Stanislav Žák

A Turing machine oracle hierarchy. II.

Commentationes Mathematicae Universitatis Carolinae, Vol. 21 (1980), No. 1, 27--39

Persistent URL: `http://dml.cz/dmlcz/105975`

COMMENTATIONES MATHEMATICAE UNIVERSITATIS CAROLINAE

21, 1 (1980)

A TURING MACHINE ORACLE HIERARCHY II+)

Stanislav ŽÁK

Abstract: We continue the investigation of the complexity measures introduced in the previous paper "A Turing machine oracle hierarchy I". Using the same principle of diagonalization we construct complexity hierarchies on the set of languages accepted by deterministic and nondeterministic Turing machines with oracles.

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

Classification: 68A20

Introduction. This paper is a continuation of [2]. Here, we construct a complexity hierarchy on the set of languages accepted by nondeterministic Turing machines of a special type with an oracle according to the first measure, introduced in [2], two hierarchies are constructed on the set of languages accepted by deterministic Turing machines with an oracle according to two first measures mentioned in Abstract of [2], and the last hierarchy is proved on the set of languages accepted by nondeterministic Turing machines with an oracle with respect to the second measure.

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

The results are 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 $L \subseteq 1^*$, $L \in \mathrm{ORACLE}(t)$ and $L \notin \bigcup \{\mathrm{ORACLE}(t_1) \mid \liminf(t(n) - t_1(n+1) - d(n)) \geq 0\}$, where d is a very small function.
We conclude the paper by a comparison of our results with results which follow from a simple diagonalization.

All preliminaries and definitions which are needed here can be found in [2]. The continuity with [2] is so close that we use a uniform numbering of theorems and lemmas common for both papers.

- - - - - - - - -

Let φ_x be the x-th function in the standard numbering of the partial recursive functions and $!\,\varphi_x(m)$ means that φ_x is defined on the natural number m. We are ready to prove the following lemma.

<u>Lemma 5</u> (for i=1,2). If $K \leq_m A$ then for each k, $k \in N$, $k \geq 1$, there is an (i,d/k,A)-recursive function d such that

(1) d is nondecreasing and unbounded, $d \leq id$,

(2) $\mathrm{Val}\ d =_{df} \{d(n) \mid n \in N\}$ is a recursive set,

(3) for each nondecreasing and unbounded recursive function c the inequality $d \preccurlyeq c$ holds.

Proof. Let us define, for $m \in N$,

$$f(m) = \sum_{i=0}^{m-1} f(i) + \sum \{\varphi_x(m) \mid 0 \leq x \leq m \wedge !\,\varphi_x(m)\} + m.$$

We see that f is an increasing function and that for each recursive function c the inequality $c \preccurlyeq f$ holds. We define for all $n \in N$ $g(n) = \min\{m \mid n \leq f(m)\}$. Since f is increasing, g is a nondecreasing surjection. Now, we are going to prove that for each nondecreasing and unbounded recursive function c the inequality $g \precsim c$ holds.

Suppose $g \not\precsim c$ for such a c. Then there are infinitely many n such that $c(n) < g(n)$. Let φ be a recursive function such that for each $m \in N$ $\varphi(m) = \max\{i \mid c(i) \leq m\}$. Clearly $\varphi(c(n)) \geq n$. Now, we have infinitely many $n \in N$ such that

$$\min\{m \mid n \leq \varphi(m)\} \leq c(n) < g(n) = \min\{m \mid n \leq f(m)\}$$

which yields a contradiction since $\varphi \precsim f$.

It is clear that there is a deterministic machine M with oracle A which constructs g i.e. for all $n \in N$ $M(1^n) = 1^{g(n)}$ and two increasing recursive functions h_1, h_2 such that for all $n \in N$ $h_i g(n) = \text{oracle}^i_M(1^n)$. This is ensured by the fact that for rewriting the word 1^n into $1^{g(n)}$ M needs to compute the numbers $f(0)$, $f(1)$, $f(2), \dots, f(g(n))$ only. Hence the number $\text{oracle}^i_M(1^n)$ depends on the number $g(n)$.

Let us define, for all $n \in N$, $d(n) = k \cdot h_i g(n)$. We see that d is nondecreasing and unbounded since both h_i and g are nondecreasing and unbounded, and that Val d is a recursive set because h_i is increasing and $\text{Val } d = \text{Val } k \cdot h_i g = \text{Val} \cdot k\, h_i$ since g is a surjection. We also see that d is $(i, d/k, A)$-recursive since, for all $n \in N$, the oracle^i complexity of the construction of $1^{d(n)} = 1^{k \cdot h_i g(n)}$ is the same as the complexity of the construction of $1^{g(n)}$ which is equal to $h_i g(n) = (d/k)(n)$.

Now, we must prove that d satisfies the condition (3) of the lemma. Let c be a nondecreasing and unbounded recursive function. Let us define for all $n \in N$,

$$h_i^{-1}(n) = \max\{m \mid h_i(m) < n\}$$

if there is such an m and $h_i^{-1}(n) = 0$ otherwise. We have $h_i h_i^{-1} \leq \mathrm{id}$. Let us write $c_i = h_i^{-1}[c/k]$ where $[\ ]$ denotes the integer part. Such a function c_i is recursive, nondecreasing and unbounded. Therefore $g \preccurlyeq c_i$ and also

$d = k \cdot h_i g \preccurlyeq k \cdot h_i c_i = k\ h_i h_i^{-1}[c/k] \leq k \cdot [c/k] \leq c.$ Q.E.D.

Definition. We say that a machine with an oracle is an r-machine if each its infinite computation contains infinitely many questions to its oracle.

Lemma 6. If $K \leq_m A$ then there is a mapping F, $F: S \to S$, such that:

(a) For each $s \in S$, $M_{F(s)}$ is an r-machine.

(b) If M_s is an r-machine, then for each $u \in \{0,1\}^+$ the equality $\mathrm{oracle}^1_{F(s)}(u) = 1+2\ \mathrm{oracle}^1_s(u)$ holds.

(c) The set $F(S) = \{F(s) \mid s \in S\}$ is recursive.

(d) F is realizable on a TM.

Proof (sketch). $M_{F(s)}$ computes in the same way as M_s except that $M_{F(s)}$ asks of A some special questions. $M_{F(s)}$ puts one of these questions before it starts processing the input word and then again each time immediately after it has asked A when simulating M_s. These questions are of the form: "Is there an infinite continuation of computation of M_s without asking A ?" (Here the König's lemma is implicitly used.) If the answer is yes then $M_{F(s)}$ stops else it continues to simulate M_s. Q.E.D.

Let us fix the mapping F from the lemma. We shall also write $F(M_s)$, $F(M_s) =_{df} M_{F(s)}$.

Lemma 7. Let A be an oracle, $K \leq_m A$. If t is an A-recursive bound then the languages $\{su \mid u \in L_t^1(s),\ s \in F(S)\}$ and $\{su \mid u \in L_t^1(s),\ s \in S_D\}$ are A-recursive.

Proof. We have to construct a deterministic Turing machine R with oracle A which decides whether the words from $\{0,1,b\}^+$ belong to the language $\{su \mid u \in L_t^1(s),\ s \in F(S)\}$ or not. Working on an input word, R starts its computation with checking whether the input word is of the form su, where $s \in F(S)$, $u \in \{0,1\}^+$. Then R computes $t(|u|)$ and constructs the tree of all computations of M_s on u with not more than $t(|u|)$ questions asked of the oracle A (on a branch). Since M_s is an r-machine ($s \in F(S)$), R can construct the tree of these computations in a finite number of steps. If among these computations there is an accepting one then $u \in L_t^1(s)$, else $u \notin L_t^1(s)$.

The proof for the deterministic case is easy. Q.E.D.

Definition. For a bound t we define

$\text{F-ORACLE}^1(t) = \{L \mid (\exists s \in F(S))\ (L = L(s) = L_t^1(s))\}$,

$\text{F-CORACLE}^1(t) = \{L_t^1(s) \mid s \in F(S)\}$.

Lemma 8. Let $\{s_i\}$ be an (A-)effective sequence of programs, where the graph of $!_{\{s_i\}}$ is (A-)recursive. Let e' be a nondecreasing and unbounded (A-)recursive function, $e' \leq id$, such that the set $\mathrm{Val}\ e' = \{e'(n) \mid n \in N\}$ is (A-)recursive. Then there is a set R of programs, a function e, a mapping z and a machine M such that:

(a) $R \subseteq 1^+$, $R = \{r_i \mid i \in N\}$ where for all i, $i \in N$, $L_{r_i} = L_{s_i}$.

(b) e is nondecreasing and unbounded, $e \leq e'$.

(c) $z: R \to N$, $(\forall r \in R)$ $(\forall j,\ 0 \leq j < z(r))$ $(e(|r1^j|) = e(|r|)$, $|r_i 1^{z(r_i)}| < |r_{i+1}|$.

(d) If $n = |r_i| + z(r_i)$ then $\neg(\exists m \in \mathrm{Val}\ e')\ (e(n) \leq m < e'(n))$, if $n = |r_i| + j$, $j < z(r_i)$, then $(\exists m \in \mathrm{Val}\ e')(e(n) \leq m < e'(n))$.

(e) M is a single-tape deterministic machine with two final states f_1, f_2 such that $L(M) = 1^+$ and for all sufficiently large m, $m \in N$, M rewrites the word $1^{e'(m)}$ to the word: if $e(m) < e'(m)$ then $1^{e(m)-1}b1^{e'(m)-e(m)}$, else $1^{e(m)}$, with using only the input cells and two adjacent cells and with using the symbols 1,b(,S) only. If $m = |r_i| + z(r_i)$ for some $i \in N$, then M finishes its computation on $1^{e'(m)}$ in f_1 iff $\neg r_i ! r_i$.

(f) R is an (A-)recursive set, e is an (A-)recursive function.

Proof. We start by the construction of words v_i.
Let $\{m_i\}$ be any sequence of natural numbers. We define
$v_1 = [\S s_1 \S 1^{n_1} \S x_1 \S 1 \S 1^{m_1} \S]$
where [ ] is a binary code of the alphabet $\{1,0,b,\S\}$ in $\{b,1\}$, n_1 is a natural number and if $1^{n_1} \in L_{s_1}$ then $x_1 = 1$, else $x_1 = 0$.
If we have v_i then we define
$v_{i+1} = [\S s_{i+1} \S 1^{n_{i+1}} \S x_{i+1} \S \overline{i+1} \S 1^{m_{i+1}} \S]$
where $\overline{i+1}$ is the binary code of i+1,

(1) $n_{i+1} = \min\{n \mid (\exists m \in \mathrm{Val}\ e')(|v_i| \leq m < e'(n))\}$,
and if $1^{n_{i+1}} \in L_{s_{i+1}}$ then $x_{i+1} = 1$, else $x_{i+1} = 0$.

It is clear that $|v_i| < |v_{i+1}|$ for all $i \in N$.

We define

(2) $R = \{1^{n_i} \mid i \in N\}$ and $L_{1^{n_i}} = L_{s_i}$.

Obviously, we have (a).

Let us define for all m, $m \in N$,

(3) $k_m = \max\{i \mid |v_i| \leq e'(m)\}$ and $e(m) = |v_{k_m}|$.

We can easily see that (b) holds.

We define a mapping z by putting for all i, $i \in N$,

(4) $z(r_i) = \min\{z \mid e'(n_i+z) \geq |v_i|\}$.

It is clear that $|r_i 1^{z(r_i)}| < |r_{i+1}|$ and also $e'(|r_i 1^{z(r_i)}|) <$
$< e'(|r_{i+1}|)$ - cf. (1).

Now, we are going to prove (c). Let us choose $r_i \in R$ and a number j, $0 \leq j < z(r_i)$, arbitrarily. We see that
$|v_i| > e'(|r_i|+j)$ (since $j < z(r_i)$, cf. (4)) $\geq e'(|r_i|) >$
$> e'(|r_{i-1}|+z(r_{i-1})) \geq |v_{i-1}|$. It is clear that $e(|r_i 1^j|) =$
$= e(|r_i|) = |v_{i-1}|$. Obviously, we have (c).

If $n = |r_i| + z(r_i)$ then $v_{k_n} = v_i$ since $|v_i| \leq e'(n_i +$
$+ z(r_i)) < n_{i+1} < |v_{i+1}|$ - see (1) and (4). Further $e'(n) =$
$= e'(n_i + z(r_i))$ is the first $m \in \mathrm{Val}\ e'$ which is not smaller than $|v_i| = |v_{k_n}| = e(n)$.
Therefore $\neg(\exists m \in \mathrm{Val}\ e')\ (e(n) \leq m < e'(n))$.

If $n = |r_i| + j$, $j < z(r_i)$, then $v_{k_n} = v_{i-1}$ since
$|v_{i-1}| \leq e'(n_{i-1} + z(r_{i-1})) < e'(n_i) \leq e'(n_i + j) < |v_i|$ -
the last inequality holds for $j < z(r_i)$. Let us put $m =$
$= e'(n_{i-1} + z(r_{i-1}))$. We see that
$e(n) = |v_{k_n}| = |v_{i-1}| \leq m < e'(n)$ and that $m \in \mathrm{Val}\ e'$.
We have proved (d).

Let us describe the main features of the action of M.

During the computation on the input word 1^a, $a \in N$, M constructs the words v_j, $|v_j| \leq a$, step by step. After construction the elements $[s_{j+1}]$, $[n_{j+1}]$, $[x_{j+1}]$, $[\widetilde{j+1}]$, M chooses m_{j+1} large enough so that all squares used during the construction of these elements or having contained the symbols of the word v_j are now occupied by the symbols of the word v_{j+1}. Let v_{j_a} be the last v_j of length not greater than a. M finishes its computation by writing the word $1^{|v_{j_a}|-1} b 1^{a-|v_{j_a}|}$ if $|v_{j_a}| < a$ or $1^{|v_{j_a}|}$ otherwise, and it finishes in f_1 iff $x_{j_a} = 0$. If $a = e'(m)$ where $m = |r_i| + z(r_i)$, then $v_{j_a} = v_i$ and M finishes in f_1 iff $x_{j_a} = x_i = 0$ iff $\neg\, r_i ! r_i$.

For proving (f) it suffices to fix the sequence of the words v_i from the construction of the machine M. Q.E.D.

<u>Theorem 3</u>. Let t be a recursive bound and d a $(1,d/8,A)$-recursive function from Lemma 5. If $K \leq_m A$ and $d \leq t$ then there is a language L such that (1) $L \subseteq 1^+$,

(2) $L \in$ F-ORACLE$^1(t)$,

(3) $L \notin$ Shadow F-CORACLE$^1(t')$

where $t'(0) = 0$ and $t'(n) = t(n-1) - d(n-1)$ for $n > 0$.

Proof. The idea of the construction of a machine X whose language has the properties stated in the theorem is similar as in the proof of Theorem 2.

Let us put Q = S and, for $q \in Q$, L_q = Shadow $L^1_{t'}(F(q))$. Such a set Q is recursive and the graph of the relation $!_Q$ is A-recursive (Lemmas 6 and 7).

Let $\{s_i\}$ be an effective sequence of programs from S in which each s, $s \in S$, occurs infinitely many times.

Let us put, for all $i \in N$, L_{s_i} = Shadow $L^1_{t'}(F(s_i))$ and $e' = \log^{(4)} \circ d$. We see that the sequence $\{s_i\}$ and the function e' satisfy the conditions of Lemma 8. Therefore there is a set R, a function e, a mapping z and machine M with properties (a) - (f) from this lemma.

It is clear that the set R and the graph of the relation $!_R'$ are A-recursive languages (cf. Lemmas 6,7,8) and that no program from Q diagonalizes R (Lemma 1). We also know that e is nondecreasing, unbounded and A-recursive and that $e \leq id$. Therefore we may apply the rtp-lemma and we are allowed to choose an rtp with e on Q,R which is constructive in the sense of this lemma.

Now, we are ready to construct the machine X and to prove that its language has the properties (1),(2) from Theorem 1.

X starts to process the input word 1^n by constructing the number t(n). During no computation on 1^n X asks A more than t(n) times. We have $L(X) \subseteq 1^+$ and $L(X) \in ORACLE^1(t)$.

Then X constructs the word $1^{e'(n)}$ - this is not of the 1-complexity greater than d(n)/8 - and then X computes in the same way as the machine M from Lemma 8. It constructs the number e(n) - this is not also of the 1-complexity greater than d(n)/8.

(1) If $\neg(\exists m \in Val\ e')(e(n) \leq m < e'(n))$

(2) then X accepts iff M has finished its computation on $1^{e'(n)}$ in the state f_1,

(3) else X computes further as follows: X writes the

program $q = RTP(e(n)) \in Q = S$ (this is of the 1-complexity not greater than $d(n)/8$). Then, after having nondeterministically rewritten the input word to any word from $\{0,1\}^{n+1}$, X computes according to the program $F(q)$ as the universal machine U from Lemma 3. X accepts iff there is an accepting computation of U on some word u from $\{0,1\}^{n+1}$ of the 1-complexity not greater than $t(n) - d(n)$. Formally:

(4) $1^n \in L(X) \leftrightarrow (\exists u \in \{0,1\}^{n+1})(\mathrm{oracle}^1_U(F(q)u) \leq tn -$
$- d(n))$.

We can easily see that X is an r-machine and that $L(X) \in \mathrm{ORACLE}^1(t - 5 \cdot d/8)$.

Now, we want to apply Theorem 1. We have defined the sets Q, R and the mappings RTP, e, z.

First, we shall verify that $r1^{z(r)} \in L(X) \leftrightarrow \neg\, r!r$ holds for all sufficiently large $r \in R$. Let us choose $r_i \in R$ arbitrarily and put $n = |r_i| + z(r_i)$. During the computation on the input word 1^n X finds that $\neg(\exists m \in \mathrm{Val}\ e')(e(n) \leq m < e'(n))$ according to Lemma 8 d. Therefore X accepts iff M has finished its computation on $1^{e'(n)}$ in the state f_1 - see (2). Thus according to Lemma 8 e X accepts $r_i 1^{z(r_i)}$ iff $\neg\, r_i\ r_i$.

Secondly, we shall prove that for all sufficiently large $r \in R$ and for all numbers j, $0 \leq j < z(r)$, the condition $r1^j \in L(X) \leftrightarrow RTP(e(\ r\ ))!r1^{j+1}$ holds. Let us arbitrarily choose a program $r_i \in R$ and a natural number j, $0 \leq j < z(r_i)$, and put $n = |r_i| + j$. During the computation on the input word 1^n, X finds that $(\exists m \in \mathrm{Val}\ e')(e(n) \leq m < e'(n))$ according to Lemma 8 d. Therefore X computes according to (3). Obviously, the following statements are equivalent.

(i) $r_i 1^j \in L(X)$,

(ii) $(\exists u \in \{0,1\}^{n+1})\ (\mathrm{oracle}_U^1(F(q)u) \leq t(n) - d(n))$,

according to (4),

(iii) $(\exists u \in \{0,1\}^{n+1})\ (\mathrm{oracle}_{F(q)}^1(u) \leq t(n) - d(n) =$
$= t'(n+1))$ - see Lemma 3,

(iv) $(\exists u \in \{0,1\}^{n+1})\ (u \in L_{t'}^1(F(q)))$,

(v) $1^{n+1} \in$ Shadow $L_{t'}^1(F(q)) = L_q = L_{RTP(e(n))} =$
$= L_{RTP(e(|r_i|))}$,

(vi) $RTP(e(|r_i|))!r_i 1^{j+1}$.

The language L(X) satisfies the conditions (1),(2) of Theorem 1.

Hence

(5) $L(X) \notin E\,\mathcal{L}(Q) = E$ Shadow F-CORACLE$^1(t')$.

We have constructed the machine X such that its language L(X) does not belong to the set E Shadow F-CORACLE$^1(t')$. For proving that L(X) belongs to F-ORACLE^1 (t) we construct a new machine M. M works on the input word 1^n as the machine X until before the moment when X computes as the universal machine U on the words from $\{0,1\}^{n+1}$ according to the code F(q) where q is the result of the testing process.
After having nondeterministically written any word from $\{0,1\}^{n+1}$, M asks a trivial (formal) question and then M also works as the machine U but according to the code q. M accepts iff there is an accepting computation of the machine M_q of 1-complexity not greater than $(t(n) - d(n) - 1)/2$. We see that each computation of M on 1^n is of 1-complexity not greater than $[d(n)/2 + (t(n) - d(n) - 1)/2]$. Therefore each computation of the machine F(M) on the same word 1^n is of 1-com-

plexity not greater than $1 + 2\cdot[\ldots] = t(n)$ - see the construction of the mapping F in the proof of Lemma 6. Hence $L(F(M)) \in$ F-ORACLE^1 (t).

The fact $L(X) = L(F(M))$ can be easily seen by taking into account the construction of F. The result of the application of the operator F on the tree of all computations of the machine M is the same as the result of the application of F only on the subtrees of all computations of the machine M_q on the words from $\{0,1\}^{n+1}$.

We have $L(X) = L(F(M)) \in$ F-CORACLE^1(t). Q.E.D.

Theorem 4. Let A be an oracle, $K \not\leq_m A$, and t a recursive bound. The following sets contain languages over the alphabet $\{1\}$:

(1) ORACLE^2(t) - Shadow CORACLE^2(t′),

(2) D-ORACLE^1(t) - D-CORACLE^1(t′),

(3) D-ORACLE^2(t) - D-CORACLE^2(t′),

where $t'(n+1) = t(n) - d(n)$ for all $n \in N$, and d is a (1,d/8,A)- or (2,d/8,A)-recursive function from Lemma 5, providing $d \not\leq t$.

The proof is similar as in the previous case. It suffices to delete all references to operator F in the previous proof, to replace the words and symbols "oracle^1", "ORACLE^1", "L^1_t " etc. by the words and symbols "oracle^2", "ORACLE^2", "$L^2_{t'}$" etc., respectively, for case (1), and to omit all the text after (5), Now, instead of "F-ORACLE^1(t)", "$L^1_{t'}(f(q))$" and so on we write "ORACLE^2(t)", "$L^2_{t'}(q)$" and so on.

For case (2) and (3), instead of "S" and "U" we write "S_D" and "U_D", respectively. After having tested, the new machine X deterministically rewrites the word 1^n to the word 1^{n+1} and computes in the same way as the machine U_D. X is deterministic.

Remark. By application of Theorem 4, we can easily prove that also the set

$ORACLE^2(t)$ - Shadow $\bigcup\{ORACLE^2(t_1) \mid t(n) - t_1(n+1) \not\geq d(n)\}$

contains a language over $\{1\}$. A similar corollary can be proved for the case of oracle^1 measure and the classes F-ORACLE^1(t) and also for the deterministic cases (without "Shadow") for i=1,2.

Example. Languages over the alphabet $\{1\}$ are also contained in $ORACLE^2(n+\log^{(k)}n)$ - Shadow $ORACLE^2(n)$ for $k>0$, and $D\text{-}ORACLE^i(n+\log^{(k)}n) - D\text{-}ORACLE^i(n)$, for $i=1,2$, $k>0$.

A trivial diagonalization yields results such as $D\text{-}ORACLE^1(2+2\,n) - D\text{-}ORACLE^1(n) \neq \emptyset$. Remark (b) after Lemma 4 gives trivial results for i=2.

References

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

[2] ŽÁK S.: A Turing machine oracle hierarchy I, Comment. Math. Univ. Carolinae 21(1980), 11-26.

Ústav výpočtové techniky ČVUT

Horská 3

12800 Praha 2

Československo

(Oblatum 4.6. 1979)