1-1-2007

A Note on the Karp-Lipton Collapse for the Exponential Hierarchy

Chris Bourke
University of Nebraska - Lincoln, bourke@cse.unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/csetechreports
Part of the Computer Sciences Commons

http://digitalcommons.unl.edu/csetechreports/7

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Technical reports by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.
A Note on the Karp-Lipton Collapse for the Exponential Hierarchy

Chris Bourke
Department of Computer Science & Engineering
University of Nebraska
Lincoln, NE 68503, USA
Email: cbourke@cse.unl.edu
January 10, 2007

Abstract

We extend previous collapsing results involving the exponential hierarchy by using recent hardness-randomness trade-off results. Specifically, we show that if the second level of the exponential hierarchy has polynomial-sized circuits, then it collapses all the way down to $\text{MA}$. 

Introduction

Much consideration has been given to the proposition that certain complexity classes may be Turing reducible to sparse sets. Equivalently, what happens if certain complexity classes have polynomially-sized (non-uniform) circuits? Such research has proven fruitful in giving evidence that such reductions and circuits do not exist for many interesting complexity classes.

The first such result, the Karp-Lipton collapse [6], showed that if $\text{NP} \subseteq P/\text{poly}$ then the entire polynomial hierarchy collapses to the second level ($\Sigma_2^p \cap \Pi_2^p$). This collapse has since been improved ([7, 2, 3] Köbler & Watanabe improved it to $\text{ZPP}^{\text{NP}}$, Cai, with Sengupta, improved it to $\Sigma_2^p$, and Chakaravarthy & Roy improved it further to $\text{O}^{\text{NP}}_2$). In the same paper, they showed a stronger hypothesis results in a stronger collapse; that if $\text{EXP} \subseteq P/\text{poly}$ then $\text{EXP} = \Sigma_2^p \cap \Pi_2^p$.

But what if even larger classes have polynomially sized circuits—do similar collapses occur? In fact, they do. Buhrman and Homer [1] strengthened the Karp-Lipton collapse to one higher level of the exponential hierarchy. This hierarchy is a natural exponential analog of the polynomial hierarchy inductively defined with NP oracles. That is, $\text{EXP}, \text{EXP}^{\text{NP}}, \text{EXP}^{\text{NPNP}},$ etc; and $\text{NEXP}, \text{NEXP}^{\text{NP}}, \text{NEXP}^{\text{NPNP}}$ along with their complements.

Theorem 1 (Buhrman & Homer [1]).

$$\text{EXP}^{\text{NP}} \subseteq P/\text{poly} \Rightarrow \text{EXP}^{\text{NP}} = \Sigma_2^p \cap \Pi_2^p$$
In contrast, however, Kannan [5] was able to provably separate the exponential hierarchy from $P/\text{poly}$. Specifically, he showed that any level of the exponential hierarchy above $\text{EXP}^{\text{NP}}$ is not contained in $P/\text{poly}$. Improving this separation may be exceedingly difficult, as the oracle construction of Wilson [9] shows that $\text{EXP}^{\text{NP}}$ has polynomial-sized (in fact linear-sized) circuits (relative to this oracle).

Research in the area of derandomization has used similar hypotheses to get conditional hardness-randomness trade-off results. That is, assuming the existence of a hard Boolean function (e.g. $\text{EXP} \not\subseteq P/\text{poly}$), one can construct pseudorandom generators from their truth table and derandomize some probabilistic complexity class like $\text{BPP}$ or $\text{MA}$. A more recent result of Impagliazzo, Kabanets and Wigderson, shows that for the case of $\text{MA}$, such circuit complexity lower bounds are actually necessary for derandomization.

**Theorem 2** (Impagliazzo, Kabanets & Wigderson [4]).

$$\text{NEXP} \subset P/\text{poly} \iff \text{NEXP} = \text{MA}$$

We observe that the containments, $\text{MA} \subseteq \Sigma_2^P \cap \Pi_2^P \subseteq \text{EXP}$, mean that the collapse also implies $\text{NEXP} = \text{EXP}$.

**Main Result**

The collapse in Theorem 2 is, in a sense, incomplete. In particular, it does not immediately follow that an inclusion in $P/\text{poly}$ one higher level in the exponential hierarchy would cause a similar collapse. We extend this result by showing that such a collapse does indeed hold.

**Definition 1.** A language $L \subseteq \{0, 1\}^*$ is in $\text{EXP}^{\text{NP}}$ if it is decidable in deterministic exponential time with an oracle for $\text{NP}$. Additionally, $L \in \text{EXP}^{\text{NP}[z(n)]}$ if $L \in \text{EXP}^{\text{NP}}$ and $L$ is computable using at most $z(n)$ $\text{NP}$ queries on inputs of length $n$.

**Theorem 3.** For any time-constructible function $z(n)$,

$$\text{EXP}^{\text{NP}[z(n)]} \subset P/\text{poly} \Rightarrow \text{EXP}^{\text{NP}[z(n)]} = \text{EXP}$$

It follows from Theorem 2 and standard hierarchy inclusions that this implies an even stronger collapse.

**Corollary 4.**

$$\text{EXP}^{\text{NP}[z(n)]} \subset P/\text{poly} \Rightarrow \text{EXP}^{\text{NP}[z(n)]} = \text{MA}$$

Clearly, $\text{EXP} \subset \text{EXP}^{\text{NP}[z(n)]}$ so it suffices to show that the assumption implies $\text{EXP}^{\text{NP}[z(n)]} \subset \text{EXP}$. We proceed by proving a series of lemmas.

**Lemma 5.**

$$\text{EXP}^{\text{NP}[z(n)]} \subset P/\text{poly} \Rightarrow \text{NEXP} \subset P/\text{poly}$$
Proof. This follows from a simple padding argument; any set \( A \in \text{NEXP} \) can be decided by an \( \text{EXP} \) machine with a single (though exponentially long) query to \( \text{NP} \), i.e. we pad out the input \( (x, 1^{2|x|}) \) in \( \text{EXP} \) time, then a query to \( \text{NP} \) (say to \( \text{SAT} \)) runs in polynomial time with respect to \( |(x, 1^{2|x|})| \). Thus \( \text{NEXP} \subseteq \text{EXP}^{\text{NP}[1]} \) and by the assumption, \( \text{NEXP} \subseteq \text{P}/\text{poly} \).

Lemma 6. \( \text{EXP}^{\text{NP}[z(n)]} \subseteq \text{P}/\text{poly} \Rightarrow \text{NEXP} = \text{EXP} \)

Proof. It follows from Lemma 5 and Theorem 2.

We are now able to mimic the argument of Krentel [8] who showed that any \( \text{OptP} \) function is computable by a \( \text{P} \) machine with access to an \( \text{NP} \) oracle (i.e. \( \text{OptP} = \text{FP}^{\text{NP}} \)). For completeness, we give the following definitions which are also analogous to those presented in [8].

Definition 2. A \( \text{NEXP} \) metric Turing machine \( N \) is a non-deterministic, exponentially time-bounded Turing machine such that every branch writes a binary number and accepts. For each \( x \in \Sigma^* \) we write \( \text{Opt}_N(x) \) for the largest value on any branch of \( N(x) \)

Definition 3. A function \( f : \Sigma^* \rightarrow \mathbb{N} \) is in \( \text{OptEXP} \) if there is a \( \text{NEXP} \) metric Turing machine such that \( f(x) = \text{Opt}_N(x) \) for all \( x \in \Sigma^* \). The function \( f \) is in \( \text{OptEXP}[z(n)] \) if \( f \in \text{OptEXP} \) and the length of \( f(x) \) is bounded by \( z(|x|) \) for all \( x \in \Sigma^* \).

Lemma 7. Any \( f \in \text{EXP}^{\text{NP}[z(n)]} \) can be computed as \( f(x) = h(x, g(x)) \) where \( g \in \text{OptEXP}[z(n)] \) and \( h \) is computable in \( \text{EXP} \) time with respect to \( |x| \). That is, \( \text{EXP}^{\text{NP}} = \text{OptEXP} \).

Proof. Let \( f \in \text{EXP}^{\text{NP}[z(n)]} \) and \( M \) be the machine computing \( f \). Note that \( M \) is an \( \text{EXP} \) machine making \( z(n) \) queries to an \( \text{NP} \) set (without loss of generality, say \( \text{SAT} \)). Algorithm 1 presents a \( \text{NEXP} \) metric Turing machine \( N \).
Input : $x \in \{0,1\}^n$
1 Compute $z(n)$
2 Non-deterministically branch for each $y \in \{0,1\}^{z(n)}$
3 Let $y = b_1 b_2 \cdots b_{z(n)}$
4 Simulate $M(x)$, constructing queries $\varphi_1, \varphi_2, \cdots, \varphi_{z(n)}$
5 foreach $\varphi_i$ such that $b_i = 1$
6 Guess a satisfying assignment for $\varphi_i$
7 if $\varphi_i \in \text{SAT}$ then
8 OUTPUT $b_1 b_2 \cdots b_{z(n)}$
9 end
10 end

Algorithm 1: A NEXP metric Turing machine computing $b_1 b_2 \cdots b_{z(n)}$

The claim that $\text{Opt}_N(x) = b_1 b_2 \cdots b_{z(n)}$ are the true oracle answers for $M(x)$ is shown by induction. Let $\varphi_1$ be the first query for $M$. If $\varphi_1 \in \text{SAT}$ then $N(x)$ on branch $100 \cdots 00$ will find a satisfying assignment and so $\text{Opt}_N(x) \geq 100 \cdots 00$ and so it must be the case that $b_1 = 1$. Conversely, if $\varphi_1 \not\in \text{SAT}$ then no branch beginning with 1 will find a satisfying assignment and so $\text{Opt}_N(x) \leq 011 \cdots 11$ and $b_1 = 0$. By induction on $i$, all of the $b_i$’s must be correct oracle answers for the computation of $M(x)$.

Therefore, given oracle answers $\text{Opt}_N(x) = b_1 b_2 \cdots b_{z(n)}$, $f$ can be computed in EXP time by simulating $M(x)$ using the bits of $\text{Opt}_N(x)$ for oracle answers. It follows, then, that $f$ can be computed by $h(x, g(x))$ with $g \in \text{OptEXP}$ and $h$ computable in EXP time.

Proof of Theorem 3. Assume $\text{EXPNP}[z(n)] \subset \text{P/poly}$ and let $f \in \text{OptEXP}$ computed by an OptEXP machine $M_f$. By Lemma 7 it suffices to show that $f$ can be computed in deterministic exponential time. Define the language $L_{M_f} = \{ \langle x, y \rangle \mid x, y \in \{0,1\}^*, M_f(x) = y \}$. Note that $L \in \text{NEXP}$: one can simply guess a (exponentially long) computation path of $M_f$ and accept if and only if $y$ is equal to the computed function value. By Lemma 6, the assumption implies that $\text{EXP} = \text{NEXP}$ thus $L_{M_f} \in \text{EXP}$.

Now consider the procedure in Algorithm 2. Here, we take the view that the polynomial advice string is a circuit. The assumption thus entails the existence of a circuit of size $p(n)$ for some fixed polynomial that computes $f$. We simply have to cycle through all possible circuits to find the right one. For each such circuit $C_i$, we must check that $M_f(x) = C_i(x)$. 


Input : $x \in \{0,1\}^*$
1 forall Circuits $C_i$ of size $\leq p(n)$ do
2 Compute $y = C_i(x)$
3 if $(x, y) \in L_{M_f}$ then
4 Store $y$
5 end
6 end
7 Among the stored strings $y$, take the lexicographically maximum, $y_{\text{max}}$
8 Output $\text{Opt}_N(x) = y_{\text{max}}$

Algorithm 2: An EXP machine computing $f(x)$

The loop in Line 1 cycles through all circuits of size $\leq p(n)$ which can be done in exponential time. Furthermore, the subroutine for deciding $L_{M_f}$ is an EXP procedure by assumption and again, Lemma 6. Thus, $f$ can be computed in deterministic exponential time and the conclusion follows.

We conclude by asking if current techniques can be combined in a more clever way to get an even bigger collapse. Can we show that $\text{EXP}^{\text{NP}} \subset \text{P}/\text{poly}$ collapses $\text{EXP}^{\text{NP}}$ to an even smaller class such as $\text{O}_2^p$?

Acknowledgements

This work was supported partially by National Science Foundation grant number CCF-0430991. I thank N. V. Vinodchandran and John Hitchcock for useful discussions.

References


