ILLiad Request Printout

Transaction Number: 427394
Username: 100934397 Name: William Gasarch
ISSN/ISBN:
NotWantedAfter: 11/09/2009
Accept Non English: Yes
Accept Alternate Edition: No
Request Type: Article - Article

Loan Information

LoanAuthor:
LoanTitle:
LoanPublisher:
LoanPlace:
LoanDate:
LoanEdition:
NotWantedAfter: 11/09/2009

Article Information

PhotoJournalTitle: Information and Control
PhotoJournalVolume: 58
PhotoJournalIssue:
Month:
Year: 1983
Pages:
Article Author: Gasarch and Homer
Article Title: Relativizations Comparing NP and Exponential time

Citation Information

Cited In:
Cited Title:
Cited Date:
Cited Volume:
Cited Pages:

OCLC Information

ILL Number:
OCLC Number:
Lending String: Direct Request
Original Loan Author:
Original Loan Title:
Old Journal Title:
Call Number: UMCP EPLS Periodical Stacks Q350.15 v.56-v.59 (1983)
Location:

Notes

Relativizations Comparing $NP$ and Exponential Time

W. Ian Gasarch*

Aiken Computation Lab, Harvard University,
Cambridge, Massachusetts 02138

AND

Steven Homer*

Department of Computer Science, Boston University,
Boston, Massachusetts 02215

The possible relationships between $NP$ and $EXP_k = \bigcup_{c=0}^{\infty} DTIME(2^{ck})$ relative to oracles are examined. It is first shown that for every oracle (including the empty set) and any $k$, $NP^A \neq EXP_k^A$. Then it is shown that all other relationships are possible under relativization. That is, for each $k > 0$ oracles $A$, $B$, and $C$ are constructed such that (i) $P^A \subseteq NP^B \subseteq EXP_k^B$, (ii) $EXP_k^A \subseteq NP^B$, and (iii) $NP^A$ and $EXP_k^A$ are incomparable with respect to inclusion. The construction of the set $A$ is especially intricate, apparently requiring a finite-injury priority argument. In each case in the constructions when a possible inclusion is ruled out, it is done in a very strong way, namely, by finding a language in one of the classes which is immune with respect to the other class.

1. Introduction

Many of the central problems of complexity theory remain open. In particular, results settling the relationship between deterministic and nondeterministic complexity classes are rare. One approach to these problems has been to look at them relative to Turing machines with oracles (see for example (Baker et al., 1975; Baker and Selman, 1979; Ladner and Lynch, 1976; Rackoff, 1982). Usually, the results obtained indicate that the question can be relativized in contrary directions. This tells us that the relativized problems most probably cannot be solved using current techniques, as most known techniques relativize.

We examine the relationships between $NP$ and the deterministic classes

* This research was supported in part by the National Science Foundation under Grant MCS-82-03482.
* This research was supported in part by the National Science Foundation under Grant MCS-82-18383.

0019-9958/83 $3.00
Copyright © 1983 by Academic Press, Inc.
All rights of reproduction in any form reserved.
Exp_k = \bigcup_{c \in \mathbb{C}}^{c^{0}} \mathbf{DTIME}(2^{c^{k}}). Obviously NP \subseteq \bigcup_{k=0}^{\infty} \text{EXP}_k; hence, we ask how NP relates to \text{EXP}_k for an arbitrary fixed k. A number of interesting early results on this topic are contained in DeKhayt (1977) and Book (1972). In addition, Heller’s thesis (1980) contains many relativized results concerning exponential time. The arguments in Theorem 3 of (Book, 1972) can be used to show that NP^A \neq \text{EXP}^A_k for every oracle A. We present a slightly different proof of this fact here. This is the only negative result we show that all of the other relativized situations are possible. Namely, oracles A, B, and C are constructed so that

(i) \( P^A \subseteq NP^A \subseteq \text{EXP}^A_k \)

(ii) \( \text{EXP}^B_k \subseteq NP^B \)

(iii) \( NP^C \) and \( \text{EXP}^C_k \) are incomparable under inclusion.

Bennett and Gill (1981) show the existence of an oracle A such that there is an NP^A set with no infinite P^A subset. The proof in (Bennett and Gill, 1981) is probabilistic and nonconstructive. In (Homer and Maass, 1983) and independently in (Schoning, 1982) a constructive proof of this fact is given. We use the methods of (Homer and Maass, 1983) to obtain similar results about NP^A and EXP^A_k. For example, an oracle C is constructed so that not only are NP^C and EXP^C_k incomparable but there exist infinite languages, \( L_1 \) and \( L_2 \) such that

(1) \( L_1 \in NP^C \) and \( L_1 \) contains no infinite EXP^C_k subset.

(2) \( L_2 \in EXP^C_k \) and \( L_2 \) contains no infinite NP^C subset.

Such an \( L \) (\( L_1 \)) is said to be immune with respect to EXP^C_k(NP^C) sets. More generally, if \( L \) is a language such that no infinite subset of \( L \) is in a particular complexity class then \( L \) is said to be immune with respect to that class. Recently, Book and Schoning (1982) have studied the notion of immunity and obtained results for a wide variety of complexity classes.

2. **Notation and First Results**

All languages considered will be subsets of \([0,1]\). We fix enumerations \( \{P^i\}_{i=0,1,2,\ldots} \) \( \{\text{NP}^i\}_{i=0,1,2,\ldots} \) of polynomial time-bounded deterministic (nondeterministic) oracle Turing machines. For any \( X \subseteq N \) we write \( P^X \) \( \text{NP}^X \) for machine \( P^i \) \( \text{NP}^i \) with oracle \( X \), or, when no confusion can result, for the language accepted by that machine. We write \( M(x) \) for the computation of machine \( M \) on input \( x \). We may assume \( \rho_i(n) = n \) bounds the time for any computation on an input of length \( n \).

For any natural number \( k \) and any set \( X \). \( \text{EXP}^k_X \) denotes the collection of languages accepted by deterministic Turing machines with oracle \( X \) which
run in time $2^{cn^2}$, where $c$ is some constant. We let $k$ denote a positive natural number which is arbitrary but fixed throughout the paper. We fix an enumeration $\{E^i\}_{i=0,1,2,\ldots}$ of deterministic oracle Turing machines with time bound $2^{cn^2}$ for some $c$. We may assume $h_i(n) = i + 2^{|n|}$ bounds the time of any computation by $E^i$ on inputs of length $n$. $P^4(NP^4, EXP^4_4)$ denotes the collection of all languages $L \in P^4(NP^4, EXP^4_4)$. For a more complete account of these definitions see Hopcroft and Ullman (1979).

$(\cdot, \cdot): N \times N \rightarrow N$ denotes a fixed recursive pairing function which is monotonic in both coordinates. $(\cdot, \cdot)$: $(0, 1)^* \times (0, 1)^*$ denotes a pairing function on strings that operates in polynomial time. We sometimes abuse notation and use an integer $i$ as one of the arguments in the $(\cdot, \cdot)$ function. In such a case we mean $i$ is in binary notation. $|x|$ is used to denote the length of string $x$. $|A|$ is used to denote the cardinality of set $A$. For any language $L$, $\overline{L}$ denotes the complement of $L$. $x * y$ denotes the concatenation of strings $x$ and $y$. The function $\log(\cdot)$ always refers to log base 2.

In Book (1972, Theorem 3) it is shown that $NP \neq EXP_k$. The proof there can be easily modified to show that for any oracle $A$, $NP^A \neq EXP^A_k$. The following argument was pointed out to us by Michael Sipser.

**Theorem 1.** For all $A$, if $EXP^A_4 \subseteq NP^4$ then $EXP^A_{k+1} \subseteq NP^4$.

**Proof.** The key observation here is that by polynomially padding sets in $EXP^A_k$ one may decrease their complexity.

Let $L \in EXP^A_{k+1}$. Define $L' = \{xO^{n+1-n} | x \in L \}$. Now $L' \in EXP^A_4$, since we can modify the machine which recognizes $L$ in $EXP^A_{k+1}$ to recognize the “padded language” $L'$ in $EXP^A_4$. Hence by our assumption $L' \in NP^4$ and so $L \in NP^4$ since $NP^4$ is closed under polynomial length padding.

As a corollary we have:

**Corollary.** For all $A$, $NP^4 \neq EXP^A_4$.

**Proof.** By a straightforward diagonalization, $EXP^A_4$ is properly contained in $EXP^A_{k+1}$ (see Hopcroft and Ullman, 1979, p. 299). Now if $NP^4 = EXP^A_4$ then $EXP^A_{k+1} \subseteq NP^4$ by the theorem. Hence we have $EXP^A_{k+1} \subseteq NP^4 \subseteq EXP^A_{k+1}$, a contradiction.

3. $P^4 \neq NP^4 \neq EXP^4_4$

One of the first and most basic oracle constructions in complexity theory was given in Baker, Gill, and Soloway (1975), where they showed the existence of an oracle $A$ such that $P^A = NP^A$. For such an oracle we have

$NP^A = P^A \subseteq EXP^A_4$ by the relativized version of the time hierarchy theorem (Hopcroft and Ullman, 1979, p. 299). We conclude that $P^A \neq NP^4 \neq EXP^A_4$, and moreover we have $NP^4$ immunity. Clearly this implies the same result for $EXP^A_1$.

**Theorem 2.** There exists an oracle $A$ such that $P^4 \neq NP^4 \neq EXP^4_1$.

1. $P^4 \neq NP^4 \neq EXP^4_1$.

2. There is an infinite $L^1 \subseteq NP^4$ with $L^1 \not\in EXP^4_1$.

3. There is an infinite $L^1 \subseteq EXP^4_1$ with $L^1 \not\in NP^4$.

**Proof:** (In the style of Soare (in press)).

$L^1 = \{O^n | \exists x \in A, |x| = n, n \text{ even}\}$

$L^2 = \{O^n | \exists x \in A, |x| = n, n \text{ even}\}$

Clearly $L^1 \subseteq NP^4$ and $L^2 \subseteq EXP^4_1$. For any $A$, we have the following requirements:

$RP_1$: $P_1 \cap \{O\}^* \text{ infinite } \Rightarrow P_1 \cap \{O\}^* \subseteq \{O\}^*$

$RNP_1$: $NP_1 \cap \{O\}^* \text{ infinite } \Rightarrow NP_1 \cap \{O\}^* \subseteq \{O\}^*$

$T_1$: $|L^1_1| > 0$ (This is to ensure the existence of $L^1_1$).

$C_1$: For almost all $x$, $NP_1(x) \subseteq O \subseteq A$. (This is to ensure the existence of $L^2_1$).

The infinitude of $L^1_1$ will follow easily from $C_1$ and be a formal requirement.

We construct $A$ in stages. $A_s$ denotes the end of stage $s$. $A = \bigcup_{s \geq 0} A_s$. The expression $A_s$ will be used in reference to a computation on an input and will mean to restrain from $A$ all strings which were not in $A_s$. $RP_1$ will act by diagonalizing. It will require that each type of restraint will not affect the other, and the code strings (requirement $C_1$) the same sets.

Both $RNP_1$ and $C_1$ operate only when the rest are changed, and all the strings they query by. Thus these requirements never have to preserve a string to preserve membership.
NP^4 = P^4 \subseteq EXP^4_k$ by the relativized version of the time hierarchy theorem (Hopcroft and Ullman, 1979, p. 299). We construct here an oracle $A$ such that $P^4 \subseteq NP^4 \subseteq EXP^4_1$, and moreover we obtain these inequalities with immunity. Clearly this implies the same result for $EXP^4_k$, $k > 1$, instead of $EXP^4_1$.

**Theorem 2.** There exists an oracle $A$ such that

1. $P^4 \subseteq NP^4 \subseteq EXP^4_1$.
2. There is an infinite $L^A_1 \in NP^A$ with no infinite $P^4$ subset.
3. There is an infinite $L^A_2 \in EXP^4_1$ with no infinite $NP^A$ subset.

**Proof:** (In the style of Soare (in press)). Let:

- $L^A_n = \{O^n \mid \exists x \in A, |x| = n, n \text{ even, } n \text{ not a power of } 2\}$
- $L^A_e = \{O^n \mid 1^n \in A\}$.

Clearly $L^A_n \in NP^A$ and $L^A_e \in EXP^4_1$ for any $A$. To ensure (1), (2), and (3) we have the following requirements:

- $RP_i$: $P^A_i \cap |O|^* \text{ infinite } \Rightarrow P^A_i \cap L^A_1 \neq \emptyset$
- $RNP_i$: $NP^A_i \cap |O|^* \text{ infinite } \Rightarrow NP^A_i \cap L^A_2 \neq \emptyset$
- $T_i$: $|L^A_i| > 1$ (This is to ensure that $L^A_i$ is infinite)
- $C_i$: For almost all $x$, $NP^A_i(x)$ accepts if\( \langle i, x \rangle \ast \langle i, x \rangle \ast 1^{2^x} \ast O \in A \) (This is to ensure that $NP^A \subseteq EXP^4_1$).

The infinitude of $L^A_1$ will follow easily from the construction and need not be a formal requirement.

We construct $A$ in stages. $A_i$ denotes the elements placed into $A$ by the end of stage $s$. $A = \bigcup_{s=0}^s A_i$. The expression "preserve a computation" will be used in reference to a computation on an oracle machine with oracle $A_i$ and will mean to restrain from $A$ all strings that the computation queried which were not in $A_i$.

$RP_i$ will act by diagonalizing. It will restrain strings from entering $A$ to preserve a computation or to prevent a certain string from entering $L^A_1$. The latter type of restraint will not affect the other requirements because $L^A_1, L^A_2$, and the code strings (requirement $C_i$) they query, operate on different sets.

Both $RNP_i$ and $C_i$ operate only when the computation by $NP^A_i$ cannot be changed, and all the strings they query have already been decided upon. Thus these requirements never have to preserve a computation, though $RNP_i$ may restrain a string to preserve membership of an element in $L^A_2$. 

...
For the most part, \( RP_i \) and \( RNP_i \) restrain strings, and \( T_i \) and \( C_i \) place strings in \( A \). The conflicts are resolved with a priority argument, in which requirements can be injured, that is, actually become unsatisfied when they were previously satisfied. We will keep track of which requirement is restraining which strings, and if a requirement wants to put into \( A \) a string restrained by another requirement of lower priority, the higher priority requirement gets its way. When this happens we say the lower priority requirement has been injured. When a requirement \( R \) restrains and/or places strings into \( A \) at stage \( s \) we say that \( R \) has received attention at stage \( s \). The priority ordering is

\[ RP_1, RNP_1, T_1, C_1, RP_2, RNP_2, T_2, C_2, \ldots. \]

Construction.

Stage 0: \( A_0 = \emptyset \).

Stage \( s + 1 \): After this stage, the question of membership in \( A \) is decided for each string of length less than \( s + 1 \), as well as for some additional strings of length greater than \( s \). We perform various actions depending on \( s \) as follows:

If \( s \) is even and not a power of 2 then we try to use it for one of the \( RP_i \) as follows:

Run \( P_i^s(O^s) \) for each \( i \leq \log s \) such that \( p_i(s) < 2^{s/2} \). If \( RP_i \) is not satisfied, then preserve \( P_i^s(O^s) \) for \( RP_i \). Find the least \( i \) such that \( RP_i \) is not satisfied, \( P_i^s(O^s) \) accepts, and \( |L_i^s| > i \). If such exists then restrain all length \( s \) strings from \( A \) for \( RP_i \). At this point \( RP_i \) is satisfied. If no such \( i \) exists then put the least string of length \( s \) that is not restrained into \( A \). (Note: Such a string must exist since the total number of strings restrained up to this point is less than

\( (\text{stages}) \times (\text{machines run per stage}) \times (\text{maximum number of queries per machine}) \)

which is less than \( s \times \log(s) \times 2^{s/2} < 2^s \).) This is done to make \( L_i^s \) infinite.

If \( s \) is odd we try to satisfy the \( RNP_i \) requirement as follows:

Let \( k = [\log(s) + 1] \).

Run \( NP_i^k(O^k) \) for each \( i \leq \log s \) such that \( p_i(k) < s \).

Note that since \( p_i(k) < s \) all the computations are automatically preserved.

Find the least \( i \) such that \( RNP_i \) is not satisfied and \( NP_i^k(O^k) \) is accepted. If \( 1^{2^i} \) is not in \( A \) already, then restrain it for \( RNP_i \). At this point \( RNP_i \) is satisfied.

We perform the next two actions regardless of whether \( s \) is even or odd.

Let \( i = |L_i^s| \), and let \( k = [\log(s) + 1] \). If requirement of higher priority than \( T_i \) then put \( w = \langle j, x \rangle \rightarrow \langle j, x \rangle \ast 1^{2^i} \ast O \) into \( A \). For any \( j < s/2 \) and any \( x \) with \( p_j(|x|) = s \), put \( w = \langle j, x \rangle \rightarrow \langle j, x \rangle \ast 1^{2^i} \ast O \) into \( A \). If requirement of higher priority than \( C_j \) or \( |w| > s \), placing \( w \) into \( A \) will not interfere with this.

End of Construction.

Lemma 1. Every requirement is injured at least restrain a finite number of strings from \( A \).

Proof. The requirements \( T_i \) and \( C_i \) need to have a restraint, so the lemma is automatically true to the \( RP_i \) and \( RNP_i \) requirements.

\( RP_i \) can only be injured by a \( T_j, j < i \), or a \( C_j \) that satisfies it never acts again, so \( T_i \) can only stop acting, but, note that at stage \( s \)

(i) \( RP_i \) needs to restrain strings of length

(ii) \( C_i \) needs to place strings of the form

where \( p_j(|x|) = s \), into \( A \).

A simple calculation reveals that eventually \( C_i \) places into \( A \) always exceed the length \( s \).

Hence there is an \( s \) such that for all stages namely, \( s \) such that all the \( T_j \) have stopped. The strings that the \( C_j \) place into \( A \) are too

\( RNP_i \) can only be injured by a \( T_j, j < i \), but often.

Each \( RP_i \), \( RNP_i \) only acts finitely often. At any stage where it can be injured, then it will be forced to have to act again. Since it acts only finitely often as a restraint.

Lemma 2. Every \( RP_i \) receives attention.

Proof. Let \( s \) be a stage such that \( s > s \) exists via Lemma 1). If \( RP_i \) ever receives permanently satisfied. Thus \( RP_i \) may receive at stage \( s \).

Lemma 3. \( L_i^s \) is infinite.
and \( C_i \) place

\[ i = |L_c^1|, \]

and let \( k = |\log(s) + 1| \). If \( 1^{2^k} \) is not restrained by a
requirement of higher priority than \( T_i \) then put \( 1^{2^k} \) into \( A \). At this point \( T_i \) is
satisfied.

For any \( j < s/2 \) and any \( x \) with \( p_j(|x|) = s \), run \( NP_j^1(x) \). If it accepts, then
put \( w = (j, x) * (j, x) * 1^{2^{|x|}} * O \) into \( A \), unless \( w \) is restrained by a
requirement of higher priority than \( C_j \) or \( |w| < s \). Note that \( w \) is of odd
length and hence does not put any elements into \( L_1^1 \) or \( L_2^1 \); and, when
\( |w| > s \), placing \( w \) into \( A \) will not interfere with any of the \( NP \) computations.

END OF CONSTRUCTION.

**Lemma 1.** Every requirement is injured only finitely often, and will only
restrain a finite number of strings from \( A \).

*Proof.* The requirements \( T_i \) and \( C_i \) never are injured or impose any
restraint, so the lemma is automatically true for them. We turn our attention
to the \( RP_i \) and \( RNP_i \) requirements.

\( RP_i \) can only be injured by a \( T_j, j < i \), or a \( C_j, j < i \). Once a \( T \) requirement
is satisfied it never acts again, so \( T_j \) can only injure \( RP_i \) once. The \( C_j \) never
stops acting, but, note that at stage \( s \)

\[ i \]

\( RP_i \) needs to restrain strings of length at most \( p_i(s) \),

\( C_j \) needs to place strings of the form \( (j, x) * (j, x) * 1^{2^{|x|}} * O \),

where \( p_j(|x|) = s \), into \( A \).

A simple calculation reveals that eventually the length of the strings that
\( C_j \) places into \( A \) always exceed the length of the strings that \( RP_i \) restrains.

Hence there is an \( s \) such that for all stages past \( s \), \( RP_i \) is never injured.

\( C_j \) places into \( A \) are too large to injure \( RP_i \).

\( RNP_i \) can only be injured by a \( T_j, j < i \), hence can be injured only finitely
often.

Each \( RP_i, RNP_i \) only acts finitely often, because it ever acts past the
stage where it can be injured, then it will be satisfied permanently and never
have to act again. Since it acts only finitely often it imposes only finite
restraint.

**Lemma 2.** Every \( RP_i \) receives attention finitely often.

*Proof.* Let \( s_0 \) be a stage such that \( s > s_0 \) \( RP_i \) does not get injured (such
exists via Lemma 1). If \( RP_i \) ever receives attention past \( s_0 \), then it will be
permanently satisfied. Thus \( RP_i \) may receive attention at most once after
stage \( s_0 \).

**Lemma 3.** \( L_i^1 \) is infinite.
Proof. We prove that $\forall i \mid L^i \mid > i$. Assume inductively that $\exists s > s_0$ such that $\mid L^s \mid > i$. Let $s_1$ be a stage such that $s_1 > s_0$, $s_1$ even, $s_1$ is not a power of 2, and none of the $RP_j$, $0 \leq j \leq i$, receive attention at $s_1$ (such exists by Lemma 2). At stage $s_1$, a string of length $s_1$ will be placed into $A$ for the first time. Hence

$O^i$ is an element of $L^i$ not in $L^i_0$.

Therefore $\mid L^i \mid > i + 1$.  

Lemma 4. $RP_i$, $RNP_i$, $T_i$, and $C_i$ are all satisfied.

Proof. In all the proofs below, let $s_0$ denote the stage past which the requirement in question, $Z$, will never be injured, and let $s_1$ denote the length of the longest string restrained by requirements of higher priority than $Z$. Both $s_0$ and $s_1$ exist by Lemma 1.

$RP_i$: Assume $P_i \cap \{0\}* \mid$ is infinite. Let $s$ be a stage such that

(a) $s_0 < s$,
(b) $i < \log s$,
(c) $p_i(s) < 2^{s/2}$,
(d) $|L^s_i| > i$,
(e) $s$ is even and not a power of 2,
(f) $P_i^s$ accepts $O^i$.

If no such $s$ exists, then $P_i^s$ only accepts odd length strings or those which are of length a power of 2, hence the requirement is satisfied. If such an $s$ exists, then at that stage $RP_i$ will act, and be satisfied forever because nothing of higher priority ever injures it.

$RNP_i$: Assume $NP_i^s \cap \{0\}* \mid$ is infinite. Let $s$ be a stage such that

(a) $s_0 < s$,
(b) $i < \log s$,
(c) $p_i(s) < 2^{s/2}$,
(d) $p_i(\log(s + 1)) < s$,
(e) $s$ is odd,
(f) $NP_i^s$ accepts $O^k$, where $k = \log s + 1$.

Such a stage must exist since $NP_i^s \cap \{0\}* \mid$ is infinite, and as $s$ goes through all the odd numbers, $\log s + 1$ goes through all the natural numbers. At stage $s$, $RNP_i$ will act and be satisfied forever, or ever injures it.

$T_i$: During any stage $s$ such that $\exists s' > s_0$, the requirement $T_i$ will be free to put strings of cardinality of $L^i_0$ until $|L^s_i| > i$, at which point

$C_i$: The only reasons not to put a code in conflicts with something of higher priority than the stage itself). By Lemma 1, the code is finite number of strings. By a simple calculation we know which a code string for $C_i$ is longer than the others put it into $A$. Hence the latter reason also strings, and the requirement is satisfied.  

4. $EXP_k^A \subseteq NP_k^{L^A}$

In this section we show that the opposite is possible as well. We obtain a stronger type of an oracle $B$ such that there is with no infinite subset of $L^B$ or $L^B$ in $EXP_k^B$, the problem to us, pointed out the following

Proposition. Let $B$ be an oracle such that $L^B$ contains no infinite subset of $L$ or $L$. Then $L$ must operate in time greater than $2^{2^{|n|}}$.

Proof. Assume there is a deterministic $T$ and operates in time $2^{2^{|n|}}$ (henceforth referred to as $T$) often. It must operate in good time on an input to the machine by having it always reject if it is the original machine accepts an infinite subset of $L$, the machine accepts an infinite subset of $L$ in good hypothesis of the theorem. In the $L^B$ case, having it reverse its answers on those inputs. The new machine accepts an infinite subset of $L^B$.

Theorem 3. There exists a recursive oracle $B$ and this inequality is witnessed by a language $L_0$ subset of $L^B$ or of $L^B$ is in $EXP_k^B$.

Proof. For clarity we present the proof a value of $k$ is similar.

We construct the set $B$ in stages. During
is \( s > s' \) such not a power \( n \) exists by for the first stage \( s, RNP_f \) will act and be satisfied forever since nothing of higher priority ever injures it.

\( T_i \): During any stage \( s \) such that \( 2^k > s_1 \), where \( k = \lceil \log(s) + 1 \rceil \), requirement \( T_i \) will be free to put strings into \( A \) and thus increase the cardinality of \( L_s^2 \) until \( |L_s^2| > l \), at which point \( T_i \) is satisfied.

\( C_i \): The only reasons not to put a code string into \( A \) are: if putting it in conflicts with something of higher priority; or, if the string is short (less than the stage itself). By Lemma 1, the former reason can only restrain a finite number of strings. By a simple calculation, there is a stage \( s \) past which a code string for \( C \) is longer than the stage number at which \( C \) acts to put it into \( A \). Hence the latter reason also only restrains finitely many strings, and the requirement is satisfied.

\[ \]

4. \( EXP^g \subset NP \)

In this section we show that the opposite inclusion of the previous section is possible as well. We obtain a stronger type of result than Theorem 2 in that we exhibit an oracle \( B \) such that there is an infinite language \( L^g \in NP^g \) with no infinite subset of \( L^g \) or \( I^g \) in \( EXP^g \). Albert Meyer, who suggested the problem to us, pointed out the following consequence:

**Proposition.** Let \( B \) be an oracle such that there is an infinite \( L \in NP^g \) with no infinite subset of \( L \) or \( I \) in \( EXP^g \). Then any deterministic algorithm for \( L \) must operate in time greater than \( 2^{cn} \) on all but a finite set of points.

**Proof.** Assume there is a deterministic Turing machine that recognizes \( L \) and operates in time \( 2^{cn} \) (henceforth referred to as "good time") infinitely often. It must operate in good time on an infinite subset of \( L \) or \( I \). Modify the machine by having it always reject if it runs in time greater than \( 2^{cn} \). If the original machine accepts an infinite subset of \( L \), then the modified machine accepts an infinite subset of \( L \) in good time, which contradicts the hypothesis of the theorem. In the \( L \) case, modify the machine further by having it reverse its answers on those inputs on which it halted in good time. The new machine accepts an infinite subset of \( L \), again a contradiction.

**Theorem 3.** There exists a recursive oracle \( B \) such that \( EXP^g \subset NP^g \), and this inequality is witnessed by a language \( L^g \in NP^g \) such that no infinite subset of \( L^g \) or of \( I^g \) is in \( EXP^g \).

**Proof.** For clarity we present the proof for \( k = 1 \). The proof for larger values of \( k \) is similar.

We construct the set \( B \) in stages. During the construction we code \( EXP^g \).
into \( NP^B \) by: for all \( i \) and \( x \), \( E_i^B \) accepts \( x \) if \( \exists w \) such that \( \langle i, x \rangle * w \in B \) and \( |w| = |\langle i, x \rangle|^4 + 1 \). Clearly this implies \( EXP^B \subseteq NP^B \). We let

\[
L^B = \{ x \mid \exists w \in B, |w| = |x|^4 \}.
\]

Note that \( L^B \in NP^B \) for all \( B \). To take care of infinite subsets of \( L^B \) and \( \overline{L^B} \) we have the following requirements:

- \( R_{(1,0)}: E_i^B \) accepts an infinite set implies \( E_i^B \cap \overline{L^B} \neq \emptyset \).
- \( R_{(2,0)}: E_i^B \) accepts an infinite set implies \( E_i^B \cap L^B \neq \emptyset \).

The requirements inherit a priority ordering from the ordering given by the pairing function \((−,−)\).

We now describe the construction of \( B \). Recall that \( h_i \) bounds the running time of machine \( E_i \). We let \( B_{k} \) denote the strings put into \( B \) through the first \( k \) stages.

**Construction.**

**Stage 0:** \( B_0 = \emptyset \).

**Stage \( s + 1 \):** (1) For each \( i \leq s \), if \( h_i(s) < 2^{s^4} \), then run \( E_i^B \) on all strings of length \( s \) and preserve each of these computations by restraining from \( B \) all the strings queried in the computation which were not in \( B \). Note that the total number of strings restrained at this stage is at most \( 2^s \cdot 2^{s^4} = 2^{s^4+1} \).

(2) Find the least \( i = (j, e) < s \) such that

- (a) \( R_j \) is not satisfied.
- (b) There is an \( x \in \Sigma^* \) such that \( |x| = s \) and \( E_j^B \) accepts \( x \).
- (c) \( h_i(s) < 2^{s^4} \).

If \( i = 1 \), then to ensure \( E_1^B \cap \overline{L^B} \neq \emptyset \) we restrain all strings of length \( s^4 \) from \( B \).

If \( i = 2 \) then to ensure \( E_2^B \cap L^B \neq \emptyset \) we place into \( B \) the least string of length \( s^4 \) that is not restrained from \( B \). There must be such a string since the total number of strings restrained from \( B \) up to this point in the construction is less than \( \sum_{i=1}^{j} 2^{i^4} < 2^{s^4} \). (Note: The \( j = 1 \) case of the previous stages restrains only strings of length less than \( s^4 \) so \( R_j \) is now said to be satisfied.

(3) For each \( E_i^B(x) \) which has just been run and which accepted \( x \), find some \( w \) such that \( |w| = |\langle i, x \rangle|^4 + 1 \), and \( \langle i, x \rangle * w \) is not restrained from \( B \). Put \( \langle i, x \rangle * w \) into \( B \). Such a \( w \) will exist, since the number of possible \( w \)'s is greater than \( 2^{s^4+1} \) which exceeds the number of strings restrained.

**End of Construction.**

We prove that each \( R_j \) is eventually satisfied. Note that each \( R_j \) is acted upon at most once. Assume that \( i = (j, e) \) and \( E_j^B \) accepts an infinite set. Let

\( s_0 \) be a stage beyond which no \( R_k \), \( k < i \), was satisfied. Let \( s > s_0 \), \( h_e(s) < 2^{s^4} \). Past \( s_0 \), all computations \( E_i^B \) is infinite, there is a stage \( s_1 > s \) where \( E_i^B \) is saturated at \( s_1 \). At this stage \( R_i \) will be acted upon and hence satisfied.

5. \( NPC \) Incomparable

Our final result shows it may be the case that one set is incomparable to the other.

**Theorem 4.** There is a recursive set \( C \) s.t.

1. \( \exists \) an infinite language \( L_1^C \) such that all its subsets are in \( EXP^C \).
2. \( \exists \) an infinite language \( L_2^C \in NP^C \), not in \( EXP^C \).

**Proof.** As usual the construction of set \( C \) is defined

\[
L_1^C = \{ O \mid \exists x \in C, |x| = n^{k+1} \}
\]

\[
L_2^C = \{ O \mid \exists x \in C \text{ and } n \text{ not in } NP^C \}
\]

(Note: The requirement on \( n \) in \( L_1^C \) is a conversation of element into \( L_1^C \) does not affect \( L_1^C \).) For no \( L_1^C \) \( (L_2^C) \) by \( L_1^C \) \( (L_2^C) \) throughout.

Clearly we have \( L_1^C \subseteq NP^C \) and \( L_2^C \subseteq EXP^C \).

The \( L_1^C \) is infinite and contains no infinite subsets in terms of requirements:

- \( R_i: E_i^C \) infinite \( \Rightarrow E_i^C \notin L_1^C \)
- \( T_i: NP^C \) infinite \( \Rightarrow NP^C \notin L_2^C \).

We will keep track of sets \( G \) and \( H \) containing strings put into \( C \), which have already been met. Also, at each stage \( n \), \( n \) be chosen large enough so that every string put into \( C \) before stage \( s \) has length less than \( n \).

**Construction.**

**Stage 0:** \( C_0 = G = H = \emptyset \), \( n_0 = 0 \).

**Stage \( s \):** There are two cases.
$s_0$ be a stage beyond which no $R_k$, $k < i$, will act, and such that for all $s > s_0$, $h_s(s) < 2^{2^i}$. Past $s_0$, all computations of $E_r^B$ are preserved. So, since $E_r^B$ is infinite, there is a stage $s_1 > s_0$ where $E_r^B$, accepts some string of length $s_1$. At this stage $R_1$ will be acted upon and hence become satisfied.

5. NP$^C$ INCOMPARABLE TO EXP$^C_k$

Our final result shows it may be the case that neither of NP$^C$ or EXP$^C_k$ is contained in the other.

**Theorem 4.** There is a recursive set $C$ such that

1. There is an infinite language $L_C^i \in NP^C$, none of whose infinite subsets are in EXP$^k_C$.

2. There is an infinite $L_C^i \in EXP^C_k$, none of whose infinite subsets are in NP$^C$.

**Proof.** As usual the construction of set $C$ is carried out in stages. We define

- $L_C^i = \{O^n \mid \exists x \in C, |x| = n^{k+1}\}$
- $L_C^i = \{O^n \mid 1^{2^{2^n}} \in C$ and $n$ is not divisible by $k+1\}$.

(Note: The requirement on $n$ in $L_C^i$ is a convenience to assure that putting an element into $L_C^i$ does not affect $L_C^i$.) For notational convenience we denote $L_1 (L_2)$ by $L_1 (L_2)$ throughout.

Clearly we have $L_1 \in NP^C$ and $L_2 \in EXP^C_k$. We need to ensure that $L_1 (L_2)$ is infinite and contains no infinite subset in EXP$^C_k$ (NP$^C$). We state this in terms of requirements:

- $R_i$: $E_i^C$ infinite $\Rightarrow E_i^C \not\subseteq L_1$
- $T_j$: NP$^C_j$ infinite $\Rightarrow$ NP$^C_j \not\subseteq L_2$.

We will keep track of sets $G$ and $H$ containing indices of requirements which have already been met. Also, at each stage $s$, we define an integer $n_s$, $n_s$ will be chosen large enough so that every string put into $C$ or restrained from $C$ before stage $s$ has length less than $n_s$. We let $C_i$ denote the set of strings put into $C$ through stage $s$ of the construction. We now describe the construction.

**Construction.**

Stage 0: $C_0 = G = H = \emptyset$, $n_0 = 0$.

Stage s: There are two cases.
Case 1. \( s \) is even. We consider oracle machines \( NP^C_{i-1}, \) where \( i \leq s/4 \) and \( i \notin G. \) Let \( n_r > n_{s-1} \) be least such that

1. \( \forall i < s/4, \ p_i(n_r) < 2^{n_i} \)
2. \( s \) is not divisible by \( k + 1 \)
3. every string in \( C_{s-1} \) has length less than \( n_r \)
4. \( 1^{2^{n_r}} \) is not restrained from \( C. \)

Find the least index \( i_0 \leq s/4 \) such that \( i_0 \notin G \) and \( NP^C_{i_0-1} \) accepts \( O^{n_0}. \) If such an \( i_0 \) exists, restrain \( 1^{2^{k}} \) from \( C \) and put \( i_0 \) into \( G. \) If no such \( i_0 \) exists, put \( 1^{2^{k}} \) into \( C \) and hence add \( O^{n_0} \) to \( L_2. \)

Case 2. \( s \) is odd. We consider oracle machines \( E^{C-1}_i, \) where \( i < s/4 \) and \( i \notin H. \) Find an integer \( n_r > 2^{(n_{s-1}+1)/2} \) such that

1. \( \forall i < s/4, \ h_i(n_r) < 2^{n_i^{1/2}} \)
2. \( (s + 1)/2 \) \( 2^{n_i^{1/2}} \) \( 2^{n_i^{1/2}} \)
3. No element of length \( n_r^{1/2} \) is in \( C_{s-1} \) or restrained from \( C. \)

For each computation \( E^{C-1}_i(O^{n_i}) \) with \( i \notin H, \ i < s/4, \) restrain from \( C \) all strings queried in the computation which are not in \( C_{s-1} \). Find the least such index \( i_0, \) such that \( E^{C-1}_i \) accepts \( O^{n_i}. \) If such an \( i_0 \) exists, put it into \( H \) and restrain from \( C \) all strings of length \( n_i^{1/2}. \)

If no such \( i_0 \) exists, put the least string of length \( n_i^{1/2} \) which is not restrained from \( C_{s-1} \) into \( C. \) (Note that such a string of length \( n_i^{1/2} \) must exist, since no string of this length is restrained from \( C \) prior to stage \( s, \) and at stage \( s \) at most \((s + 1)/2 \) \( 2^{n_i^{1/2}} \) \( < 2^{n_i^{1/2}} \) strings of length \( n_i^{1/2} \) are restrained from \( C. \) This adds \( O^{n_0} \) to \( L_1. \)

END OF CONSTRUCTION.

That both \( L_1 \) and \( L_2 \) are infinite follows from the fact that at a stage \( s \) we only consider machines with indices less than \( s/4. \) Hence, after an even stage \( s \) at least \( s/2 \) elements have been put into \( L_2, \) and after an odd stage \( s \) at least \( (s - 1)/2 \) elements have been put into \( L_1. \) So it remains to show that the requirements are all satisfied.

**Lemma 1.** Each requirement \( R_i \) is satisfied.

**Proof.** Assume not, and let \( i_0 \) be the least \( i \) with \( R_i \) not satisfied. Then \( E^{C}_{i_0} \) is an infinite subset of \( L_1, \) and \( i_0 \) is never put into \( H, \) as when an index \( j \) is put into \( H \) we ensure that \( E^C_j \notin L_1. \) Let \( s \) be an odd stage such that \( i_0 < s/4 \) and, for every \( j < i_0 \) ever put into \( H, \) \( j \) is in \( H \) before stage \( s. \) As \( i_0 \notin H, \) at every stage \( s \geq s, \) the computation \( E^{C-1}_{i_0} \) rejects \( 0^{n_0}, \) and these computations are preserved. This contradicts our assumption that \( E^{C}_{i_0} \) is an infinite subset of \( L_1. \)  

**Lemma 2.** Each requirement \( T_i \) is satisfied.

**Proof.** Assume not, and let \( i_0 \) be the least \( i \) with \( T_i \) not satisfied. Let \( s \) be any even stage such that \( i_0 \) \( NP^{C-1}_{i_0} \) must reject \( O^{n_0}, \) since otherwise at stage \( s \) from \( C, \) \( NP^{C-1}_{i_0} \) would be preserved and would have \( NP^{C-1}_{i_0} \in L_2. \) Similarly we see that \( NP^{C-1}_{i_0} \) is preserved for every \( s \geq s \) an infinite subset of \( L_2 \) and \( T_{i_0} \) is satisfied.

6. Further Research

The techniques and question explored in the above can be extended to apply at other settings. One might hope to get results in the strong form they are obtained here. Let \( A \) be constructed such that there is a \( (\text{Relativized} \sum_{p=1}^{p^*} \text{classes are defined in Bal} \) infinite subset of \( L \) is in \( \sum_{p=4}^{p^*}? \) Can \( L \) be such a \( \sum_{p=4}^{p^*}? \) Such results would be of interest to \( NP \) theory for recognizing any infinite set.

Other results relating \( NP \) and \( EXP_k \) are in Maass (1983), an oracle \( B \) is constructed such that \( K^B \) has an infinite set in \( NP \) and \( EXP_k \) also contains an infinite set in \( EXP_k \) with \( K \). The author has shown that for almost all oracles, with impartiality, and has generalized other deterministic vs nondeterministic questions (Gasarch, in press).

Acknowledgments

The authors would like to thank Ron Book, Hans Buhr, and Steven Sipser for helpful discussions and communications.

Received March 31, 1983; accepted June 28, 1983.

References

Baker, T. Gill, J. and Solovay, R. (1975), Relativization of Questions about \( NP \).
Lemma 2. Each requirement $T_i$ is satisfied.

Proof. Assume not, and let $i_0$ be the least number such that $T_{i_0}$ is not satisfied. Let $s$ be any even stage such that $i_0 \leq s/4$. Then at any even $s_1 \geq s$, $NP_{i_0}^{s_1-1}$ must reject $O^{s_1-1}$, since otherwise at stage $s_1$, $1^{2s_1}$ would be restrained from $C$, $NP_{i_0}^{s_1-1}(O^{s_1})$ would be preserved (since $n_i \geq 1$) and we would have $NP_{i_0}^{s_1} \subseteq L_2$. Similarly we see that the rejecting computation $NP_{i_0}^{s_1-1}(O^{s_1})$ is preserved for every $s_1 \geq s$ and so $NP_{i_0}^{s_1}$ cannot be an infinite subset of $L_2$ and $T_{i_0}$ is satisfied.

6. Further Research

The techniques and question explored in this paper might well be looked at in other settings. One might hope to get previously known relativization results in the strong form they are obtained here. For example, can an oracle $A$ be constructed such that there is a language $L$ in $\sum_{2}^{P,A} - \sum_{1}^{P,A}$? (Relativized $\sum_{p}^{P,A}$ classes are defined in Baker and Selman, 1979) and no infinite subset of $L$ is in $\sum_{1}^{P,A}$? Can $L$ be such that no infinite subset of $L$ or $\bar{L}$ is in $\sum_{1}^{P,A}$? Such results would be of interest in that they imply that any $NP^{A}$ algorithm for recognizing any infinite subset of $L$ (or $\bar{L}$) must fail.

Other results relating $NP$ and $EXP_A$ are also possible. In Homer and Maass (1983), an oracle $B$ is constructed such that $P^{B} \neq NP^{B}$ and every infinite set in $NP^{B}$ has an infinite $P^{B}$ subset. Similar questions might be asked here. For example, can one construct a set $B$ such that $NP^{B} \subseteq EXP^{B}$ and every infinite set in $EXP^{B}$ contains an infinite $NP^{B}$ subset? The first author has shown that for almost all oracles $A$, $EXP^{A}$ and $\sum_{p}^{P,A}$ are incomparable, with immunity, and has generalized the theorems in this paper to other deterministic vs nondeterministic questions. These results will appear in (Gasarch, in press).

Acknowledgments

The authors would like to thank Ron Book, Harry Lewis, Albert Meyer, and Michael Sipser for helpful discussions and communications.

Received March 31, 1983; Accepted June 28, 1983

References

The Complexity of Evaluating Relational Queries

STAVROS S. COSMAKIS
Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts

A sequence of results which characterize expressibility, which is related to the evaluation of relational queries consider the complexity of joins. It is shown that testing whether a given relation equals some other given relation in a language that is equal to the intersection of a co-NP—includes both NP and co-NP, and within a different context, see Papadimitriou and Yannakakis (1992) Annual ACM Sympos. on the Theory of Comput. pp. 255–260). It is shown that testing inclusion is in the complexity of the resulting query. This is examined.

1. INTRODUCTION

The relational algebra is known to be a suitable formalism for expressing database queries. But exactly how hard is the expressibility, Codd showed in his classical paper to a version of first-order logic. In terms of computational complexity, there have been several results on the relational algebra on finite relations embodies some Zariski power. Already in (Aho et al., 1979; and Codd et al, 1980) shown that evaluation as well as testing of conjunctive relational queries are hard combinatorially, and there were results suggesting that the join of relations in relational algebra is even in certain weak senses of the word (Hachihara et al., 1981) (a polynomial time algorithm for computing the join) (Honeyman, 1980), and that project-join queries have a given conjectured result for inclusion (Maier, 1985).

In some sense, relational algebra seems to be a completely different problem from, say, the ordinary algebra (for example, matrix multiplication, exponentiation). In ordinary algebra, as in relational algebra.