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 EPSL Periodical Stacks Q350.I5 v.56-v.59 (1983)
Location:

Notes

9/11/2009 7:42:06 AM System 1. No Matching Bib/2. No ISBN, ISSN, or OCLCNo in request.

INFORMATION AND CONTROL 58, 88–100 (1983)

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 $EXP_k^A = \bigcup_{c=0}^{\infty} DTIME(2^{cn^k})$ 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) $P^A \subsetneq NP^A \subsetneq EXP_k^A$, (ii) $EXP_k^B \subsetneq NP^B$, and (iii) NP^C and EXP_k^C 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 unrelativized 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

$EXP_k = \bigcup_{c=0}^{\infty} DTIME(2^{cn^k})$. Obviously NP
how NP relates to EXP_k for an arbitrary fix
early results on this topic are contained in
(1972). In addition, Heller's thesis (1980) co
concerning exponential time. The arguments
can be used to show that $NP^A \neq EXP_k^A$ for
slightly different proof of this fact here. This
show that all of the other relativized situation
A, B, and C are constructed so that

(i) $P^A \subsetneq NP^A \subsetneq EXP_k^A$
(ii) $EXP_k^B \subsetneq NP^B$
(iii) NP^C and EXP_k^C are incomparable

Bennett and Gill (1981) show the existence
is an NP^A set with no infinite P^A subset. T
1981) is probabilistic and nonconstructive. In
independently in (Schoning, 1982) a constru
We use the methods of (Homer and Maass,
about NP^A and EXP_k^A. For example, an orac
only are NP^C and EXP_k^C incomparable but t
and L_2 such that

(1) $L_1 \in NP^C$ and L_1 contains no infi
(2) $L_2 \in EXP_k^C$ and L_2 contains no in

Such an $L_1(L_2)$ is said to be immune w
More generally, if L is a language such tha
particular complexity class then L is said to
class. Recently, Book and Schoning (198
immunity and obtained results for a wide va

2. NOTATION AND FIRS

All languages considered will be subsets
$\{P_i^{(\,)}\}_{(i=0,1,2,\ldots)}$ $(\{NP_i^{(\,)}\}_{(i=0,1,2,\ldots)})$ of po
ministic (nondeterministic) oracle Turing ma
P_i^X (NP_i^X) for machine $P_i^{(\,)}$ $(NP_i^{(\,)})$ with orac
result, for the language accepted by that r
computation of machine M on input x.
bounds the time for any computation on an

For any natural number k and any set X
languages accepted by deterministic Turing

$EXP_k = \bigcup_{c=0}^{\infty} DTIME(2^{cn^k})$. Obviously $NP \subseteq \bigcup_{k=0}^{\infty} EXP_k$; hence, we ask how NP relates to EXP_k for an arbitrary fixed k. A number of interesting early results on this topic are contained in Dekhytar (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 $NP^A \neq EXP_k^A$ 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) $P^A \subsetneq NP^A \subsetneq EXP_k^A$

(ii) $EXP_k^B \subsetneq NP^B$

(iii) NP^C and EXP_k^C 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_k^A. For example, an oracle C is constructed so that not only are NP^C and EXP_k^C 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_k^C subset.

(2) $L_2 \in EXP_k^C$ and L_2 contains no infinite NP^C subset.

Such an $L_1(L_2)$ is said to be immune with respect to $EXP_k^C(NP^C)$ 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 $\{P_i^{(\,)}\}_{(i=0,1,2,\ldots)}$ $(\{NP_i^{(\,)}\}_{(i=0,1,2,\ldots)})$ of polynomial time-bounded deterministic (nondeterministic) oracle Turing machines. For any $X \subseteq N$ we write P_i^X (NP_i^X) for machine $P_i^{(\,)}$ $(NP_i^{(\,)})$ 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 $p_i(n) = i + n^i$ bounds the time for any computation on an input of length n.

For any natural number k and any set X, EXP_k^X denotes the collection of languages accepted by deterministic Turing machines with oracle X which run in time 2^{cn^k}, 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^k} for some c. We may assume $h_i(n) = i + 2^{in^k}$ bounds the time of any computation by E_i^X on inputs of length n. $P^X(NP^X, EXP_k^X)$ denotes the collection of all languages $L \in P^X(NP^X, EXP_k^X)$. For a more complete account of these definitions see Hopcraft and Ullman (1979).

$(\ , \ )$: $N \times N \to N$ denotes a fixed recursive pairing function which is monotonic in both coordinates. $\langle \ , \ \rangle$: $\{0,1\}^* \times \{0,1\}^*$ 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 $\langle \ , \ \rangle$ 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, $\bar{L}$ denotes the complement of L. $x * y$ denotes the concatenation of strings x and y. The function log( ) 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, $NP^A \neq EXP_k^A$. The following argument was pointed out to us by Michael Sipser.

THEOREM 1. *For all A, if $EXP_k^A \subseteq NP^A$ then $EXP_{k+1}^A \subseteq NP^A$.*

Proof. The key observation here is that by polynomially padding sets in EXP_k^A one may decrease their complexity.

Let $L \in EXP_{k+1}^A$. Define $L' = \{xO^{n^{k+1}-n} \mid |x| = n \text{ and } x \in L\}$. Now $L' \in EXP_k^A$, since we can modify the machine which recognizes L in EXP_{k+1}^A to recognize the "padded language" L' in EXP_k^A. Hence by our assumption $L' \in NP^A$ and so $L \in NP^A$ since NP^A is closed under polynomial length padding. ∎

As a corollary we have:

COROLLARY. *For all A, $NP^A \neq EXP_k^A$.*

Proof. By a straightforward diagonalization, EXP_k^A is properly contained in EXP_{k+1}^A (see Hopcraft and Ullman, 1979, p. 299). Now if $NP^A = EXP_k^A$ then $EXP_{k+1}^A \subset NP^A$ by the theorem. Hence we have $EXP_{k+1}^A \subseteq NP^A \subsetneq EXP_{k+1}^A$, a contradiction. ∎

3. $P^A \subsetneq NP^A \subsetneq EXP_k^A$

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

$NP^A = P^A \subsetneq EXP_k^A$ by the relativized version
(Hopcraft and Ullman, 1979, p. 299). We c
that $P^A \subsetneq NP^A \subsetneq EXP_1^A$, and moreover we
immunity. Clearly this implies the same res
EXP_1^A.

THEOREM 2. *There exists an oracle A su*

(1) $P^A \subsetneq NP^A \subsetneq EXP_1^A$.

(2) *There is an infinite $L_1^A \in NP^A$ wit*

(3) *There is an infinite $L_2^A \in EXP_1^A$ wi*

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

$L_1^A = \{O^n \mid \exists x \in A, |x| = n, n$ ever

$L_2^A = \{O^n \mid 1^{2^n} \in A\}$.

Clearly $L_1^A \in NP^A$ and $L_2^A \in EXP_1^A$ for any
have the following requirements:

RP_i: $P_i^A \cap \{O\}^*$ infinite $\Rightarrow P_i^A \cap \bar{L}$

RNP_i: $NP_i^A \cap \{O\}^*$ infinite $\Rightarrow NP_i^A$

T_i: $|L_2^A| > i$ (This is to ensure t

C_i: For almost all x, $NP_i^A(x)$
$O \in A$. (This is to ensure th

The infinitude of L_1^A will follow easily fro
be a formal requirement.

We construct A in stages. A_s denotes th
end of stage s. $A = \bigcup_{s=0}^{\infty} A_s$. The expressio
be used in reference to a computation on a
and will mean to restrain from A all strin
which were not in A_s.

RP_i will act by diagonalizing. It will re
preserve a computation or to prevent a cert
latter type of restraint will not affect the ot
and the code strings (requirement C_i) the
sets.

Both RNP_i and C_i operate only when th
changed, and all the strings they query b
Thus these requirements never have to pres
may restrain a string to preserve membersh

$NP^A = P^A \subsetneq EXP_k^A$ by the relativized version of the time hierarchy theorem (Hopcraft and Ullman, 1979, p. 299). We construct here an oracle A such that $P^A \subsetneq NP^A \subsetneq EXP_1^A$, and moreover we obtain these inequalities with immunity. Clearly this implies the same result for EXP_k^A, $k > 1$, instead of EXP_1^A.

THEOREM 2. *There exists an oracle A such that*

(1) $P^A \subsetneq NP^A \subsetneq EXP_1^A$.

(2) *There is an infinite* $L_1^A \in NP^A$ *with no infinite* P^A *subset.*

(3) *There is an infinite* $L_2^A \in EXP_1^A$ *with no infinite* NP^A *subset.*

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

$$L_1^A = \{O^n \mid \exists x \in A, |x| = n, n \text{ even}, n \text{ not a power of } 2\}$$

$$L_2^A = \{O^n \mid 1^{2^n} \in A\}.$$

Clearly $L_1^A \in NP^A$ and $L_2^A \in EXP_1^A$ for any A. To ensure (1), (2), and (3) we have the following requirements:

RP_i: $P_i^A \cap \{O\}^*$ infinite $\Rightarrow P_i^A \cap \overline{L_1^A} \neq \varnothing$

RNP_i: $NP_i^A \cap \{O\}^*$ infinite $\Rightarrow NP_i^A \cap \overline{L_2^A} \neq \varnothing$

T_i: $|L_2^A| > i$ (This is to ensure that L_2^A is infinite)

C_i: For almost all x, $NP_i^A(x)$ accepts iff $\langle i, x\rangle * \langle i, x\rangle * 1^{2^{|x|}} * O \in A$. (This is to ensure that $NP^A \subseteq EXP_1^A$.)

The infinitude of L_1^A will follow easily from the construction and need not be a formal requirement.

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

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_1^A. The latter type of restraint will not affect the other requirements because L_1^A, L_2^A, and the code strings (requirement C_i) they query, operate an on different sets.

Both RNP_i and C_i operate only when the computation by NP_i^A 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 $\overline{L_2^A}$.

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 prority ordering is

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

Construction.

Stage 0: $A_0 = \varnothing$.

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 $P_i^{A_s}(O^s)$ for each $i \leqslant \log s$ such that $p_i(s) < 2^{s/2}$. If RP_i is not satisfied, then preserve $P_i^{A_s}(O^s)$ for RP_i. Find the least i such that RP_i is not satisfied, $P_i^{A_s}(O^s)$ accepts, and $|L_1^{A_s}| > i$. 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

(stages) $*$ (machines run per stage) $*$ (maximum number of queries per machine)

which is less than $s * \log(s) * 2^{s/2} < 2^s$.) This is done to make L_1^A infinite.

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

Let $k = \lceil \log(s) + 1 \rceil$.

Run $NP_i^{A_s}(O^k)$ for each $i \leqslant \log s$ 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 $NP_i^{A_s}(O^k)$ is accepted. If 1^{2^k} 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 $i = |L_2^{A_s}|$, and let $k = \lceil \log(s) + 1 \rceil$. If
requirement of higher priority than T_i then pu
satisfied.

For any $j < s/2$ and any x with $p_j(|x|) = s$,
put $w = \langle j, x\rangle * \langle j, x\rangle * 1^{2^{|x|}} * O$ into A,
requirement of higher priority than C_j or |
length and hence does not put any elemen
$|w| > s$, placing w into A will not interfere wi

END OF CONSTRUCTION.

LEMMA 1. *Every requirement is injured*
restrain a finite number of strings from A.

Proof. The requirements T_i and C_i ne
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
is satisfied it never acts again, so T_j can onl
stops acting, but, note that at stage s

(i) RP_i needs to restrain strings of le

(ii) C_j needs to place strings of the
where $p_j(|x|) = s$, into A.

A simple calculation reveals that eventua
C_j places into A always exceed the length
Hence there is an s such that for all stag
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 $T_j, j < i$, b
often.

Each RP_i, RNP_i only acts finitely often.
stage where it can be injured, then it will be
have to act again. Since it acts only finit
restraint. ∎

LEMMA 2. *Every RP_i receives attention*

Proof. Let s_0 be a stage such that $\forall s >$
exists via Lemma 1). If RP_i ever receives
permanently satisfied. Thus RP_i may rece
stage s_0. ∎

LEMMA 3. L_1^A *is infinite.*

Let $i = |L_2^{A_s}|$, and let $k = \lceil \log(s) + 1 \rceil$. 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 $NP_j^{A_s}(x)$. If it accepts, then put $w = \langle j, x \rangle * \langle j, x \rangle * 1^{2^{|x|}} * O$ 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^A or L_2^A; 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 $p_i(s)$,

(ii) C_j needs to place strings of the form $\langle j, x \rangle * \langle j, x \rangle * 1^{2^{|s|}} * O$, 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, namely, s such that all the T_j have stopped acting, and large enough so that the strings that the C_j place 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 if 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 $\forall s > s_0$ 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_1^A is infinite.*

Proof. We prove that $\forall i \; |L_1^A| > i$. Assume inductively that $\exists s > s_0$ such that $|L_1^{A_{s_0}}| > i$. 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, $0 \leqslant j \leqslant i$, 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

$$O^{s_1} \text{ is an element of } L_1^{A_{s_1}} \text{ not in } L_1^{A_{s_0}}.$$

Therefore $|L_1^{A_{s_1}}| > i + 1$. ∎

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 $P_i \cap \{0\}^*$ 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) $|L_1^{A_s}| > i$,

(e) s is even and not a power of 2,

(f) $P_i^{A_s}$ accepts O^s.

If no such s exists, then $P_i^{A_s}$ 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 $NP_i^A \cap \{0\}^*$ 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) $p_i(\lceil \log(s) + 1 \rceil) < s$,

(e) s is odd,

(f) $NP_i^{A_s}$ accepts O^k, where $k = \lceil \log s + 1 \rceil$.

Such a stage must exist since $NP_i^A \cap \{0\}^*$ is infinite, and as s goes through all the odd numbers, $\lceil \log s + 1 \rceil$ goes through all the natural numbers. At

stage s, RNP_i will act and be satisfied foreve
ever injures it.

T_i: During any stage s such that
requirement T_i will be free to put strings
cardinality of L_2^A until $|L_2^A| > i$, at which po

C_i: The only reasons not to put a co
in conflicts with something of higher priorit
than the stage itself). By Lemma 1, the for
finite number of strings. By a simple calc
which a code string for C is longer than the
put it into A. Hence the latter reason al
strings, and the requirement is satisfied. ∎

4. $EXP_k^B \subsetneq NP$

In this section we show that the opposite
is possible as well. We obtain a stronger ty
that we exhibit an oracle B such that there i
with no infinite subset of L^B or $\overline{L^B}$ in EXP
the problem to us, pointed out the following

PROPOSITION. *Let B be an oracle such t
with no infinite subset of L or $\overline{L}$ in EXP_k^B.
for L must operate in time greater than 2^{cn^k}*

Proof. Assume there is a deterministic Tu
and operates in time 2^{cn^k} (henceforth referr
often. It must operate in good time on an i
the machine by having it always reject if it
the original machine accepts an infinite s
machine accepts an infinite subset of L in
hypothesis of the theorem. In the $\overline{L}$ case,
having it reverse its answers on those inputs
The new machine accepts an infinite subset

THEOREM 3. *There exists a recursive o
and this inequality is witnessed by a languag
subset of L^B or of $\overline{L^B}$ is in EXP_k^B.*

Proof. For clarity we present the proof
values of k is similar.

We construct the set B in stages. During stage s, RNP_i 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_2^A until $|L_2^A| > i$, 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. $EXP_k^B \subsetneq NP^B$

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 $L^B \in NP^B$ with no infinite subset of L^B or $\overline{L^B}$ in EXP_k^B. 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 $L \in NP^B$ with no infinite subset of L or $\bar{L}$ in EXP_k^B. Then any deterministic algorithm for L must operate in time greater than 2^{cn^k} 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^k} (henceforth referred to as "good time") infinitely often. It must operate in good time on an infinite subset of L or $\bar{L}$. Modify the machine by having it always reject if it runs in time greater then 2^{cn^k}. 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 $\bar{L}$ 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 $\bar{L}$, again a contradiction. ∎

THEOREM 3. *There exists a recursive oracle B such that $EXP_k^B \subsetneq NP^B$, and this inequality is witnessed by a language $L^B \in NP^B$ such that no infinite subset of L^B or of $\overline{L^B}$ is in EXP_k^B.*

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_1^B

643 58/1-3-7

into NP^B by: for all i and x, E_i^B accepts x iff $\exists w$ such that $\langle i, x\rangle * w \in B$ and $|w| = |\langle i, x\rangle|^4 + 1$. Clearly this implies $EXP_1^B \subseteq NP^B$. 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:

$R_{(1,s)}$: E_s^B accepts an infinite set $\Rightarrow E_s^B \cap \overline{L^B} \neq \varnothing$.

$R_{(2,s)}$: E_s^B accepts an infinite set $\Rightarrow E_s^B \cap L^B \neq \varnothing$.

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: $B_0 = \varnothing$.

Stage $s+1$: (1) For each $i \leqslant s$, if $h_i(s) < 2^{s^2}$, then run $E_i^{B_s}$ 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_s. Note that the total number of strings restrained at this stage is at most $2 * 2^s * 2^{s^2} < 2^{s^3}$.

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

(a) R_i is not satisfied.

(b) There is an $x \in \sum^*$ such that $|x| = s$ and $E_e^{B_s}$ accepts x.

(c) $h_s(s) < 2^{s^2}$.

If $j = 1$, then to ensure $E_e^B \cap \overline{L^B} \neq \varnothing$ we restrain all strings of length s^4 from B.

If $j = 2$ then to ensure $E_e^B \cap L^B \neq \varnothing$ 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 $\sum_{i=1}^{s} 2^{s^3} < 2^{s^4}$. (Note: The $j = 1$ case of the previous stages restrains only strings of length less than s^4.) R_i is now said to be satisfied.

(3) For each $E_i^{B_s}(x)$ 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_i is eventually satisfied. Note that each R_i is acted upon at most once. Assume that $i = (j, e)$ and E_e^B accepts an infinite set. Let

s_0 be a stage beyond which no R_k, $k < i$, v
$s > s_0$, $h_e(s) < 2^{s^2}$. Past s_0, all computations
E_e^B is infinite, there is a stage $s_1 > s_0$ where E_e^B
s_1. At this stage R_i will be acted upon and h

5. NP^C INCOMPARABLE

Our final result shows it may be the case t
contained in the other.

THEOREM 4. *There is a recursive set C s*

(1) *There is an infinite language* L_1^C
subsets are in EXP_k^C.

(2) *There is an infinite* $L_2^C \in EXP_k^C$, *no*
in NP^C.

Proof. As usual the construction of set
define

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

$$L_2^C = \{O^n \mid 1^{2^{n^k}} \in C \text{ and } n \text{ is not}$$

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

Clearly we have $L_1 \in NP^C$ and $L_2 \in EXP$
(L_2) is infinite and contains no infinite subse
in terms of requirements:

R_i: E_i^C infinite $\Rightarrow E_i^C \not\subseteq L_1$

T_i: NP_i^C infinite $\Rightarrow NP_i^C \not\subseteq L_2$.

We will keep track of sets G and H cont
which have already been met. Also, at each s
n_s will be chosen large enough so that every
from C before stage s has length less than n
strings put into C through stage s of the con
construction.

CONSTRUCTION.

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

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 $s > s_0$, $h_e(s) < 2^{s^2}$. Past s_0, all computations of E_e^B are preserved. So, since E_e^B is infinite, there is a stage $s_1 > s_0$ where $E_e^{B_{s_1}}$ accepts some string of length s_1. At this stage R_i will be acted upon and hence become satisfied. ∎

5. NP^C Incomparable to EXP_k^C

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

THEOREM 4. *There is a recursive set C such that*

(1) *There is an infinite language $L_1^C \in NP^C$, none of whose infinite subsets are in EXP_k^C.*

(2) *There is an infinite $L_2^C \in EXP_k^C$, none of whose infinite subsets are in NP^C.*

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

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

$$L_2^C = \{O^n \mid 1^{2^{n^k}} \in C \text{ and } n \text{ is not divisible by } k+1\}.$$

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

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

R_i: E_i^C infinite $\Rightarrow E_i^C \not\subseteq L_1$

T_i: NP_i^C infinite $\Rightarrow NP_i^C \not\subseteq L_2$.

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_s denote the set of strings put into C through stage s of the construction. We now describe the construction.

CONSTRUCTION.

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

Stage s: There are two cases.

Case 1. s is even. We consider oracle machines $NP_i^{C_{s-1}}$, where $i \leqslant s/4$ and $i \notin G$. Let $n_s > n_{s-1}$ be least such that

(1) $\forall i \leqslant s/4, p_i(n_s) < 2^{n_s}$

(2) s is not divisible by $k+1$

(3) every string in C_{s-1} has length less than n_s

(4) $1^{2^{n_s^k}}$ is not restrained from C.

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

Case 2. s is odd. We consider oracle machines $E_i^{C_{s-1}}$, where $i < s/4$ and $i \notin H$. Find an integer $n_s > 2^{(n_{s-1})^k}$ such that

(1) $\forall i < s/4, h_i(n_s) < 2^{(n_s^{(k+1)})/2}$

(2) $((s+1)/2)(2^{(n_s^{(k+1)})/2} < 2^{n_s^{(k+1)}}$

(3) No element of length n_s^{k+1} is in C_{s-1} or restrained from C.

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

If no such i_0 exists, put the least string of length n_s^{k+1} which is not restrained from C_{s-1} into C. (Note that such a string of length n_s^{k+1} must exist, since no string of this length is restrained from C prior to stage s, and at stage s at most $((s+1)/2)\, 2^{(n_s^{(k+1)})/2} < 2^{n_s^{(k+1)}}$ strings of length n_s^{k+1} are restrained from C.) This adds O^{n_s} 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 $E_{i_0}^C$ 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 $E_j^C \not\subset L_1$. 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 $s_1 \geqslant s$, the computation $E_{i_0}^{C_{s_1-1}}$ rejects $0^{n_{s_1}}$, and these computations are preserved. This contradicts our assumption that $E_{i_0}^C$ is an infinite subset of L_1. ∎

LEMMA 2. *Each requirement T_i is satisfie*

Proof. Assume not, and let i_0 be the lea
satisfied. Let s be any even stage such that i_0
$NP_{i_0}^{C_{s_1-1}}$ must reject $O^{n_{s_1}}$, since otherwise at st
from C, $NP_{i_0}^{C_{s_1-1}}(O^{n_{s_1}})$ would be preserved
would have $NP_{i_0}^C \not\subset L_2$. Similarly we see t
$NP_{i_0}^{C_{s_1-1}}(O^{n_{s_1}})$ is preserved for every $s_1 \geqslant s$ an
subset of L_2 and T_{i_0} is satisfied. ∎

6. FURTHER RESE

The techniques and question explored in t
at in other settings. One might hope to get
results in the strong form they are obtained h
A be constructed such that there is a
(Relativized $\sum_n^{p,A}$ classes are defined in Ba
infinite subset of L is in $\sum_1^{p,A}$? Can L be suc
$\bar{L}$ is in $\sum_1^{p,A}$? Such results would be of inte
NP^A algorithm for recognizing any infinite s

Other results relating NP and EXP_k are
Maass (1983), an oracle B is constructed
infinite set in NP^B has an infinite P^B subs
asked here. For example, can one construct
and every infinite set in EXP_k^B contains an
author has shown that for almost all oracles
parable, with immunity, and has generalize
other deterministic vs nondeterministic questi
(Gasarch, in press).

ACKNOWLEDGME

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

RECEIVED March 31, 1983; ACCEPTED June 28, 1983

REFERENCES

BAKER, T., GILL, J., AND SOLOVAY, R. (1975), Re
SIAM J. Comput. **4** (4), 431–442.

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 $i_0 \leqslant s/4$. Then at any even $s_1 \geqslant s$, $NP_{i_0}^{C_{s_1-1}}$ must reject $O^{n_{s_1}}$, since otherwise at stage s_1, $1^{2^{n^k_{s_1}}}$ would be restrained from C, $NP_{i_0}^{C_{s_1-1}}(O^{n_{s_1}})$ would be preserved (since $n_{s_1+1} > p_{i_0}(n_{s_1})$) and we would have $NP_{i_0}^C \not\subset L_2$. Similarly we see that the rejecting computation $NP_{i_0}^{C_{s_1-1}}(O^{n_{s_1}})$ is preserved for every $s_1 \geqslant s$ and so $NP_{i_0}^C$ 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 $\sum_n^{p,A}$ 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_k 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 $NP^B \subsetneq EXP_k^B$ and every infinite set in EXP_k^B contains an infinite NP^B subset? The first author has shown that for almost all oracles A, EXP_k^A and $\sum_n^{p,A}$ 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

BAKER, T., GILL, J., AND SOLOVAY, R. (1975), Relativizations of the $P = ?NP$ question, *SIAM J. Comput.* **4** (4), 431–442.

BAKER, T., AND SELMAN, A. (1979), A second step toward the polynomial hierarchy, *Theoret. Comput. Sci.* **8** 177–187.

BENNET, C. G., AND GILL, J. (1981), Relative to a random oracle A, $P^A \neq NP^A \neq co-NP^A$ with probability 1, *SIAM J. Comput.* **10** (1), 96–113.

BOOK, R, (1972), On languages accepted in polynomial time, *SIAM J. Comput.* **1** (4), 281–287.

BOOK, R., AND SCHONING, U. (1982), Immunity, unpublished.

DEKHTYAR, M. I. (1977), On the relation of deterministic and nondeterministic complexity classes, Lecture Notes in Computer Science 45, pp. 282–287, Springer-Verlag, New York/Berlin.

HELLER, H. (1980), "Relativized Polynomial Hierarchies Extending Two Levels," Thesis, Technische Universitat, Munich.

HOPCRAFT, J., AND ULLMAN, J. (1979), "Introduction to Automata Theory, Languages and Computation," Addison–Wesley, Reading, Mass.

HOMER, S., AND MAASS, W. (1983), Orace dependent properties of the lattice of *NP* sets, *Theoret. Comput. Sci.* **24** 279–289.

LADNER, R., AND LYNCH, N. (1976), Relativization of questions about log space computability, *Math. Systems Theory* **10** 19–32.

RACKOFF, C. (1982), Relativized questions involving probabilistic algorithms, *J. Assoc. Comput. Mach.* **29** 261–268.

SCHONING, U. (1982), Relativization and infinite subsets of *NP* sets, unpublished.

SOARE, R. I., (in press), Recursively enumerable sets and degrees, "Omega Series in Logic," Springer-Verlag, Berlin/New York.

GASARCH, W. I. (in press), Ph. D. thesis, Harvard University, Cambridge, Mass.

INFORMATION AND CONTROL **58**, 101–112 (1983)

The Complexity of Evaluatin

STAVROS S. COSM

Laboratory for Computer Science, Massach
Cambridge, Massac

A sequence of results which characterize ex
related to the evaluation of relational queries co
joins is proved. It is shown that testing whethe
given relation equals some other given relation
languages that are equal to the intersection of a
co-*NP*—it includes both *NP* and co-*NP*, and w
different context, see Papadimitriou and Yan
Annual ACM Sympos. on the Theory of C
pp. 255–260). It is shown that testing inclusi
respect to a fixed relation (or of relations wit
complete. The complexity of estimating the num
examined.

1. INTRODUCT

The relational algebra is known to be a
expressing database queries. But exactly
expressibility, Codd showed in his classical
to a version of first-order logic. In terms of
other hand, there have been several resul
algebra on finite relations embodies so
power. Already in (Aho *et al.*, 1979; and C
shown that evaluation as well as testing
relational queries are hard combinatorial
were results suggesting that the join of rel
even in certain weak senses of the word (
al., 1981) (a polynomial time algorithm fo
(Honeyman, 1980), and that project-join q
given conjectured result for inclusion (Mai

In some sense, relational algebra seems
way different from, say, the ordinary alg
exponentiation). In ordinary algebra, as i