# Superpolynomial Circuits, Almost Sparse Oracles and the Exponential Hierarchy

Harry Buhrman* Steven Homer†

**Abstract**

Several problems concerning superpolynomial size circuits and superpolynomial-time advice classes are investigated. First we consider the implications of NP (and other fun- damental complexity classes) having circuits slightly bigger than polynomial. We prove that if such circuits exist, for example if NP has $n^{logn}$ size circuits, the exponen- tial hierarchy collapses to the second level. Next we con- sider the consequences of the bottom levels of the exponen- tial hierarchy being contained in small advice classes. Again various collapses result. For example, if $EXP^{NP}$ is contained in $EXP/poly$ then $EXP^{NP} = EXP$. Finally, we consider the alternating $2^{polylog}$-time hierarchy. The properties of this hierarchy underlie many of the previous results.

## 1 Introduction

In research from the early 1980's to the present, there has been considerable interest in the implications of efficient reductions of $NP$-complete sets (and other hard sets) to sparse sets. To a certain extent this is due to the equivalence between reductions to sparse sets and small circuits. Namely, a problem is polynomial time (Turing) reducible to a sparse set if and only if the set has polynomial-size circuits. Thus the reduction of an $NP$-complete set to a sparse set would imply that every $NP$ set is efficiently computable non-uniformly by a circuit family. This basic intuition would still hold if the reductions were from $NP$-complete sets to sets which are "almost" sparse. For example, if $SAT$ is reducible to a set $S$ of density $n^{log\ n}$ then one could still store the elements of $S$ up to some reasonable length in a small table and then solve most interesting instances of $SAT$ using the table and the given reduction. Our interest is in whether such reductions are possible and what evidence can be found for (or against) their existence. In particular, we would like to determine just how efficiently hard problems in fundamental complexity classes can be computed using non-uniform methods.

In this paper we study several such question concerning small circuits and non-uniform complexity classes. We begin by considering the consequences of complexity classes having circuit families whose size is superpolynomial, but still relatively small. In particular, we

*University of Amsterdam, Depts of Mathematics and Computer Science, Plantage Muidergracht 24, 1018 TV Amsterdam. This research was done while visiting the Boston University Computer Science Department with the support of NSF Grant CCR-8814339, The Netherlands Organization for Scientific Research (NWO) grant SIR 13-603 and NWO-programma voor korte reisbeurzen.

†Boston University, Computer Science Department, Boston MA 02215. Supported in part by National Science Foundation Grant CCR-8814339.

want to examine whether *NP* or *PSPACE* has $n^{logn}$ (or $n^{log^k n}$ for some $k$) size circuits. We give strong evidence that such circuit families do not exist. Next we consider the sizes of circuit families (or equivalently the non-uniform complexity classes) recognizing sets in the *EXP*-time hierarchy. (This is the hierarchy built using alternating machines where the length of the computations is $2^p$ for some fixed polynomial $p$.) Some classes in this hierarchy are already known not to have small circuits. For the other classes in this hierarchy, where this is not known, we prove that if they do have small circuits then the hierarchy collapses. Finally we examine the relationship between the almost-polynomial time hierarchy and non-uniform complexity measures. The almost-polynomial time hierarchy is constructed just as the polynomial-time hierarchy, only the length of the computations are $2^{polylog}$. The properties of this hierarchy, vis a vis non-uniform measures such as advice classes, underlie the proofs of most of the previous results. Again we are able to conclude that the hierarchy collapses from assumptions concerning the small non-uniform complexity of uniform classes.

Our starting point for this work is the classic paper of Karp and Lipton [12]. This was the first extensive study of the consequences of complexity classes having polynomial-sized circuits. For example, Karp and Lipton (together with Sipser) proved that if every set in *NP* has polynomial sized circuits then the polynomial hierarchy collapses (to the second level). They also proved similar results involving collapses which follow from classes such as *PSPACE* and *EXP* (= DTIME($2^{poly}$)) having polynomial-sized circuits. Using the connection between small circuits, non-uniform advice classes and reductions to sparse sets, the study of small circuits also yields results about the existence of sparse hard sets for complexity classes.

In Section 3 we extend the previous results concerning sparse oracles and polynomial-size circuits to somewhat larger non-uniform computations. We explore the implications of the central uniform complexity classes such as *NP* and *PSPACE* having small, but superpolynomial, circuits and being reducible to sets slightly more dense than P-sparse. The results imply the collapse of the exponential hierarchy under assumptions of small non-uniform complexity. For example,

- if *NP* has $n^{log^k n}$ size circuits, for any fixed $k$, then the *EXP*-time hierarchy collapses to the second level.
- if *NP* is polynomial-time Turing reducible to an "almost sparse" set in *NP* (that is, one of density $n^{log^k n}$) then there is a further collapse of the *EXPTIME* hierarchy.
- if *PSPACE* has a $\leq_T^p$-hard set of density $n^{log^k n}$ then *EXPSPACE* is contained in a fixed level of the *EXPTIME* hierarchy.

The exponential hierarchy studied here is the hierarchy formed within exponential space by starting with *EXP* (= TIME($2^{poly}$)) and iterating using *NP* oracles. So the first few level of this hierarchy are *EXP*, *NEXP*, $NEXP^{NP}$, and $NEXP^{NP^{NP}}$. This hierarchy has been studied by Hemachandra [7], Hartmanis, Immerman and Sewelson [6], and Wilson [17]. The question of whether the various levels of this hierarchy can have polynomial sized circuits is of interest. The results of Kannan [10] prove that there are sets in level $NEXP^{NP}$ which do not have small (that is, polynomial-sized) circuits. On the other hand the relativizations of Wilson [17] show that there are oracles relative to which $EXP^{NP}$

(and hence $EXP$ and $NEXP$) does have P-sized circuits. Here we prove that if this is in fact the case and $EXP^{NP}$ has polynomial circuits then $EXP^{NP} = NEXP = EXP = \Sigma_2^p \cap \Pi_2^p$. In this case, by Kannan's results mentioned above, the bottom levels of the hierarchy collapse to the second level of the polynomial hierarchy. However there is still one other level of the hierarchy above this bottom level which is not contained in the $PH$. It is a complex situation which is fully explained in Section 4.

Finally, in Section 5, we extend the results of Section 4 from polynomial-size advice to $2^{polylog}$-size advice. We prove several results about the almost -polynomial time hierarchy using this weaker hypothesis. This hierarchy is defined just as the polynomial hierarchy except that the computations have $2^{polylog}$ rather than polynomial length. Among the results are,

- if there is a $\leq_{btt}^p$-hard set for $PSPACE$ of density $2^{polylog}$ then $EXPSPACE = EXPTIME$.
- if $EXP^{NP}$ has $2^{polylog}$-size circuits then the almost-polynomial time hierarchy collapses to the second level and includes $EXP^{NP}$.

We begin with some definitions and preliminary results.

## 2 Preliminary Results

We follow what is now fairly standard notation in naming the exponential time classes.

$E = DTIME(2^{linear})$, $\quad EXP = DTIME(2^{poly})$,
$NE = NTIME(2^{linear})$, $\quad NEXP = NTIME(2^{poly})$,
$EXPSPACE = DSPACE(2^{poly})$.

We will consider exponential hierarchies based on these time classes. These hierarchies are most conveniently defined in terms of alternating Turing machine computations.

**Definition 2.1** *For a function $f$, $\Sigma_i^f(\Pi_i^f)$ denotes the class of languages accepted by a $\Sigma_i(\Pi_i)$ alternating Turing machine which runs in time f(n). For a class of functions $F$, $\Sigma_i^F = \cup_{f\in F}\Sigma_i^f$ and $\Pi_i^F = \cup_{f\in F}\Pi_i^f$.*

Several important examples are,

1. Letting $F = \{p(n)|\ p$ is a polynomial$\}$ we obtain the usual levels of the polynomial-time hierarchy.
2. Letting $F = \{2^{p(n)}|$ p is a polynomial$\}$ we obtain the $EXP$-hierarchy, one of two exponential time hierarchies studied in this paper. In terms of the more modern oracle notation, the first few levels of this hierarchy are $EXP = \Sigma_0^{EXP}, NEXP = \Sigma_1^{EXP}, NEXP^{NP} = \Sigma_2^{EXP}, NEXP^{NP^{NP}} = \Sigma_3^{EXP}, \ldots$. The whole hierarchy is contained in $EXPSPACE$.

3. Letting $F = \{2^{cn} | c \in I\!N\}$ we obtain the $E$-hierarchy. Its first few levels are $E, NE, NE^{NP}, NE^{NP^{NP}}, \ldots$.

We will have occasion to consider classes between the levels of these two exponential hierarchies. These are commonly denoted by $\Delta$ classes defined by, for $F \in \{E, EXP\}$, $\Delta_0^F = F$, $\Delta_1^F = F^{NP}, \Delta_2^F = F^{NP^{NP}}, \ldots$. So for all $i$, $\Sigma_i^F \cup \Pi_i^F \subseteq \Delta_i^F \subseteq \Sigma_{i+1}^F \cap \Pi_{i+1}^F$.

One of the primary contributions of this paper is relating central facts about these hierarchies to non-uniform computations, reductions to sets of subexponential density and in particular to small circuits. We now briefly review the definitions of these non-uniform complexity measures. For more detailed definitions see Balcázar, Díaz and Gabarró [4].

**Definition 2.2** *An advice function is a function $f : I\!N \rightarrow \Sigma^*$. Let $C$ be a complexity class and $F$ be a class of advice functions. The class $C/F$ is the collection of all sets $A$ such that for some $B \in C$ and some $f \in F$, $A = \{x | <x, f(|x|)> \in B\}$.*

In this paper we will mainly consider three nonuniform classes: $P/poly$, $EXP/poly$ and $E/lin$. Here *poly* denotes the class of functions whose output length is bounded by some polynomial in the length of the input, and *lin* denotes the class of functions whose output length is bounded by some linear function in the length of the input.

We will be considering efficient reductions of complete sets to sets of different subexponential densities. The following definition specifies our measure of a set's density.

**Definition 2.3** *Let $f$ be a nondecreasing function, $f : I\!N \rightarrow I\!N$. A set $S \subseteq \Sigma^*$ is $f$-sparse if $||\{x \in A \mid |x| \leq n\}|| < f(n)$ for all $n$.*
*If $F$ is a class of functions, we say $S$ is $F$-sparse if $S$ is $f$-sparse for some $f \in F$.*

$S$ is said to be *sparse* if $S$ is $p(n)$-sparse, for some polynomial $p$. Informally, sets which, for some fixed $k$, are $n^{log^k n}$-sparse, will be called *almost sparse*. Since $n^{log^k n} = 2^{log^{k+1} n}$, we will use $n^{log^k n}, 2^{log^k n}$, and less formally, $2^{polylog}$ interchangeably. $S$ is said to be *dense* if there is some $\epsilon > 0$ such that $S$ is not $2^{n^\epsilon}$-sparse.

Familiarity with the standard definitions of Boolean circuit complexity is assumed here. A class $C$ is said to have $f$-size circuits if for every set $A \in C$ there is a sequence $\{D_n\}$ of circuits which recognize $A$ and such that for all $n$, $D_n$ has size $\leq f(n)$.

We end this section by presenting some straightforward relationships between these closely related notions of non-uniform computation. It is well-known that $P/poly$ is equal to the class of problems which have polynomial-size circuits, or equivalently to the class of problems which are polynomial-time Turing reducible to some sparse set. There is a similar, though somewhat weaker, relationship concerning almost sparse sets which is captured in the following proposition.

**Proposition 1** *Let $A$ be a set with $2^{polylog}$-circuits. Then $A$ is reducible to a sparse set in time $2^{polylog}$.*

For clarity and also for later use it is helpful to break the proof into two parts.

*Part 1*: If $A$ has $2^{polylog}$-circuits then $A \in P/2^{polylog}$.

**Proof**: Let $\{C_n\}$ be an encoding of a circuits family recognizing $A$ with $|C_n| \leq 2^{log^k n}$, for some fixed $k$. Define the advice function $f(n) = C_n$, so $|f(n)| \leq 2^{log^k n}$. Clearly, for $x$ of length $n$, $x \in A$ if and only if $x$ is accepted by the circuit coded by $f(n)$. Since the circuit value problem is in $P$, whether $x \in A$ can be decided using the advice function $f(n)$ and the polynomial-time predicate for the circuit value problem. □

*Part 2*: If $A \in P/2^{polylog}$ then $A$ is reducible to a sparse set via a $2^{polylog}$-time Turing reduction.

**Proof**: Assume $A$ is in the class $P/2^{polylog}$ via the advice function $f$ and the set $B \in P$. Define $S = \{<x, 0^n> | x \text{ is a prefix of } f(n)\}$. As there are at most $k^2$ many elements in $S$ of length $k$, $S$ is sparse. Now, given $x$, a straightforward prefix search can be carried out using $S$ as an oracle to obtain $f(|x|)$ in $n^{log^k n}$ steps, for some fixed $k$. Having done this, we have $x \in A$ if and only if $<x, f(|x|)> \in B$, and this last can be determined in time $2^{polylog}$ relative to $|x|$. □

## 3 Small Circuits for the Polynomial-Time Hierarchy

In this section we consider the consequences of (*NP* or *PSPACE*) hard problems having circuits whose size grows slightly faster than any polynomial. This is closely related to these problems having reductions to almost sparse sets, that is sets whose densities are bounded by $2^{log^k \, n}$ for some $k$. Our results prove that such reductions do not exist unless certain seemingly different exponential time classes collapse. Hence the results can be considered as evidence that these circuit families and the corresponding reductions to almost sparse sets probably do not exist.

There have been many results examining the implications of *NP* being polynomial-time reducible to a sparse set. Besides the Karp-Lipton-Sipser results mentioned previously. the best known of these is the theorem of Mahaney [14] proving that if *NP* is polynomial-time many-one reducible to a sparse set then *P*=*NP*. The motivation here does not arise from circuit complexity but rather from the isomorphism conjecture of Berman and Hartmanis. Mahaney's theorem is a consequence of this conjecture and so provides some evidence for it. Recently there have been a series of papers (by Ogiwara and Watanabe [16] , Lozano and Ogiwara [15] and Homer and Longpré [13] ) filling in some of the gaps for the various truth-table reductions lying between many-one and Turing reductions to sparse sets.

For the most part we will be concerned here with polynomial-Turing reductions, but we begin with a proposition linking our work to the recent research on bounded truth-table reductions mentioned above. Recall that a set $A$ is bounded truth-table reducible to a set $B$ ($A \leq^p_{btt} B$) if, for some fixed $k$, there is a polynomial time, non-adaptive reduction from $A$ to $B$ which asks at most $k$ queries to $B$ on any input. (For a precise definition of the various truth-table reductions see Homer/Longpré [13] .) We first consider the consequences of *NP* being btt-reducible to an almost sparse set and obtain the equality of *EXP* and *NEXP* under this assumption. Using the methods developed in this paper, a similar theorem for *PSPACE* appears at the end of this paper (Theorem 9).

**Proposition 2** *NP has no* $\leq^p_{btt}$ *hard set* $S$ *with* $||S^{\leq n}|| \leq n^{\log^k n}$ *unless*

- $NP \subseteq \bigcup_{t>0} \mathrm{DTIME}\left(n^{\log^t n}\right)$ *and*

- *EXP = NEXP*.

**Proof**: The main Theorem (Theorem 5) of Homer and Longpré, when applied to btt reductions, states that, if $NP \leq^p_{btt} S$ with $||S^{\leq n}|| \leq n^{log^k n}$ then $NP \subseteq \bigcup_{t>0}$ DTIME $4^k k! n^{log^{kt} n} n^k = \bigcup_{t>0} DTIME(n^{\log^{kt} n})$. The following padding argument now yields *EXP* = *NEXP*.

Let $A \in NEXP$, say $A \in NTIME(2^{n^l})$. Define $B = \{x0^{2^{|x|^l}} | x \in A\}$. Then $B \in NTIME(n) \Rightarrow B \in DTIME(n^{log^u n})$ for some $u$. But then $A \in DTIME((2^{n^l})^{log^u 2^{n^l}}) = DTIME((2^{n^l})^{n^{ul}}) = DTIME(2^{n^{l+ul}}) \subseteq EXP$ as needed. □

The next theorem extends the methods of Karp and Lipton to include circuits of slightly larger than polynomial size. We show that if an *NP*-complete set has circuits of size $2^{polylog}$ then the exponential time hierarchy collapses to the second level.

**Theorem 1** *NP-complete sets do not have circuits of size* $2^{polylog}$ *unless the exponential hierarchy collapses to* $NEXP^{NP}$.

**Proof**: Assume that a $2^{polylog}$-size family of circuits exists for SAT. Then, by Proposition 1, SAT is reducible to a *P*-sparse set in time $n^{log^k n}$, for some fixed $k$. In fact, we will prove our result starting from the slightly weaker hypothesis that SAT is reducible to a set $S$ of density $n^{(log^k n)}$ in time $n^{log^k n}$, for some $k$. And from this weaker hypothesis we will obtain the hierarchy collapse.

The proof uses techniques originated by Karp and Lipton [12] and Balcázar [3]. The main idea is to use a $NEXP^{NP}$machine to find a large initial segment of $S$ and thus a large initial segment of $SAT$. We then show how a $NEXP^{NP}$machine with a large initial segment of $S$ can simulate a $NEXP^{NP^{NP}}$computation, collapsing the hierarchy.

Let $A$ be a set in $NEXP^{NP^{NP}}$and let $M_j$ be a *NEXP*-machine such that $A = L(M_j)$ with a $NP^{NP}$oracle. Furthermore assume that $M_j$ runs in time $2^{n^l}$. We view queries to a $NP^{NP}$oracle as queries to a *NP* machine $M_1$ that on its turn is allowed to make queries to $SAT$. Since the running time of $M_j$ is bounded by $2^{n^l}$ it can query strings to $M_1$ of length at most $2^{n^l}$. We know that $M_1$ has a running time bounded by a polynomial it follows that $M_1$ queries $SAT$ strings of length at most $2^{pn^l}$ for some $p$.

In order to be able to simulate $M_j$ with a $NEXP^{NP}$machine we must compute $SAT$ up to length $2^{pn^l}$. We know that $SAT$ is reducible to $S$ in time $n^{log^k n}$, where the density of $S$ is $n^{\log^k n}$, for some fixed $k$. Let $M_k$ witness the $n^{log^k n}$ reduction from $SAT$ to $S$ and assume that $M_k$ runs in time $n^{log^c n}$. So if we know the elements of $S$ up to length $2^{p^2 n^{l+cl}}$ we're done. There are at most $n^{\log^k n}$ strings in $S$ of length $n$ so there are at most $2^{p^{lk+2} n^{(l+cl)(1+k)}}$ strings of length $2^{p^2 n^{l+cl}}$ in $S$, which is $\leq 2^{n^i}$ for some $i$. Now define the string $z$ by,

$$z = x_1 \# x_2 \# \ldots \# x_{2^{n^i}} \text{ where } x_i \in S \ (i = 1 \ldots 2^{n^i})$$

$z$ codes S up to the desired length and the length of $z$ is less then $2^{n^{2i}}$. The idea now is to first find $z$ and than replace machine $M_1$ that queries $SAT$ with an *NP* machine that makes no such queries. First we will show that if $z$ is known than there exists a *NP* machine $M_{1'}$ such that $x \in L(M_1, SAT) \Leftrightarrow <x, z> \in L(M_{1'})$. Machine $M_{1'}$ behaves as follows:

input $<x,z>$
nondeterministically simulate $M_1$ on input $x$
if $M_1$ queries a string $y$ to $SAT$ do the following:
  simulate machine $M_k$ (that witnesses the reduction of $SAT$ to S) on input $y$
  if $M_k$ queries a string $y'$ to $S$ then
    run through $z$ and find out if $y'$ is in $z$
  if $y'$ is in $z$ proceed the simulation of $M_k$ in the $YES$ state
  otherwise proceed the simulation of $M_k$ in the $NO$ state
if $M_k$ accepts continue the simulation of $M_1$ in $YES$ state
otherwise proceed in the $NO$ state

If $z$ is correct, i.e. it is a coding of an initial segment of $S$, then $M_{1'}$ accepts $<x,z>$ iff $x \in L(M_1, SAT)$. Furthermore for $M_{1'}$ to answer the queries $y$ made by $M_1$ to $SAT$, $M_{1'}$ only needs to simulate the computations $M_k(y)$ where oracle queries are answered via a search through $z$ which is linear in the length of $z$. So $M_{1'}$ is an $NP$ machine.

We now use an $EXP^{NP}$ computation to find a string $z'$ which, when used by $M_k$ in place of the $S$-oracle, computes SAT. Of course, the $z$ defined above would work as $z'$, but we may find a different sufficient $z'$ as well. Once a $z'$ is found, we use it to simulate $M_j$ with $M_1$ replaced by $M_{1'}$. We take advantage of the fact that $SAT$ is self-reducible.

Let $M_{self}$ be an oracle Turing machine which, when equipped with a $SAT$ oracle, witnesses the self-reducibility of $SAT$. More precisely, $M_{self}$ works as follows. For an input $u$, $M_{self}(u)$ computes by generating two queries $u_1$ and $u_2$, both shorter than $u$, with the property that $u \in SAT$ iff $u_1 \in SAT$ or $u_2 \in SAT$. In line (*) of the simulation below, we write that $M_{self}(u)$ accepts with the help of $M_k^{z'}$ to mean that $M_{self}(u)$ computes and generates the two oracle queries $u_1$ and $u_2$ and then either $M_k^{z'}(u_1)$ or $M_k^{z'}(u_2)$ accepts. Recall that $n^{log^c n}$ bounds the running time of $M_k$.
The whole simulation works as follows:

input $x$
guess a string $z'$ of length $2^{n^{2i}}$ and pad $z'$ with 0's so that its length is $2^{n^{4ci}}$
query $NP$ oracle: Is there a string $u, |u| \leq 2^{n^i}$ such that
  (*) $M_{self}(u)$ accepts with the help of $M_k^{z'} \Leftrightarrow M_k^{z'}(u)$ rejects?
  Note that $M_k^{z'}(x)$ means every query from $M_k$ will be
  answered $NO$ if it isn't in $z'$
if the answer is $YES$ then REJECT (wrong $z'$)
if the answer is $NO$ then:
simulate $M_j$ on input $x$
if $M_j$ queries a $y$ to its oracle then query $<y, z'>$ to $M_{1'}$
ACCEPT iff $M_j$ accepts

Since $M_k$ runs in time $n^{log^c n}$, the above query to the $NP$ oracle can be checked to be an $NP$ computation relative to the length of its input (which is $2^{n^{4ci}}$). So the above simulation can be done in $EXP^{NP}$. The only thing remaining to show is that the $NP$ oracle correctly decides if the $z'$ gives answers which cause the oracle machine $M_k$ to correctly compute $SAT$. Line (*) of the above simulation ensures that this is so. Note that the $z'$

used in the above simulation may not code exactly the same strings as $z$ but serves the same purpose from the point of view of $M_k$'s computations. □

Note: In the above theorem, since we show that $\Sigma_3^{EXP} \subseteq \Sigma_2^{EXP}$, we actually have that $\Sigma_2^{EXP} = \Pi_2^{EXP}$. This same comment applies to the other results in this section.

**Corollary 1** *If NP has a $\leq_T^p$ hard set $S$, and $||S^{\leq n}|| \leq 2^{polylog}$, then the EXP hierarchy collapses to $NEXP^{NP}$.*

**Proof**: The proof of Theorem 1 yields the desired conclusion from the hypothesis that the reduction to the almost sparse set $S$ was possible in time $2^{log^k n}$. This is weaker than our assumption here that the reduction is in $PTIME$. □

In Theorem 1 no assumption was made concerning the almost sparse set $S$. Using methods of Kadin[9] we obtain a better collapse of the exponential hierarchy if we assume that the set $S$ is in $NP$. In the next theorem $EXP^{NP}[poly]$ refers to the class of problems computable by an exponential time oracle Turing machine where, for any input $x$, the number of oracle queries is bounded by a fixed polynomial in $|x|$. Hemachandra[7] has shown that $EXP^{NP}[poly] = P^{NE}$

**Theorem 2** *NP has no $\leq_T^p$ complete set $S$ with $||S^{\leq n}|| \leq n^{log^l n}$ unless the exponential hierarchy collapses to $EXP^{NP}[poly]$*

**Proof**: Let $A$ be a set in $NEXP^{NP}$such that $A = L(M_0, SAT)$, for some $NEXP$ machine $M_0$. We construct a machine $M_1$ from $M_0$ such that $L(M_1, S) = L(M_0, SAT)$. $M_1$ is basically the same machine as $M_0$. $M_1$ nondeterministicly simulates $M_0$ until $M_0$ queries a query to $SAT$. Instead of querying it to $SAT$, $M_1$ uses the $\leq_T^p$ reduction from $SAT$ to $S$ to answer the query to $SAT$ with queries to $S$. Note that $M_1$ only queries $S$ up to strings of length $2^{n^k}$ for some $k$.

The idea now is to generate two $NP$ machines, $M_{no}$ and $M_{yes}$ which accept $\overline{S}$ and $S$ respectively, up to length $2^{n^k}$. Once defined, an $EXP^{NP}[poly]$ machine can generate a machine $M_2(x)$, using oracle substitution, that accepts iff $M_1^S(x)$ accepts. Now the $EXP^{NP}[poly]$ machine queries its $NP$ oracle one more time to decide if $M_2(x)$ accepts.

$M_{yes}$ is the $NP$ machine that accepts $S$, since $S$ is in $NP$. $M_{no}$ is a machine that on input $<0^i, 1^j, x>$ guesses $i$ strings of length at most $j$, verifies that the strings are in $S$, and accepts iff $x$ is not one of the $i$ guessed strings. If $|x| \leq j$ and $i = (Census)_S(j)$, then $M_{no}$ accepts $<0^i, 1^j, x>$ iff $x \in \overline{S}$.

The $EXP^{NP}[poly]$ machine needs to be able to compute the census of $S$ up to length $2^{n^k}$. There are at most $n^{log^l n+1}$ strings of length $\leq n$ in $S$ so the census of $S$ up to $2^{n^k}$ is bounded by $2^{n^{2lk}}$. Using binary search only $n^{2lk}$ queries are needed to compute the census of $S$. □

Using a similar, but slightly more complicated, argument we can apply our methods to $PSPACE$.

**Theorem 3** *PSPACE does not have a $\leq_T^p$ hard set $S$ with $||S^{\leq n}|| \leq 2^{polylog}$ unless $EXPSPACE \subseteq NEXP^{NP}$.*

**Proof**: Let $A$ be a set in *EXPSPACE* via a Machine $M_A$ using no more than $2^{p(n)}$ space. we have to show that $A \in \Sigma_2^{EXP}$. Consider the set $A' = \{<x, 0^{2^{p(|x|)}-|x|}>| \ x \in A\}$. $A'$ is in *PSPACE*. Under the assumption that *PSPACE* has a $\leq_T^p$ hard set $S$ with $||S^{\leq n}|| \leq 2^{log^k n}$ we can show that $A'$ is in $\Sigma_2^{2^{polylog}}$. To do this we use the same approach as in the previous theorem. The only difference is that we use a different self-reduction. We consider Quantified Boolean Formulas, a *PSPACE*-complete problem. *QBF* is self-reducible. To show this observe that if we encounter a $\forall x_1 Q_{x_2} \ldots Q_{x_n} \phi(x_1, x_2, \ldots, x_n)$ formula the self-reduction queries $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 0, x_2, \ldots, x_n)$ and $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 1, x_2, \ldots, x_n)$ to its oracle and accepts iff both are in *QBF*. Where $Q_x$ is either a universal or a existential quantifier and $x = 1(0)$ means everywhere in the formula $x$ has been replaced by true (false). If we encounter a existential quantifier the self-reduction becomes $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 0, x_2, \ldots, x_n)$ or $Q_{x_2} \ldots Q_{x_n} \phi(x_1 = 1, x_2, \ldots, x_n)$ and accepts if at least one of the queries is in *QBF*. Let $M_k$ be the polynomial time machine witnessing the reduction from *QBF* to $S$. A $\Sigma_2^{2^{polylog}}$ machine guesses a string $z$ of appropriate length, uses its *NP* oracle to check that it is a correct $z$ and accepts iff $M_k^z$ accepts. Now we have that $A' \in \Sigma_2^{2^{polylog}}$ from this it follows that $A$ is in $\Sigma_2^{EXP}$. □

## 4 Small Circuits and the EXP-Time Hierarchy

In this section we examine the consequences of $EXP^{NP}$ and $E^{NP}$ being efficiently computable using polynomial advice. By results of Kannan [10] , any level of the exponential hierarchies above the $\Delta_1$ level (that is $E^{NP}$ and $EXP^{NP}$) is not in *P/poly*. Chris Wilson, in [17], constructs an oracle relative to which $E^{NP}$ has polynomial-size circuits. So to extend Kannan's result to $E^{NP}$ and $EXP^{NP}$ is likely quite difficult. Nonetheless our first result gives evidence for these classes not being in *P/poly* (or even in *EXP/poly*). We prove that if there are polynomial-size circuits for $EXP^{NP}$ then then the bottom levels of the *EXP* hierarchy collapse into the polynomial hierarchy. From this the same result for the *E*-hierarchy and several others results follow.

In light of the results of Section 1, it is also reasonable to consider the implications of $2^{polylog}$-size advice for exponential classes as well. We postpone these considerations until the next section, as they involve alternating Turing machines with computations of length $2^{polylog}$ which is studied there.

Let $\{M_e\}$ be an enumeration of the problems in *NEXP*. Define the following set $U$: $U = \{<e, x, t, j, b>| M_e(x)$ accepts in $\leq \ t$ steps and the $j^{th}$ bit of the leftmost accepting path is $b$ $(b \in \{0, 1\})\}$

We begin with two lemmas which show that $U$ is $EXP^{NP}$-complete.

**Lemma 3.1** $U \in EXP^{NP}$

**Proof**: To determine if $< e, x, t, j, b >$ is in $U$ we use a prefix search to determine the first $j$ bits of the leftmost accepting computation of $M_e(x)$. We then ask if the $j^{th}$ bit is a $b$. The following algorithm computes the leftmost path via an $EXP^{NP}$ computation.

Input $< e, x, l, j, b >$.
Set $z = \lambda$ (the empty string).

While $|z| \leq j$,
determine if there is an accepting computation of $M_e(x)$ whose first $|z|+1$ bits are $z0$. (This is done by a query to the *NP* oracle.)
If yes, then set $z = z0$. Otherwise, see if there is an accepting computation of $M_e(x)$ whose first $|z|+1$ bits are $z1$. If so, set $z = z1$.

After concluding this algorithm, $z$ is equal to the first $j$ bits of the leftmost accepting path of $M_e(x)$, if the computation accepts. Otherwise $z = \lambda$. We can now test if the $j^{th}$ bit of $z$ is $b$. If so, we accept. Clearly, this algorithm can be done in $EXP^{NP}$ as needed. □

**Lemma 3.2** *If $U$ is in EXP then $EXP = EXP^{NP}$.*

**Proof**: Let $M$ be a Turing machine witnessing the computation of a problem $L$ in $EXP^{SAT}$. (Without loss of generality, we assume the *NP* oracle is $SAT$.) So $M$ runs in exponential time and has access to a $SAT$ oracle. The idea is to convert $M$ from an oracle machine to an *NEXP* machine without an oracle. Then, using $U$ we find the leftmost accepting path through this *NEXP* machine in exponential time. From the leftmost path we are able to read off whether or not the original machine $M$ accepted. Now given an input $x$ to $M$, consider the following *NEXP* computation.

Carry out $M$'s computation on $x$ substituting the following nondeterministic procedure for all oracle calls. When a query $z$ to $SAT$ is reached, nondeterministically guess a possible satisfying assignment for the formula $z$. Consider these guesses as ordered lexicographically. So the leftmost (that is lexicographically least) assignment assigns 0's to each variable and the rightmost assignment assigns 1's to each variable. There are three cases:

1. If the guessed assignment satisfies $z$ then continue on with $M$'s simulation as if the oracle query had answered "yes".

2. If the assignment fails to satisfy $z$ and the assignment is not all 1's (that is, not the rightmost) then the computation halts and rejects.

3. If the guessed assignment fails to satisfy $z$ and is the rightmost assignment then continue on with $M$'s simulation as if the oracle query had answered "no"

Finally, when $M$ during the above simulation halts either accepting or rejecting, the simulation does one final nondeterministic step (branching either left or right) and then halts. The left (right) branch halts in an accepting state if and only if $M$ halts and rejects (accepts).

Note that the above simulation is in *NEXP* since the length of $M$'s computation is exponential and each oracle query $z$ becomes an *NP* computation, but on a (possibly) exponential length formula (in $|x|$). For any input $x$, the *NEXP* computation always has an accepting path and furthermore the leftmost accepting path reflects a computation of $M$ on input $x$ with the correct answers to the oracle queries. (To see this note that when an oracle query is reached, there is always a guessed assignment to the variables of the query which results in the simulation of $M$ continuing along this path. Either the query is satisfiable, in which case (by 1. above) the leftmost path will be a satisfying assignment and $M$'s computation will continue on in a "yes" state, or the query is not satisfiable, in which case (by 3. above) the leftmost path will be the assignment of all 1's to the queries variables and the computation will continue with $M$ in a "no" state.)

Now, in a straightforward manner, we can use an exponential sequence of queries to $U$ to determine the leftmost accepting path of the above $NEXP$ simulation of $M^{SAT}(x)$. Since, by assumption $U$ is in $EXP$, this path can be found in exponential time. Moreover, this path reflects the true computation of $M(x)$ with a $SAT$ oracle. From this leftmost path we can immediately read off whether or not $M$ accepts $x$ by seeing if the last bit of the path is a 1, in which case $M$ accepts, or a 0, in which case $M$ rejects. $\Box$

**Theorem 4** *If $EXP^{NP}$ is in $EXP/poly$ then $EXP^{NP} = EXP$.*

**Proof**: By Lemma 3.2 it is sufficient to prove that $U \in EXP$. The assumption entails the existence of a function $h$ such that $|h(n)| \leq p(n)$ (for some fixed polynomial $p(n)$) and a set $B$ in $EXP$ such that $x \in U$ iff $<x, h(|x|)> \in B$. The idea of the proof is to cycle through all of the possible advice strings for $U$, using the structure of $U$ to find the correct advice in exponential time. The following algorithm recognizes $U$ in deterministic time $2^{poly}$.

input$<e,x,t,j,b>$ (let $n = |<e,x,t,j,b>|$)
foreach string $y'$ with $|y'| \leq p(n)$ do
let $z = \lambda$
repeat until $|z| > t$
if$<<e,x,t,|z|,0>,y'> \in B$ then $z = z0$
else $z = z1$
end repeat
check whether $z$ is an accepting computation of $M_e(x)$
if so, store $z$
end for
find the smallest $z$ among all the stored $z's$ if one exists
accept iff the $j^{th}$ bit of this $z = b$

$\Box$

Recall that the $\Delta$-levels of the $EXP$ hierarchy are defined to be $\Delta_i^{EXP} = EXP^{NP^{i-1}}$, where $NP^i$ is a tower of $i$ $NP$'s. The next corollary says that Theorem 4 applies to any $\Delta$-level of the hierarchy.

**Corollary 2** *If $\Delta_i^{EXP}$ in $EXP/poly$ $i \geq 1$ then $\Delta_i^{EXP} = EXP$.*

**Proof**: For $i = 1$ this is a restatement of Theorem 4 since $\Delta_1^{EXP} = EXP^{NP}$. We show that it is also true for $\Delta_2^{EXP}$. A straightforward inductive argument (omitted here) completes the proof. The basic observation is that the proof of Theorem 4 relativizes. So if $\Delta_3^{EXP^{SAT}}$ is in $EXP^{SAT}/poly$ then $\Delta_3^{EXP^{SAT}} = EXP^{SAT}$. But $\Delta_1^{EXP^{SAT}} = \Delta_2^{EXP}$ and $EXP^{SAT} = EXP^{NP} = \Delta_1^{EXP}$. So if $\Delta_2^{EXP}$ is in $EXP/poly$ then $\Delta_1^{EXP}$ is also in $EXP/poly$. Another application of Theorem 4 now yields $\Delta_2^{EXP} = EXP$. $\Box$

Next we show that if $EXP^{NP}$ has polynomial size circuits, the $EXP$ hierarchy collapses into the polynomial hierarchy.

**Corollary 3** *If $EXP^{NP}$ is in $P/poly$ then $EXP^{NP} = \Sigma_2^P \bigcap \Pi_2^P$*

**Proof**: Since $P/poly \subseteq EXP/poly$, Theorem 4 yields $EXP^{NP} = EXP$. From Karp and Lipton[12] we know that if $EXP \subseteq P/poly$ then $EXP = \Sigma_2^P \bigcap \Pi_2^P$. □

This same result can now be proved for the $E$-hierarchy.

**Corollary 4** *If $E^{NP}$ is in $P/poly$ then $E^{NP} = \Sigma_2^p \bigcap \Pi_2^p$*

**Proof**: We will prove that if $E^{NP} \subseteq P/poly$ then $EXP^{NP} \subseteq P/poly$. The result then follows from the previous corollary. So let $A \in EXP^{NP}$, say $A \in DTIME(2^{n^t})$. We will show that $A \in P/poly$. Define $B = \{x0^{(|x|^{n^t}-|x|)} | x \in A\}$. It is straightforward to check that $B \in E^{NP}$ and so, by assumption, $B \in P/poly$, say via advice function $h$ and polynomial-time predicate $D$. Now define the polynomial-length bounded advice function $h'$ by, for any integer $n$, $h'(n) = h(n^t)$ and define a polynomial-time predicate $D'$ by, on input $<w,n>$, $D'$ carries out the computation of $D$ on input $<w0^{(|w|^t-|w|)},n>$. Then for any $z$, $z \in A \leftrightarrow w = z0^{|w|^t-|w|} \in B \leftrightarrow <w,h(|w|)> \in D \leftrightarrow <z,h'(|z|)> \in D'$. So $A$ is in $P/poly$ as claimed. □

Using our results we can now prove that $NE^{NP}$ does not has P-size circuits. This result was proved earlier by Kannan[10] using different methods.

**Corollary 5** *$NE^{NP}$ does not have P-size circuits.*

**Proof**: Suppose $NE^{NP}$ does have P-size circuits. Then by Karp and Lipton [12] we have that the $PH$ collapses to $\Sigma_2^P \bigcap \Pi_2^P$. From this an easy padding argument yields that $EH$ collapses to $\Sigma_2^{EXP} \bigcap \Pi_2^{EXP}$. So in particular $\Delta_3^{EXP} \subseteq \Sigma_2^{EXP}$. Together with corollary 2 and corollary 3 this would imply that $\Sigma_2^{EXP} \subseteq \Sigma_2^P$. Since this violates the hierarchy theorem for alternating Turing machines we have a contradiction. □

A consequence of these results is that we have a very strange situation under the assumption that $EXP^{NP}$ has $P$-size circuits. Namely, the $EXP$-hierarchy collapses into two levels:

- $EXP^{NP}$ collapses to $EXP = \Sigma_2^P \cap \Pi_2^P$.
- $EH$ collapses to $NEXP^{NP}$.

This seems a strange, yet not contradictory, conclusion.

The next result proves a similar theorem for the $E$-hierarchy. It is actually a corollary to the proof of Theorem 4. Recall that $E/lin$ is the set of problems computable in $DTIME(2^n)$ with a linear size amount of advice. The class $E/lin$ is not very natural, but the result allows us to link our methods to two earlier lines of research.

**Corollary 6** *If $E^{NP} \subseteq E/lin$ then $E^{NP} = E$.*

**Proof**: The proof is exactly the same as that of Theorem 4 except that $EXP$ is replaced by $E$ and $NEXP$ by $NE$ throughout. Some simple algebra, together with the assumption that the class is contained in $E/lin$ (rather than $EXP/poly$) allows us to achieve the collapse to $E$. $\Box$

The results in this section relate to some of those obtained by Allender and Watanabe in [1] . There they considered a property $Q$ which posits that for every honest function $f : \Sigma^* \rightarrow 0^*$ there is a polynomial-time computable weak inverse $g$ such that for all $x \in f(\Sigma^*), f(g(x)) = x$. They prove that property $Q$ is equivalent to the property of $NE$ computations that, "Every $NE$ predicate is $E$-solvable". Meaning that, given an $NE$ predicate $R$, there is an $E$-computable function which, for any input $x$ to $R$, computes a witness to $R(x)$, if one exists. Allender and Watanabe (Proposition 2, [1] ) proved that $E = E^{NP} \Rightarrow$Q$\Rightarrow E = NE$. It is not known if any of these arrows are reversible.

Our results, when looked at in a similar light, can be seen as running parallel to theirs. Consider the property $Q'$ stating that for every honest function $f : \Sigma^* \rightarrow 0^*$, the function $g$ which computes, for any $x \in f(\Sigma^*)$, the least $y$ such that $f(y) = x$, is polynomial time computable. Using the techniques in this section one can show that $Q'$ is equivalent to the property that the leftmost path through an $NE$ computation can be found in $E$. Given this, lemmas 1 and 2 above imply that $E = E^{NP} \Leftrightarrow Q'$. The proof of Theorem 4 actually shows that if $E^{NP} \subseteq E/lin$ then $Q'$. So our results, when stated in this fashion, say that $E^{NP} \subseteq E/lin \Rightarrow Q' \Leftrightarrow E = E^{NP}$.

There is also a relationship between our results and the existence of sparse sets in the polynomial hierarchy. One of the main results in the paper of Hartmanis, Immerman and Sewelson [6] is that $E = NE$ if and only if there are no sparse sets in $NP - P$. This result, together with our Theorem, immediately yields,

**Corollary 7** *If $E^{NP} \subseteq E/lin$ then there are no sparse sets in $\Delta_1^P - P$.*

# 5 The Almost-Polynomial Time Hierarchy and Open Problems

## 5.1 The Almost-Polynomial Time Hierarchy

On examining the central proofs of the previous two sections, it becomes apparent that while the results are stated about the exponential hierarchy, they could equally well be applied to the subexponential hierarchy based on $2^{polylog}$-length alternating computations. Results concerning this hierarchy are actually stronger in the sense that they imply the previous results concerning the $EXP$-hierarchy (usually via a simple padding argument). We chose to state the results for classes in the exponential hierarchy as these classes have previously been extensively studied. In this section we restate our main results for $2^{polylog}$ length computations and obtain from them new results concerning $PSPACE$ and $EXPTIME$.

Let $PL$ denote the class of $2^{polylog}$ functions. That is $PL = \{2^{p(logn)} | p \text{ is a polynomial}\}$. As before, we consider the hierarchy based on alternating $PL$ length computations, whose levels are $\Sigma_i^{PL}$, $\Pi_i^{PL}$ and $\Delta_i^{PL}$ (for all $i$). This hierarchy is not as central or well-studied as the exponential hierarchy. Nonetheless the $PL$-hierarchy has some interesting properties.

First of all there is a natural relationship between this hierarchy and the exponential hierarchy. In essence the $PL$-hierarchy plays the same role for the $EXP$-hierarchy as the polynomial hierarchy does for the $E$-hierarchy. The relationship between the polynomial-time and the $E$-hierarchy was extensively studied by Hartmanis, Immerman and Sewelson in [6] and one can obtain analogous results for the $2^{polylog}$ vs. the $EXP$-hierarchy. For example, there are no sparse sets in $\Sigma_1^{PL} - PL$ if and only if $\Sigma_1^{EXP} = EXP$. Furthermore, possibly unlike the exponential hierarchy, this hierarchy obeys downward separation: if $\Sigma_{i+1}^{PL} \subseteq \Sigma_i^{PL}$ then for all $j > i, \Sigma_j^{PL} \subseteq \Sigma_i^{PL}$. A wholly different aspect of this hierarchy arises in the recent paper by Feige, Goldwasser, Lovász, Safra and Szegedy [5]. They show that if the approximation of the size of the maximum clique of a graph within a factor of $2^{log^{1-\epsilon} n}$ is in $PL$ then $\Sigma_1^{PL} = \Sigma_0^{PL} = DTIME(2^{polylog})$. This provides strong evidence against the existence of such an approximation algorithm.

The following two results concerning the $PL$-hierarchy are proved exactly as the previous results (Theorem 1 and Corollary 1) in Section 3 about the exponential hierarchy. In each case, we use arithmetic properties of $2^{polylog}$ functions in place of those of $EXP$-functions. We omit the proofs here.

**Theorem 5** *$NP$-complete sets do not have circuits of size $2^{polylog}$ unless the $2^{polylog}$ hierarchy collapses to $\Sigma_2^{PL}$.*

**Corollary 8** *If $NP$ has a $\leq_T^p$ hard set $S$, and $||S^{\leq n}|| \leq 2^{polylog}$ then the $2^{polylog}$ hierarchy collapses to $\Sigma_2^{PL}$ and $PH \subseteq \Sigma_2^{PL}$.*

In Section 3 we investigated the implications of there being almost-polynomial size circuit families for $NP$ and $PSPACE$. In Section 4 we considered polynomial size circuits for the $EXP$-hierarchy, and in particular for $EXP^{NP}$. Having introduced the $PL$-hierarchy, we can now bring these ideas full circle and prove the next result which investigates whether $EXP^{NP}$ has almost-polynomial size circuits. Again, by results of Kannan [10], any class in the $EXP$-hierarchy above $EXP^{NP}$ is known not to have such circuits. As a corollary, our theorem yields a new proof of this fact.

**Theorem 6** *If $EXP^{NP}$ has circuits of size $2^{polylog}$ then $EXP^{NP} \subseteq \Sigma_2^{PL}$.*

**Proof**: Without loss of generality we assume that a Turing machine has only 1 tape and 1 head and accepts by entering a unique accepting state. By a configuration we mean a binary string $c$ coding the state, the tape contents and the place of the head on the tape. For a $NEXP$ machine the length of a configuration $c$ is always bounded by $2^{p(n)}$ for some fixed polynomial $p$. A legal transition from configuration $c_i$ to configuration $c_j$ (with respect to some Turing machine $M_e(x)$) is one where the state, the tape contents and the place of the head change accordingly $M_e$'s transition relation. Note that in order to check one bit of $c_j$ one only needs to consider a constant number of bits of $c_j$ and $c_{j-1}$. For a more detailed description of this see, for example, [4]. As in the proof of Theorem 4 we need to define a complete set for $EXP^{NP}$. We use almost the same set $U$ as in theorem 4. Define $U'$ to be the following set:
$U' = \{<e, x, t, j, k, b> |$the $k^{th}$ bit of the $j^{th}$ configuration of $M_e(x)$'s leftmost accepting path of length $\leq t$ equals $b\}$

A same sort of argument as used in lemma 3.1 shows that $U'$ is in $EXP^{NP}$ and a simple $\leq_m^p$ reduction from $U$ to $U'$ shows that $U'$ is complete for $EXP^{NP}$. Under the assumption that $U'$ has a circuit family of size $2^{polylog}$ we show that it is in $\Sigma_3^{PL}$. Together with Theorem 5 we then have that $EXP^{NP} \subseteq \Sigma_3^{PL} \subseteq \Sigma_2^{PL}$.

**CLAIM 6.1** *If $U'$ has a circuit family of size $2^{polylog}$ then $U'$ is in $\Sigma_3^{PL}$*

**Proof**: We provide a $\Sigma_3^{PL}$ algorithm for $U'$. We first show how to determine, via an *NP* computation, if a circuit $z$ recognizes the bits in a sequence of configurations comprising an accepting path in a *NEXP* computation. In the following description of the computation of an *NP* machine, the input $z$ represents a circuit, possibly recognizing $U'$. So $z$ takes inputs of the form $< e, x, t, j, k, b >$.
Input $z, e, x$. Accept if,

1. There is a place in the first configuration, as determined by bits of this configuration which are decided by inputs ot $z$, which incorrectly represents the initial configuration of $M_e(x)$, or

2. there is an integer $i$ ( the number of a configuration) and a position $k$ in configuration $i$, such that the $k^{th}$ bit in configuration $i$, as determined by $z$, does not follow from the corresponding bits in the $(i-1)^{th}$ configuration (again as determined by $z$) via a legal transition of $M_e(x)$, or

3. the last configuration of $M_e(x)$ contains a state other than the accepting state.

Since $M_e$ is a *NEXP* machine, in 2 above the integers $i$ and $k$ have length polynomial in the length of $x$. Hence the above computation can be carried out by an *NP* oracle. This oracle will reject $z, e, x$ iff $z$ correctly determines the bits in a sequence of configurations representing some accepting computation of $M_e(x)$.

Now we have to check for a leftmost computation. The above procedure ensures that the computation is correct but it might not be the leftmost. In order to check that a $z$ codes the leftmost computation we query a $NP^{NP}$ oracle if there exists a $z'$ that also codes a correct computation and codes a computation that is to the left of the computation that $z$ codes. More precisely:

input $z, e, x$,
guess a $z' \neq z$
check that $z'$ codes a correct accepting computation of $M_e(x)$
(This uses a query to an *NP* oracle.)
guess a place in the computation of $M_e(x)$ where $z'$ codes a left branch
and $z$ a right. Furthermore check that $z$ and $z'$ code the
same computation up to this point.
The last check requires again an *NP* query.
If all of this holds accept, else reject.

Combining the two algorithms we obtain a $\Sigma_3^{PL}$-algorithm for $U'$:

input$< e, x, t, j, k, b >$
guess a $z$
check that $z$ codes a correct accepting computation of $M_e(x)$

(This check needs a query to a $NP$ oracle.)
check that $z$ codes the leftmost accepting computation of $M_e(x)$
(This check needs a query to a $NP^{NP}$ oracle.)
Now use $z$ to determine if $<e, x, t, j, k, b> \in U'$ and accept accordingly

□

Now we apply Theorem 5 to obtain the inclusion in $\Sigma_2^{PL}$. □

**Corollary 9** *(Kannan [10]) $NEXP^{NP}$ does not have circuits of size $2^{polylog}$.*

**Proof**: The main observation is again that the above proof relativizes. So if $EXP^{NP^{NP}}$ has circuits of size $2^{polylog}$ then it collapses to $\Sigma_2^{PL}$. Furthermore by Theorem 1 we have that if $NEXP^{NP}$ (and thus $NP$) has circuits of size $2^{polylog}$ then the exponential hierarchy collapses to $NEXP^{NP}$. So $\Sigma_2^{EXP} = NEXP^{NP} = EXP^{NP^{NP}} \subseteq \Sigma_2^{PL}$. But this is in contradiction with the hierarchy theorem for alternating Turing machines. □

The next two theorems (and their proofs) are analogous to those proved earlier (Theorems 2 and 3) for the EXP-hierarchy. They will allow us to obtain a new result about $btt$-hard sets for $PSPACE$.

**Theorem 7** *$NP$ has no $\leq_T^p$-complete set $S$ with $||S^{\leq n}|| \leq 2^{poly}$ unless the $2^{polylog}$ hierarchy collapses to $\Delta_1^{PL}[polylog]$.*

**Theorem 8** *$PSPACE$ does not have a $\leq_T^p$ hard set $S$ with $||S^{\leq n}|| \leq 2^{polylog}$ unless $\bigcup_{k>0} DSPACE(2^{\log^k n}) \subseteq \Sigma_2^{PL}$.*

Combining this last Theorem with Proposition 2 of Section 3 we obtain a new result concerning almost sparse $\leq_{btt}^p$-hard sets for PSPACE. The results provides strong evidence that such sets do not exist. Of course, a proof that they don't would imply $P \neq PSPACE$.

**Theorem 9** *If $PSPACE \leq_{btt}^p S$ with $||S|| \leq 2^{polylog}$ then*

- *$PSPACE \subseteq DTIME(2^{polylog})$ and*
- *$EXPSPACE = EXPTIME$.*

**Proof**: If an $S$ exists as in the hypothesis, then clearly $S$ is $\leq_{btt}^p$-hard for $NP$. So by Proposition 2, $NP \subseteq DTIME(2^{polylog})$. This implies $NTIME(2^{polylog}) = DTIME(2^{polylog})$ by the closure properties of $2^{polylog}$ functions. Since the $PL$-hierarchy obeys downward separation, this implies that the whole $PL$-hierarchy collapses to $DTIME(2^{polylog})$. Now, by the previous Theorem, our hypothesis implies that $PSPACE \subseteq \Sigma_2^{PL} = DTIME(2^{polylog})$. From this inclusion, a straightforward padding argument yields $EXPSPACE = PSPACE$. □

## 6 Open Problems

There are a number of interesting problems here which bear further study. In Section 3, Corollary 1 we consider reductions to almost sparse sets $S$. It would be useful to know an upper bound on the complexity of $S$. In particular, can $S$ be computed in *EXP* ?

We have considered reductions to almost-sparse sets and the existence of circuit families of size $2^{polylog}$. From these we have proved collapses within the *EXP*-hierarchy. What happens if the sets or circuit families are larger than $2^{polylog}$, yet still subexponential ? One would suspect that a collapse in a suitably large superpolynomial hierarchy would result. For example, using the methods here, it is possible to extend our results to some densities larger than $2^{polylog}$. From the assumption that SAT is $\leq_T^P$-reducible to a set $S$ of density $2^{logn^{loglogn}}$ we can prove that the double exponential hierarchy collapses to the second level. We don't know if there is a corresponding result for any subexponential density.

Corollary 3 yielded strong consequences from the existence of polynomial-size circuits for $EXP^{NP}$. In particular, under this assumption, $EXP^{NP} = EXP = \Sigma_2^p \cap \Pi_2^p$. Similarly, it was previously known that if *EXP* has polynomial size circuits then $EXP = \Sigma_2^p \cap \Pi_2^p$. But what follows from the intermediate assumption that *NEXP* has polynomial-size circuits ? The clear expectation is that then *NEXP*=*EXP*, but this remains an open question. Interestingly, this question seems intertwined with the question of how difficult it is to find an accepting path in an accepting *NEXP* computation. Analyzing the proof of Theorem 4 leads to the conclusion that if one could find such an accepting path in *NEXP* then the $NEXP = EXP$ conclusion would likely follow from the assumption that $NEXP \subseteq P/poly$. This same question was raised by Babai, Fortnow and Lund in [2] where it arises when considering upper bounds for the power of the prover in multiple prover interactive proofs. If accepting paths in *NEXP* computations can be found in *NEXP* then *NEXP* provers suffice for MIP proofs. The current best upper bound is $EXP^{NP}$.

## References

[1] Allender E. & O. Watanabe. *Kolmogorov Complexity and Degrees of Tally Sets*. Information and Computation, 86 (1990) pp160–187.

[2] Babai, L., L. Fortnow & C. Lund. *Non-deterministic exponential time has two-prover interactive protocols*. In Proc. 31st IEEE Symp. on Foundations of Computer Science, (1990) pp16–25.

[3] Balcázar J.L. *Self-reducibility*. Journal of computer and system sciences 41 (1990) pp367–388.

[4] Balcázar J.L., J. Díaz & J. Gabarró. *Structural Complexity I*. W. Brauer, G. Rozenberg & A. Salomaa (eds.) EATCS Monographs on Theoretical Computer Science 11 (1988) Springer Verlag.

[5] Feige U., S. Goldwasser , L. Lovász, S. Safra & M. Szegedy *Approximating Clique is Almost NP-complete*. manuscript, 1991.

[6] Hartmanis J., N. Immerman & V. Sewelson. *Sparse Set in NP-P: EXPTIME versus NEXPTIME*. Information and Control, 65 (1985) pp158–181.

[7] Hemachandra L.A. *Counting in Structural Complexity Theory* . Ph.D. thesis Cornell university (1987).

[8] Homer S. & L. Longpré. *On Reductions of NP Sets to Sparse Sets*. To appear in Proc. Structure in Complexity Theory sixth annual conference (1991) in Chicago Il.

[9] Kadin J. $P^{NP[logn]}$ *and sparse Turing complete sets for NP*. Proc. Structure in Complexity Theory second annual conference, IEEE Computer Society Press (1987) pp33–40.

[10] Kannan R. *Circuit-size Lower Bounds and Non-reducibility to Sparse Sets*. Information and control 55 (1982) pp40–46.

[11] Karp R.M. *Reducibility among combinatorial problems*. Complexity of Computer Computations, R.E. Miller & J.W. Thatcher eds. Plenum N.Y. pp85–103.

[12] Karp R.M. & R.J. Lipton. *Some connections between nonuniform and uniform complexity classes*. Proc. 21 st IEEE Found. Comput. Sci. (1980) pp302–309.

[13] Longpré L. & S. Homer. *On Reductions of NP Sets to Sparse Sets*. To appear in Proc. Structure in Complexity Theory sixth annual conference (1991) in Chicago Il.

[14] Mahaney S.R. *Sparse complete sets for NP: Solution of a conjecture by Berman and Hartmanis*. Journal of Computer and System Sciences 25 (1982) pp130–143.

[15] Ogiwara M. & A. Lozano. *On One Word-Decreasing Self-Reducible Sets*. To appear in Proc. Structure in Complexity Theory sixth annual conference (1991) in Chicago Il.

[16] Ogiwara M. & O. Watanabe *On polynomial time bounded truth-table reducibility of NP sets to sparse sets*. Proc. 22nd Annual ACM Symposium on Theory of Computing (1990) pp457–467.

[17] Wilson C.B. *Relativized Circuit Complexity* J. Comp. Sys. sci. 31 (1985) pp169–181.