from Claim 1. □

Finally consider the following program.

```
M_R^A
    Input z = ⟨e, x⟩;
    z' = ⟨e, x, 2^{|x|^e}⟩;
    y = λ;
    for i = 1 to |z'| do
    If M_Q(⟨z, i, |z|⟩) then y = y1 else y = y0;
    If f_K'(⟨y, z'⟩) ∈ A accept else reject
    end.
```

It follows from Claim 3 that $M_Q(\langle z, i, |z|\rangle)$ returns the ith bit of a string y such that $f_{K'}(\langle y, z'\rangle) \notin S$. From the definition of K' we have that $f_{K'}(\langle y, z'\rangle) \in A$ if and only if z' in K'. Since $f_{K'}(\langle y, z'\rangle) \notin S$ it follows that $f_{K'}(\langle y, z'\rangle) \in A - S$ if and only if $z' \in K$. All statements hold only for large enough x. So for fixed exponential time machine e and almost all x, $M_R(\langle e, x\rangle)$ accepts if and only if $\langle e, x, 2^{|x|^e}\rangle \in K$, which is true if and only if e accepts x. This concludes the proof. □

Note that the construction of the proof does *not* give an explicit reduction from K to $A - S$, since all statements only hold for all but finitely many x given a program e. This may mean that the construction may be mistaken on an infinite set of strings of the form $\langle e, x, 2^{|x|^e}\rangle$. However it is guaranteed that for a fixed e the number of mistakes is finite, therefore we can use the construction to reduce any set in exponential time (including K) to $A - S$.

The proof of Theorem 2 clearly relativizes. The question is therefore legitimate how far we can go down to many-one reductions with relativizing proofs. The following Theorem shows that, at least with relativizing techniques Theorem 2 is optimal.

Theorem 3 *There exist an oracle A and sets B^A and S^A such that B^A is many-one complete for* EXP^A*, S^A is of sub-exponential density and $S^A \in \mathrm{P}^A$, and $B^A - S^A$ is not tt-complete for* EXP^A.

Proof. The proof is by diagonalization. We construct a set B such that K^A is encoded in B^A, so that B^A is many-one complete. S^A is the set $\{x \mid x \in A\}$, so that $S^A \in \mathrm{P}^A$. Then the construction just needs to meet the requirements that: (1) $B^A - S^A$ is not tt-complete for EXP^A, which is done by diagonalizing against the ith tt-reduction at appropriate times on input $0^{b(i)}$ for some rapidly growing function $b(i)$ of i, and (2) that S^A remains of sub-exponential density, which is achieved by adding only strings queried by the ith reduction. Assume an enumeration $\{M_i\}_i$ of polynomial time truth-table reductions that can (adaptively) query an oracle before or during the computation of the truth-table. These machines will effectively work with two oracles A and B where A is the oracle queried directly and B is the oracle that is used for evaluating the truth table. The queries to A may be adaptive, but the queries to B must be non-adaptive. In particular, the queries to B are answered only after all queries to A have been made. The set B itself may depend on A, so that in fact M_i queries A and B^A. We denote this by M_i^{A,B^A}. We assume that M_i is time bounded by n^i for almost all inputs x.

We present the diaganolization in stages:

> **Stage** 0: $A_0 = \emptyset$; $B_0 = \emptyset$;
> **Stage** $i+1$:
> Let $b(i) = \min\{n \mid n > 2^{b(i-1)^{i-1}} \wedge 2^n > n^i\}$;
> $C = \emptyset$;
>
> Run $M_i^{A_i \cup C, \emptyset}(0^{b(i)})$ until M_i produces a next query q;
> **If** $|q| > b(i)$ add q to C and continue in the YES state
> **else** query q to A_i and continue in the corresponding state
> **until** $M_i^{A_i \cup C, \emptyset}$ halts;
>
> $A_{i+1} = A_i \cup C \cup \{y \mid y$
> is in the truth-table produced by $M_i^{A_i \cup C, \emptyset}(0^{b(i)}) \wedge |y| > b(i)\}$;
> $B_{i+1}^{A_{i+1}} = B_i^{A_i} \cup \{\langle 1, x\rangle \mid x \in K^{A_{i+1}} \wedge b(i-1)^{i-1} < |x| \leq b(i)^i\}$;
> $B_{i+1}^{A_{i+1}} = B_{i+1}^{A_{i+1}} \cup \{0^{b_i}\}$ if and only if $M_i^{A_{i+1}, B_{i+1}^{A_{i+1}} - A_{i+1}}(0^{b(i)})$ rejects.
> **end of construction**.

We assume wlog that $(\forall x, i)[0^{b(i)} \neq \langle 1, x\rangle]$. First, it is clear that B^A is exponential-time hard, since the strings of K^A are coded consecutively in B^A in such a way that strings added at stage $i+1$ do not influence the coding phase at stage i, i.e., $(\forall x)[(\exists i)[\langle 1, x\rangle \in B^{A_i}]$ iff $\langle 1, x\rangle \in B^A$ iff $x \in K^A]$. Next, $B^A \in EXP^A$ since computing $\langle 1, x\rangle \in B^A$ requires computing $x \in K^A$, which is linear exponential time, and computing $0^{b(i)}$ in B^A requires simulation of M_i on input 0^{b_i} with queries up to length $b(i)^i$ which can be done in linear exponential time relative to A because of the choice of $b(i)$.

Finally, as $S^A = \{x \mid x \in A\}$, $B^A - S^A$ is not tt-complete for EXP^A for if it were, there would be some tt-reduction M_i reducing B^A to $B^A - S^A$. However $0^{b(i)} \in B^A$ if and only if $M_i^{A_{i+1}, B_{i+1}^{A_{i+1}} - A_{i+1}}(0^{b(i)}) = M_i^{A, B^A - A}(0^{b(i)}) = M_i^{A, B^A - S^A}(0^{b(i)}) = 0$, a contradiction. □

Using the above ideas in combination with the techniques developed in [BHT98], we are able to show that similar results hold for 2 truth table reductions: if A is 2 truth-table complete for EXP then $A - S$ is still 2 truth-table complete, if we assume hard sets in E exists, and unconditionally Turing complete. However the proof is very long and consists of an elaborate case analysis. The details will appear in the final version of this paper.

5 Turing Complete Sets

In this section we make extensive use of the fact that the bits of an exponential time computation are themselves computable by an exponential time machine and can thus be retrieved by querying an exponential time complete set. This observation has been made before and forms the core observation of one part of [BvMFT00]. It is also the starting point of the proof of the theorem that follows, and the proof is relatively short and illuminating, so we isolate this in the following Lemma.

Lemma 2 *Let A be an exponential time complete set. Then for any x, $x \in A$ can be computed by an oracle machine that uses oracles $A - \{x\}$ and $A \cup \{x\}$.*

Proof. Let $M_A(x) = x \in A$ and M_A runs in time $2^{|x|^c}$. Let $M_B(\langle x, i, j \rangle) = \langle q, b \rangle$, where q is the state of M_A on input x at time i and b is the tape cell contents of M_A on input x at time i at position j. Without loss of generality we assume that q also carries information about the tape head being present or not at time i in position j. Let $M_{B'}$ be the machine that can be used as an oracle to reconstruct the output of M_B. $M_B(\langle x, i, j \rangle) = \langle\langle M_{B'}(\langle x, i, j, k \rangle) \mid k \rangle\rangle$ As $M_{B'}$ is also an exponential time machine, there is a polynomial time query machine Q that decides $M_{B'}(\langle x, i, j, k \rangle) = 1$ with the help of oracle A, that is $Q^A(\langle x, i, j, k \rangle) = M_{B'}(\langle x, i, j, k \rangle)$. Define M_P, the prover, to be the polynomial time oracle machine that with the help of oracle $A - \{x\}$ simulates Q^A. That is, M_P runs the program of Q, asking queries of $A - \{x\}$. Whenever x is queried, M_P directly continues in the NO state instead of querying x. Note that, if $x \notin A$, then for any input $\langle x, i, j, k \rangle$, $M_P^{A-\{x\}}$ returns the correct answer. However if $x \in A$, then either $M_P^{A-\{x\}}(\langle x, i, j, k \rangle)$ is the k-th bit of $\langle \text{REJECT}, 0 \rangle$ for $i \geq 2^{|x|^d}$ or there is an inconsistency in the bits returned by the computation. That is, there is some pair in$\{(\langle i, j \rangle, \langle i-1, \ell \rangle) \mid \ell \in \{j-1, j, j+1\}\}$ such that $\langle\langle M_P^{A-\{x\}}(\langle x, i, j, k \rangle) \mid k \rangle\rangle$ is not consistent with $\langle\langle M_P^{A-\{x\}}(\langle x, i-1, \ell, k \rangle) \mid k \rangle\rangle$ according to the program of M_A. The program M_C that, on input x, checks $M_P^{A-\{x\}}(\langle x, i, j, k \rangle)$ for all relevant values of x, i, j, and k and outputs this inconsistent pair $\langle i, j \rangle$ or returns all zeros if this pair is nonexistent is *also* an exponential time computable machine. So, we can define a machine $M_{C'}$ such that $\langle\langle M_{C'}(\langle x, k \rangle) \mid k \rangle\rangle = M_{C'}$. Let Q' be a polynomial time query machine that decides $M_{C'}(\langle x, k \rangle) = 1$ with the help of oracle A. Let M_F, the falsifier, be the oracle machine that with the help of oracle $A \cup \{x\}$ on input x computes the bits of $M_C(x)$. $M_F^{A\cup\{x\}}$ simulates Q', but whenever x is queried it immediately continues in the YES state. Then the following situations can occur. If $x \notin A$ then all computations of M_P are correct. No matter which $\langle i, j \rangle$ is computed by M_F, there will be no inconsistency in the computation, so checking $M_P^{A-\{x\}}(\langle x, i, j, k \rangle)$ and $M_P^{A-\{x\}}(\langle x, i-1, \ell, k \rangle)$ for all relevant k and $\ell \in \{j-1, j, j+1\}$ will not reveal such an inconsistency. If $x \in A$, then either $M_P^{A-\{x\}}(\langle x, i, j, k \rangle)$ returns the bits of $\langle \text{ACCEPT}, 0 \rangle$ for $i \geq 2^{|x|^d}$ or there is an inconsistency somewhere in the computation reconstructed by $M_P^{A-\{x\}}$. As $M_F^{A\cup\{x\}}(x, k)$ is now correct for all k, it will find this pair $\langle i, j \rangle$ and checking $M_P^{A-\{x\}}(\langle x, i, j, k \rangle)$ and $M_P A - \{x\}(\langle x, i-1, \ell, k \rangle)$ will reveal the inconsistency. $\square$

Now we want to prove that a Turing complete set A remains complete if we take out a log-dense set S instead of a set just missing a single string. The proof of this theorem is an extension of the proof of Lemma 2 in the sense that now machines M_P and M_F can be defined that make assumptions on the membership of a set of strings that cannot be queried instead of just one string. As a polynomial time oracle machine on input x can only query strings that are polynomially larger than $|x|$ and S is log-dense, the number of strings that an oracle machine can query on input x is only logarithmic in $|x|$. Such strings can be in or out of S and in or out of A. If the strings are in S, then they cannot be queried and M_P^{A-S} and M_F^{A-S} then need to make assumptions on membership of such strings in S. We will replace M_P and M_F by a polynomial number of query machines $M_{P[w]}$ and $M_{F[w]}$, where $w \in \{0,1\}^{c \log |x|}$, and $M_{P[w]}$ assumes that $x \notin A$ and that the

i-th query that is *in* S is answered w_i—all other strings can be queried. $M_{F[w]}$ assumes that $x \in A$ and that the ith query that is in S is answered w_i.

The actual proof is a bit more complicated, but this is the rough idea. We now turn to the statement of the theorem and its proof.

Theorem 4 *Let A be Turing complete for exponential time and let S be a polynomial time computable log-dense set. The set $A - S$ is also complete for exponential time.*

Proof. We show the existence of a polynomial time oracle machine M_Q that for almost all x can decide membership in A by adaptive queries to oracle $A - S$.

Fix x large enough and let M_A be the machine that for some constant c decides $x \in A$ in time $2^{|x|^c}$. Let s be a constant such that $\|S^{\leq n}\| \leq s \log n$.

Without loss of generality we assume that any query machine that follows in this proof queries x on input x. Now let M_P be a polynomial time query machine that on input $\langle x, i, j\rangle$ returns the bits $\langle i, j\rangle$ of the computation of M_A on input x as in the proof of Lemma 2 and let M_F, also as in the proof of Lemma 2 be the machine that tries to show an inconsistency in M_A's output for some values $\langle i, j\rangle$. We skip the construction of specific exponential time computable sets that are spelled out in the proof of Lemma 2 for M_P since these can be defined analogously. The sets for M_F are slightly more interesting and therefore we will spend some attention to these sets below. Let us first define a polynomial number of query machines that query $A - (S \cup \{x\})$ as follows for $w \in \{0,1\}^r$, where r is some large enough constant such that S has less than $r \log|x|$ strings up to lengths that can be queried by any of the machines involved.

$M_{P[w]}^{A-(S\cup\{x\})}$: $j = 0$; Repeat simulate the program of M_P until the next query q. If $q \notin S \cup \{x\}$ query q_i. If $q = x$ then continue in the NO state. If $q \in S$ then if $w[j] = 1$ continue in the YES state else continue in the NO state; $j = j + 1$.

Note that all possible settings of the strings that can be queried on input x there is a $w \in \{0,1\}^r$ such that w represents this setting, so if indeed $x \notin A$ then at least one of $M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i, j, k\rangle)$ returns correct and consistent bits for all i, j, and k. If $x \in A$, then there for every $w \in \{0,1\}^r$ either $(\forall i > 2^{|x|^d} \forall j)[M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i, j, k\rangle)]$ is the k-th bit of $\langle \text{ACCEPT}, 0\rangle$, or some pair $\langle i, j\rangle$ and $\langle i{-}1, \ell\rangle$ leads to inconsistent data. For given w checking $M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i, j, k\rangle)$ for *all* relevant $\langle i, j, k\rangle$ searching for an inconsistency is again an exponential time computation, so there exists an exponential time machine M_C that on input $\langle x, w\rangle$ searches for a pair $\langle i, j\rangle$ such that $\langle\langle M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i, j, k\rangle) \mid k\rangle\rangle$ and $\langle\langle M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i-1, \ell, r\rangle) \mid r\rangle\rangle$ are inconsistent with the program of M_A for some $\ell \in \{j-1, j, j+1\}$. Again, there exists a machine $M_{C'}$ that returns the bits of the computation of M_C our oracle machine M_F will again be such that $M_F^A(\langle x, w\rangle) = M_C(\langle x, w\rangle)$.

Define the following query machines $M_{F[w,w']}$ for $w, w' \in \{0,1\}^r$:

> $j = 0$;
> Repeat
> Simulate the program of $M_F(\langle x, w\rangle)$ until the next query q.
> If $q \notin S \cup \{x\}$ query q_i.
> If $q = x$ then continue in the YES state.
> If $q \in S$ then if $w[j] = 1$ continue in the YES state
> else continue in the NO state; $j = j + 1$.

As noted, if $x \notin A$ then there is a w such that $M_{P[w]}(\langle x, i, j, k\rangle)$ returns correct and consistent bits for all i, j, and k. However, if $x \in A$, then every w such that $[M_{P[w]}^{A-(S\cup\{x\})}(\langle x, 2^{|x|^d}, 0, k\rangle)]$ is the k-th bit of $\langle \text{REJECT}, 0\rangle$ or we must have some i and j such that $\langle\langle M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i, j, k\rangle) \mid k\rangle\rangle$ is inconsistent with $\langle\langle M_{P[w]}^{A-(S\cup\{x\})}(\langle x, i, \ell, r\rangle) \mid r\rangle\rangle$ for some $\ell \in \{j-1, j, j+1\}$. *and* there must be some w' such that $M_{F[w',w]}$ uses the correct oracle and demonstrates this inconsistency for w. □

Using the previous Theorem as a base case we can show the following extension, just as in [BvMFT00]. The details will be in the full version of the paper.

Theorem 5 *Let A be Turing complete for Δ_k^{EXP} and let S be a polynomial time computable log-dense set. The set $A - S$ is also complete for Δ_k^{EXP}.*

6 Non-relativization and EEXPSPACE

We will show in this section that Theorem 4 and 5 do not relativize. The construction of the oracle is similar to the one in [BvMFT00], and as in that paper we get an absolute result for EEXPSPACE: There exists a set in $A \in$ EEXPSPACE, that is 3 truth-table complete, and a log-dense set $S \in$ P such that $A - S$ is not Turing complete for EXP anymore.

Theorem 6 *There exists an oracle A and a set B that is Turing complete for* EXP^A *and a log-dense set S^A computable in polynomial time, such that $B^A - S^A$ is not Turing complete for* EXP^A.

Proof. Buhrman et al. [BvMFT00] show that there exists a set complete for EEXPSPACE that is not autoreducible. Details of the construction learn that the set is complete by the reduction that on input x first queries $0^{b(|x|)}$ where b is some fast growing function that divides ranges of diagonalization and then queries either $\langle 0, x\rangle$ or $\langle 1, x\rangle$ depending on the outcome of the first query. It turns out that membership of $0^{b(|x|)}$ and membership of the strings $\langle 0, x\rangle$, $\langle 1, x\rangle$ that can be queried on input $0^{b(|x|)}$ can be set in such a way that if $0^{b(|x|)}$ may not be queried, then the reduction diagonalized against in that region cannot determine membership of $0^{b(|x|)}$ in the set, yet membership in K of a string x can always be coded *either* on $\langle 0, x\rangle$ if $0^{b(|x|)}$ is not put in the set or on $\langle 0, x\rangle$ if $0^{b(|x|)}$ is put in the set, so that the set is indeed complete.

We note the following about their proof. First of all the function b can be taken to be any function, provided that it creates large enough gaps. So the subset $\{0^{b(n)} | n \in \omega\}$ can be taken to meet any given denseness condition, and can be taken to be easily computable. Second, the only reason that the set is very hard to compute is that for each reduction a great many strings have to be examined to determine the so-called type of the reduction, i.e., which decides whether $0^{b(n)}$ is in or out of the set. Now suppose we follow this construction for our set B, but now stage by stage also constructing an oracle A, by putting in a string x in the set and then simultaneously putting $0^{2^{|x|^2}}$ in the oracle. Then the set B becomes:

1. Exponential time computable, since on input x we can query $0^{2^{|x|^2}}$ to determine $x \in B$.
2. Exponential time hard, since K^X is linear exponential time computable for any oracle X, so at any stage coding of K can still be done since $x \in K$ is encoded in the oracle (and therefore in B) at a length that cannot be queried by the linear exponential time computing K^A.
3. Not auto-reducible, since any Turing reduction fails on some string $0^{b(n)}$

Therefore, $(B - \{0^{b(n)} \mid n \in \omega\})^A$ is not complete for EXP^A, since B does not reduce to this set. $\square$

As the oracle constructed in the previous proof itself is EEXPSPACE computable, this also gives.

Theorem 7 *There exists a set in $A \in$ EEXPSPACE, that is 3 truth-table complete, and a log-dense set $S \in$ P such that $A - S$ is not Turing complete for* EEXPSPACE.

7 Conclusions and Open Questions

Theorems 7 and 5 show that the robustness property studied in this paper can be used for proving lower bounds and separating complexity classes. This is because an answer to the question whether the complete sets for EEXP are robust will *either way* separate complexity classes. If on one hand for every 3 truth-table complete set $A \in$ EEXP, and every log-dense set $S \in$ P, $A - S$ is still Turing complete for EEXP then by Theorem 7 it follows that EEXP $\neq$ EEXPSPACE and via padding P $\neq$ PSPACE. However on the other hand if there exists a Turing complete set $A \in$ EEXP, and a log-dense set $S \in$ P, such that $A - S$ is not Turing complete for EEXP by Theorem 5 it follows that for every k, $\Delta_k^{EXP} \neq$ EEXP and by padding that $\Delta_k^P \neq$ EXP.

These results are analogous to the results about autoreducibility and complete sets [BvMFT00]. However studying the Robustness of complete sets has the advantage that we can set several parameters: the type of reduction, the complexity class we study, and the density of the set $S \in$ P. As it turns out for various settings of these parameters we get separation of complexity classes. In addition to the autoreducibility results we get that in order to prove that EXP does not have polynomial size circuits one could show that the Turing complete sets for EXP are robust against removal of a polynomially dense set in P.

With this respect a gap remains between the robustness of Turing complete sets against removal of log-dense sets in P Theorem 4 and poly-dense removal, implying EXP $\notin$ P/poly.

question	yes	no
Are $\leq_m$-complete sets in EXP robust against subexp-dense P sets?	open	EXP $\subseteq 2^{o(n)}$-circuits
Are $\leq_T$-complete sets in EXP robust against sparse P sets?	EXP $\not\subseteq$ P/poly	open
Are $\leq T$-complete sets in EEXP robust against log-dense P sets?	P $\neq$ PSPACE	PH $\neq$ EXP

References

[All96] E. Allender. Circuit complexity before the dawn of the new millenium. In *Proceedings 16th Foundations of Software Technology and Theoretical Computer Science*, volume 1180 of *Lecture Notes in Computer Science*, pages 1–18. Springer Verlag, 1996.

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

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–441, Dec. 1975.

[BHT98] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness and structure of complete sets. *SIAM Journal on Computing*, 27(3):637–653, 1998.

[BvMFT00] H. Buhrman, D. van Melkebeek, L. Fortnow, and L. Torenvliet. Using autoreducibility to separate complexity classes. *Siam Journal on Computing*, 29(5):1497–1520, 2000.

[Deg73] A.N. Degtev. tt and m-degrees. *Alg. Log.*, 12:143–161, 1973.

[HO02] L.A. Hemaspaandra and M. Ogihara. *The Complexity Theory Companion.* Springer Verlag, 2002.

[IW97] R. Impagliazzo and A. Wigderson. P=BPP unless E has sub-exponential circuits. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 220–229. ACM Press, 1997.

[Pos44] E. Post. Recursively enumerable sets of integers and their decision problems. *Bull. Amer. Math. Soc.*, 50:284–316, 1944.

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

[Vol98] H. Vollmer. Relating polynomial time to constant depth. *Theoretical Computer Science*, 207(1):159–170, 1998.