2007

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

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

University of Nebraska–Lincoln, Computer Science and Engineering
Technical Report TR-UNL-CSE-2007-004

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

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, $\mathsf{EXP}, \mathsf{EXP}^{\mathsf{NP}}, \mathsf{EXP}^{\mathsf{NP}^{\mathsf{NP}}}$, etc; and $\mathsf{NEXP}, \mathsf{NEXP}^{\mathsf{NP}}, \mathsf{NEXP}^{\mathsf{NP}^{\mathsf{NP}}}$ along with their complements.

Theorem 1 (Buhrman & Homer [1])**.**

$$\mathsf{EXP}^{\mathsf{NP}} \subset \mathsf{P/poly} \Rightarrow \mathsf{EXP}^{\mathsf{NP}} = \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$$

In contrast, however, Kannan [5] was able to provably separate the exponential hierarchy from $\mathsf{P/poly}$. Specifically, he showed that any level of the exponential hierarchy above $\mathsf{EXP}^{\mathsf{NP}}$ is *not* contained in $\mathsf{P/poly}$. Improving this separation may be exceedingly difficult, as the oracle construction of Wilson [9] shows that $\mathsf{EXP}^{\mathsf{NP}}$ 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. $\mathsf{EXP} \not\subset \mathsf{P/poly}$), one can construct pseudorandom generators from their truth table and derandomize some probabilistic complexity class like BPP or MA. A more recent result of Impagliazzo, Kabanets and Wigderson, shows that for the case of MA, such circuit complexity lower bounds are actually necessary for derandomization.

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

$$\mathsf{NEXP} \subset \mathsf{P/poly} \Leftrightarrow \mathsf{NEXP} = \mathsf{MA}$$

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

Main Result

The collapse in Theorem 2 is, in a sense, incomplete. In particular, it does not immediately follow that an inclusion in $\mathsf{P/poly}$ 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 $\mathsf{EXP}^{\mathsf{NP}}$ if it is decidable in deterministic exponential time with an oracle for NP. Additionally, $L \in \mathsf{EXP}^{\mathsf{NP}[z(n)]}$ if $L \in \mathsf{EXP}^{\mathsf{NP}}$ and L is computable using at most $z(n)$ NP queries on inputs of length n.

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

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

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

Corollary 4.

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

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

Lemma 5.

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

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

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

Proof. It follows from Lemma 5 and Theorem 2. $\square$

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

Definition 2. A 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 $\mathsf{Opt}_N(x)$ for the largest value on any branch of $N(x)$

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

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

Proof. Let $f \in \mathsf{EXP}^{\mathsf{NP}[z(n)]}$ and M be the machine computing f. Note that M is an EXP machine making $z(n)$ queries to an NP set (without loss of generality, say SAT). Algorithm 1 presents a NEXP metric Turing machine N.

Input : $x \in \{0,1\}^n$

Compute $z(n)$

Non-deterministically branch for each $y \in \{0,1\}^{z(n)}$

Let $y = b_1 b_2 \cdots b_{z(n)}$

Simulate $M(x)$, constructing queries $\varphi_1, \varphi_2, \cdots, \varphi_{z(n)}$

foreach φ_i *such that* $b_i = 1$ **do**

  Guess a satisfying assignment for φ_i

  if $\varphi_i \in \mathsf{SAT}$ **then**

    Output $b_1 b_2 \cdots b_{z(n)}$

  end

end

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

The claim that $\mathsf{Opt}_N(x) = b_1 b_2 \cdots b_{z(n)}$ are the true oracle answers for $M(x)$ is shown by induction. Let φ_1 be the first query for M. If $\varphi_1 \in \mathsf{SAT}$ then $N(x)$ on branch $100 \cdots 00$ will find a satisfying assignment and so $\mathsf{Opt}_N(x) \geq 100 \cdots 00$ and so it must be the case that $b_1 = 1$. Conversely, if $\varphi_1 \notin \mathsf{SAT}$ then no branch beginning with 1 will find a satisfying assignment and so $\mathsf{Opt}_N(x) \leq 011 \cdots 11$ 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 $\mathsf{Opt}_N(x) = b_1 b_2 \cdots b_{z(n)}$, f can be computed in EXP time by simulating $M(x)$ using the bits of $\mathsf{Opt}_N(x)$ for oracle answers. It follows, then, that f can be computed by $h(x, g(x))$ with $g \in \mathsf{OptEXP}$ and h computable in EXP time. $\square$

Proof of Theorem 3. Assume $\mathsf{EXP}^{\mathsf{NP}[z(n)]} \subset \mathsf{P/poly}$ and let $f \in \mathsf{OptEXP}$ 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 $L \in \mathsf{NEXP}$: 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 $\mathsf{EXP} = \mathsf{NEXP}$ thus $L_{M_f} \in \mathsf{EXP}$.

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** $\langle x, y \rangle \in L_{M_f}$ **then**

4 Store y

5 **end**

6 **end**

7 Among the stored strings y, take the lexicographically maximum, $y_{\max}$

8 Output $\mathsf{Opt}_N(x) = y_{\max}$

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. $\square$

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 $\mathsf{EXP}^{\mathsf{NP}} \subset \mathsf{P/poly}$ collapses $\mathsf{EXP}^{\mathsf{NP}}$ to an even smaller class such as $\mathsf{O}_2^{\mathsf{P}}$?

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

[1] Harry Buhrman and Steven Homer. Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In *Proceedings of the 12th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 116–127, London, UK, 1992. Springer-Verlag.

[2] Jin-Yi Cai. $\mathsf{S}_2^{\mathsf{P}} \subseteq \mathsf{ZPP}^{\mathsf{NP}}$. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, page 620, Washington, DC, USA, 2001. IEEE Computer Society.

[3] Venkatesan T. Chakaravarthy and Sambuddha Roy. Oblivious symmetric alternation. In *23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 230–241. Springer, 2006.

[4] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[5] Ravindran Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55:40–46, 1982.

[6] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th annual ACM symposium on Theory of computing*, pages 302–309, New York, NY, USA, 1980. ACM Press.

[7] Johannes Köbler and Osamu Watanabe. New collapse consequences of NP having small circuits. *SIAM Journal on Computing*, 28(1):311–324, 1999.

[8] Mark W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36(3):490–509, 1988.

[9] Christopher B. Wilson. Relativized circuit complexity. *Journal of Computer and System Sciences*, 31:169–181, 1985.