ELSEVIER

Information Processing Letters 61 (1997) 189–194

Information Processing Letters

Revisiting a result of Ko

Patrizio Cintioli [a], Riccardo Silvestri [b,*]

[a] Dipartimento di Matematica e Fisica, Università di Camerino, Via Madonna delle Carceri, I-62032 Camerino (MC), Italy
[b] Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", Via Salaria 113, I-00198 Roma, Italy

Received 15 June 1996; revised 1 November 1996
Communicated by L.A. Hemaspaandra

Abstract

In this paper we analyze Ko's Theorem 3.4 in [9]. We extend point (b) of Ko's Theorem by showing that $P_{1\text{-help}}(UP \cap co\text{-}UP) = UP \cap co\text{-}UP$. As a corollary, we get the equality $P_{help}(UP \cap co\text{-}UP) = P_{1\text{-help}}(UP \cap co\text{-}UP)$, which is, to our knowledge, a unique result of type $P_{1\text{-help}}(\mathscr{C}) = P_{help}(\mathscr{C})$, for a class $\mathscr{C}$ that would not be equal to P. With regard to point (a) of Ko's Theorem, we observe that it also holds for the classes UP_k and for FewP. In spite of this, we prove that point (b) of Theorem 3.4 fails for such classes in a relativized world. This is obtained by showing the relativized separation of $UP_2 \cap co\text{-}UP_2$ from $P_{1\text{-help}}(NP \cap co\text{-}NP)$. Finally, we suggest a natural line of research arising from these facts.

Keywords: Theory of computation; Computational complexity; Relativizations; Robust algorithms

1. Introduction

Schöning [10] proposed a notion of an oracle set helping the computation of a language. He introduced the basic concept of a *robust machine*, that is, a deterministic oracle Turing machine that always recognizes the same language, independent of the oracle that is used. The oracle is only for possibly speeding up the computation. Precisely, a language L is said to be recognized in polynomial time with the *help* of an oracle H if there is a robust machine M recognizing L such that M with oracle H runs in polynomial time. The first basic result obtained by Schöning states that the class P_{help} of languages recognized in polynomial time with the help of some oracle is equal to $NP \cap co\text{-}NP$. Thus, the nontrivial languages which can be helped belong to the quite narrow and little known domain $NP \cap co\text{-}NP - P$. This fact led Ko [9] to introduce a notion of partial helping, called *one-sided helping*, in which the oracle is requested to speed up the computation only when the input belongs to the language. Ko proved that the class $P_{1\text{-help}}$ of languages recognized in polynomial time with the one-sided help of some oracle is equal to NP (see [7] for a recent survey). In this work we consider Theorem 3.4 in [9] and point out some questions about it.

Theorem 1 (Ko [9], Theorem 3.4).
(a) $UP \subseteq P_{1\text{-help}}(UP)$.
(b) $UP \cap co\text{-}UP = P_{help}(UP \cap co\text{-}UP)$.

Here, for a given class $\mathscr{C}$ of oracle languages, $P_{help}(\mathscr{C})$ ($P_{1\text{-help}}(\mathscr{C})$) denotes the class of languages recognized in polynomial time with the help (one-sided help) of some oracle in $\mathscr{C}$. We extend point (b)

* Corresponding author. Email: silvestri@dsi.uniroma1.it.

PII S0020-0190(97)00197-4

of the theorem by proving that it holds for one-sided helping, that is $P_{1\text{-help}}(UP \cap \text{co-}UP) = UP \cap \text{co-}UP$ giving as a corollary $P_{1\text{-help}}(UP \cap \text{co-}UP) = P_{\text{help}}(UP \cap \text{co-}UP)$, or equivalently, $P_{1\text{-help}}(UP \cap \text{co-}UP) = \text{co-}P_{1\text{-help}}(UP \cap \text{co-}UP)$. This is to our knowledge the only case in which $P_{1\text{-help}}(\mathscr{C}) = P_{\text{help}}(\mathscr{C})$, for a class $\mathscr{C}$ that would not be equal to P. For point (a) of Theorem 3.4 [9], we observe that the original proof also works for the classes UP_k and FewP, that is $UP_k \subseteq P_{1\text{-help}}(UP_k)$ and $\text{FewP} \subseteq P_{1\text{-help}}(\text{FewP})$. These give the inclusions $UP_k \cap \text{co-}UP_k \subseteq P_{\text{help}}(UP_k)$ and $\text{FewP} \cap \text{co-FewP} \subseteq P_{\text{help}}(\text{FewP})$. The question of validity of the inverse inclusion of point (a) of Theorem 3.4 is open, as any of those listed above. However it has been proved that there exist oracles for which these inclusions are proper [5,6]. The existence of an oracle for which UP is properly included in $P_{1\text{-help}}(UP)$ has also been proved in [4]. We prove here the existence of an oracle for which $UP_2 \cap \text{co-}UP_2$ is not contained in $P_{1\text{-help}}(NP \cap \text{co-}NP)$, which implies the relativized separations of $UP_k \cap \text{co-}UP_k$ from $P_{\text{help}}(UP_k \cap \text{co-}UP_k)$, for every $k \geqslant 2$, and of $\text{FewP} \cap \text{co-FewP}$ from $P_{\text{help}}(\text{FewP} \cap \text{co-FewP})$. These facts show that point (b) of Theorem 3.4 in [9] fails for the classes UP_k and FewP, at least in a relativized world. Finally, we suggest a natural line of research arising from our results.

2. Notations and preliminaries

Let $\Sigma = \{0, 1\}$ be the binary alphabet. For any word $x \in \Sigma^*$, let $|x|$ be the length of x. For any n, let Σ^n be the set of all the words of length n over Σ. We denote the usual *pairing* functions by

$\langle \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \to \Sigma^*$ and

$\langle \cdot, \cdot, \cdot \rangle : \Sigma^* \times \Sigma^* \times \Sigma^* \to \Sigma^*$.

For any class of languages $\mathscr{C}$, let $\text{co-}\mathscr{C} = \{L \mid \overline{L} \in \mathscr{C}\}$, where $\overline{L}$ denotes the complement of L. For any integer $k \geqslant 1$, UP_k is the class of languages accepted by nondeterministic polynomial-time Turing machines which, for every input, have at most k accepting paths [3]. In particular, $UP = UP_1$. FewP is the class of languages accepted by nondeterministic polynomial-time Turing machines such that for some fixed polynomial q and for every input x the machine has at most $q(|x|)$ accepting paths [1] (for more on these classes see [8]).

If M is a Turing machine, $L(M)$ is the language accepted by M. A *robust* machine is a deterministic oracle Turing machine M such that for every oracle A, $L(M^A) = L(M^{\emptyset})$. We say that an oracle A *helps* a robust machine M if M^A runs in polynomial time. We denote by $P_{\text{help}}(A)$ the class of languages accepted by robust machines helped by oracle A, and we let $P_{\text{help}}(\mathscr{C}) := \bigcup_{A \in \mathscr{C}} P_{\text{help}}(A)$, for a class $\mathscr{C}$ [10]. An oracle A *one-sidedly helps* a robust machine M if there exists a polynomial p such that for every $x \in L(M^{\emptyset})$, $M^A(x)$ halts in $p(|x|)$ steps. We denote by $P_{1\text{-help}}(A)$ the class of languages accepted by robust machines one-sidedly helped by oracle A, and we let $P_{1\text{-help}}(\mathscr{C}) := \bigcup_{A \in \mathscr{C}} P_{1\text{-help}}(A)$, for a class $\mathscr{C}$ [9].

3. UP ∩ co-UP, helping and one-sided helping

First, we list two well-known results, Theorem 2 and Fact 3: Theorem 2 is contained in [9].

Theorem 2 (Ko [9]). *For any complexity class $\mathscr{C}$ closed with respect to the $\leqslant_T^p$-reducibility,*

$P_{1\text{-help}}(\mathscr{C}) \subseteq \mathscr{C}$.

Fact 3. *The class* $UP \cap \text{co-}UP$ *is closed w.r.t.* $\leqslant_T^p$, *or equivalently*, $P^{UP \cap \text{co-}UP} \subseteq UP \cap \text{co-}UP$.

Now we are in position to extend point (b) of Theorem 1 to one-sided helping.

Theorem 4. $P_{1\text{-help}}(UP \cap \text{co-}UP) = UP \cap \text{co-}UP$.

Proof. ($\subseteq$) It follows from the fact that $UP \cap \text{co-}UP$ is closed w.r.t. $\leqslant_T^p$, as stated in Fact 3, and from the fact that any class $\mathscr{C}$ closed w.r.t. $\leqslant_T^p$ is such that $P_{1\text{-help}}(\mathscr{C}) \subseteq \mathscr{C}$, as stated in Theorem 2.

($\supseteq$) It follows from $P_{\text{help}}(UP \cap \text{co-}UP) \subseteq P_{1\text{-help}}(UP \cap \text{co-}UP)$ and from Theorem 1(b). □

From the above result and Theorem 1 we immediately obtain the following.

Corollary 5.
(a) $P_{1\text{-help}}(UP \cap \text{co-}UP) = P_{\text{help}}(UP \cap \text{co-}UP)$.
(b) $P_{1\text{-help}}(UP \cap \text{co-}UP) = \text{co-}P_{1\text{-help}}(UP \cap \text{co-}UP)$.

Examining the proof of $\mathrm{UP} \subseteq \mathrm{P}_{1\text{-help}}(\mathrm{UP})$ in [9], we deduce that the same idea works in proving the following theorem that also follows from a stronger result obtained by Yamakami in an unpublished manuscript [11].

Theorem 6.
(a) *For any* $k \geqslant 1$, $\mathrm{UP}_k \subseteq \mathrm{P}_{1\text{-help}}(\mathrm{UP}_k)$
(b) $\mathrm{FewP} \subseteq \mathrm{P}_{1\text{-help}}(\mathrm{FewP})$.

The other directions of all the inclusions relating to Theorems 1 and 6 are open, however it has been proved that there exist oracles for which all of these inclusions are proper [4–6]. Given that $\mathrm{P}_{\mathrm{help}}(\mathscr{C}) = \mathrm{P}_{1\text{-help}}(\mathscr{C}) \cap \text{co-}\mathrm{P}_{1\text{-help}}(\mathscr{C})$ for any class $\mathscr{C}$, Theorem 6 implies that, for every $k \geqslant 1$, $\mathrm{UP}_k \cap \text{co-}\mathrm{UP}_k \subseteq \mathrm{P}_{\mathrm{help}}(\mathrm{UP}_k)$ and $\mathrm{FewP} \cap \text{co-FewP} \subseteq \mathrm{P}_{\mathrm{help}}(\mathrm{FewP})$. In the next section we prove that there exist oracles for which also these inclusions are proper.

4. Relativized separations

In this section we prove the existence of an oracle for which $\mathrm{UP}_2 \cap \text{co-}\mathrm{UP}_2$ is not contained in $\mathrm{P}_{1\text{-help}}(\mathrm{NP} \cap \text{co-NP})$. From this we get the relativized separations of $\mathrm{UP}_k \cap \text{co-}\mathrm{UP}_k$ from $\mathrm{P}_{\mathrm{help}}(\mathrm{UP}_k \cap \text{co-}\mathrm{UP}_k)$, for every $k \geqslant 2$, and of $\mathrm{FewP} \cap \text{co-FewP}$ from $\mathrm{P}_{\mathrm{help}}(\mathrm{FewP} \cap \text{co-FewP})$, that is, point (b) of Theorem 1 fails for the classes UP_k and FewP, at least in a relativized world.

To relativize the notion of a robust machine, we consider deterministic Turing machines that have access to two oracles (for instance, by two separate oracle tapes). For any two oracles X and Y, we denote by $M^{X,Y}$ the machine M having access to the oracles X and Y. We say that M is an *X-robust* machine if for every oracle Y, $L(M^{X,Y}) = L(M^{X,\emptyset})$. An oracle H *one-sidedly helps* an X-robust machine M if there exists a polynomial p such that for every $x \in L(M^{X,\emptyset})$, $M^{X,H}(x)$ halts in $p(|x|)$ steps. Let $\mathrm{P}^X_{1\text{-help}}(\mathscr{C})$ be the class of languages accepted by X-robust machines one-sidedly helped by oracles in the class $\mathscr{C}$.

In order to obtain the above separations it is convenient to characterize the class $\mathrm{P}_{1\text{-help}}(\mathrm{NP} \cap \text{co-NP})$ and its relativizations in terms of simple Turing machines. The characterization is similar to that given in [2] for general helping classes.

Theorem 7. *For every oracle* X, *a language* L *is in* $\mathrm{P}^X_{1\text{-help}}((\mathrm{NP} \cap \text{co-NP})^X)$ *if and only if there exist two polynomial-time deterministic oracle Turing transducers* R_1, R_2 *and a polynomial* p *such that the following holds*:
(a) $x \in L$ *implies*
(i) $(\exists! y_1)(\exists y_2)[|y_1| = |y_2| = p(|x|) \wedge R^X_2(\langle x, y_1, y_2\rangle) = 1]$ [1] (*unambiguity*),
(ii) $(\forall y_1, y_2)[R^X_2(\langle x, y_1, y_2\rangle) \neq 0 \wedge R^X_1(\langle x, y_1\rangle) \neq 0]$ (*correctness*),
(iii) $(\forall y_1, y_2)[R^X_2(\langle x, y_1, y_2\rangle) = 1 \Rightarrow R^X_1(\langle x, y_1\rangle) = 1]$ (*heredity*).
(b) $x \notin L$ *implies*
(i) $(\exists! y_1)(\exists y_2)[|y_1| = |y_2| = p(|x|) \wedge R^X_2(\langle x, y_1, y_2\rangle) = 0]$ (*unambiguity*)
(ii) $(\forall y_1, y_2)[R^X_2(\langle x, y_1, y_2\rangle) \neq 1 \wedge R^X_1(\langle x, y_1\rangle) \neq 1]$ (*correctness*).

Proof. Let $L \in \mathrm{P}^X_{1\text{-help}}((\mathrm{NP} \cap \text{co-NP})^X)$ via an X-robust machine M with one-sided helper $H \in (\mathrm{NP} \cap \text{co-NP})^X$. Let q be a polynomial such that for every $x \in L$, $M^{X,H}(x)$ halts in $q(|x|)$ steps. Without loss of generality, we assume that there is a polynomial t such that M, on every input x, makes only queries of length $t(|x|)$ to the second oracle. Since $H \in (\mathrm{NP} \cap \text{co-NP})^X$, there exists a polynomial-time deterministic oracle Turing transducer S and a polynomial r such that, for every x,

$$x \in H \Leftrightarrow (\exists z)\big[\,|z| = r(|x|) \wedge S^X(x, z) = 1\big]$$

and

$$x \notin H \Leftrightarrow (\exists z)\big[\,|z| = r(|x|) \wedge S^X(x, z) = 0\big].$$

Let p be the polynomial such that $p(n) = q(n) \cdot r(t(n))$ for every n. Define R_2 and R_1 as follows:

$R^X_2(\langle x, y_1, y_2\rangle) :=$ if $|y_1| \neq p(|x|)$ or $|y_2| \neq p(|x|)$, then output "#". Otherwise, let $y_2 = z_1 z_2 \cdots z_{q(|x|)}$ with $|z_1| = \cdots = |z_{q(|x|)}| = r(t(|x|))$ and simulate $M^X(x)$, for at most $q(|x|)$ steps, answering the ith query w_i to the second oracle by the ith symbol of y_1, and check the correctness of the answer by the ith certificate, for H, contained in y_2, that is, check whether or not

[1] The quantifier "$\exists!$" means 'there exists a unique'.

$S^X(w_i, z_i) = i$th symbol of y_1. Let m be the number of queries to the second oracle made during this simulation. If either there is a $j > m$ such that the jth symbol of y_1 is not 0 or some check is not successful, then output "♯". Otherwise, if the simulated computation $M^X(x)$ halts and accepts, then output "1", or else output "0".

$R_1(\langle x, y_1\rangle) :=$ if $|y_1| \neq p(|x|)$ then output "♯". Otherwise, simulate $M^X(x)$, for at most $q(|x|)$ steps, answering the ith query to the second oracle by the ith symbol of y_1. Output "1" if $M^X(x)$ halts and accepts, output "♯" otherwise.

We have to prove that R_1, R_2, and p satisfy the conditions above. Let m be the number of queries that $M^{X,H}(x)$ makes to H within $q(|x|)$ steps and let w_i be the ith of such queries. Let $\overline{y_1}$ be the word such that the ith symbol of $\overline{y_1}$ is equal to $H(w_i)$ if $i \leqslant m$ and it is equal to 0 otherwise. For every $i = 1, \ldots, m$ let z_i be a word such that $|z_i| = r(t(|x|))$ and $S^X(w_i, z_i) = H(w_i)$. Let

$$y_2 = z_1 \cdots z_m 0^{p(|x|) - m \cdot r(t(|x|))}.$$

Now, it is easy to see that $R_2^X(\langle x, \overline{y_1}, y_2\rangle) = L(x)$ and that if for some y' and y'' it holds that $R_2^X(\langle x, y', y''\rangle) = L(x)$ then $y' = \overline{y_1}$. This shows that the two unambiguity conditions are satisfied. The verification of the validity of the other conditions is routine.

Conversely, let L, R_1, R_2, and p satisfy the above conditions. Define

$$H := \{\langle x, u\rangle \mid (\exists v, y)[\,|uv| = |y| = p(|x|) \wedge R_2^X(\langle x, uv, y\rangle) = 1]\}.$$

Since R_2 satisfies the unambiguity conditions, for every x there is a unique word y_x of length $p(|x|)$ for which $(\exists y)[\,|y| = p(|x|) \wedge R_2^X(\langle x, y_x, y\rangle) = L(x)]$. To show that H belongs to $(\mathrm{NP} \cap \text{co-NP})^X$ it suffices to observe that a polynomial-time nondeterministic Turing machine with oracle X, on input $\langle x, u\rangle$, can guess y_x (together with a y such that $R_2^X(\langle x, y_x, y\rangle) \in \{0, 1\}$), successively if $R_2^X(\langle x, y_x, y\rangle) = 1$ and u is a prefix of y_x then outputs "1", otherwise outputs "0". Now, an X-robust machine M that recognizes L can be defined as follows: $M^{X,A}$ on input x does a prefix search, by the second oracle A, to compute a word y of length $p(|x|)$, successively if $R_1^X(\langle x, y\rangle) = 1$ then halts and accepts, otherwise uses R_2 to compute $L(x)$. Since R_1 and R_2 satisfy the correctness and unambiguity conditions, it is easy to see that M is indeed an X-robust machine. When the second oracle of M is H, the word computed by the prefix search is equal to y_x. Moreover, if $x \in L$ then R_1 and R_2 satisfy the heredity condition which means that $R_1^X(\langle x, y_x\rangle) = 1$. It follows that H one-sidedly helps M. □

Now, we are ready to prove the following.

Theorem 8. *There exists an oracle A such that*

$$(\mathrm{UP}_2 \cap \text{co-UP}_2)^A \not\subseteq \mathrm{P}^A_{\text{1-help}}\big((\mathrm{NP} \cap \text{co-NP})^A\big).$$

Proof. For the sake of convenience, we consider oracles as functions from Σ^* to $\{0, 1, \sharp\}$. For any $n \in \mathbb{N}$, $u, v \in \{0, 1\}^*$ with $|u| = |v| = n$, $b \in \{0, 1, \sharp\}$, and for any oracle function $A: \Sigma^* \to \{0, 1, \sharp\}$, we denote by $A_n^b[u]$ the oracle function defined as follows:

$$A_n^b[u](x) := \begin{cases} b & \text{if } x = u, \\ \sharp & \text{if } |x| = n \text{ and } x \neq u, \\ A(x) & \text{otherwise.} \end{cases}$$

Likewise, we denote by $A_n^b[u, v]$ the oracle function defined as follows:

$$A_n^b[u, v](x) := \begin{cases} b & \text{if } x = u \text{ or } x = v, \\ \sharp & \text{if } |x| = n \text{ and } x \neq u, v, \\ A(x) & \text{otherwise.} \end{cases}$$

For any oracle E, word x, and for any oracle Turing machine R, we denote by $Q(R^E(x))$ the set of queries made by the computation $R^E(x)$.

Let $\{(R_1, R_2)_i\}_{i \geqslant 0}$ be a list of all the pairs of polynomial-time deterministic oracle Turing transducers. For any pair $(R_1, R_2)_i$, let p_i be a polynomial such that, for any n, $p_i(n)$ bounds the running time of $R_1^E(\langle x, y\rangle)$ and $R_2^E(\langle x, y, z\rangle)$ for all words x, y, z with $|x| = |y| = |z| = n$ and for all oracles E. Without loss of generality, we assume that $Q(R_1^E(\langle x, y\rangle)) \subseteq Q(R_2^E(\langle x, y, z\rangle))$ for all words x, y, z. For every oracle function E we define the following language: $T(E) := \{0^n \mid \exists y \mid y| = n \wedge E(y) = 1\}$. We will construct an oracle A in such a way that for every positive integer n, $1 \leqslant |\{y|$

$|y| = n \wedge A(y) \neq \sharp\}| \leqslant 2$, and for any y_1, y_2, with $|y_1| = |y_2|$, $A(y_1) = 1 \Rightarrow A(y_2) \neq 0$. This will ensure that $T(A) \in (\mathrm{UP}_2 \cap \text{co-UP}_2)^A$. The construction of the oracle will be done by stages. At any stage k we diagonalize against the pair $(R_1, R_2)_k$ defining a suitable oracle function A_k.

Begin Construction

Stage 0. Let A_0 be the oracle function such that, for every n, $A_0(0^n) = 0$, $A_0(1^n) = 0$, and $A_0(y) = \sharp$ elsewhere. Let $l(0) := 0$.
Stage k. Let $l(k)$ be large enough so that
(1) $p_{k-1}(l(k-1)) < l(k)$,
(2) $2^{l(k)} \cdot p_k(l(k)) < 2^{l(k)}(2^{l(k)} + 1)/2$.

Let R_1, R_2 be the two transducers of the kth pair and let $n = l(k)$. If there exists a word u of length n such that $R_1^{A_n^1[u]}$ and $R_2^{A_n^1[u]}$ do not satisfy conditions (a) and (b) of Theorem 7 w.r.t. 0^n and $T(A_n^1[u])$, then we simply set $A_k := A_n^1[u]$ and go to the next stage.

Otherwise, for any word $u \in \Sigma^n$ we denote by $y_1(u)$ and $y_2(u)$ two words in Σ^n for which $R_2^{A_n^1[u]}(\langle 0^n, y_1(u), y_2(u)\rangle) = 1$. Let $G_n = (V_n, E_n)$ be the directed graph defined as:

$$V_n := \Sigma^n,$$

$$E_n := \Big\{(u, v) \mid v \in Q\Big(R_2^{A_n^1[u]}(\langle 0^n, y_1(u), y_2(u)\rangle)\Big)\Big\}.$$

Every vertex u has at most $p_k(n)$ outgoing edges and $|V_n| = 2^n$. Thus $|E_n| \leqslant 2^n p_k(n)$ which is less than $|V_n|(|V_n| + 1)/2$, that is, the number of all the unordered pairs of vertices of V_n. Hence, there exist two vertices u, v (possibly $u = v$) for which $(u, v) \notin E_n$ and $(v, u) \notin E_n$. Now two cases can occur:

Case 1: $y_1(u) \neq y_1(v)$. In this case the unambiguity of condition (a) is violated, because

$$R_2^{A_n^1[u,v]}(\langle 0^n, y_1(u), y_2(u)\rangle) = 1 \quad \text{and}$$

$$R_2^{A_n^1[u,v]}(\langle 0^n, y_1(v), y_2(v)\rangle) = 1.$$

So we set $A_k := A_n^1[u, v]$.

Case 2: $y_1(u) = y_1(v)$. Observe that

$$v \notin Q\Big(R_1^{A_n^1[u]}(\langle 0^n, y_1(u)\rangle)\Big) \quad \text{and}$$

$$u \notin Q\Big(R_1^{A_n^1[v]}(\langle 0^n, y_1(v)\rangle)\Big).$$

Furthermore, $R_1^{A_n^1[u]}(\langle 0^n, y_1(u)\rangle) = 1$ by the heredity of condition (a). We claim that

$$u \notin Q\Big(R_1^{A_n^1[u]}(\langle 0^n, y_1(u)\rangle)\Big).$$

Suppose the contrary and consider the computation $R_1^{A_n^1[u]}(\langle 0^n, y_1(u)\rangle)$. It makes a first set of queries all different from v, since $(u, v) \notin E_n$, and all receiving answer "$\sharp$", then it queries u. But this first part of the computation must be equal to that of $R_1^{A_n^1[v]}(\langle 0^n, y_1(v)\rangle)$, because $y_1(u) = y_1(v)$, so $R_1^{A_n^1[v]}(\langle 0^n, y_1(v)\rangle)$ queries u, contradicting the fact that $(v, u) \notin E_n$. This implies that $R_1^{A_n^0[u,v]}(\langle 0^n, y_1(u)\rangle) = 1$, which violates the correctness of condition (b) w.r.t. 0^n and $T(A_n^0[u, v])$. So we set $A_k := A_n^0[u, v]$.

End Construction

Set $A := \lim_k A_k$. This limit exists since for any x there is an h such that, for any $k \geqslant h$, $A_k(x) = A_h(x)$. Moreover, it is easy to verify that

$$T(A) \in (\mathrm{UP}_2 \cap \text{co-UP}_2)^A - \mathrm{P}^A_{1\text{-help}}\big((\mathrm{NP} \cap \text{co-NP})^A\big). \quad \square$$

Corollary 9. *There exists an oracle A for which*
(i) $(\mathrm{UP}_k \cap \text{co-UP}_k)^A \not\subseteq \mathrm{P}^A_{1\text{-help}}((\mathrm{UP}_k \cap \text{co-UP}_k)^A)$, *for every* $k \geqslant 2$,
(ii) $(\mathrm{FewP} \cap \text{co-FewP})^A \not\subseteq \mathrm{P}^A_{1\text{-help}}((\mathrm{FewP} \cap \text{co-FewP})^A)$.

Clearly, this implies the analogous separations for two-sided helping.

5. Comments and open questions

We feel that these results, although easy, can stimulate new lines of research, like the search for other classes $\mathscr{C}$ for which $\mathrm{P}_{1\text{-help}}(\mathscr{C}) = \mathrm{P}_{\text{help}}(\mathscr{C})$, or prove that for any other class not equal to $\mathrm{UP} \cap$ co-UP, this is not true, at least in a relativized world.

Since

$$P_{1\text{-help}}(\mathscr{C}_1 \cup \mathscr{C}_2) = P_{1\text{-help}}(\mathscr{C}_1) \cup P_{1\text{-help}}(\mathscr{C}_2),$$

it makes sense to search for the largest class $\mathscr{C}$ for which $P_{1\text{-help}}(\mathscr{C}) = P_{\text{help}}(\mathscr{C})$. At the present, we only know that this class is included between UP $\cap$ co-UP and NP $\cap$ co-NP.

The equality $P_{1\text{-help}}(\mathscr{C}) = P_{\text{help}}(\mathscr{C})$ is equivalent to $P_{1\text{-help}}(\mathscr{C}) = \text{co-}P_{1\text{-help}}(\mathscr{C})$, so

$$P_{1\text{-help}}(\text{UP} \cap \text{co-UP}) = \text{co-}P_{1\text{-help}}(\text{UP} \cap \text{co-UP}).$$

For this result, the property of being closed with respect to the $\leqslant^p_T$-reducibility does not seem to be crucial: for example, it could be the case that

$$P_{1\text{-help}}(\text{NP} \cap \text{co-NP}) \neq \text{co-}P_{1\text{-help}}(\text{NP} \cap \text{co-NP}).$$

It seems natural to search for sufficient conditions on a class $\mathscr{C}$ which make $P_{1\text{-help}}(\mathscr{C}) = \text{co-}P_{1\text{-help}}(\mathscr{C})$ true.

Finally, we propose the following questions: is it true that, for any class $\mathscr{C}$,

$$\mathscr{C} = P_{\text{help}}(\mathscr{C}) \Rightarrow \mathscr{C} = P_{1\text{-help}}(\mathscr{C}) \quad \text{or}$$

$$P_{\text{help}}(\mathscr{C}) = P_{1\text{-help}}(\mathscr{C}) \Rightarrow \mathscr{C} = P_{\text{help}}(\mathscr{C})?$$

Acknowledgments

We thank the editor and two anonymous referees for very useful comments.

References

[1] E. Allender, The complexity of sparse sets in P, in: *Proc. 1st Structure in Complexity Theory*, Lecture Notes in Computer Science, Vol. 223 (Springer, Berlin, 1986) 1–11.

[2] J.L. Balcázar, Self-reducibility, *J. Comput. System Sci.* 41 (1990) 367–388.

[3] R. Beigel, L. Hemachandra and G. Wechsung, On the power of probabilistic polynomial time: $P^{NP[\log]} \subseteq PP$, in: *Proc. 4th IEEE Structure in Complexity Theory Conf.* (1989) 225–227.

[4] J. Cai, L.A. Hemachandra and J. Vyskŏc, Promise problems and access to unambiguous computation, in: K. Ambos-spies, S. Homer and U. Schöning, eds., *Complexity Theory* (Cambridge University Press, Cambridge, 1993) 101–146.

[5] P. Cintioli, Classi helping relativizzate, Tesi di Dottorato, Dipartimento di Matematica, Università di Siena, 1996.

[6] P. Cintioli and R. Silvestri, Helping by unambiguous computations and probabilistic computations, *Math. Systems Theory*, to appear.

[7] L. Hemachandra, Fault-tolerance and complexity, in: *Proc. 20th Internat. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 700 (Springer, Berlin, 1993) 189–202.

[8] D.S. Johnson, A catalog of complexity classes, in: J. van Leeuwen, ed., *Handbook of Theoretical Computer Science*, Vol. A (Elsevier, Amsterdam, 1990) 67–161.

[9] K. Ko, On helping by robust machines, *Theoret. Comput. Sci.* 52 (1987) 15–36.

[10] U. Schöning, Robust algorithms: A different approach to oracles, *Theoret. Comput. Sci.* 40 (1985) 57–66.

[11] T. Yamakami, Polynomial helpers of robust machines, Manuscript, Gunma University, Japan, 1990.