Stanislav Žák

A Turing machine oracle hierarchy. I.

Commentationes Mathematicae Universitatis Carolinae, Vol. 21 (1980), No. 1, 11--26

Persistent URL: `http://dml.cz/dmlcz/105974`

COMMENTATIONES MATHEMATICAE UNIVERSITATIS CAROLINAE

21, 1 (1980)

A TURING MACHINE ORACLE HIERARCHY I [+)]

Stanislav ŽÁK

Abstract: We introduce three complexity measures of computations on Turing machines with oracles. The complexity of a computation on a Turing machine with an oracle is given either by the number of interactions with the oracle during the computation, or by the sum of lengths of questions asked by the machine of its oracle during the computation, or by the maximum of lengths of these questions.

For oracles of a minimal level, using a principle of diagonalization we construct a complexity hierarchy for the case of the third measure. The case of the first and second measures is postponed in the following paper.

Key words: Diagonalization, Turing machine, oracle, complexity hierarchy.

Classification: 68A20

Introduction. One of the classical problems of the theory of computational complexity is to find the least enlargement of the complexity bound which increases the computing power. This work investigates this problem for the case of three complexity measures of computations on Turing machines (TM's) with oracles defined in Abstract. In our approach the complexity of a computation depends neither on the amount of

---

+) An abridged version of this work can be found in Proceedings of the symposium MFCS'79.

the tape required by the computation nor on the number of its steps.

For oracles of a minimal level which can decide the acceptance of words on Turing machines without oracles, we construct a hierarchy on the set of languages accepted by the nondeterministic Turing machines with an oracle according to the third measure (see Abstract). The hierarchy for the deterministic case is the same.

Now, we give a brief description of our main result. For an oracle A, for the third oracle measure and for t a function on natural number we define: ORACLE(t) is the class of all languages accepted by the nondeterministic Turing machines with oracle A such that if they accept a word of length n then they also accept it by a computation of the complexity not greater than t(n). The result is of the form: If the set of pairs (T,u), where T is a Turing machine without oracle and u is a word accepted by T, is m-reducible ([1]) to A and if t is a recursive function with $\lim t = \infty$, then there is a language L such that $L \subseteq 1^*$, $L \in$ ORACLE(t) and $L \notin \bigcup \{ORACLE(t_1) \mid \liminf (t(n) - t_1(n+1)) \geq 0\}$.

The author does not know any similar results in the literature, and so we compare our result only with trivial propositions which follow from a simple diagonalization.

The paper is a by-product of the work ([3]) on the same problem for the case of space complexity of computations on Turing machines and is based on the same principle of diagonalization. It consists of two chapters. The first chapter is concerned with diagonalization and the second contains all complexity results.

Chapter 1. The aim of this chapter is to introduce a principle of diagonalization. The first part of this chapter contains a basic definition of a mapping called the result of the testing process (rtp) and a theorem which exhibits the logical structure of the diagonalization principle without taking care of existence and complexity aspects. The second part of the chapter contains a lemma which ensures the existence of the rtp-mappings and introduces first complexity aspects. The proofs of the theorem and of two lemmas of the chapter can be found in [3].

Let us first recall some usual definitions and conventions. An alphabet is a nonempty finite set of symbols, all alphabets are subsets of a fixed infinite set containing, among others, the symbols b, 0, 1, 2, S. A string or a word over an alphabet is a finite sequence of its symbols, $|u|$ is the length of the word u. A language over an alphabet is a set of strings over this alphabet. If X is an alphabet then X^* (X^+, X^n) is the language of all words (of positive length, of the length n, respectively) over X. Two words may be concatenated which yields a similar operation for languages. N denotes the set of natural numbers. If a is a symbol and $i \in N$ then a^i is a string of a's of the length i. By a function or by a bound we always mean a mapping of N into itself. The identity function will be denoted id, and the integer part of the binary logarithm will be denoted log. For two functions f, g we shall write $f \leq g$ iff $(\forall n)(f(n) \leq g(n))$ and $f \preccurlyeq g$ iff $(\exists n_0)(\forall n \geq n_0)(f(n) \leq g(n))$. From time to time in the following text we shall use the if .. then .. else construction well-known from the programming languages.

We shall call two languages L_1, L_2 equivalent ($L_1 \sim L_2$) iff they differ only in a finite number of strings. If W is a class of languages then EW will be the class of all languages for which there are equivalent languages in W.

By a program system we mean a pair (P,F) where P is a language and F is a mapping of P into a set of languages over an alphabet. In this context, P is called the set of progams and its elements are called programs. In what follows if we use the phrase "Let P be a set of programs", we implicitly understand that P is the first item of a program system. Its second item will have the general denotation L and L_p will mean the language which corresponds to the program $p \in P$. The set of all such L_p for all $p \in P$ will be denoted $\mathcal{L}(P)$.

For a program p and a word u, we say that p accepts u $(p!u)$ iff $u \in L_p$.

<u>Definition</u>. Let p be a program and Q a set of programs. We say that p diagonalizes Q iff there is a finite set F such that $(\forall q \in Q-F)(p!q \longleftrightarrow \neg\, q!q)$.

<u>Lemma 1</u>. Let p be a program and Q a set of programs. If there are infinitely many programs from Q with the same language as the program p then p does not diagonalize Q.

<u>Definition</u>. Let Q, R be sets of programs, e a function and RTP a mapping of N without some initial segment into the set Q. If for all $q \in Q$ the sets

$$R_q = \{ r \in R \mid RTP(e(|r|)) = q \wedge \neg(q!r \longleftrightarrow \neg\, r!r) \}$$

are infinite then RTP is called the result of the testing process with the function e on the sets Q, R in short, rtp with e on Q, R.

<u>Theorem 1</u>. Let Q, R be sets of programs, RTP an rtp with e on Q, R, X a program and z a mapping from R into N. If for all $q \in Q$ there are infinitely many $r \in R_q$ such that

(1) $X!r1^{z(r)} \longleftrightarrow \neg\, r!r$,

(2) $(\forall j,\ 0 \leq j < z(r))(X!r1^{j} \longleftrightarrow RTP(e(|r|))!r1^{j+1})$,

then $L_X \notin E\mathcal{L}(Q)$.

The next lemma concerns Turing machines and Turing machines with oracles, considered as accepting devices.

We say that a TM T accepts a word u if there is a computation of T on u which stops in a final (accepting) state. If T is a deterministic single-tape TM and accepts a word u, then T(u) denotes the word written on the tape after the computation of T on u has been finished.

A Turing machine with oracle A ($A \subseteq N$) is a Turing machine which has among its tapes a fixed one, on which a special symbol S may be written. The set of states of the machine includes three special states q, YES, NO. If it enters the state q, then in the next step if the number of occurrences of S on its fixed tape belongs to A, it must enter the state YES, otherwise the state NO.

A function e will be called (A-)recursive if there is a deterministic Turing machine T (with oracle A) such that for all $n \in N$ $T(1^n) = 1^{e(n)}$.

A language over an alphabet X is called recursively enumerable (A-recursively enumerable) if it is accepted by a Turing machine (Turing machine with oracle A) and it is called (A-)recursive if moreover its complement in X^* is also (A-) recursively enumerable.

If P is a set of programs then $!_P$ is the binary relation $\{(p,u) | p \in P,\ p!u\}$. - The graph of the binary relation H on a set of strings is the set $\{u2v | (u,v) \in H\}$.

Let A be an oracle. We say that a sequence $\{a_i\}$ of words over an alphabet is (A-)effective iff there is a deterministic Turing machine (with oracle A) rewriting the unary code of any natural number i to the word a_i.

Let L be a language over an alphabet X and T a deterministic Turing machine (with an oracle) which has two final states f_1, f_2. We say that T decides L if for each u, $u \in X^+$, T finishes its computation on the input word u in the state f_1 or f_2 according to whether $u \in L$ or $u \in X^+ - L$, resp.

<u>Lemma 2 (rtp-lemma)</u>. (a) Let Q,R be nonempty sets of programs and e a function. If $\lambda \notin Q$ and no program from Q diagonalizes R and if $e \leq id$ and $\lim e = \infty$ then there is an rtp with e on Q, R.

(b) Let A be an oracle. If, in addition, the sets Q, R, $Q \subseteq \{b,1\}^+$, are (A-)recursively enumerable languages and the graphs of the relations $!_Q$, $!_R$ are (A-)recursive and also the function e is (A-)recursive then there is a deterministic Turing machine T (with oracle A) with one tape and with one head such that

(1) during the computation on the input word 1^k, T uses only the input cells and two adjacent cells,

(2) T writes only the symbols 1, b (1,b,S),

(3) there is a constant c such that the mapping RTP = $= \{(k,T(1^k)) | k \in N,\ k \geq c\}$ is an rtp with e on Q, R.

In fact, we have two lemmas - the version without an oracle and the relativized version.

Sketch of the proof of (a). Since no $q \in Q$ diagonalizes R, R is infinite. Let $\{r_j\}$ be a sequence of all programs from R and $\{q_i\}$ a sequence of all programs from Q with infinitely many occurrences of each of them.

Let us observe the following sequence of pairs:

$(q_1,r_1),\dots$

$\dots$

$(q_{i-1},r_1),(q_{i-1},r_2),(q_{i-1},r_3),\dots,(q_{i-1},r_{j_{i-1}}),$

$(q_i,r_1),(q_i,r_2),(q_i,r_3),\dots,(q_i,r_{j_i}),$

$(q_{i+1},r_1),(q_{i+1},r_2),(q_{i+1},r_3),\dots,(q_{i+1},r_{j_{i+1}}),$

$(q_{i+2},r_1),\dots$

$\dots,$

where for all i, r_{j_i} is the first of those r_j such that $\neg(q_i!r_j \leftrightarrow \neg r_j!r_j) \wedge e(|r_j|) \geq \max\{|q_k| + |r_\ell| \mid$ the pair (q_k,r_ℓ) precedes the pair (q_i,r_1) in our sequence$\}$.

We construct a mapping RTP from N into Q. For $k \in N$, we find the first pair (q_i,r_j) from our sequence such that $|q_i| + |r_j| > k$ and we define $RTP(k) = q_i$.

Let us try to find the value $RTP(e(|r_{j_i}|))$, $i \in N$. We know that for each pair (q_k,r_ℓ) which precedes the pair (q_i,r_1) in our sequence the inequality $|q_k| + |r_\ell| \leq e(|r_{j_i}|) < |q_i| + |r_{j_i}|$ holds. Thus according to the definition of RTP, $RTP(e(|r_{j_i}|)) = q_i$.

Let us define, for all $q \in Q$, $R'_q = \{r_{j_i} \mid q = q_i\}$.

Since $e \leq id$ and $e(|r_{j_{i+1}}|) \geq |q_i| + |r_{j_i}|$ the inequality

$|r_{j_{i+1}}| > |r_{j_i}|$ holds. Therefore the sets R'_q are infinite. Let R_q be the sets from the definition of rtp. We have proved $R'_q \subseteq R_q$. Thus R_q are infinite and the mapping RTP is an rtp with e on Q, R. Q.E.D.

The proof of (b) is based on the same idea, the only change is made that the pairs (q_i, r_j) are embedded into some complicated words.

Chapter 2. In this chapter, by an oracle machine we shall mean a deterministic or nondeterministic single-tape single-head Turing machine with an oracle such that its tape is infinite in both directions and input words are over the alphabet $\{0,1\}$ only.

We shall say that a machine M asks its oracle if M is in the state q. By the length of such a question we shall mean the number of symbols S currently written on the tape.

For an oracle machine M and for an input word u, we define $oracle^1_M(u) = oracle^2_M(u) = oracle^3_M(u) = \infty$ if M does not accept u, and otherwise

$oracle^1_M(u)$ = the minimal number of questions asked by M of its oracle during an accepting computation of M on u,

$oracle^2_M(u) = \min\{s \mid s$ = sum of lengths of all questions asked by M of its oracle during an accepting computation of M on u$\}$,

$oracle^3_M(u) = \min\{s \mid s$ = maximum of lengths of questions asked by M during an accepting computation of M on u$\}$.

In short, the complexity of the acceptance of a word by a machine is given by the complexity of the most modest accepting computation.

In what follows, by a machine we shall mean a nondeterministic machine with a fixed oracle A.

Lemma 3 (universal machine). There is a recursive set S, $S \subseteq 1^{+}\{b,1\}^{*}$, in a one-one correspondence with the set of all machines, and a machine U such that for all i=1 .. 3, for all $s \in S$ and for all input words u the equality $\text{oracle}_{U}^{i}(su) = \text{oracle}_{M_s}^{i}(u)$ holds (where M_s stands for the machine corresponding to s).

Proof. Trivial. S is the usual set of codes of machines.

For the case of deterministic machines there is a deterministic universal machine U_D and a recursive set S_D, $S_D \subseteq S$, in a one-one correspondence with the set of all deterministic machines, such that an analogue of the lemma holds.

Let us fix the set S from Lemma 3. In what follows, the language accepted by the machine M_s will be denoted $L(M_s)$ or L(s). For i=1 .. 3, we shall also write $\text{oracle}_{s}^{i}(u)$ instead of $\text{oracle}_{M_s}^{i}(u)$ where $s \in S$ and $u \in \{0,1\}^{*}$.

Definition (for i=1,2,3). (a) If t is a bound and M_s a machine then by i-t-cut off of the language L(s) we mean the set $L_t^i(s) = L_t^i(M_s) = \{u \mid \text{oracle}_s^i(u) \leq t(|u|)\}$.

(b) We say that a machine M_s accepts its language within i-bound t if $L(s) = L_t^i(s)$.

(c) For a bound t we define

$\text{ORACLE}^i(t) = \{L \mid (\exists s \in S)(L = L(s) = L_t^i(s))\}$,

$\text{CORACLE}^i(t) = \{L_t^i(s) \ s \in S\}$,

$\text{D-ORACLE}^i(t) = \{L \mid (\exists s \in S_D)(L = L(s) = L_t^i(s))\}$,

$\text{D-CORACLE}^i(t) = \{L_t^i(s) \mid s \in S_D\}$.

We can easily see that D-ORACLE^3(t) = ORACLE^3(t), E ORACLE^i(t) = ORACLE^i(t) and for each recursive bound t, CORACLE^i(t) = ORACLE^i(t).

Let us repeat some standard definitions. For $A,B \subseteq N$, we shall write $A \leq_m B$ iff there is a recursive function f such that for all x, $x \in N$, $x \in A \leftrightarrow f(x) \in B$.

K will be a set of natural numbers, $K = \{\langle T,u \rangle \mid T$ is a TM without oracle, $u \in \{0,1\}^*$, T accepts $u\}$ where $\langle\ \rangle$ is a standard coding.

<u>Definition</u> (for i=1,2,3). Let A be an oracle and f, g functions. We say that f is (i,g,A)-recursive iff there is a deterministic machine T with oracle A such that $L(T) = 1^*$, for all $n \in N$ $T(1^n) = 1^{f(n)}$ and $L(T) = L_g^i(T)$. - We say that f is (i,rec,A)-recursive iff it is (i,h,A)-recursive for a recursive function h.

<u>Lemma 4</u> (for i=2,3). If $K \leq_m A$ and if t is an A-recursive bound then the language $L = \{su \mid u \in L_t^i(s),\ s \in S\}$ is A-recursive.

If moreover t is (i,rec,A)-recursive then there is a recursive function f such that the language L can be decided by a Turing machine D for which the equality $L(D) = L_f^i(D)$ holds.

Proof. We have to construct a deterministic Turing machine D with oracle A which decides whether the words from $\{0,1,b\}^*$ belong to L. - Working on an input word, D starts its computation with checking whether the input word is of the form su where $s \in S$, $u \in \{0,1\}^*$. Then D computes the number $t(|u|)$ and D asks of A each question of the length not greater than $t(|u|)$ and lists all answers. Then D converts the code s to the code s' such that (a) $u \in L(M_{s'}) \leftrightarrow u \in$ $\in L_t^i(M_s)$, (b) $M_{s'}$ never asks A (this means that the state q is not accessible in its finite control). The list mentioned above is incorporated in the finite control of such an $M_{s'}$. $M_{s'}$ computes on u in the same manner as $M_{s'}$ does but $M_{s'}$ does not need to ask A, it has its list. After having constructed the code s', D asks A whether $M_{s'}$ accepts u or not, and decides.

The existence of an appropriate recursive function f is clear. Q.E.D.

A similar lemma holds for the deterministic case.

Remark. (a) If $K \leq_m A$ and t is an A-recursive bound then the sets $L_t^i(s)$, i = 2,3, $s \in S$, are A-recursive. This follows from Lemma 4.

(b) Lemma 4 yields also trivial separation result such as for i=2,3 $\mathrm{ORACLE}^i(f) - \mathrm{CORACLE}^i(t) \neq \emptyset$. It seems that f is not a small function (with respect to t).

(c) We also see that $\mathrm{ORACLE}^3(t) \supseteq \mathrm{ORACLE}^2(t)$ and that for t a recursive bound $\mathrm{ORACLE}^1(t+1) \supsetneq \mathrm{ORACLE}^i(t)$ for i=2,3.

For a word u, a language L, a family of languages $\mathcal{L}$, we define Shadow u = $1^{|u|}$, Shadow L = {Shadow u | $u \in L$} and Shadow $\mathcal{L}$ = {Shadow L | $L \in \mathcal{L}$} .

Definition. Let L_1, L_2 be languages. We say that a mapping h, $h: L_1 \to L_2$, is (A-)realizable if there is a deterministic Turing machine T (with oracle A) such that for all $u \in L_1$ the equality $h(u) = T(u)$ holds.

Now, we shall concentrate on the case of oracle^3 measure which is the simplest one.

Theorem 2. Let t be a recursive bound, $\lim t = \infty$. If $K \leq_m A$ then there is a language L such that (1) $L \subseteq 1^+$,

(2) $L \in ORACLE^3(t)$.

(3) $L \notin$ Shadow $ORACLE^3(t')$, where $t'(n+1) = t(n)$ for all $n \in N$.

Proof. First, we shall choose a set Q of programs such that $\mathcal{L}(Q)$ = Shadow $CORACLE^3(t')$ and a set R of programs both satisfying the conditions of the rtp-lemma. Secondly, we shall construct a machine X such that X accepts its language $L(X)$ within 3-oracle bound t and this language has the properties (1),(2) from Theorem 1. By application of this theorem we shall get $L(X) \notin E\,\mathcal{L}(Q) = E$ Shadow $CORACLE^3(t') \supseteq$ $\supseteq$ Shadow $ORACLE^3(t')$.

Let us write $Q = S$ and, for $q \in Q$, L_q = Shadow $L^3_{t'}(q)$. Q is a recursive set and, according to Lemma 4, the graph of the relation $!_Q$ is A-recursive.

Now, we are going to construct the set R. Let $\{s_i\}$ be an effective sequence of programs from S in which each s, $s \in S$, occurs infinitely many times. - There is a realizable mapping h, $h: S \rightarrow S$, such that for all s, $s \in S$, $L(h(s))$ = = Shadow $L^3_{t'}(s)$.

Let f be the function from Lemma 4. We define $r_1 = 1$,

$z(r_i) = \min\{n \mid t(|r_i| + n) \geq f(|h(s_i)r_i|)\}$, $r_{i+1} = 1^{|r_i| + z(r_i) + 1}$.

The sequence r_1, $z(r_1)$, r_2, $z(r_2)$, ..., r_i, $z(r_i)$ can be constructed recursively, therefore the set $R = \{r_i \mid i \in N\}$ is recursive. Let us define, for $i \in N$, $L_{r_i} = L_{s_i} = L(h(s_i)) =$ $=$ Shadow $L^3_{t'}(s_i)$. – We see that the graph of the relation $!_R$ is A-recursive. Moreover, no $q \in Q$ diagonalizes R – see the construction of R, of $\{s_i\}$ and Lemma 1.

Let us define, for $i \in N$,

$$e(|r_i|) = \min (\{|r_i|\} \cup \{t(|r_i|+j) \mid 0 \leq j < z(r_i)\})$$

and $e(n) = n$ for $n \in N$ such that $n \neq |r_i|$ for each $i \in N$. We see that e is recursive, $e \leq id$, $\lim e = \infty$.

We have an rtp RTP with e on Q, R constructive in the sense of the rtp-lemma.

Now, we are ready to construct the machine X. X starts its computation with checking whether the input word is of the form 1^n. Then X computes the number $t(n)$ and will never ask of A a question of the length greater than $t(n)$. We have $L(X) \subseteq 1^+$, and $L(X) \in \mathrm{ORACLE}^3(t)$. Then X recursively constructs the sequence $r_1, z(r_1), r_2, z(r_2), \ldots$.

(1) If the input word 1^n is of the form $r_i 1^{z(r_i)}$ then X accepts iff $\neg\, r_i ! r_i$ iff $r_i \notin L(h(s_i))$ = Shadow $L^3(s_i)$. The possibility to process in this manner is ensured by the definition of the function z and by Lemma 4 – the machine from that lemma decides whether $r_i \notin L(h(s_i))$ without asking of A a question of a length greater than $f(|h(s_i)r_i|) \leq$ $\leq t(|r_i|+z(r_i)) = t(n)$.

If the input word 1^n is of the form $r_i 1^j$, $0 \leq j < z(r_i)$, then X constructs a segment of the tape of the length $e(|r_i|)$ and within this segment X tests. Let $RTP(e(|r_i|)) =$ $= q \in S = Q$ be the resulting program.

The possibility to process in this manner is ensured by the definition of the function e - during the construction of $RTP(e(|r_i|))$ X is never required to ask of A a question of a length greater than t(n); this follows from the construction of e and from the fact that X tests within the length $e(|r_i|)$ - see the condition (1) of the rtp-lemma.

Then X nondeterministically rewrites the input word 1^n to any word from $\{0,1\}^{n+1}$ and on this word computes in the same way as the universal machine U (Lemma 3) according to the program q. If there is an accepting computation of U on a word from $\{0,1\}^{n+1}$ according to the program q and this computation does not require to ask of A a question of a length greater than t(n) then X accepts. Formally:

(2) $1^n \in L(X) \longleftrightarrow (\exists u \in \{0,1\}^{n+1})(\text{oracle}_U^3(RTP(e(|r_i|))u) \leq t(n))$

Now, we must prove that $L(X) \notin E\,\mathcal{L}(Q) =$
$= E$ Shadow $CORACLE^3(t')$.

We shall apply Theorem 1. We have the sets Q, R and the mappings RTP, e, z. We prove that the language L(X) satisfies the conditions of this theorem. The equivalence $r1^{z(r)} \in$ $\in L(X) \longleftrightarrow \neg\, r!r$ (condition (1)) holds for all programs from R except a finite number of them. This is clear from the construction of X - see (1). Now, we are going to demonstrate that for all sufficiently large r, $r \in R$,

$(\forall j,\ 0 \leq j < z(r))\ (r1^j \in L(X) \longleftrightarrow RTP(e(|r|))!r^{j+1})$

(condition (2)) holds. Let us arbitrarily choose a sufficiently large r from R and a number j, $0 \leq j < z(r)$, and let us write $n = |r| + j$. We know that the following statements are equivalent:

(i) $r1^j \in L(X)$,

(ii) $(\exists u \in \{0,1\}^{n+1})$ $(oracle^3_U(RTP(e(|r|))u) \leq t(n))$ - according to the fact that $j < z(r)$ and to the construction of X - see (2),

(iii) $(\exists u \in \{0,1\}^{n+1})$ $(oracle^3_q(u) \leq t(n) = t'(n+1)$ - see Lemma 3, $q = RTP(e(|r|))$,

(iv) $(\exists u \in \{0,1\}^{n+1})$ $(u \in L^3_{t'}(q))$,

(v) $1^{n+1} \in$ Shadow $L^3_{t'}(q) = L_q$,

(vi) $q!1^{n+1}$,

(vii) $RTP(e(|r|))!r1^{j+1}$.

The language L(X) satisfies the conditions of Theorem 1 and therefore $L(X) \notin E\,\mathcal{L}(Q)$ = E Shadow $CORACLE^3(t') \supseteq$ $\supseteq$ Shadow $ORACLE^3(t')$. Q.E.D.

From the fact $D\text{-}ORACLE^3(t) = ORACLE^3(t)$ follows that the same theorem also holds for the deterministic case.

Remark. Condition (3) in Theorem 2 may be changed as follows:

$L \notin$ Shadow $\bigcup\{ORACLE^3(t_1) \mid \liminf(t(n) - t_1(n+1)) \geq 0\}$.

Example. $ORACLE^3(n+1) - ORACLE^3(n) \neq \emptyset$.

R e f e r e n c e s

[1] ROGERS H.Jr.: Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.

[2] SIMON I.: On some subrecursive reducibilities, Tech. Rep. STAN-CS-77-608, June 1977.

[3] ŽÁK S.: A Turing machine space hierarchy, Kybernetika 15,2(1979), 100-121.

Ústav výpočtové techniky ČVUT
Horská 3
12800 Praha 2
Československo

(Oblatum 4.6. 1979)