An. şt. Univ. „Al. I. Cuza" Iaşi
Informatică, 35 (1989)

SET LIMITED RELATIVIZATION

BY

MARIUS ZIMAND

1. Introduction. In striking contrast to the situation in recursive function theory, the relativization principle does not hold in complexity theory. For example, Baker, Gill and Solovay [1] established the existence of two sets A and B such that $P(A)=\mathrm{NP}(A)$, $P(B)\neq \mathrm{NP}(B)$. Such contradictory relativization results are now common place in the study of complexity classes. These results suggests that usual proof techniques which carry over when introducing oracles, like simulation of diagonalization, are of no use proving the separation or the collapsing of the classes admitting conflicting relativization.

A more recent trend in the study of complexity classes is the development of restricted relativizations with the property that any separation of relativized classes implies the separation of the unrelativized classes. To be more specific, we state the notions of positive and negative relativizations. Consider the problem $C=?D$, where C and D are complexity classes. A restriction R, placed on the way the machines specifying the classes C and D access the oracle, yields a positive relativization of the problem $C=?D$ if the following is true : $C=D$ if and only if $C_R(A)=D_R(A)$, for every set A. In a similar way, a restriction R yields a negative relativization of the problem $C=?D$ if $C\neq D$ if and only if $C_R(A)\neq D_R(A)$. Observe that if a positive (negative) relativization exists, then finding a set A such $C_R(A)\neq D_R(A)$ $(C_R(A)=D_R(A))$ implies $C\neq D$, $(C=D)$. If there is one restriction that yields both a positive and a negative relativization of the $C=?D$ problem, then it is said that this problem admits a relativization principle. In this case one can solve the $C=?D$ problem by solving the relativized problem $C_R(A)=?_R D_R(A)$ for some set A. We consider in this work the $P=?\mathrm{NP}$ and the $\mathrm{NP}=?\text{co}-\mathrm{NP}$ problem. Book, Long and Selman [5] have developed the following positive relativizations for these two problems: $P=\mathrm{NP}$ if and only if for all sets A, $P(A)=\mathrm{NP}_B(A)$, and $\mathrm{NP}(A)=\text{co-}\mathrm{NP}(A)$ if and only if for all sets A, $\mathrm{NP}_B(A)=\text{co-}\mathrm{NP}_B(A)$, where $\mathrm{NP}_B(A)$ denotes the class of languages L such that $L\in \mathrm{NOP}(A)$ is witnessed by a machine M, which makes only a polynomially bounded number of queries on all input x. In [4], a more intricate version of the above restriction is shown to yield a relativization principle of the $P=?\mathrm{NP}$ problem. Another kind of restrictions which yield positive relativizations for various problems are considered in [6]. They are called there qualitative restrictions because they restrict the shape of the computation tree. For example, a nondeterministic oracle machine is confluent if for each computational configuration I all the computation paths branching out from I either lead to the same query configuration or do not lead to queries at all. It is shown in [6] that the existence of an oracle A such that $NPC(A) \neq NP_B(A)$ implies $P \neq NP$ and $NP \neq co\text{-}NP$, where $NPC(A)$ denotes the class of languages accepted by polynomial time confluent machines. For more results on positive and negative relativizations, the reader should consult the excelent survey of Book [2].

We propose here a new kind of restriction which yields what we call set limited relativization. An oracle Turing machine has access to the oracle B restricted by the set A if the queries "$y \in$ oracle set B" are answered by YES if $y \in A \cap B$ and by NO if $y \notin A \cap B$. Thus, M has the capacity of answering to questions "$y \in B$" only if $y \in A$. We say that a set A realizes a positive relativization of the question $C=?D$ if $C=D$ if and only if for all sets B, $C(A \cap B) = D(A \cap B)$. In a similar way we define the fact that A realizes a negative relativization of the $C=?D$ problem. Our goal is to characterize the class of sets A which realize positive relativizations of the $P=?NP$ and of the $NP=?co\text{-}NP$ problems. If $P \neq NP$ ($NP \neq co\text{-}NP$) then it is easy to see that any set A realizes a positive relativization of the $P=?NP$ ($NP=?co\text{-}NP$) problem. If $P=NP$ ($NP=co\text{-}NP$) then a set A realizes a positive relativization of the $P=?NP$($NP=?co\text{-}NP$) problem if and only if every subset of A is self-P-printable (self-NP-printable). We also construct two r.e. sets C and D which realize the following weaker form of negative relativizations of the $P=?NP$ and $NP=?co\text{-}NP$ problems: $P=NP$ ($NP=co\text{-}NP$) if and only if for all r.e. sets A, $P(C \cap A) = NP(C \cap A)$ ($NP(D \cap A) = co\text{-}NP(D \cap A)$).

The rest of this section is devoted to fixing the notation and terminology for the main notions of interest for this work. Sets are generally assumed to be over the alphabet $\Sigma = \{0, 1\}$. If $A \subseteq \Sigma^*$, then the cardinality of A is denoted by $\|A\|$. For a string $x \in \Sigma^*$, $|x|$ denotes the length of x. For any set $A \subseteq \Sigma^*$ and $n \in N$ (N is the set of natural numbers), let $A^n = \{x | x \in A$ and $|x| \leqslant n\}$. A set $S \subseteq \Sigma^*$ is sparse if there is a polynomial p such that $\|S^n\| \leqslant p(n)$ for all $n \in N$. The class NP consists of the languages accepted by nondeterministic polynomial time Turing machines and the class P consists of the languages accepted by deterministic polynomial time Turing machines. A set A is in co-NP if its complementary $\bar{A}$ is in NP. Oracle machines are defined in the usual way [8]. For any oracle machine M and set $A \subseteq \Sigma^*$, the notation M^A denotes that M is using A as its oracle set, and $L(M^A)$ denotes the language accepted by M relative to A. For every set A, $NP(A) = \{L(M^A) \mid M$ is a nondeterministic polynomial-time oracle machine$\}$ and $P(A) = \{L(M^A) \mid M$ is a deterministic polynomial time oracle machine$\}$. The class co-NP(A) consists of the sets having their complementary in NP(A).

A set A is self-P-printable (self-NP-printable) if there is a single-valued function f computable in deterministic (nondeterministic) polynomial time relative to the oracle A, such that on all inputs $n \in N$, $f^A(0^n) = x_1\$ x_2\$ \ldots x_k$, where $A^n = \{x_1, x_2, \ldots, x_k\}$. A set A belongs to K[log, poly] (we say that A has small Kolmogorov complexity) if there exist an universal Turing machine U and constants c and k such that for every $x \in A$ there exists a string y, where y has length at most $c.\log|x|$ and U generates x on input y in at most $|x|^k$ steps. The last notions defined above relate

less to the computational complexity of sets but rather to the complexity of their internal organization (for details see [3] and [9]).

For the notions regarding the theory of recursive function theory (like the priority method etc.) see for example [8].

2. Results. Our first goal is to characterize the sets realizing positive relativizations of the $P=?NP$ and $NP=?co\text{-}NP$ questions. We need the following definition.

Definition. *a) A set $A \subseteq \Sigma^*$ is fully self-P-printable if every subset B of A is self-P-printable.*

b) A set $A \subseteq \Sigma^*$ is fully self-NP-printable if every subset B of A is self-NP-printable.

The next proposition settles the existence of fully self-P-printable sets (and implicitely of fully self-NP-printable sets) by showing that this class coincides with K[log, poly].

Proposition 2.1. *A set A is fully self-P-printable if and only if $A \in K$[log, poly].*

Proof. If $A \in K$[log, poly] then any subset B of A also belongs to K[log, poly]. Hence B is self-P-printable [3]. Now suppose A is fully self-P-printable and $A \notin K$[log, poly]. Let $\bar{K}$ be the complementary of K[log, poly] and consider B a subset of A, $B \in \bar{K}$ and card $(B^{\leqslant n}) \leqslant \log n$ for all n. Since A is fully self-P-printable, it follws that B is self-P-printable which is equivalent with $B \in K^B$[log, poly] ([3], K^B[log, poly] is obtained as K[log, poly] only that U is an oracle transducer which use B as the oracle). Consequently for every $x \in B$ there is $y \in \Sigma^*, |y| \leqslant c.\log|x|$ and $U^B(y) = x$ within $|x|^k$ steps. We show that by adding a small number of bits of information to the input y, we can completely avoid the use of the oracle B in the computation $U^B(y) = x$. Let $q_1, q_2, \ldots, q_m$ be the strings queried during this computation and $q_{i_1}, q_{i_2}, \ldots, q_{i_s}$ be the strings from the above list which are in B. It follows that $m \leqslant |x|^k$ (since $U^B(xy) = x$ within $|x|^k$ steps) and $s \leqslant k \log|x|$ (since card $(B^{\leqslant n}) \leqslant \log n$). Now consider the transducer N which on input $z\$j_1\$j_2 \ldots \$j_t$ simulates $U^{(\cdot)}(z)$ and answer the i-th query to the oracle by YES if $i \in \{j_1, j_2, \ldots, j_t\}$ and by NO otherwise. It is clear that $N(y\$i_1\$i_2 \ldots \$i_s) = x$. It follows that $x \in K$[log, poly] which contradicts $B \in \bar{K}$.

We shall also use the following result of Hartmanis and Hemachandra [7].

Lemma 2.2. *a) If $P=NP$ and S is a sparse set, than $P(S)=NP(S)$ if and only if S is self-P-printable.*

b) If $NP=co\text{-}NP$ and S is a sparse set, then $NP(S) = co\text{-}NP(S)$ if and only if S self-NP-printable.

Proposition 2.3. Let A be a non-sparse set. Then there exists a set C such that $P(A \cap C) \neq NP(A \cap C)$.

Proof. It is sufficient to find a subset B of A such that $P(B) \neq NP(B)$. The construction of B is achieved by a slight variation of Baker, Gill and Solovay [1] construction of a set D with the property $P(D) \neq NP(D)$. Consider the set $L(B) = \{x | \exists y, |y| = |x|, y \in B\}$. Clearly $L(B) \in NP(B)$ for all $B \subseteq \Sigma^*$. We construct a subset B of A such that $L(B) \notin P(B)$. Let $\{P_i\}_i$ be an enumeration of polynomial time deterministic Turing machines with oracle. We may assume that P_i runs in time $n^i + i$ for all inputs of length n, for all n.

Construction of B:

Stage 0. $B(0)=\emptyset$, $n_0=0$

Stage $i+1$, $i\geq 0$. Take n sufficiently large such that $n>n_i$ and $\|A^n\|=n^i+i$. If P_i accepts 0^n relative to $B(i)$, then let $B(i+1)=B(i)$ and $n_{i+1}=n$. If P_i does not accept 0^n relative to $B(i)$, then let z be a string of length n in A such that P_{i+1} does not query the oracle about z during this computation. The condition imposed on A assures the existence of the string z. In this case let $B(i)=B(i-1)\cup\{z\}$ and $n_{i+1}=n$.

End of construction.

Let $B=\cup_{i\geq 0}B(i)$. It is clear that $B\subseteq A$ and it follows by standard arguments that $L(B)\notin P(B)$.

Proposition 2.4. *Let A be a non-sparse set. Then there exists a set C such that $NP(A\cap C)\neq$ co-NP$(A\cap C)$.*

Proof. It follows by a similar modification of the corresponding construction of Baker, Gill and Solovay in [1] of a set D such that NP$(D)\neq$ co-NP(D).

Theorem 2.5. a) *If NP$\neq P$ then any set $A\subseteq\Sigma^*$ realizes a positive relativization of the NP $=P$ question.*

b) *If NP $=P$ then set $A\subseteq\Sigma^*$ realizes a positive relativization of the NP $=P$ question if and only if A is fully self-P-printable.*

Proof. If NP$\neq P$, we may take $C=\emptyset$ and then NP$(A\cap C)\neq P(A\cap C)$. The other implication is trivial since the assertion $NP\neq P$ holds true.

Consider now the case NP $=P$. For the implication from right to left, observe that for any set C, $A\cap C\subseteq A$ and thus $A\cap C$ is self-P-printable. Consequently NP$(A\cap C)=P(A\cap C)$ (see [3]). For the other direction take into account Proposition 2.3. It follows that A is sparse and hence, for any $C\subseteq\Sigma^*$, $A\cap C$ is sparse. But it is proven in [7] that if $P=$NP and B is a sparse set then $P(B)=$ NP(B) if and only if B is self-P-printable. It follows that $A\cap C$ is self-P-printable for all sets $C\subseteq\Sigma^*$ which means that A is fully-self-P-printable.

Theorem 2.6. a) *If NP$\neq$co$-$NP, then any set A realizes a positive relativization of the NP$=$co$-$NP question.*

b) *If NP$=$co$-$NP, then a set A realizes a positive relativization of the NP$=$co$-$NP question if and only if A is fully-self-NP-printable.*

Proof. Similar to the above proof, but now we use the fact that if NP$=$co$-$NP and B is any sparse set than NP$(B)=$co$-$NP(B) if and only if B is self-NP-printable ([7]).

We prove next the existence of sets achieving weaker form of negative relativization, in which the oracle sets are selected only from the class of r.e. sets. More specific, we construct two sets C and D such that $P\neq$NP if and only if for every r.e. set A_k, $P(C\cap A_k)\neq$NP$(C\cap A_k)$, and $NP\neq$co-NP if and only if for every r.e set A_k, NP$(C\cap A_k)\neq$co-NP$(C\cap A_k)$. Since, as far as we know, all the oracles constructed in the literature to achieve the separation or the collapsing of various complexity classes are r.e., it is natural to consider the weaker form of negative relativization specified above.

Theorem 2.7. *There exists an infinite r.e. set C such that $P\neq$NP if and only if for every r.e. set A_k, $P(C\cap A_k)\neq$NP$(C\cap A_k)$.*

We first prove the following lemma.

Lemma 2.8. *There exists an infinite r.e. set C such that for every r.e. set A_k either $C\cap A_k$ is finite or NP$(C\cap A_k)\neq P(C\cap A_k)$.*

Proof. 1. We consider an enumeration $(A_k)_{k\in N}$ of the class of r.e sets. Let $A_{k,n}$ denote the strings enumerated in A_k within n steps. Let $(P_i)_{i\in N}$ be an enumeration of polynomial bounded deterministic Turing machines. W.l.o.g assume that $P_i(x)$ works in time bounded by $p_i(x)=|x|^i+i$. Observe that $p_i(n)<p_{i+1}(n)$, $\forall i\geq 0, \forall n$.

2. The construction of C is done in stages, keeping into account the following objectives

I. If A_k is sparse, then we make $C\cap A_k$ finite.

II. If A_k is not sparse, then we make $L(C\cap A_k)\notin((C\cap A_k)$, where $L(C\cap A_k)=\{x\mid\exists y, |y|=|x|, y\in C\cap A_k\}$. Since $L(C\cap A_k)\in$ NP$(C\cap A_k)$, we achieve NP$(C\cap A_k)\neq P(C\cap A_k)$.

We denote $C_n=$ the strings entered in C by the end of stage m. The construction uses the finite injury priority method. The requirements are $R(i,k)$: if A_k is not sparse, $L(C\cap A_k)\neq L(P_i^{C\cap A_k})$, and the order of priorities is $R(j,n)>R(i,k)$ if $\langle j,h\rangle<\langle i,k\rangle$. (We use Cantor bijections $\langle .,..,..,..\rangle: N^m\to N$). During the construction we keep the indicators witness(i,k) which either have the special value "undefined" or witness$(i,k)=z$. The latter variant means that at the stage n when witness(i,k) is set z, $z\in(L(C_n\cap A_{k,n})-L(P_i^{C_n\cap A_{k,n}}))\cup(L(P_i^{C\cap A_{k,n}})-L(C_n\cap A_{k,n}))$. We say that witness $(i,k)=z$ is valid for requirement $R(i,k)$ at stage n. For each requirement $R(i,k)$ we also keep a length restriction $r(i,k)\in N_A$ such that if at stage n we have achieved $L(C_n\cap A_{k,n})\neq L(P_i^{C_n\cap A_{k,n}})$ via witness$(i,k)=z$, then $r(i,k)=p_i(|z|)$. One can observe that $r(i,k)$ is the longest possible length of any queried word used in the computation of $P_i^{C_n\cap A_{k,n}}(z)$. Hence if at later stages no string with length less than $r(i,k)$ enters in C or A_k, then $L(C\cap A_n)=L(P_i^{C\cap A_k})$ via z.

The construction of C.

Stage 0. $C_0=\emptyset$, witness $(i,k)=$ 'undefined'; $r(i,k)=0$, $\forall(i,k)\in N^2$.

Stage $n=\langle i,k,s\rangle$. a) If $i>k$ then go to next stage.

b) If card $(A_{k,n}\cap\Sigma^m)\leq\max(k\cdot p_n(m), m+i)$, $\forall m\leq n$, then go to next stage.

c) If witness$(i,k)=z$ and z is valid for $R(i,k)$ at stage n then go to next stage.

d) If $n\leq\max\{r(j,h)\mid\langle j,h<\langle i,k\rangle\}$ then go to next stage.

e) Let x be the minimal string (in lexicographic order) with the properties: $n\geq|x|>\max\{r(j,h)\mid\langle j,h\rangle<\langle i,k\rangle\}$ and either

i) $P_i^{C_{n-1}\cap A_{k,n}}(x)=$ accept, or

ii) $P_i^{C_{n-1}\cap A_{k,n}}(x)=$ reject and there is y, $|y|=|x|$, $y\in A_{k,n}$, $y\notin A_{p,n}$ for all $p<k$ with the property card $(A_{p,n}\cap\Sigma|x|)\leq p_n(|x|)$, such that $P_i^{C_{n-1}\cup\{y\}\cap A_{k,n}}(x)=$ reject.

We set witness$(i,k)=$, $r(i,k)=|x|^i+i$ and in case i) $C_n=C_{n-1}$ and in case ii) $C_n=C_{n-1}\cup\{y\}$

End of stage n.

Let us observe that if e) is reached, then there exists a x with the required properties and this x is valid for $R(i,k)$ at stage n. Indeed if there is a x, $|x|=m$, such that $P_i^{C_{n-1}\cap A_{k,n}}(x)=$ reject then there are at most m^i+i queried strings. If $A_{k,n}$ satisfies test b), then we can find a string y with the properties required at e. ii), among the strings not queried during the computation of $P_i^{C_{n-1}\cap A_{k,n}}(x)$.

Since, either case i) occurs and then $x \in L(P_i^{C_n \cap A_{k,n}}) - L(C_n \cap A_{k,n})$ or case ii) occurs and then $x \in L(C_n \cap A_{k,n}) - L(P_i^{C_n \cap A_{k,n}})$.

Fact 1. If A_k is sparse, then for every i, $r(i, k)$ stabilizes at some value.

Proof. If A_k is sparse, there is m such that card $(A_k \cap \Sigma^n) \leqslant p_m(n)$, $\forall n$. Hence beginning with stage $m_0 = \max(m, i)$, A_k does not pass the test at b), so $r(i, k)$ does not modify any more.

Fact 2. If A_k is not sparse, then $\forall i \leqslant k$, $r(i, k)$ stabilizes and $R(i, k)$ is eventually satisfied.

Proof. We proceed by induction on $n = \langle i, k\rangle$ with A_k not sparse. Observe that if $i > k$, then $r(i, k) = 0$ at all stages, since point a) is never passed over. Let $n = \langle i, k\rangle$ be the smallest value with A_k not sparse and let $A_{j_1}, A_{j_2}, \ldots, A_{j_s}$ be the sparse sets with $j_i < k$, $\forall i = 1, s$ and $n_0 \in N$ such that card $(A_{j_i} \cap \Sigma^m) \leqslant$ $\leqslant p_{n_0}(m)$, $\forall m$, $i = 1, s$. There is some stage n' such that $r(j, h)$ stabilized at n', for all $\langle j, h\rangle < \langle i, k\rangle$ (if there is such a pair then $j > h$ or A_h is sparse and we use Fact 1). Let m be such that card $(A_k \cap \Sigma^m) \geqslant \max(k, p_{n_0}(m), m^k + k)$ and m is greater than the stable values of (j, h), $\forall \langle j, h\rangle < \langle i, k\rangle$ (there is such a m since A is not sparse). Let n of the form $n = \langle i, k, s\rangle$ be the lowest stage with the properties $n \geqslant m$ and $A_{k,n} \cap \Sigma^m = A_k \cap \Sigma^m$. If at this stage $n = \langle i, k, s\rangle$ witness$(i, k) = z$ and at later stages $n'' \geqslant n$, $n'' = \langle i, k, s''\rangle$, z remains valid for $R(i, k)$ at stage n'', then $r(i, k)$ does not modify itself anymore and $R(i, k)$ is satisfied. If there is a stage $n'' = \langle i, k, s''\rangle$, $n'' \geqslant n$ with witness(i, k) not valid for $R(i, k)$ at n'' or witness(i, k)='undefined', then at this stage n'' witness(i, k) is assigned a value which remains valid at all later stages (since it has a length less than m and $r(j, h)$ are stable for $\langle j, h\rangle < \langle i, k\rangle$). Hence $r(i, k)$ stabilizes (at later stages of the form $\langle i, k, t\rangle$, point c) is not passed over), and $R(i, k)$ is satisfied (see the observation before the construction of C).

The induction step is similar, by considering a step n' at which $r(j, h)$ have stabilized for all $\langle j, h\rangle < \langle i, k\rangle$. The existence of n' comes either from Fact 1, for A_h sparse, from the induction hypothesis, if A_h is not sparse.

Fact 3. If *A_k is sparse then $A_k \cap C$ is a finite set.*

Proof. Let n be such that card $(A_k \cap \Sigma^m) \leqslant p_n(m)$, for all m. At stages $n' \geqslant n$, a word from A_k is introduced in C only for the sake of satisfying a requirement $R(j, h)$ with j and h less than k ($h < k$ from condition c, ii, the only point where C gets new strings, and $j \leqslant h$ from a). Let m be a stage such that $r(j, h)$ have stabilized for all pairs (j, h) with j and h less than k. Hence at no stage superior to $\max(n, m) + 1$, does a string from A_k enter C. So $C \cap A_k$ is finite.

Fact 4. If A_k is not sparse, $L(C \cap A_k) \notin P(C \cap A_k)$.

Proof. Suppose there is some i with $L(C \cap A_k) = L(P_i^{C \cap A_k})$. There is an index k_1 such that $k_1 \geqslant i$ and $A_{k_1} = A_k$. By Fact 2, the requirement $R(i, k_1)$ is satisfied, so $L(C \cap A_{k_1}) \neq L(P_i^{C \cap A_{k_1}})$. But $A_k = A_{k_1}$.

Fact 5. C is an infinite set.

Proof. Let i be such that $L(P_i^A) = \emptyset$, for all A. For all A_k that is not sparse, $k \geqslant i$, the requirement $R(i, k)$ is satisfied (Fact 2). This can be accomplished only by introducing some string in C at a stage of the form (i, k, s). Since the number of sets A_k that are not sparse is infinite, it follows that C is infinite.

Proof of Theorem 2.7. By taking $A_k = \emptyset$, we obtain the implication from right to left. Conversely, let C be the set from Lemma 2.8. Then if $P \neq NP$ and $C \cap A_k$ is finite then clearly $P(C \cap A_k) \neq NP(C \cap A_k)$. If $C \cap A_k$ is infinite then, by Lemma 2.8, $P(C \cap A_k) \neq NP(C \cap A_k)$.

Now we consider the NP =?co-NP problem.

Theorem 2.9. *There exists an infinite r.e. set D such that* NP$\neq$co-NP *if and only if for every r.e. set A_k,* NP$(D \cap A_k) \neq$ co-NP$(D \cap A_k)$.

This theorem is an immediate consequence of the following lemma (in the same way Lemma 2.8 implies Theorem 2.7).

Lemma 2.10. *There exists an infinite r.e. set D such that for every r.e. set A_k either $D \cap A_k$ is finite or* NP $(D \cap A_k) \neq$ co-NP$(D \cap A_k)$.

The proof is similar to the proof of Lemma 2.8.

BIBLIOGRAPHY

1. Baker, T., Gill, J. and Solovay R. — *Relativizations for the P= ? NP question*, SIAM J. Computing 4 (1975), 431—442.
2. Book, R. — *Towards a theory of relativizations: positive relativizations*, STACS-87, LNCS 247, Springer-Verlag, 1987, 1—21.
3. Book, R., Balcazar, J. — *Sets with small generalized Kolmogorov complexity*. Acta Informatica 23 (1986), 679—688.
4. Book, R., Long T. — *Relativizing the P = ? NP problem*, Univ. of California at Santa Barbara, preprint, 1986.
5. Book, R., Long, T. and Selman, A. — *Quantitative relativization of complexity classes*, SIAM J. Comput, 13 (1984), 461—487.
6. Book, R., Long, T. and Selman, A. — *Qualitative relativization of complexity classes* J. Comput. System Sci., 30 (1985), 395—413.
7. Hartmanis, J., Hemachandra, L. — *On sparse oracles separating feasible complexity classes*, Proc. STACS-86, LNCS 210, 321—330.
8. Soare, R. — *Recursively enumerable sets and degrees*, Springer Verlag, 1987.
9. Zimand M. — *On relativizations with restricted number of acceses to the oracle set*, Math. Systems Theory, 20 (1987), 1—11.

Received 15.VI.1989

Universitatea Bucureşti
Centrul de Calcul
Str. Academiei 14,
R-70109, Bucureşti