On the Cutting Edge of Relativization: The Resource Bounded Injury Method

Harry Buhrman[1]
Dept. Llenguatges i Sist. Informàtics
Univ. Polytècnica de Catalunya
Pau Gargallo 5
08028 Barcelona, Spain
e-mail: harry@goliat.upc.es

Leen Torenvliet
Dept. of Math. and Comp. Science
University of Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
e-mail: leen@fwi.uva.nl

November 22, 1994

[1]also supported by the Netherlands Organization for Scientific Research (NWO) TALENT stipendium.

Abstract

In this report we present a new method of diagonalization that is a refinement of the well-known finite injury priority method discovered independently by Friedberg and Muchnik in 1957. In the *resource bounded injury method*, it is necessary in addition to proving that the number injuries for a given requirement is finite to carefully count these injuries and prove that this number does not exceed a bound given by the index of the requirement. The method is used to construct an oracle relative to which the polynomial time hierarchy collapses to an extent that the second level of this hierarchy (P^{NP^A}) captures nondeterministic exponential time. This oracle is an answer to an open problem posed by Heller in 1984 that has thus far resisted existing methods and that has recently regained interest by work of Fu et. al. and by work of Homer and Mocas. Moreover, our oracle provides a constructive counterexample to Sewelson's conjecture that does not make use of information theoretical lowerbounds, and answers several other questions.

1 Introduction

Since the emergence of easy-to-understand non-relativizing techniques, the most prominent of which is the one connected to the interactive proof systems [Sha92], the construction of oracles has dramatically lost popularity in the complexity theory community. Relations between complexity classes relativizing 'both ways' were formerly considered a demonstration of the difficulty of proving a relation between these complexity classes in the real world. Random relativizations [BG81] relativize only one way, but unfortunately the random oracle hypothesis, in which Bennet and Gill conjecture that a relation that holds between complexity classes in a random relativized world also holds between these classes in the real world, was also provided with counterexamples soon after it's statement. Positive relativizations [BGS75, Boo81, Boo89] put a restriction on the oracle model that guarantees that the construction of an oracle relative to which a certain relation between complexity classes holds implies that this relation holds in the real world. A relatively new insight [AIV93], connecting local computation to oracle results, shows that there may be hope yet for oracles satisfying certain conditions. In this paper Arora et. al. consider oracles that satisfy the so-called Cook-Levin property. A relation between complexity classes that holds relative to such oracles also holds in the real world. Here the access mechanism is however by no means restricted.

As things stand, oracles are allowed in papers only if they solve long standing open problems [FFK92] or present new techniques. In this paper, we intend to do both.

The question that is central in this paper was first posed by Heller in 1984 [Hel84]. In that paper, Heller constructed an oracle A relative to which the first level of the Exponential Time Hierarchy collapses down to the second level of the Polynomial Time Hierarchy ($\Sigma_2^{P,A} = \Delta_2^{EXP,A}$). In [Hel86], he observed that his construction can be altered to bring the second level of this hierarchy down to R^{NP}. In the final

In a different vain, it is well-known [SFM78, Zák83] that the time hierarchy for nondeterministic machines is tight. Therefore the question arizes where we know that $NP \neq NEXP$, whether $P^{NP} \neq P^{NEXP}$. Fu et al. [FLZ92] attack this problem and show that $P^{NP} \neq P^{NEXP}$ if the P^{NP} machine is only allowed $n^{o(1)}$ queries or an unbounded number of parallel queries. To demonstrate their inability of improving upon their construction they restate Hellers problem in the form "By now we are not able to separate NE from P^{NP}, nor can we obtain an oracle A with $NE^A \subseteq P^{NP^A}$."(In our oracle notation). Soon after that however, Mocas [Moc93] improved upon this result by showing that in fact $P^{NP} \neq P^{NEXP}$ if the P^{NP} machine is only allowed n^k queries for some fixed k.

In this paper we answer Hellers open problem, by constructing an oracle A such that $P^{NP^A} = NEXP^A$. As a consequence it follows that the result in [Moc93] is optimal for relativizing proof techniques. The construction of this oracle answers most of the other open questions from Heller [Hel84]. Among other things, it shows that there is a relativized world where Sewelsons conjecture fails.

This paper is organized as follows. In the next section we give some definitions and notations. Then we discuss why the oracle that we wish to construct might escape standard construction methods, and introduce the variation of the finite injury method in the title of this paper. Section 4, contains the actual oracle construction, and Section 5 discusses some conclusions and problems that remain open.

2 Definitions and Notations

We assume the reader familiar with standard notions in structural complexity theory, as are defined e.g. in [BDG88]. Nonetheless, we will in this section, recall some notions that we feel are not common knowledge, and fix on some notation.

Sets are denoted by capital letters and are subsets of Γ^*, where $\Gamma = \{0, 1\}$. The cardinality of a set A is denoted as $\|A\|$. Strings are denoted as small letters $x, y, u, v, \ldots$.

The length of a string x is denoted by $|x|$. For a set A and $n \in \omega$, we let the notation A^n stand for the set consisting of all the strings in A of length n.

We assume standard enumerations of recursively presentable classes by (oracle) Turing machines. An *oracle* machine is a multi-tape Turing machine with an input tape, an output tape, work tapes, and a *query* tape. Oracle machines have three distinguished states QUERY, YES and NO, which are explained as follows: at some stage(s) in the computation the machine may enter the state QUERY, and then goes to the state YES, or goes to the state NO, depending on the membership of the string currently written on the query tape in a fixed *oracle* set. Computations of a nondeterministic machine can be ordered by ordering the state set of that machine. An accepting computation that is minimal in this sense can thus be defined. Such a computation is called the *leftmost* computation of M on input x. If no accepting computation exists then the leftmost computation is simply the minimal (then rejecting) computation. If the machine is deterministic then the leftmost computation is the unique computation. We let $M(x)$ stand for the leftmost computation of machine M on input x. If a Turing machine accepts (rejects) a string

x, we will write $M(x) = 1$ ($M(x) = 0$). We use the same notation for oracle machines ($M^A(x) = 0/1$). We let $Q(M^A(x))$ be the set of queries that is asked in the leftmost computation with oracle A. The *length* of a computation, i.e. the number of steps, is denoted by $|M^A(x)|$. For a nondeterministic machine this stands for the length of its leftmost accepting computation if it exists, and 0 otherwise. The set of strings recognized by a Turing (oracle) machine M (with oracle A), is called the *language* of M (relative to A) and is denoted by $L(M)$ ($L(M, A)$). A relativized complexity class is denoted by writing the oracle as a superscript, e.g. P^A. A complexity class may also appear as a superscript to another complexity class, denoting the class of languages emerging from the operation of equipping a machine from the class with an oracle from the superscript class, e.g. P^{NP} is the class of languages recognized by deterministic polynomial time oracle machines with oracle SATISFIABILITY.

The main complexity classes in this paper are P, NP,EXP, and $NEXP$, where exponential time is taken to be two to the power polynomial (2^{n^i+i}, for $i \in \omega$). These classes are members of the so-called Exponential Time Hierarchy and the Polynomial Time Hierarchy respectively, which we define below.

The Polynomial Time Hierarchy was introduced by Stockmeyer [Sto76] and consists of the infinite collection of classes Σ_i^P, Δ_i^P and Π_i^P, which are defined inductively as follows:

$\Sigma_0^P = \Pi_0^P = \Delta_0^P = \Delta_1^P = P$

$\Sigma_{i+1}^P = NP^{\Sigma_i^P}$

$\Pi_{i+1}^P = \text{co}-\Sigma_{i+1}^P$

$\Delta_{i+1}^P = P^{\Sigma_i^P}$

The Exponential Time Hierarchy as an analog of the Polynomial Time Hierarchy is defined as follows:

$\Sigma_0^{EXP} = \Pi_0^{EXP} = \Delta_0^{EXP} = \Delta_1^{EXP} = EXP$

$\Sigma_{i+1}^{EXP} = NEXP^{\Sigma_i^P}$

$\Pi_{i+1}^{EXP} = \text{co}-\Sigma_{i+1}^{EXP}$

$\Delta_{i+1}^{EXP} = EXP^{\Sigma_i^P}$

Relativizations of these hierarchies will be denoted by, e.g. $\Sigma_i^{EXP,A}$, where the interpretation is that the highest level oracle is changed from NP to NP^A.

3 The Problem and the Method

In this section we will briefly explain our feeling that the oracle we are after deserves a new method of construction, and describe this method.

3.1 Natural Boundaries

Our construction of the oracle A for which $P^{NP^A} = NEXP^A$ proceeds by encoding a standard universal set for $NEXP^A$, K^A, into the oracle in such a way that membership of strings in K^A can be decided by a polynomial time oracle machine with an NP^A oracle. Coding $NEXP^A$ into P^{NP^A} implies that $EXP^A = NEXP^A$. It is not known whether the Exponential Time Hierarchy has the downward separation property (or upward collapse property) as does the Polynomial Time Hierarchy, i.e. if $\Sigma_i^P = \Sigma_{i+1}^P$ for some i then $\Sigma_i^P = \Sigma_j^P$ for all $j \geq i$. For the Polynomial Time Hierarchy this property relativizes, i.e. holds relative to *any* oracle. If the Exponential Time Hierarchy should also have this property, then it follows from a relativization of the deterministic time hierarchy theorem [HS65] that $P^{NP} \neq NEXP$, since $P^{NP} \neq EXP^{NP}$.

Hartmanis et. al. [HIS85] observed that if the Exponential Time Hierarchy should have this property then this property does not relativize. They constructed an oracle A such that $EXP = NEXP$, yet the second level of the Exponential Time Hierarchy is proper, i.e. $NEXP \neq NEXP^{NP}$.

A much weaker property for the Exponential Time Hierarchy is the following statement. $EXP = NEXP \Rightarrow NEXP = EXP^{NP}$ as was conjectured by Sewelson [Sew83]. In a relativized world where Sewelsons Conjecture holds, $P^{NP} \neq NEXP$, since $P^{NP} \neq EXP^{NP}$, again using a relativization of the time hierarchy theorem for deterministic machines. Fortunately, Sewelson's Conjecture has also met with counterexamples [IT89] though these oracles are harder to construct and the particular one referenced makes use information theoretical lower bound results.

Heller [Hel84] first put forward a technique of oracle construction based on the preservation of accepting computations. Some authors [TV86], who claim independent discovery of this technique, dub this technique the Stable Query Method. The idea of the method is that whenever an accepting computation of a nondeterministic oracle machine is possible, then the oracle is adjusted such that this computation is an actual computation. I.e. queries that need a positive answer in this computation are put in the oracle, and queries that need a negative answer are reserved for the complement. This technique is probably not sufficient to achieve our result since a well-understood modification can be applied to EXP computations resulting in about the same number of additions and reservations. for a machine M operating with oracle ^{A}, instead of trying to fix an accepting computation for machine M, the method tries to fix some computation by subsequently adapting the oracle such that a next queried formula q is satisfiable (i.e. the answer to this query is YES). For each individual query this implies putting in the oracle (or reserving for the complement) a number of strings proportional to the length of the query. As M has (linear) exponential time, the length and number of these queries is bounded by 2^n. Hence the number of strings that need be preserved is also exponential. (This description of the stable query method is in fact exactly as appeared in [Tor88].) Of course there can be no oracle relative to which $P^{NP} = EXP^{NP}$.

The earlier mentioned result in [Moc93], because the proof relativizes, puts another restriction on the simplicity of our construction, though this restriction is not directly connected to the resource bounded injury method. If it holds in any relativized world that $P^{NP}[n^k] \neq NEXP$, and $P_{tt}^{NP} \neq NEXP$ then an oracle A for which K^A can be recognized in polynomial time with the help of an NP^A oracle has to be encoded such that the oracle machine can neither be replaced by a nonadaptive machine nor by one that uses less then n queries on inputs of length n. In many 'oracle papers' the number of queries needed to retrieve the information is less than that. (Most of the times even one.)

3.2 Resource Bounded Injury

A first solution to Post's problem was obtained by Friedberg [Fri57] and independently Muchnik [Muc56]. They presented a means of construction that later became known as the 'Finite Injury Priority Method'. Usually, an r.e. set that has some property with respect to all recursive oracle machines is constructed by stages. The set of all oracle machines form an infinite set of requirements that have to be satisfied by the construction. At each stage one of these requirements is satisfied, and so the process guarantees that in the limit all the requirements are satisfied. The difference between earlier constructions and the finite injury priority method is that requirements that are satisfied at some stage, may become unsatisfied again at some later stage. Here indexing the requirements becomes of major importance. A requirement may be unsatisfied at some stage only if this action is taken to satisfy a requirement of higher priority (= with smaller index). For a given requirement there are only finitely many requirements of higher priority (indexing starts with 0). If there are infinitely many stages at which a given requirement may be satisfied, it follows that every requirement eventually will be satisfied permanently (i.e. not be injured after being satisfied). Soare's book [Soa87] contains an excellent exposé of different methods of constructing r.e. sets. In complexity theory, especially in oracle construction, it is seldomly necessary to resort to more complicated methods of construction then standard, slow, so-called *wait-and-see* arguments. The method, at stage s takes into consideration all requirements with index less than s, and satisfies from the subset of these requirements that can be satisfied the one with the smallest index.

Satisfying a requirement usually means adding a (set of) string(s) to the oracle under construction. The requirements in our (coding) construction are indexed with (the length of) strings. Requirement $R_{|x|}$ asks that strings x of length $|x|$ be correctly encoded in the oracle. That is if $x \in K^A$ then there has to be some code that says that it is and if $x \notin K^A$ than there has to be some code that it isn't. In recursive function theory this code can, because recursive machines have unbounded time, be spread out through the entire oracle. In our case however, we need to be able to retrieve the encoding, by a resource bounded machine (in particular a polynomial time oracle machine). Therefore, all of the encoding pertinent to x has to be done 'close to' x. That is, the length of the encoding strings may not be greater than a fixed polynomial in the length of x. On a binary alphabet there are only $2^{p(|x|)}$ strings available of length $p(|x|)$. Therefore a method that permits injuries, that is reencoding of x, should also have an implicit bound on the number of times that an injury is permitted. Otherwise the space of encoding would simply fill up. Because our method has such a built in security, we call the method 'resource bounded injury'

4 The Oracle

4.1 Encoding and Retrieval

Let $K^A = \{<i,x,n> : M_i^A(x) = 1 \wedge |M_i^A(x)| \leq n\}$, be the standard complete set for $NEXP^A$. We plan to encode K^A into A such that it can be retrieved by a polynomial time bounded machine with the help of an NP^A oracle. The encoding of a string x will exist of strings $<x,y,z>$, where $|y| = |z| = |x|^3$. On input x, the polynomial time oracle machine will retrieve the maximal (in the lexicographical order) string z such that $<x,y,z> \in A$ and accepts iff this z is odd.

We will define a deterministic polynomial time bounded oracle machine M and a nondeterministic polynomial time bounded oracle machine N such that M can, with the help of oracle $L(N^A)$ retrieve the information encoded as described above. On input x, our machine M starts first by querying $<x>$. On this input N guesses a string $<x, y, z>$ of the appropriate length and accepts if this string is in A. If N has no accepting computation (i.e. the answer to the query is NO), then x is not encoded and M rejects. Otherwise, M sets $z = 0\underbrace{1\ldots1}_{|x|^3-1}$ and queries $<x, z>$. On this input N guesses strings y and z' with $|y| = |z'| = |x|^3$ such that $z' > z$, and accepts if $<x, y, z'>$ is in A. If the answer to M's query is YES, then M proceeds by setting z to $10\underbrace{1\ldots1}_{|x|^3-2}$ and otherwise it sets z to $00\underbrace{1\ldots1}_{|x|^3-2}$. Thus, M can, using binary search, determine the largest z of length $|x|^3$ such that there exists a y of the same length for which $<x, y, z> \in A$ (in $O(|x|^3)$ steps). Then it accepts iff this z is odd.

4.2 Construction

We first give an informal description of the construction. The construction of the oracle proceeds by stages. Let A_s be the oracle after stage s. Let M_K^X be an oracle machine computing K^X, and M_C^X and N^X be oracle machines computing the P^{NP} algorithm described above. Let $C^X = L(M_C, L(N, X))$ be the set of strings encoded in the oracle. The goal of stage $s+1$ is to achieve that at the end of this stage $C^{A_{s+1}} \cap \Gamma^n = K^{A_{s+1}} \cap \Gamma^n$, for some n depending on the stage. To achieve this goal the construction builds a set B at stage $s+1$ which is initially empty. Then it repeatedly selects a string $<x, y, z>$ such that

- $x \in (K^{A_s \cup B} \vartriangle C^{A_s \cup B}) \cap \Gamma^n$
- $<x, y, z> \notin Q(M_K^{A_s \cup B}(u))$ for any u with $|u| \le |x|$
- $x \notin (K^{A_s \cup B \cup \{<x,y,z>\}} \vartriangle C^{A_s \cup B \cup \{<x,y,z>\}}) \cap \Gamma^n$.

Then it proceeds by setting B to $B \cup \{<x, y, z>\}$. Stage $s+1$ ends when no more x in $(K^{A_s \cup B} \vartriangle C^{A_s \cup B}) \cap \Gamma^n$ can be found. I.e. when all strings of length n are correctly encoded in the oracle. We will prove that indeed each stage ends after a limited number of steps.

We call the need for the construction to correctly encode all strings of length n a *requirement*. We say that requirement R_n is *satisfied at stage s* if $C^{A_s} \cap \Gamma^n = K^{A_s} \cap \Gamma^n$. A requirement *requires attention* at stage s if it is not satisfied. A requirement that was satisfied at stage s may require attention at some stage $t > s$, due to changes in the oracle. We say that this requirement is *injured*.

After stage s the construction acts upon the highest priority injury by setting n to the minimal m for which an injury occurs. This can be recognized by the fact that $\Gamma^m \cap (K^{A_{s+1}} \vartriangle C^{A_{s+1}}) \neq \emptyset$. Note that since $\|A_{s+1}\| < \infty$ such an m *must* exist for any given s. We will prove that each requirement R_n will be satisfied and injured only a limited number of times (i.e. the encoding fits in the encoding space). As R_n is satisfied only if all strings of length n are correctly encoded, this proves the correctness of the construction. First we will present the construction slightly more formal.

Construction

Stage 0: $n_0 = 1$; $m_0 = 4$; $A = \emptyset$

Stage s+1: $t = 0$;$B_{s+1,0} = \emptyset$;

while $\exists x \in (K^{A_s \cup B_{s+1,t}} \vartriangle C^{A_s \cup B_{s+1,t}}) \cap \Gamma^{n_s}$ *do*

$B_{s+1,t+1} = B_{s+1,t} \cup \{<x, \min\{y : |y| = m_s \wedge (<x, y, \Gamma^*> \cap (\cup_{|u| \leq n} Q(M_K^{A_s \cup B_{s+1,t}}(u))) = \emptyset)\}, \min\{z : z > \max\{v : (\exists u)<x, u, v> \in A_s \wedge |u| = |v| = m_s, 0\} \wedge |z| = m_s \wedge z = z'1 \leftrightarrow M_K^{A_s \cup B_{s+1,t}}(x) = 1\}>\}$;

If no such string $<x, y, z>$ exists, then the construction ends.

Otherwise $t = t + 1$;

endwhile

$A_{s+1} = A_s \cup B_{s+1,t}$; $n_{s+1} = \min\{p : (C^{A_{s+1}} \vartriangle K^{A_{s+1}}) \cap \Gamma^p \neq \emptyset\}$; $m_{s+1} = n_{s+1}^3 + 3$;

$s = s + 1$

End of construction

4.3 Correctness

The correctness proof consists of a series of lemmas and corollaries. We begin by showing that a string y that avoids all query sets of strings of length less than or equal to n_s can always be selected.

Lemma 1 $(\forall X \subseteq \Gamma^*)(forall x \in \Gamma^n)(forall m > 2n+1)(exists y \in \Gamma^m)[<x, y.\Gamma^*> \cap \bigcup_{|u| \leq n} Q_K^X(u) = \emptyset$

Proof. On input u of length $\leq n$ the number of steps of M_K^X is bounded by 2^n. It follows that the number of queries in the leftmost computation on this input is bounded by 2^n. As there are at most 2^{n+1} strings u of length $\leq n$, and there are more than 2^{2n+1} strings of length m, y exists. □

It follows from this lemma that the encoding of a level (one stage) does not take too many encoding strings. For each string x of length n_s we may find first that it is not in $K^{A_s \cup B_{s+1,t}}$ at some point, and then have to encode this in $B_{s+1,t+1}$. Next we may find for some $t' > t$ that $x \in K^{A_s \cup B_{s+1,t'}}$ and have to encode this in $B_{s+1,t'+1}$. As in subsequent iterations y is picked such that the new string in $B_{s+1,t''+1}$ for $t'' \geq t$ is not queried in the leftmost computation of $M_k^{A_s \cup B_{t''+1}}$ this is the last time that we need to encode x. The total number of strings that will enter the oracle to encode a given string x is thus limited by two times the number of times that the construction visits the level $|x|$.

Corollary 2 $(\forall s \in \omega)(\forall x \in \Gamma^*)\|{<}x, \Gamma^*, \Gamma^*{>}\| \cap A_s \leq \|\{s : n_s = |x|\}\| \times 2$

From the fact that there is an upper bound on the number of strings added for each x at stage s it follows that there is an upper bound for $B_{s+1,t}$

Corollary 3 $(\forall s, t)\|B_{s+1,t}\| \leq 2^{n_s+1}$

Proof. The cardinality of $B_{s+1,t}$ is maximal if the construction succeeds in finding a string ${<}x, y, z{>}$ to encode each time such a string is necessary. We have observed (resulting in Corollary 2) that this can happen twice for each individual string and there are 2^{n_s} strings of length n_s. $\square$

The following lemma is central to the correctness of the construction. It states that the number of times that the strings of a given length are encoded in the oracle is limited. And therefore that the room to code is present.

Lemma 4 $(\forall m)\|\{s : n_s \leq m\}\| \leq 2^{m^2+1}$

Proof. We prove this by induction on m. Let $T_m = \{t_0, t_1, \ldots, t_k\}$ be the maximal set such that $n_{t_i} = m$ for $t_i \in T_m$. Thus $\|\{s : n_s \leq m\}\| = \|T_m\| + \|\{s : n_s < m\}\|$. For $m = 1$, since there are no stages s such that $n_s < 1$ we have $\|\{s : n_s \leq 1\}\| = \|T_1\|$. From the construction it follows that $(\forall t_i)C^{A_{t_i}} \bigcap \Gamma^1 \subseteq C^{A_{t_{i+1}}} \bigcap \Gamma^1$. Since there are only 2 strings of length 1, $\|T_1\| \leq 3$.

Suppose $(\forall r < m)\|\{s : n_s \leq r\}\| \leq 2^{r^2+1}$. Observe that $(\forall i, m)(\exists s)t_i < s < t_{i+2^m+2}$ such that $n_s < m$. This is because if $(\forall t_i, s)t_i < s < t_{i+1} \wedge n_s > m$ then $C^{A_{t_i}} \bigcap \Gamma^m = C^{A_s} \bigcap \Gamma^m \subseteq C^{A_{t_{i+1}}} \bigcap \Gamma^m$. Since there are only 2^m strings of length m, this observation yields an upper bound on $\|T_m\|$ as the induction hypotheses gives us an upper bound on how often $n_s < m$. We find that $\|T_m\| \leq \|\{s : n_s <$

$m\}\| * (2^m+1) + (2^m+1)$. Since $\|\{s : n_s < m\}\| \leq 2^{(m-1)^2+1}$, it follows that $\|\{s : n_s \leq m\}\| \leq 2^{(m-1)^2+1} + (2^m+1) + (2^{(m-1)^2+1}) * (2^m+1) = (2^{(m-1)^2+1}) * (2^m+2) + (2^m+1)$. Since $(m > 1)$ this is $\leq (2^{(m-1)^2+1}) * (2^m+4) = 2^{m^2-m+2} + 2^{m^2-2m+4} \leq 2^{m^2+1}$, again because $(m > 1)$.

□

From this Lemma and Corollary 3 it follows that indeed each stage ends because the level is correctly encoded in the extension of the oracle. It follows that.

Corollary 5 $(\forall s)(C^{A_{s+1}} \triangle K^{A_{s+1}}) \cap \Gamma^{n_s} = \emptyset$

Proof. For each stage $s+1$ where $n_s = m$, the set $B_{s+1,t}$ has less than 2^{m+1} strings by Corollary 3. As there are at most 2^{m^2+1} such stages by Lemma 4, subsequent z are selected minimal, and for each x and y there are 2^{n^3+3} different z available, it follows that a stage can never end for lack of code strings. Hence $(C^{A_{s+1}} \triangle K^{A_{s+1}}) \cap \Gamma^{n_s} = \emptyset$ at the end of each stage $s+1$. □

Corollary 6 $\lim_{s\to\infty} n = \infty$

Proof. This follows directly from Lemma 4. □

From these lemmas the correctness of the construction follows. The oracle is constructed through a recursive procedure, hence it is recursively enumerable. As the construction can 'fall back' to a length n at any point in the construction (through a chain of changes in the oracle) we can not conclude recursiveness.

Theorem 7 *There exists a recursively enumerable oracle A such that $P^{NP^A} = NEXP^A$*

5 Some Consequences and Questions

5.1 Consequences

Corollary 8 *There exists an r.e. oracle A such that $P^{NP^A} = P^{NEXP^A}$.*

Proof. The oracle constructed in Theorem 7 makes $P^{NP^A} = NEXP^A$. It now follows that $P^{NP^A} = P^{NEXP^A}$. □

The previous corollary suggests that there is no tight hierarchy theorem (at least not one that relativizes) for $P^{NTIME(f(n))}$ time classes, for suitable functions f. This contrasts the fact that for $NTIME(f(n))$ there does exist a tight hierarchy [SFM78, Zák83].

Hemachandra [Hem89] proved that the strong exponential hierarchy collapses ($P^{NEXP} = NP^{NEXP}$). Because the proof-technique for that theorem relativizes it follows that (solving another open problem by Heller)

Corollary 9 *There exists an r.e. oracle A such that $P^{NP^A} = NP^{NEXP^A}$.*

An equivalent statement is the following:

Corollary 10 *There exists an r.e. oracle A such that $EXP^{NP^A}[poly] = P^{NP^A}$*

Proof. Again use the techniques in [Hem89] together with oracle A in Theorem 7. □

Again this is a strange but not quite contradictory consequence. We know from the deterministic hierarchy theorem that $EXP \not\subseteq P$ and that this theorem relativizes. The previous corollary shows that if the exponential time oracle machines are restricted to querying only polynomially many queries (as are the polynomial time oracle machines), then the two classes coincide relative to A.

The following corollaries are also mentioned as open problem in [Hel84].

Corollary 11 *There exists an r.e. oracle A such that $EXP^A = NEXP^A$ and $EXP^{NP^A} = \Sigma_2^{EXP}$, but $NEXP^A \subsetneq EXP^{NP^A}$*

Proof. Take again oracle A in the proof of theorem 7. It follows from the fact that the Polynomial Time Hierarchy collapses to P^{NP^A} by padding that the Exponential Time Hierarchy collapses to EXP^{NP^A}. Furthermore a relativized theorem for deterministic oracle machines implies $NEXP^A \neq EXP^{NP^A}$. □

This corollary implies:

Corollary 12 *[IT89] There exists an r.e. oracle A such that Sewelson's conjecture fails relative to A.*

Another consequence is:

Corollary 13 *[BH91] There exists an r.e. oracle A such that $P^{NP^A} \not\subseteq P_{tt}^{NP^A}$.*

5.2 Open Questions

In this paper we develop a new technique called resource bounded injury method. We use this technique to show a collapse between P^{NP} and $NEXP$. The result suggests that it is probably hard to separate these two classes. This result fits in between the previous results: $EXP^A = NP^A$ [Dek77] and $EXP^{NP^A} = \Sigma_2^{P,A}$ [Hel84]. These results together with our result show that all the possible collapses between the second level of the Exponential Time Hierarchy and the second level of the Polynomial Time Hierarchy can occur relative to an oracle. We leave open however this question for higher levels of these two hierarchies. Results along this way imply that the Polynomial Time Hierarchy extends more than 2 levels and may yield different proofs of this than appear in [Ko89]. It may also be the case that research along these lines points the way to new results in lower bounds for circuits.

Acknowledgement

We thank Judy Goldsmith for giving us access to her expert source of oracle knowledge and Peter van Emde Boas and Andy Klapper for proofreading earlier versions of this paper.

Bibliography

[AIV93] S. Arora, R. Impagliazzo, and U. Vazirani. On the role of the Cook-Levin theorem in complexity theory. manuscript, 1193.

[BDG88] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I.* Springer-Verlag, 1988.

[BG81] C. Bennett and J. Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{Co-NP}^A$ with probability 1. *SIAM J. Comput.*, 10(1):96–113, February 1981.

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

[BH91] S.R. Buss and L. Hay. On truth-table reducibility to SAT. *Information and Computation*, 90(2):86–102, February 1991.

[Boo81] R.V. Book. Bounded query machines: on NP and PSPACE. *Theoretical Computer Science*, 15:27–39, 1981.

[Boo89] R.V. Book. Restricted realtiviazations of complexity classes. In J. Hartmanis, editor, *Computational Complexity Theory, Proc. Symposia in Applied Mathematics*, pages 47–74. American Mathematical Society, 1989.

[Dek77] M.I. Dekhtyar. On the relation of deterministic and nondeterministic complexity classes. In *Proceedings Mathematical Foundations of Computer Science, Lecture Notes in Computer Science vol. 45*, pages 282–287, Berlin, 1977. Springer-Verlag.

[FFK92] S. Fenner, L. Fortnow, and S.A. Kurtz. The isomorphism conjecture holds relative to an oracle. In *Proc. 33rd IEEE Symposium Foundations of Computer Science*, pages 30–39, 1992.

[FLZ92] B. Fu, H. Li, and Y. Zhong. Some properties of exponential time complexity classes. In *Proc. Structure in Complexity Theory seventh annual conference*, pages 50–57. IEEE computer society press, 1992.

[Fri57] R.M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability. In *Proc. Nat. Acad. Sci.*, volume 43, pages 236–238, 1957.

[Hel84] Hans Heller. On relativized polynomial and exponential computations. *SIAM J. Comput.*, 13(4):717–725, november 1984.

[Hel86] Hans Heller. On relativized exponential and probabilistic complexity classes. *Information and Control*, 71(3):231–243, December 1986.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[HIS85] J. Hartmanis, N. Immerman, and V. Sewelson. Sparse sets in NP-P: EXPTIME versus NEXPTIME. *Information and Control*, 65(2/3):158–181, May/June 1985.

[HS65] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117:285–306, 1965.

[IT89] R. Impagliazzo and G. Tardos. Decision versus search problems in superpolynomial time. In *Proc. 30th IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1989.

[Ko89] K. Ko. Distinguishing conjunctive and disjunctive reducibilities by sparse sets. *Information and Computation*, 81:62–87, 1989.

[Moc93] S. Mocas. *Exponential Time Classes from Polynomial Time Classes.* PhD thesis, Northeastern University, 1993.

[Muc56] A.A. Muchnik. On the unsolvability of the problem of reducibility in the theory of algorithms. *Dokl. Acad. Nauk SSSR*, 108:194–197, 1956.

[Sew83] V. Sewelson. *A study of the Structure of NP.* PhD thesis, Cornell University, Ithaca. New York 14853, August 1983. TR83-575.

[SFM78] J.I. Seiferas, M.J. Fischer, and A.R. Meyer. Separating nondeterministic time complexity classes. *J. ACM*, 25(1):146–167, January 1978.

[Sha92] A. Shamir. IP=PSPACE. *Journal of the ACM*, 4:869–877, October 1992.

[Soa87] Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Perspectives in Mathematical Logic. Springer-Verlag, 1987.

[Sto76] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[Tor88] Leen Torenvliet. A second step towards the strong p-time hierarchy. *Mathematical Systems Theory*, 21:99–123, January 1988.

[TV86] L. Torenvliet and P. Van Emde Boas. Diagonalisation methods in a polynomial setting. In *Proc. Structure in Complexity Theory, Lecture Notes in Computer Science vol. 223*, pages 331–334, Berlin, 1986. Springer-Verlag.

[Zák83] S. Zák. A Turing machine hierarchy. *Theoretical Computer Science*, 26:327–333, 1983.