Publ. RIMS, Kyoto Univ.
28 (1992), 603–614

Some Descriptive-Set-Theoretical Problems in Complexity Theory

By

Hisao TANAKA*

Abstract

We bring some descriptive-set-theoretical problems into complexity theory. We here deal with the uniformization problem and the separation problem. It is shown that 1) there exists an oracle A such that for some set $S \in \mathbf{P}[A]$ the uniformizator U, is not in $\mathbf{NP}[A]$, 2) there is an oracle A such that Sep ($\mathbf{NP}[A]$) does not hold and hence so does not Unif ($\mathbf{coNP}[A]$), and 3) there is an oracle A such that Sep($\mathbf{NEXT}$ $[A]$) does not hold and hence so does not Unif($\mathbf{coNEXT}$ $[A]$).

Introduction

The uniformization problem and the separation problem are central subjects in descriptive set theory. We bring these problems into complexity theory. Let $S \subseteq \Sigma^* \times \Sigma^*$ be given, and suppose that a subset U of S satisfies the following condition:

$$\exists y(\langle x, y\rangle \in S) \Rightarrow \exists ! y(\langle x, y\rangle \in U).$$

Then, we say "U uniformizes S", and call U an uniformizator for S. U is the graph of a partial function defined on the domain $\{x : \exists y(\langle x, y\rangle \in S)\}$ of S. The problem is how to get U from S. This means the following: Let S be a set in a complexity class. Then, in what complexity class can we find an uniformizator for S? (In descriptive set theory, one of the most famous results on this subject is the Novikov-Kondo-Addison Uniformization Theorem: Every Π^1_1 set S can be uniformized by a Π^1_1 set. (Here, in a typical case $S \subseteq N^N \times N^N$. See [Sh 67].)) We shall show in this paper that there is an oracle A such that for some $S \in \mathbb{P}[A]$ its typical uniformizator U_s (defined below) is not in $\mathbb{NP}[A]$. Further, it is shown that there is an oracle A such that Unif($\mathbf{coNP}[A]$) does not hold. (The notations used here will be explained in the following sections.)

The separation principle is as follows: Let $\boldsymbol{C}$ be a complexity class. Sep($\boldsymbol{C}$) asserts that for any disjoint X, $Y \in \boldsymbol{C}$ there is a set $Z \in \boldsymbol{C} \cap \mathbf{co}\boldsymbol{C}$ such that $X \subseteq Z$ and $Z \cap Y = \emptyset$. Then, the separation principle for $\boldsymbol{C}$: Sep($\boldsymbol{C}$) holds.

Communicated by S. Takasu, September 2, 1991.
1991 Mathematics Subject Classifications: 03E15, 03D15, 68Q15
* College of Engineering, Hosei University, Koganei, Tokyo 184

(A typical example in descriptive set theory is $\mathrm{Sep}(\Sigma^1_1)$. Of course, in the latter theory, subsets of N^N are considered.) In recursion theory, we have $\neg\mathrm{Sep}(\Sigma^0_1)$ under considering subsets of N ([Kl 52; p.311]). In this paper, we shall show that there is an oracle A such that $\neg\mathrm{Sep}(\mathbf{NP}[A])$. This implies the above mentioned result: $\neg\mathrm{Unif}(\mathbf{coNP}[A])$. The latter holds for the class $\mathbf{coNEXT}[A]$ with some another oracle A.

Also, we shall present some problems concerning our subjects.

§1. Preliminaries

We use the standard notions and notations for complexity theory. Let $\Sigma=\{0, 1\}$ be the alphabet and let Σ^* be the set of all finite strings with the empty string λ. We denote strings by $u, v, w, x, y, \ldots$, and sets of strings by $A, B, \ldots, X, Y, \ldots$ We use the following canonical linear ordering (denoted by $<$) of all strings:

$$\lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots$$

For a string x, $|x|$ denotes the length of x. Then, $|x|\leqq n$ implies $x<0^{n+1}$. π is a pairing function on Σ^* which is one-to-one, onto, and computable in polynomial time:

$$\pi(\pi_0(z), \pi_1(z))=z \quad \text{for any} \quad z\in\Sigma^*,$$

where π_0 and π_1 are the inverse functions of π which are also polynomial time computable. We abbreviate $\langle x, y\rangle$ for $\pi(x, y)$. So, the sets S and U used in the introduction may be considered as subsets of Σ^*.

It is assumed that the reader is familiar with the complexity classes $\mathbf{P}$, $\mathbf{NP}$, $\mathbf{DEXT}$, $\mathbf{NEXT}$, and $\mathbf{PSPACE}$, and their relativized classes such as $\mathbf{P}[A]$, etc. For more information about properties of these classes, see Balcázar-Díaz-Gabarró's textbooks [BDG 88] and [BDG 90]. For a class $\boldsymbol{C}$, let $\mathbf{co}\boldsymbol{C}=\{X: \neg X\in\boldsymbol{C}\}$, where $\neg X=\Sigma^*-X$.

The other notions and notations used in this paper will be explained in the following sections. For more information about descriptive-set-theoretical notions, see [Mo 80].

§2. The Uniformization Problem

We treat with complexity classes of sets of strings such as $\mathbf{P}$, $\mathbf{P}[A]$, etc. Let $\boldsymbol{C}$ and $\boldsymbol{K}$ be such classes.

Definition. $\mathrm{Unif}(\boldsymbol{C}; \boldsymbol{K})$ expresses the statement asserting that every set in $\boldsymbol{C}$ can be uniformized by a set in $\boldsymbol{K}$, namely:

For every $S\in\boldsymbol{C}$ there is a set U in $\boldsymbol{K}$ which satisfies the following condition

(1) $U \subseteq S$, and (2) $\exists y(\langle x, y\rangle \in S) \Rightarrow \exists! y(\langle x, y\rangle \in U)$.

So, we have

(3) $\exists y(\langle x, y\rangle \in S) \Leftrightarrow \exists! y(\langle x, y\rangle \in U) \Leftrightarrow \exists y(\langle x, y\rangle \in U)$.

Further, let $\mathrm{Unif}(\boldsymbol{C})$ mean $\mathrm{Unif}(\boldsymbol{C}; \boldsymbol{C})$. □

We interchangeably use a predicate $S(x, y)$ and its corresponding set $S = \{\langle x, y\rangle : S(x, y)\}$.

Proposition 2.1. *If $\boldsymbol{C}$ is closed under the operations of conjunction, complementation, and the bounded quantifier of the form $\forall z < y$, then* $\mathrm{Unif}(\boldsymbol{C})$ *holds.*

Proof. Let $S \in \boldsymbol{C}$. Define the set U_s as follows:

(4) $U_s(x, y) \Leftrightarrow S(x, y) \wedge \forall z\ (z < y \Rightarrow \neg S(x, z))$.

Clearly, U_s uniformizes S. By the assumption, U_s is in $\boldsymbol{C}$. □

For later usage we call U_s the *bottom curve* of S. Now, let **DEXT** be the class of sets which are accepted by deterministic 2^{lin} time bounded Turing machines, where 2^{lin} means 2^{cn} for some constant numbers c (see [BDG 88]). Then

Corollary 2.2. $\mathrm{Unif}(\mathbf{DEXT})$ *holds.*

Proof. **DEXT** is closed under the three operations stated in Proposition 2.1, since the number $\#\{z : z < y\} \doteq 2^{|y|+1}$. □

Since $z < y$ implies $|z| \leqq |y|$, we have

(4′) $U_s(x, y) \Leftrightarrow S(x, y) \wedge \forall z[|z| \leqq |y| \Rightarrow (z < y \Rightarrow \neg S(x, z))]$.

So, if $\mathbf{NP} = \mathbf{coNP}$ [$\mathbf{P} = \mathbf{NP}$], then $\mathrm{Unif}(\mathbf{NP})$ [resp. $\mathrm{Unif}(\mathbf{P})$] holds. Hence, by [BGS 75] we obtain

Proposition 2.3. *There is a recursive oracle A such that* $\mathrm{Unif}(\mathbf{NP}[A])$, $\mathrm{Unif}(\mathbf{coNP}[A])$, *and* $\mathrm{Unif}(\mathbf{P}[A])$ *hold.* □

A partial function $f : \Sigma^* \to \Sigma^*$ is polynomial time computable if its domain (denoted by $\mathrm{Dom}(f)$) is in **P** and if there is a polynomial time bounded Turing machine M such that $f(x)$ is computed by M for every $x \in \mathrm{Dom}(f)$. An uniformizator of a set S is the graph of a partial function. Then,

Problem 1. *Can every set in* **P** *be uniformized by a polynomial time computable partial function?*

This problem is open for us, yet. However, if we impose restrictions on $\mathbf{P}$, then it has an affirmative answer:

Proposition 2.4. *Let* $\log\mathbf{P} = \{Q : Q(x, y) \Leftrightarrow |y| \leqq c \cdot \log |x| \wedge S(x, y)$ *for some* $c > 0$ *and* $S \in \mathbf{P}\}$. *Then, every* $Q \in \log\mathbf{P}$ *can be uniformized by a polynomial time computable partial function.*

Proof. Since $\#\{y : |y| \leqq c \cdot \log |x|\} \leqq 2|x|^c$, the domain $\{x : \exists y\, Q(x, y)\}$ is in $\mathbf{P}$, and the least y (with respect to $<$) such that $Q(x, y)$ holds can be found in polynomial time of $|x|$. So, the bottom curve U_Q is a polynomial time computable partial function. □

On the contrary above, we have:

Proposition 2.5. *There is a recursive oracle* A *such that some set* S *in* $\mathbb{P}[A]$ *cannot be uniformized by any partial function computable in polynomial time relative to* A.

Proof. Take a recursive A such that $K(A)$ is not in $\mathbb{P}[A]$, where $K(A) = \{\langle i, x, 0^n\rangle$: some computation of NP_i^A accepts x in $\leqq n$ steps$\}$ ([BGS 75]). Here $NP_i^{\sim}$ is the i-th nondeterministic polynomial time bounded oracle Turing machine, where the index i denotes the code of this machine and further it is identified with some natural number. Since $K(A) \in \mathbb{NP}[A]$, there is a polynomial $p(n)$ and a predicate $R \in \mathbb{P}[A]$ such that

$$x \in K(A) \Leftrightarrow \exists y[|y| \leqq p(|x|) \wedge R(x, y)].$$

Take $S(x, y) \Leftrightarrow |y| \leqq p(|x|) \wedge R(x, y)$. Clearly S is in $\mathbb{P}[A]$. Since the domain of S is $K(A)$ and since $K(A)$ is not in $\mathbb{P}[A]$, S cannot be uniformized by any partial function computable in polynomial time relative to A. □

In contrast with this proposition, the following problem seems to be rather difficult to solve:

Problem 2. *Is there any oracle* A *such that* $\mathrm{Unif}(\mathbb{P}[A])$ *does not hold*?

In the proof of Proposition 2.5, it may be possible that the graph of some partial function which uniformizes S is in $\mathbb{P}[A]$. In the next section, we shall show a theorem concerning this problem.

Now, by what set can a set in $\mathbb{P}$ be uniformized? The following gives an answer for it:

Proposition 2.6. $\mathrm{Unif}(\mathbb{P}; \mathbf{coNP})$ *holds.*

Proof. Let $S \in \mathbf{P}$. Then, by (4′) we have $U_s \in \mathbf{coNP}$. □

However it remains open to know whether every set in **P** can be uniformized by a set in **P**:

Problem 1′ (A weak form of **Problem 1**). *Does* Unif(**P**) *hold*?

Now, since **PSPACE** is closed under nondeterministic computation ([Sa 70]), by (4′) we have

Proposition 2.7. Unif(**PSPACE**) *holds.* □

§3. A Partial Answer for Problem 2

There is an oracle A such that for some $S \in \mathbf{NP}[A]$ the bottom curve U_s is not in $\mathbf{NP}[A]$: Take an oracle A such that $\mathbf{NP}[A] - \mathbf{coNP}[A]$ is not empty ([BGS 75]), and let $E \in \mathbf{NP}[A] - \mathbf{coNP}[A]$. Now, define S by

$$S(x, y) \Leftrightarrow (x \in E \wedge y = 0) \vee y = 1.$$

Then, the bottom curve U_s is not in $\mathbf{NP}[A]$. For, suppose $U_s \in \mathbf{NP}[A]$. Since $x \in \neg E \Leftrightarrow \langle x, 1\rangle \in U_s$, E would be in $\mathbf{coNP}[A]$, a contradiction. (This proof is based on a conversation with H. Enderton about a similar subject.) Now, how about an $S \in \mathbf{P}[A]$? We have

Theorem 3.1. *There is a recursive oracle A and a set $S \in \mathbf{P}[A]$ such that the bottom curve U_s of S is not in $\mathbf{NP}[A]$.*

Proof. We modify a proof in [BGS 75]. For any $X \subseteq \Sigma^*$, let

$$L(X) = \{\langle x, y\rangle : \exists z (z < y \wedge \langle x, z\rangle \in X)\}.$$

We will define an oracle A by stages such that

$$L(A) \notin \mathbf{coNP}[A]. \tag{5}$$

Let $A(s)$ be the set of strings put into A before stage s, and let $A(0) = \emptyset$. We also define natural number n_s at each stage s. Let $n_0 = 0$. Further, let $\ell(n) = |\langle 1^n, 0^{n+1}\rangle|$. We may assume that $\ell(n)$ is a linear function. (By taking a special pairing function, we have $\langle 1^n, 0^{n+1}\rangle = 1^{2n+1}$.)

Stage $s \geqq 0$. Let n be the least number n such that $n > n_s$ and $p_s(\ell(n)) < 2^n$, where $p_s(n)$ is the polynomial that is the time bound function of the s-th nondeterministic polynomial time bounded oracle Turing machine $NP_s^{\sim}$. First, let $B(s)$ be the set obtained from $A(s)$ by adding $\langle 1^n, 0^{n+1}\rangle$. Then, run the s-th machine $NP_s^{B(s)}$ on $\langle 1^n, 0^{n+1}\rangle$. If it accepts, then take an accepting computation

and add to $B(s)$ a string of the form $\langle 1^n, u\rangle$ to make $A(s+1)$, where u is the least string of length n such that $\langle 1^n, u\rangle$ is not queried during the computation. (Such a string exists.) Namely, $A(s+1)=B(s)\cup\{\langle 1^n, u\rangle\}$. If it rejects, then nothing to do and let $A(s+1)=B(s)$. Let $n_{s+1}=2^n$. The strings added to A at stage s do not affect any computation performed at any earlier stage $<s$. [For, consider the stage $s-1$, and let $n_s=2^m$. Then $m>n_{s-1}$, and $p_{s-1}(\ell(m))<2^m=n_s<n$. On the other hand, if $|u|=n$, then $|\langle 1^n, 0^{n+1}\rangle|>n$. So, $p_{s-1}(\ell(m))<|\langle 1^n, 0^{n+1}\rangle|$.]

As usual, define A as the union of all $A(s)$'s. Then, clearly A is a recursive set. For any s, we have the following equivalences:

$$\begin{aligned}
&NP_s^A \text{ accepts } \langle 1^n, 0^{n+1}\rangle \Leftrightarrow NP_s^{B(s)} \text{ accepts } \langle 1^n, 0^{n+1}\rangle\\
&\Leftrightarrow \exists u(|u|=n \wedge \langle 1^n, u\rangle \in A)\\
&\Leftrightarrow \exists u(u<0^{n+1} \wedge \langle 1^n, u\rangle \in A) \Leftrightarrow \langle 1^n, 0^{n+1}\rangle \in L(A)\\
&\Leftrightarrow \langle 1^n, 0^{n+1}\rangle \notin \neg L(A).
\end{aligned}$$

So, we have (5): $L(A)\notin \mathbf{coNP}[A]$.

Now consider the bottom curve U_A of A:

$$U_A(x, y)\Leftrightarrow \langle x, y\rangle \in A \wedge \forall z(z<y \Rightarrow \neg(\langle x, z\rangle \in A)).$$

$L(A)$ is the set of all points $\langle x, y\rangle$'s which are above the curve U_A. Let R be the set of all points $\langle x, y\rangle$'s which are below or on the curve U_A. That is,

$$\langle x, y\rangle \in R \Leftrightarrow \exists z[y\leqq z \wedge \langle x, z\rangle \in U_A].$$

However, if $\langle x, z\rangle \in U_A$ then $|z|\leqq |x|+1$. Therefore, we have

$$\langle x, y\rangle \in R \Leftrightarrow \exists z[|z|\leqq |x|+1 \wedge (y\leqq z \wedge \langle x, z\rangle \in U_A)]. \tag{6}$$

Now, suppose that U_A were in $\mathbf{NP}[A]$. Then by (6), R would also be in $\mathbf{NP}[A]$. On the other hand, the domain $D=\{x:\exists y(\langle x, y\rangle \in A)\}$ is in $\mathbf{P}[A]$. (For, given x, first check if x is of the form 1^n. If so, query $\langle 1^n, 0^{n+1}\rangle$ to A. If the answer is yes, then $x\in D$. This process is done in time of polynomial of $|x|$.) So, the cylinder $D\times\Sigma^*$ is also in $\mathbf{P}[A]$. Since $L(A)=D\times\Sigma^*-R$, $L(A)$ would be in $\mathbf{coNP}[A]$. This contradicts (5). Hence, U_A can not be in $\mathbf{NP}[A]$. So, taking $S=A$ our theorem is proved, since always $A\in\mathbf{P}[A]$. □

In this theorem, the uniformizator concerned is a particular one, and S may possibly be uniformized by a set in $\mathbf{P}[A]$. In fact, the $S(=A)$ used in the proof of this theorem can be uniformized by a set in $\mathbf{P}[A]$. Our wish (Problem 2) is to have A and S such that any uniformizator of S is not in $\mathbf{P}[A]$ (or not in $\mathbf{NP}[A]$).

§4. The Separation

The separation principle is closely related to the uniformization problem.

Let $\boldsymbol{C}$ be a complexity class of sets of strings. The Separation Principle for $\boldsymbol{C}$ "Sep($\boldsymbol{C}$) holds" was already defined in the Introduction. It is well-known ([Mo 80]) that under some very mild condition

(7) $$\text{Unif}(\boldsymbol{C}) \text{ implies } \text{Sep}(\mathbf{co}\boldsymbol{C}).$$

So, by showing that $\neg\text{Sep}(\mathbf{co}\boldsymbol{C}[A])$ for some oracle A, we can conclude that there is an oracle A such that $\text{Unif}(\boldsymbol{C}[A])$ does not hold. Now clearly, if $\boldsymbol{C}$ is closed under the operation of complementation, then $\text{Sep}(\boldsymbol{C})$ hold. So, the principle becomes our subject of discussion for such classes that are not closed under complementation.

Now, let $\boldsymbol{C}$ and $\boldsymbol{E}$ be complexity classes such that $\boldsymbol{E} \subset \boldsymbol{C}$, and let B and C be disjoint sets in $\boldsymbol{C}$. B and C are $\boldsymbol{E}$-separable if there is a set $R \in \boldsymbol{E}$ such that $B \subseteq R$ and $R \cap C = \emptyset$. Otherwise, we say that B and C are $\boldsymbol{E}$-*inseparable*. Then, by using a proof-method in [BGS 75], we obtain

Theorem 4.1. *There is a recursive oracle A such that* $\mathbf{NP}[A]$ *has two disjoint sets which are* $\mathbf{P}[A]$*-inseparable.*

Proof. As before, let $A(s)$ be the set of strings put into A before stage s and let $A(0) = \emptyset$. We define a natural number n_s, and let $n_0 = 0$.

Stage $s \geqq 0$. Let m be the least number such that $m > n_s$ and $p_s(m) < 2^{m-1}$, where $p_s(n)$ is the polynomial time bound function of the s-th deterministic polynomial time bounded oracle Turing machine $P_s^{\sim}$. Run $P_s^{A(s)}$ on 0^m. If it rejects [accepts] the input, then add to A the least string of the form $0y$ [resp. $1y$] of length m not queried in the computation in order to make $A(s+1)$. Such a string $0y$ or $1y$ exists. Let $n_{s+1} = 2^{m-1}$.

Now, let A be the union of all sets $A(s)$'s. Clearly, A is a recursive set. The above computation at stage s remains unchanged when oracle A is used instead of $A(s)$. Let $K_0(A) = \{x : \exists y(0y \in A \wedge |0y| = |x|)\}$, and $K_1(A) = \{x : \exists y(1y \in A \wedge |1y| = |x|)\}$. Clearly, $K_0(A)$ and $K_1(A)$ are disjoint and both are in $\mathbf{NP}[A]$. Now, suppose that there were a set $R \in \mathbf{P}[A]$ such that $K_0(A) \subseteq R$ and $R \cap K_1(A) = \emptyset$. Identifying a Turing machine with the set accepted by it, let $R = P_s^A$. Then, at stage s, letting m be the m described during defining $A(s+1)$, we have

$$\begin{aligned} P_s^A \text{ rejects } 0^m &\Rightarrow \exists y(0y \in A \wedge |0y| = m) \Rightarrow 0^m \in K_0(A) \\ &\Rightarrow 0^m \in R \Rightarrow P_s^A \text{ accepts } 0^m, \text{ and} \end{aligned}$$

$$\begin{aligned} P_s^A \text{ accepts } 0^m &\Rightarrow \exists y(1y \in A \wedge |1y| = m) \Rightarrow 0^m \in K_1(A) \\ &\Rightarrow 0^m \in \neg R \Rightarrow P_s^A \text{ rejects } 0^m. \end{aligned}$$

This is a contradiction. So, there is no such R. Hence, $K_0(A)$ and $K_1(A)$ are $\mathbf{P}[A]$-inseparable. □

This theorem corresponds to the classical theorem asserting that there are two recursively inseparable recursively enumerable disjoint sets of natural numbers ([Kl 52, p.311]). And this theorem does not directly imply $\neg\mathrm{Sep}(\mathbf{NP}[A])$, because $K_0(A)$ and $K_1(A)$ may possibly be separated by a set R which is in $\mathbf{NP}[A]\cap\mathbf{coNP}[A]$ but not in $\mathbb{P}[A]$. However, modifying the proof of Theorem 6 in [BGS 75], we can obtain a recursive oracle A such that $\mathbb{P}[A]=\mathbf{NP}[A]\cap\mathbf{coNP}[A]\subset\mathbf{NP}[A]$ (here, $\subset$ denotes the proper inclusion) and such that $K_0(A)$ and $K_1(A)$ are $\mathbb{P}[A]$-inseparable. (In the proof of Theorem 6 in [BGS 75], we let the requirement $\langle i, i\rangle$ be satisfied if the following condition has been assured (after [BGS 75] here we use E instead of A): For some string x, both the computations of $P_i^{E(n)}$ and P_i^E on x are the same, and $[x\notin P_i^E\Rightarrow x\in K_0(E)]$ & $[x\in P_i^E\Rightarrow x\in K_1(E)]$ holds. Here, K_0 and K_1 are ours defined above. At stage n, to satisfy the requirement $\langle i, i\rangle$, run $P_i^{E(n)}$ on input $0^{e(n)}$. If it rejects the input, then, as in our proof above, add to E the least string of the form $0y$ of length $e(n)$ not queried in the computation; otherwise add to E such a string $1y$. The remainder is almost the same as in [BGS 75]. For detailed account, see the proof of Theorem 4.4 below.) So, these two disjoint sets are $(\mathbb{NP}[A]\cap\mathbf{coNP}[A])$-inseparable. Thus we have:

Theorem 4.2. *There is a recursive oracle A such that* $\mathrm{Sep}(\mathbb{NP}[A])$ *does not hold.* □

By combining this with (7), we have the following theorem:

Theorem 4.3. *There is a recursive oracle A such that* $\mathrm{Unif}(\mathbf{coNP}[A])$ *does not hold.* □

Now we shall show

Theorem 4.4. *There is a recursive oracle A such that* $\mathrm{Sep}(\mathbb{NEXT}[A])$ *does not hold.*

Proof. Let $L_0(A)=\{x:\exists y[0y\in A\wedge|0y|=2^{|x|}]\}$ and $L_1(A)=\{x:\exists y[1y\in A\wedge|1y|=2^{|x|}]\}$. We construct the following recursive oracle A:

(8) $\mathbb{DEXT}[A]=\mathbb{NEXT}[A]\cap\mathbf{coNEXT}[A]$, and

(9) $L_0(A)$ and $L_1(A)$ are disjoint and are $\mathbb{DEXT}[A]$-inseparable, and hence $\mathbb{DEXT}[A]\neq\mathbb{NEXT}[A]$.

These implies the negation of $\mathrm{Sep}(\mathbb{NEXT}[A])$. To construct such an A, we use the method developed in the proof of Theorem 6 in [BGS 75]. Take a recursive oracle B such that $\mathbb{P}[B]=\mathbb{NP}[B]$, then $\mathbb{DEXT}[B]=\mathbb{NEXT}[B]$ holds also ([Bo 74]). Without loss of generality we may assume that B does not contain any

string of even length. We define number-theoretic functions $d(n)$ and $e(n)$ as follows:

$$d(0)=2,\ e(0)=2^2,\ d(n+1)=2^{e(n)},\ \text{and}\ e(n+1)=2^{d(n+1)}.$$

Let $DE_i^{\sim}$ [$NE_i^{\sim}$] be the i-th deterministic [resp. nondeterministic] 2^{lin} time bounded oracle Turing machine. Here, the index i denotes the code of the machine and it is identified with a natural number, and further let $2^{c_i n}$ be the i-th machine's time bound function, where c_i is a constant number. Let $A(n)$ be the set of strings added to A before stage n, and let $A(0)=B$. We consider some requirements $R(i, j)$ depending on indicies i and j.

$R(i, i)$ is satisfied at stage n if at stage n there is a string x such that both computations of $DE_i^{A(n)}$ and DE_i^A on x are the same and the condition

$$(x \notin DE_i^A \Rightarrow x \in L_0(A)) \wedge (x \in DE_i^A \Rightarrow x \in L_1(A)) \tag{10}$$

is ensured. $R(j, k)$, where $j \neq k$, is satisfied at stage n if at stage n the condition

$$\exists x\ [\text{neither } NE_j^A \text{ nor } NE_k^A \text{ accepts } x] \tag{11}$$

is ensured. (Then, letting $S=NE_j^A$ and $T=NE_k^A$, $\neg S \neq T$. So, for such $R(j, k)$ we do not have to do anything.)

An unsatisfied requirement $R(i, i)$ is vulnerable at stage n if the following condition holds:

$$2^{c_i d(n)} < 2^{e(n)-1}\ (< d(n+1) < e(n+1)). \tag{12}$$

An unsatisfied requirement $R(j, k)$, where $j \neq k$, is vulnerable at stage n if there is a string x such that

$$e(n-1) < |x| \leqq e(n) \leqq \max\{2^{c_j|x|},\ 2^{c_k|x|}\} < e(n+1),\ \text{and} \tag{13}$$

$$\text{neither } NE_j^{A(n)} \text{ nor } NE_k^{A(n)} \text{ accepts } x. \tag{14}$$

If $\langle i, j\rangle < \langle k, m\rangle$, then we say that $R(i, j)$ has higher priority than $R(k, m)$.

Stage $n \geqq 0$. We satisfy the unsatisfied requirement of highest priority which is vulnerable at stage n. If there is no such requirement, then we skip this stage and let $A(n+1)=A(n)$.

Case 1) Such requirement is $R(j, k)$, where $j \neq k$. We do nothing, and so $A(n+1)=A(n)$. Then, by its vulnerability, there is a string x such that neither NE_j^A nor NE_k^A accepts x.

Case 2) Such requirement is $R(i, i)$. Run $DE_i^{A(n)}$ on $0^{d(n)}$. If it rejects [accepts] the input, then add to $A(n)$ the least string of the form $0y$ [resp. $1y$] of length $e(n)$ not queried in the computation in order to make $A(n+1)$. By its vulnerability, there exists such a string.

Claim 1. Every $R(i, i)$ is eventually satisfied. Otherwise, let $R(i, i)$ be the

least unsatisfied requirement. Since there are only finitely many requirements of higher priority, there is a stage n at which $R(i, i)$ is the vulnerable requirement of highest priority. Then, at such least stage $R(i, i)$ would have been satisfied.

It readily follows from Claim 1 that $L_0(A)$ and $L_1(A)$ are $\mathbf{DEXT}[A]$-inseparable and hence $\mathbf{DEXT}[A] \neq \mathbf{NEXT}[A]$.

Claim 2. If S, $\neg S \in \mathbf{NEXT}[A]$, then $S \in \mathbf{DEXT}[A]$.

Proof. There are indecies j and k such that $S = NE_j^A$ and $\neg S = NE_k^A$. Clearly, $R(j, k)$ is never satisfied. There is an m such that

(15) $\forall x[|x| \geqq e(m) \Rightarrow \exists^{\leqq 1} n(|x| \leqq e(n) \leqq \max\{2^{c_j|x|}, 2^{c_k|x|}\})]$, and

(16) if there are some requirements of higher priority than $R(j, k)$ which are ever satisfied, then they all were satisfied before stage m.

Now, let x be an arbitrary input. If $|x| \leqq e(m)$, then we decide by a finite table if $x \in S$. Otherwise, compute the least n such that $e(n) \geqq |x|$. This can be done within $O(|x|)$ time. For all strings u of length $e(\ell)$ for $\ell \leqq n-1$, we can determine whether $u \in A$ in time $2^{O(|x|)}$, because the number of such u's is less than $2^{2\cdot|x|}$, since $e(n-1) < |x|$.

Case (i) $e(n) > \max\{2^{c_j|x|}, 2^{c_k|x|}\}$. Then, the computation of machines j [resp. k] on x with oracle $A(n)$ is the same as with oracle A. Since we can know the elements of $A(n) - B$ in $2^{O(|x|)}$ time, we can construct the following oracle Turing machine $NE_{i(n)}^{A(n)}$: $NE_{i(n)}^{A(n)}$ simulates $NE_j^{A(n)}$ with the same time bound but queries no strings of length $e(s)$ for any $s < n$. Then the computation of $NE_{i(n)}^{A(n)}$ on x coincides with the one of $NE_{i(n)}^{A(n)}$. Hence

(17) $$x \in S \Leftrightarrow NE_{i(n)}^B \text{ accepts } x \Leftrightarrow \langle i(n), x, 0^{2^{c_j|x|}} \rangle \in KE(B),$$

where $KE(B) = \{\langle i, x, 0^n \rangle : NE_i^B \text{ accepts } x \text{ in } \leqq n \text{ steps}\}$. Then, $KE(B)$ can be accepted by an oracle Turing machine with oracle B in linear time. The code $i(n)$ can be obtained from the fixed code j by some minor change and by adding a finite table of length $2^{O(|x|)}$. So, by (17), whether $x \in S$ can be nondeterministically computed in time $2^{O(|x|)}$ relatively to B. Since $\mathbf{NEXT}[B] = \mathbf{DEXT}[B]$, this can be done deterministically with the same time bound.

Case (ii) Otherwise. Since $|x| \geqq e(m)$, by (15) we have:

(18) $$e(n-1) < |x| \leqq e(n) \leqq \max\{2^{c_j|x|}, 2^{c_k|x|}\} < e(n+1).$$

So, if neither $NE_j^{A(n)}$ nor $NE_k^{A(n)}$ accepts x, then $R(j, k)$ becomes vulnerable at stage n. Moreover, by (16) it then has the highest priority, and hence it is satisfied at this stage. However, we know that this $R(j, k)$ is never satisfied, a contradiction. So, either of them must accept x. Further, as in Case (i), we can know which machine accepts x in time $2^{O(|x|)}$. Suppose $NE_j^{A(n)}$ does. A may contain a string not in $A(n)$. Since $\mathbf{P}[B] = \mathbf{NP}[B]$ by our choice and since

$A(n)-B$ has been computed in time $2^{O(|x|)}$, we can deterministically find an accepting computation of $NE_j^{A(n)}$ on x in time $2^{O(|x|)}$ relatively to B (and hence to A), by using the following Lemma:

Lemma (A 2^{lin}-version of Lemma 2 in [BGS 75]). Let B be an oracle such that $\mathbf{P}[B]=\mathbf{NP}[B]$. Then, for each nondeterministic oracle Turing machine $NE_i^{\sim}$ there exists a deterministic oracle Turing machine $DE_j^{\sim}$ such that if NE_i^B accepts x then DE_j^B on x produces an accepting computation of NE_i^B as an output.

[*Outline of the proof.* We here borrow all notations such as INIT, I_k etc. (with $2^{c_i|x|}$ instead of $p_i(|x|)$) from [BGS 75] without explanation. Then, INIT $(B)\in\mathbf{P}[B]$. So, INIT(B) is accepted by a deterministic $\lambda n[n^{\ell}]$ time bounded oracle Turing machine with oracle B for some integer ℓ. Now, suppose NE_i^B accepts x. Let I_0 be the initial instantaneous description (ID) of NE_i^B on x. Then, $\langle i, x, 0^{2^{c_i|x|}}, I_0\rangle\in\mathrm{INIT}(B)$. Suppose $\langle i, x, 0^{2^{c_i|x|}}, I_0, I_1, \ldots, I_{k-1}\rangle\in\mathrm{INIT}(B)$, where k, $|I_0|, \ldots, |I_{k-1}|<2^{c_i|x|}$. There are only finitely many (determined by the code i) possible next ID's I_k. For each such I_k, whether $\langle i, x, 0^{2^{c_i|x|}}, I_0, \ldots, I_{k-1}, I_k\rangle\in\mathrm{INIT}(B)$ can be deterministically decided in time about $(2^{c_i|x|})^{3\ell}=2^{O(|x|)}$. So, we can deterministically find an accepting computation $I_0, I_1, \ldots, I_m$ (where $m<2^{c_i|x|}$) of NE_i^B on x in time $2^{O(|x|)}$ relatively to B.]

In this accepting computation, if a string w of length $e(n)$ is queried to $A(n)$, then ask if $w\in A$. (Recall that the number of w's is $\leqq 2^{O(|x|)}$.) Subcase 1): no such string w is in A. This computation coincides with a computation of NE_j^A on x. So, NE_j^A also accepts x: $x\in S$. Hence, we can deterministically decide whether $x\in S$ in $2^{O(|x|)}$ time relatively to A. Subcase 2): Otherwise. Then, $NE_j^{A(n)}$ on x queries a string w of length $e(n)$ which belongs to A. Since there is at most only one string of length $e(n)$ in A, by the above method $A(n+1)-A(n)$ can be computed deterministically in time $2^{O(|x|)}$ and hence so can $A(n+1)-B$. By (18), we can also deterministically decide in time $2^{O(|x|)}$ which of the machines accepts x, $NE_j^{A(n+1)}$ or $NE_k^{A(n+1)}$. Again by (18) we have:

$$x\in S\Leftrightarrow NE_j^A \text{ accepts } x\Leftrightarrow NE_j^{A(n+1)} \text{ accepts } x$$

or

$$x\in S\Leftrightarrow NE_k^A \text{ accepts } x\Leftrightarrow NE_k^{A(n+1)} \text{ accepts } x.$$

Either right hand side can be computed from an accepting computation which can be obtained deterministically in time $2^{O(|x|)}$ by using the method of Case (i) (which uses a finite table and $KE(B)$). Consequently, whether $x\in S$ can be decided deterministically in time $2^{O(|x|)}$ relatively to A.

Thus, we have $S\in\mathbf{DEXT}[A]$. Since recursiveness of A is clear, this completes the proof of Theorem 4.4. □

By this theorem with (7), we have

Theorem 4.5. *There is a recursive oracle A such that* Unif(**coNEXT**$[A]$) does not hold.

§5. Other Problems

There is another descriptive-set-theoretical principle called the reduction principle. Let $\mathcal{C}$ be a complexity class of sets of strings, and let

$$\begin{aligned}\mathrm{Red}(\mathcal{C}) \Leftrightarrow \text{For any } X,\ Y \in \mathcal{C} \text{ there exist } X_1,\ Y_1 \in \mathcal{C} \text{ such that}\\ X_1 \subseteq X,\ Y_1 \subseteq Y,\ X_1 \cup Y_1 = X \cup Y, \text{ and } X_1 \cap Y_1 = \emptyset.\end{aligned}$$

Then, the reduction principle for $\mathcal{C}$ is that $\mathrm{Red}(\mathcal{C})$ holds. Under very mild condition, we have

(7′) $\mathrm{Unif}(\mathcal{C})$ implies $\mathrm{Red}(\mathcal{C})$, and $\mathrm{Red}(\mathcal{C})$ implies $\mathrm{Sep}(\mathbf{co}\mathcal{C})$.

We know no information on the reduction principle for familiar complexity classes except for a few things. For examples,

Problem 3. *Does* Red(**NP**) *hold*?

Problem 4. *Does* Red(**NEXT**) *hold*?

Acknowledgments. Most of this work was done while the author was on leave and stayed at the Department of Mathematics, University of California, Santa Barbara. The author would like to thank Professor Ronald V. Book and the Department of Mathematics for their kind hospitality. Also, the author thanks the anonymous referee for his comments.

References

[BGS 75] Baker, T., Gill, J., and Solovay, R., Relativizations of the $P = ?$ NP question, *SIAM J. Comput.*, 4 (1975), 431–442.

[BDG 88] Balcázar, J.L., Díaz, J., and Gabarró, J., *Structural Complexity* I, Springer-Verlag, Berlin etc., 1988.

[BDG 90] ———, *Structural Complexity* II, Springer-Verlag, Berlin etc., 1990.

[Bo 74] Book, R.V., Comparing complexity classes, *J. Comput. System Sci.*, 9 (1974), 213–229.

[Kl 52] Kleene, S.C., *Introduction to Metamathematics*, North-Holland Publ. Co., Amsterdam etc., 1952.

[Mo 80] Moschovakis, Y.N., *Descriptive Set Theory*, North-Holland Publ. Co., Amsterdam etc., 1980.

[Ro 67] Rogers, H.Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill Book Co., New York etc., 1967.

[Sa 70] Savitch, W.J., Relationships between nondeterministic and deterministic tape complexities, *J. Comput. Syst. Sci.*, 4 (1970), 177–192.

[Sh 67] Shoenfield, J.R., *Mathematical Logic*, Addison-Wesley Publ. Co, Reading Massachusetts etc., 1967.

Note added in proof: V.G. Kanoveĭ gives a counter example for Problem 1 (in a private communication).