Publl. RIMS, Kyoto Univ.
21 (1985), 439–453

Some Hierarchies of Relativized Time-Bounded Complexity Classes[1]

By

Hisao TANAKA[*], Masa-aki IZUMI[*][†] and Nobuyuki TAKAHASHI[**]

Abstract

Some results on relativized time-bounded complexity classes are presented. There can be many kinds of hierarchies of complexity subclasses of relativized NP. For brevity, let P(A, k) [NP(A, k)] be the relativized complexity class DTIME$^A(n^k)$ [resp. NTIME$^A(n^k)$] with respect to oracle set A. (For $k=1$, replace n^k by $2n$). Then for example: 1). There is an oracle set A such that for all $k>0$ P(A, k) is properly contained in NP(A, k) and NP(A, k) is properly contained in P(A, $k+1$). 2) For each $k>0$, there is an oracle set D (depending on k) such that for any $i\leq k$ P(D, i)$\neq$NP(D, i) but for all $j>k$ P(D, j)=NP(D, j). Besides, we show a theorem which is a higher level analog to a theorem of Book, Wilson and Mei-Rui [3].

§1. Introduction

One of the most important problems in the theory of computation is to find the precise relationship between the two kinds of computation: one is deterministic computation and the other is nondeterministic one. A fundamental open question is to determine whether or not

$$\text{(1)} \qquad \mathrm{P}=\mathrm{NP},$$

where P [NP] is the class of languages accepted in polynomial time by deterministic [resp. nondeterministic] Turing machines. Baker, Gill and Solovay [1] (and others) show that the corresponding relativized question to (1) holds for some oracle sets on one hand and does not hold for some other oracle sets on the other hand. That is, they construct oracle sets A and B such that

Communicated by S. Takasu, January 15, 1984. Revised July 5, 1984.
[1] The authors would like to thank the referee for his kind and valuable advice.
[*] College of Engineering, Hosei University, Koganei, Tokyo 184, Japan.
[**] Yokohama College of Commerce, Yokohama 230, Japan.
[†] Current address: Department of Information Scieince, Tokyo Institute of Technology.

(2) $$\mathrm{P}(A)=\mathrm{NP}(A) \quad \text{and} \quad \mathrm{P}(B)\neq\mathrm{NP}(B).$$

This result gives a strong influence to methods which will be employed to solve (1).

Now, for the sake of simplicity, let $\mathrm{P}(k)$ [$\mathrm{NP}(k)$] be the complexity class $\mathrm{DTIME}(n^k)$ [resp. $\mathrm{NTIME}(n^k)$] for $k>1$. (For $k=1$, replace n^k by $2n$). Then, although it has been shown by Paul, Pippenger, Szemerédi and Trotter [5] that $\mathrm{P}(1)\neq\mathrm{NP}(1)$, we do not know whether

(3) $$\mathrm{P}(k)=\mathrm{NP}(k)$$

holds for any integer $k>1$, yet. We suspect that these questions will be difficult to solve. So, we here consider their relativized versions and show that there can be various hierarchies of complexity subclasses of relativized NP. Namely, it will be shown that (3) holds for some oracle sets B on one hand but does not hold for other oracle sets A on the other hand. In fact, we shall construct such oracles A and B independent of k (Theorems 1 and 2). Significance is their independence of k. Proofs are easy if their independency of k is not required. Although our method of the proof for Theorem 1 might be called a kind of the dovetailing methods (Kintala and Fischer [4; pp. 49–50]), we will describe its full proof. Because the polynomial degree of the time-bound function $f_e(n)$ of the e-th polynomial-time-bounded oracle Turing machine does not monotonically increase as e increases. So, our construction of the oracle set A will be a little better involved. Further, it will be shown that there can be other kinds of hierarchies, too (Theorem 3). Moreover, these results can be extended to some classes of exponential time-bounded complexity (Theorems 4, 5 and 6). Finally, we shall show a theorem which is higher level analog to a theorem of Book, Wilson and Mei-Rui [3] (Theorem 7).

§ 2. Preliminaries

We mostly use standard terminology and notation for complexity theory. See, e.g., [1]–[4]. Let Σ be an alphabet. We assume $\Sigma=\{0, 1\}$. Σ^* denotes the set of all finite strings consisting of members of Σ. A subset L of Σ^* is called a language and we denote the complement of L by $\bar{L}$: $\bar{L}=\Sigma^*-L$. For x in Σ^*, $|x|$ denotes the length of x. Our model for computation is the oracle Turing machine. An *oracle Turing machine* (abbreviated by OTM) is a multitape Turing machine with an query tape and with three special internal states called the query state $q_?$, the yes state q_Y and the no state q_N. When an OTM M is associated with an oracle set $X \subseteq \Sigma^*$, we denote it by M^X and call an OTM with oracle X. When an OTM M^X enters the state $q_?$, the machine asks the oracle X whether the string written on its query tape belongs to X. If the string is in X, then M^X enters q_Y, otherwise it enters q_N.

If the next-move-operation of M is single-valued, then we call M to be *deterministic*, otherwise M to be *nondeterministic*. Now, suppose that M^X runs on an input $x \in \Sigma^*$ and it halts after some running time. (We always consider OTM's that halt for every input.) If the final state in the computation is a special state called an accepting state, then we say M^X *accepts* x. For a nondeterministic OTM, there can be many computation processes (called computation paths). Then M^X accepts x if one of computation paths enters an accepting state. Otherwise we say it *rejects* x. Let L be a language. L is *accepted* by an M^X (denoted by $L = T(M^X)$) if the following condition holds: For all strings x x is in L iff M^X accepts x.

Let ω be the set of all natural numbers, and let f be a function from ω to ω. An OTM M is *f-time-bounded* if every computation of M on any input x halts in fewer than $f(|x|)$ steps, whatever oracle X is used. For $\mathscr{F}$ a family of functions, an OTM M is called to be *$\mathscr{F}$-time-bounded* if M is f-time-bounded for some f in $\mathscr{F}$. Let X be an oracle set. DTIME(X, f) [DTIME$(X, \mathscr{F})$] is the class of languages accepted by deterministic f [resp. $\mathscr{F}$]-time-bounded OTM's with oracle X. NTIME(X, f) and NTIME$(X, \mathscr{F})$ are their nondeterministic counterparts. By convention, e.g., DTIME(X, n^k) is DTIME(X, f), where $f(n) = n^k$ for k a constant. For brevity, we use the following notation:

$\mathrm{P}(X, k) = \mathrm{DTIME}(X, n^k) = \cup \{\mathrm{DTIME}(X, c \cdot n^k): c > 0\}$ for $k > 1$ (by the speed-up theorem),

$$\mathrm{P}(X, 1) = \cup \{\mathrm{DTIME}(X, c \cdot n): c > 0\} (= \mathrm{DTIME}(X, 2n)),$$
$$\mathrm{P}(X) = \cup \{\mathrm{P}(X, k): k > 0\},$$
$$\mathrm{DEXT}(X, k) = \cup \{\mathrm{DTIME}(X, 2^{cn^k}): c > 0\},$$
$$\mathrm{DEXT}(X) = \mathrm{DTIME}(X, 2^{\mathrm{lin}}) = \cup \{\mathrm{DTIME}(X, 2^{cn}): c > 0\},$$
$$\mathrm{DEPT}(X) = \mathrm{DTIME}(X, 2^{\mathrm{poly}}) = \cup \{\mathrm{DEXT}(X, k): k > 0\}.$$

NP(X, k), NP$(X, 1)$, NP(X), NEXT(X, k), NEXT(X) and NEPT(X) are the corresponding nondeterministic complexity classes to the above, respectively. We also use P$(X, k+1/2)$ whose definition is clear. Nonrelativized classes P, P(k), etc. are used, too.

For a positive real number α, $\lceil\alpha\rceil$ denotes the least integer larger than or equal to α. "$O(f(n))$-time-bounded" means that $c\cdot f(n)$-time-bounded for some constant $c>0$. Similarly for the word "$O(f(n))$ steps".

§3. Two Hierarchies of Subclasses of Relativized NP

In this section we shall construct oracle sets A and B as stated in §1. We need a fact that is a folklore-type lemma:

Lemma A. *Let m and k be fixed positive integers. Let $f(n)=n^{m+1/k}$ and let $f'(n)=n^m\cdot\lceil n^{1/k}\rceil$. Then there is a cf-time-bounded* TM *T for some constant c such that for any input w T outputs a string v (e.g., of the form $w0^i$) whose length is $f'(|w|)$.* □

Let $\lambda ki[\langle k, i\rangle]$ be a recursive pairing function from $(\omega-\{0\})\times\omega$ onto ω. If $e=\langle k, i\rangle$ we write $(e)_0=k$ and $(e)_1=i$. Let P_e $[NP_e]$ be the e-th deterministic [resp. nondeterministic] polynomial-time-bounded OTM, and let f_e be its time-bound. We may assume that $f_e(n)=c_e n^k$, where $k=(e)_0$. Further we may assume that $c_e^{(k+1/2)/k}$ is an even integer for some purpose. (We could take $c_e=1$ if $k>1$. But, in order to uniformly describe our construction for all $k\geq 1$ and to want to extend Theorems 1 and 2 to Theorems 4 and 5, respectively, we do not so.) For any oracle X and any positive integer k, let

$$L_k(X)=\{0^m\colon \exists u\in X[|u|=m^k]\}.$$

Clearly $L_k(X)$ is in NP(X, k). Now we have the following theorem:

Theorem 1. *There is an oracle set A such that for all $k>0$ $\mathrm{P}(A, k)\subsetneq\mathrm{NP}(A, k)\subsetneq\mathrm{P}(A, k+1)$. That is, we obtain the following hierarchy:*

$$\mathrm{P}(A, 1)\subsetneq\mathrm{NP}(A, 1)\subsetneq\mathrm{P}(A, 2)\subsetneq\mathrm{NP}(A, 2)\subsetneq\cdots$$

(So, for this A, clearly $\mathrm{P}(A)=\mathrm{NP}(A)$ holds.)

Proof. We construct the required oracle A in stages as in proofs described in e.g., [1], [3] or [4]. Let $A(s)$ be the set of all strings placed into A prior to stage s. We construct A in such a way that: for all $k\geq 1$

(a) $L_k(A)\neq T(P^A_{\langle k,i\rangle})$ for any i and hence $L_k(A)\notin\mathrm{P}(A, k)$ and so $\mathrm{P}(A, k)\neq\mathrm{NP}(A, k)$, and

(b) $\mathrm{NP}(A, k)\subseteq\mathrm{P}(A, k+1/2)$ and hence $\mathrm{NP}(A, k)\subsetneq\mathrm{P}(A, k+1)$.

For (a), our construction causes each machine P^A_e with $(e)_0=k$ to reject

$L_k(A)$ by placing a string u into A at an appropriate odd stage $2s+1$. Then we cancel the index e at this stage and set $q_e=2s+1$. u must be chosen in such a way that u has never been queried in any computation performed at any earlier stage $<2s+1$. To ensure the existence of such a string u we wait until the conditions (i)–(iv) (see below) are satisfied. To obtain A to be independent of k, we consider the number $b=\max\{(a)_0\colon a\leq e\}$ and choose u depending on this b.

For (b), consider an arbitrary language L in NP(A, k). Let $L=T(NP^A_{\langle k,i\rangle})$. We want a deterministic $O(n^{k+1/2})$-time-bounded OTM M^A simulating $NP^A_{\langle k,i\rangle}$. For this purpose we encode full information about nondeterministic acceptance of a ~~s~~tring x by $NP^A_{\langle k,i\rangle}$ into A as a string y of even length. Roughly speaking, at a certain even stage $2s$ we place a string y of the form $y=0^k10^i1x10^n$ into A when and only when $NP^A_{\langle k,i\rangle}$ accepts x. In order that every string placed into A at later odd stages may not queried in any computation of $NP^{A(2s)}_{\langle k,i\rangle}$ on x, the length of every string queried in any such computation must be less than $(2s)^{k/(k+1/2)}$. For this purpose, we impose the following condition on y:

$$|y|=(c_{\langle k,i\rangle}|x|^k)^{(k+1/2)/k}.$$

Then we get a desired deterministic machine M^A as is shown in later proof. Of course, we also must impose a further condition on y in order to ensure that A is independent of k. Now we describe our construction for A in detail. Let $A(0)=A(1)=\emptyset$ and start at stage 1. Recall q_e is the odd stage at which the index e is cancelled. As convention, put $q_{-1}=0$.

(I) Stage $2s+1$. Let e be the first uncancelled index at this stage and put $e=\langle k, i\rangle$. Let $b=\max\{(a)_0\colon a\leq e\}$. Note that $b\geq 1$. Suppose the following conditions are satisfied:

(i) $2s+1>q_{e-1}$,

(ii) $(2s+1)^{b/(k\cdot(b+1/2))}$ is an (odd) integer. For brevity, we denote it by m.

(iii) $m^k>f_{e'}(q_{e'})$ for all $e'<e$, and

(iv) $f_e(m)=c_e\cdot m^k<2s+1<2^{m^k}$.

Then we execute the following procedure. Run the deterministic machine $P_e^{A(2s+1)}$ on the string 0^m. If the machine rejects 0^m, then we add to A a string u of odd length m^k not queried during the computation: $A(2s+2)=A(2s+1)\cup\{u\}$. By (iv) such a string u exists. If 0^m is accepted let $A(2s+2)=A(2s+1)$. Then cancel the index e at this stage in either case, and set $q_e=2s+1$.

If at least one of the conditions (i)–(iv) does not hold, then we skip this stage and set $A(2s+2)=A(2s+1)$. Clearly every index will eventually cancelled.

(II) Stage $2s+2$. Let e be the last (i.e. largest) index that was cancelled at an odd stage before $2s+2$. Let $d=\max\{(a)_0\colon a\leq e\}$. Consider a string y such that

$$(1)\quad y=0^k10^i1x10^n \mathbin{\&} k\leq d \mathbin{\&} 2s+2=|y|=c_{\langle k,i\rangle}^{(k+1/2)/k}\cdot|x|^k\cdot\ulcorner|x|^{1/2}\urcorner \quad \text{for some } k,\ i,\ n\in\omega \text{ and } x\in\Sigma^*.$$

For such a string y, run $NP_{\langle k,i\rangle}^{A(2s+2)}$ on the string x. Note that the length of a string queried in any computation of this machine on x is less than

$$c_{\langle k,i\rangle}\cdot|x|^k\leq(c_{\langle k,i\rangle}\cdot|x|^k\cdot\ulcorner|x|^{1/2}\urcorner)^{k/(k+1/2)}\leq(2s+2)^{d/(d+1/2)}<2s+2.$$

If some computation of this machine accepts x, then we place y into A. Otherwise, we do nothing.

Let $A'(2s+3)$ be the set of all strings placed into A by performing the above procedure for all y satisfying (1), and set $A(2s+3)=A(2s+2)\cup A'(2s+3)$. If there is no such e or there is no such y satisfying (1), then we skip this stage, and set $A(2s+3)=A(2s+2)$.

Define A by $A=\bigcup_{s=0}^{\infty}A(s)$. We show that A is a desired oracle set. First we show that, in (I), the string u placed into A at the stage q_e is not queried in any computation performed at any earlier stage $<q_e$. For, length of any string queried at odd stage $q_{e'}$ for any $e'<e$ is less than

$$f_{e'}(q_{e'}^{b'/(j\cdot(b'+1/2))})<f_{e'}(q_{e'})<m^k=|u|,$$

where $j=(e')_0$ and $b'=\max\{(a)_0\colon a\leq e'\}$. So, u is not queried at any earlier odd stage. At an even stage $2t<q_e$ let e' be the largest index cancelled at an odd stage $<2t$ and let $d'=\max\{(a)_0\colon a\leq e'\}$. Hence $d'\leq b=\max\{(a)_0\colon a\leq e\}$. Then by (II) with $s=t-1$, length of any string queried at stage $2t$ is less than

$$(2t)^{d'/(d'+1/2)}\leq(2s+1)^{b/(b+1/2)}=|u|.$$

So, u is not queried at any earlier even stage.

Next, in (II), y is not queried at any earlier stage $<2s+2$. Because, length of any string queried at any earlier stage is less than $2s+2$ which is the length of y.

Thus, for each stage s at which the described procedure is executed indeed, any computation performed with oracle $A(s)$ at stage s is the same as one with A.

Now, to prove (a), let i be arbitrary and consider the index $e=\langle k, i\rangle$. m is as in (I). Then P_e^A rejects 0^m iff $P_e^{A(q_e)}$ rejects 0^m iff $\exists u\in A[|u|=m^k]$ iff $0^m\in L_k(A)$.

So, $P^A_{\langle k,i\rangle}$ does not accepts $L_k(A)$.

To prove (b), let $L \in \mathrm{NP}(A, k)$ be arbitrary, and let $L = T(NP^A_{\langle k,i\rangle})$. We define a deterministic $O(n^{k+1/2})$-time-bounded OTM M with oracle A as follows: Let $e' = \langle k, i\rangle$. Then $q_{e'}$ is determined. Given a string x, first M generates the string y such that

(2) $$y = 0^k 10^i 1x10^n \quad \text{and} \quad q_{e'} < |y| = c^{(k+1/2)/k}_{\langle k,i\rangle} \cdot |x|^k \cdot \ulcorner |x|^{1/2} \urcorner .$$

By Lemma A, this is done in $O(|x|^{k+1/2})$ steps. There are only finitely many x's for which there is no y satisfying (2). So, we can make a finite table so that M accepts x iff x is in L for such x's. Now, let $2s+2 = |y|$. Let e be the last index cancelled before stage $2s+2$. Then $e' \leq e$, since $q_{e'} < 2s+2$. So $k \leq d$, where $d = \max\{(a)_0 : a \leq e\}$. Hence (1) holds for these $2s+2$, y, k, i, n and x. If y is in A, then M^A accepts x, otherwise M^A rejects x. Thus M^A accepts x iff y is in A iff $NP^{A(2s+2)}_{\langle k,i\rangle}$ accepts x iff $NP^A_{\langle k,i\rangle}$ accepts x iff x is in L. That is, M^A accepts L. Obviously, M is deterministic $O(n^{k+1/2})$-time-bounded OTM. So, L is in $\mathrm{P}(A, k+1/2)$ and hence $\mathrm{NP}(A, k) \subseteq \mathrm{P}(A, k+1/2)$. This completes the proof of Theorem 1. □

THEOREM 2. *There is an oracle set B such that for all $k \geq 1$ $\mathrm{P}(B, k) = \mathrm{NP}(B, k)$.*

Proof. Let $B(s)$ be the set of all strings placed into B before stage s, and let $B(0) = \emptyset$.

Stage s. Consider a string y such that

(3) $$y = 0^k 10^i 1x10^n \quad \text{and} \quad s = |y| = c_{\langle k,i\rangle}|x|^k \quad \text{for some } k \geq 1,\ i, n \in \omega \text{ and } x \in \Sigma^* .$$

Run $NP^{B(s)}_{\langle k,i\rangle}$ on x. Length of strings queried in any computation of $NP^{B(s)}_{\langle k,i\rangle}$ on x is less than s. If x is accepted we place y into B. Otherwise we do nothing. We execute this procedure for every y satisfying (3), and let $B'(s+1)$ be the set of all strings added to B at this stage. Let $B(s+1) = B(s) \cup B'(s+1)$. If there is no such y we let $B(s+1) = B(s)$.

Define $B = \bigcup_{s=0}^{\infty} B(s)$. Then B is a desired oracle set. □

So, by not all oracles Paul–Pippenger–Szemerédi–Trotter's Theorem $\mathrm{P}(1) \neq \mathrm{NP}(1)$ can be relativized.

§4. Another Type of Hierarchies on Subclasses of Relativized NP

In this section we prove the following theorem:

Theorem 3. *For each* $k \geq 1$, *there is an oracle set* D_k *such that* $\mathrm{P}(D_k, k) \neq \mathrm{NP}(D_k, k)$ *but* $\mathrm{P}(D_k, k+1) = \mathrm{NP}(D_k, k+1)$.

That $\mathrm{P}(k) = \mathrm{NP}(k)$ implies $\mathrm{P}(k+1) = \mathrm{NP}(k+1)$ is a folkloretype proposition. By Theorem 3 with this proposition, we get the following hierarchy:

$$(\forall i)[1 \leq i \leq k \rightarrow \mathrm{P}(D_k, i) \neq \mathrm{NP}(D_k, i)] \& (\forall j)[j > k \rightarrow \mathrm{P}(D_k, j) = \mathrm{NP}(D_k, j)].$$

Proof of Theorem 3. Let $k \geq 1$ be fixed. To simplify our proof, we assume $k \geq 2$. We use subenumerations $\{P_e : (e)_0 = k \text{ or } k+1\}$ and $\{NP_e : (e)_0 = k \text{ or } k+1\}$, where P_e and NP_e are OTM's stated in the preceding section. This time, we may assume that the time-bound functions of P_e and NP_e for $e = \langle l, i \rangle$, where $l = k$ or $k+1$, are $f_e(n) = n^l$, since $l \geq 2$. As before, $D = D_k$ will be constructed at stages. Let $D(s)$ be the set of all strings placed into D prior to stage s. We construct D in such a way that

(a) $L_k(D) \neq T(P^D_{\langle k,i \rangle})$ for any i and hence $L_k(D) \notin \mathrm{P}(D, k)$, so $\mathrm{P}(D, k) \neq \mathrm{NP}(D, k)$, and

(b) $\mathrm{NP}(D, k+1) \subseteq \mathrm{P}(D, k+1)$.

At each stage, some strings (possibly none) are placed into D and some other strings (possibly none) are reserved for the complement $\bar{D}$. (a) will be satisfied at some odd stages and (b) at some even stages. First of all we consider (b). Let $M_0 = NP_{\langle k+1, i \rangle}$ be arbitrary. We want a deterministic $O(n^{k+1})$-time-bounded OTM M with oracle D which simulates M_0^D. In order to encode full information about nondeterministic acceptance of a string x by M_0^D into D, we consider the following strings at even stage $2s$:

$$(*) \qquad y = 0^i 1 x 1 0^n, \quad \text{where} \quad |y| = s,$$

and we place a string $w = y0^l$ into D when and only when "$M_0^{D(2s)}$ accepts x". In order that w may not be queried at any earlier stages, we take $l = |y|^{k+1}$. We have a difficulty to overcome. In Theorem 1, for each nondeterministic $O(n^k)$-time-bounded OTM we only need a deterministic $O(n^{k+1/2})$-time-bounded OTM simulating it. This time we must have a deterministic $O(n^{k+1})$-time-bounded OTM which simulates a given nondeterministic $O(n^{k+1})$-time-bounded OTM.

So, it is difficult to expect that every string newly placed into D or newly reserved for $\bar{D}$ is never queried at any earlier stage. That is, some unaccepting computation performed at even stage $2s$ with oracle $D(2s)$ may query some strings which will be placed into D or reserved for $\bar{D}$ at some future stages. Then, we must construct D so that the computation with $D(2s)$ still remains unaccepting even when the full oracle D is employed instead of $D(2s)$. And all accepting computation with oracle $D(2s)$ still must be accepting one even when the oracle D is used. These will be solved by using an idea due to [3].

For (a), no difficulty occurs and a standard technique is applicable. So, let us turn to describe the construction of D in details. Let $D(0)=\emptyset$.

(I) Stage $2s$. Consider the following condition

$$s^{k+1}\cdot 2^s < 2^{2(s-2)}. \tag{1}$$

If (1) is not satisfied we skip this stage and set $D(2s+1)=D(2s)$. Suppose (1) holds. (Clearly all sufficiently large s's satisfy (1). By this, the number of strings reserved at all earlier even stages including $2s$ is less than 2^{2s}, as will be seen later). Consider strings y such that

$$y=0^i1x10^n \text{ for some } i,\ n\in\omega \text{ and } x\in\Sigma^*, \text{ where } |y|=s \tag{2}$$

Let

$$y_1, y_2, \dots, y_r \quad (r\geq 1 \text{ by (1)}) \tag{3}$$

be an enumeration of all such y's. We put $y_j=0^{i(j)}1x_j10^{n(j)}$ for $1\leq j\leq r$. Set $D_0(2s)=D(2s)$ and consider y_j. For simplicity, we write $y_j=y=0^i1x10^n$. Run $\mathrm{NP}^{D_{j-1}(2s)}_{\langle k+1,i\rangle}$ on x. The length of a string queried in any computation of this machine is less than s^{k+1}. If x is accepted we add to $D_{j-1}(2s)$ the even length string $y0^m$, where $m=|y|^{k+1}$. Every string of this length is not queried in any computation at any earlier (including this) stage. (See the condition (i) below). Further we reserve all unreserved strings queried in this accepting computation for $\bar{D}$. If no computation of this machine accepts x, then determine whether the addition of some unreserved strings to $D_{j-1}(2s)$ will lead to acceptance. (This and following ideas are due to [3; p. 579].) If so (case (a)), then place those unreserved strings queried in an accepting computation into D and $\bar{D}$ appropriately so that acceptance is preserved. Place $y0^m$ into D. Let $D_j(2s)$ be the set obtained from $D_{j-1}(2s)$ by adding all such strings as above. If not (case (b)), then we only do reservation of $y0^m$ for $\bar{D}$. Let $D(2s+1)=D_r(2s)$ and go to the next stage. Note that any computation of $NP^{D_j(2s)}_{\langle k+1,i\rangle}$ on x performed as above is not affected by any possible future addition to D. And for the case of the above illustration

(4) $$y0^m \text{ is in } D \quad \text{iff} \quad NP^{D_J(2s)}_{\langle k+1,i\rangle} \text{ accepts } x\,.$$

(II) Stage $2s+1$. Let i be the first uncancelled index. Suppose the following three conditions hold:

(i) $f_{\langle k,j\rangle}(q_j)<\min\{s+s^{k+1}, (2s+1)^k\}$ for all $j<i$.

(Note that $s+s^{k+1}$ is the length of $y0^m$, where $|y|=s$ and $m=s^{k+1}$.)

(ii) $f_{\langle k,i\rangle}(2s+1)=(2s+1)^k<(s+1)+(s+1)^{k+1}$ $(<(2s+1)^{k+1})$,

(iii) $(2s+1)^{k+1}+2^{2s}<2^{(2s+1)^k}$.

Run $P^{D(2s+1)}_{\langle k,i\rangle}$ on $z=0^{2s+1}$. We reserve all unreserved strings queried in the computation for $\bar{D}$. If the machine rejects z, then we take a string u of length $(2s+1)^k$ such that it is not queried in the computation and is unreserved at the beginning of this stage. (Existence of such a string u is proven in Claim 1 below.) Set $D(2s+2)=D(2s+1)\cup\{u\}$. If the machine accepts z, then we reserve all strings of length $(2s+1)^k$ for $\bar{D}$ and $D(2s+2)=D(2s+1)$. In either case we cancel the index i and set $q_i=2s+1$. If one of the conditions (i)–(iii) does not hold, then we skip this stage and let $D(2s+2)=D(2s+1)$. Clearly each index i will eventually be cancelled.

Define D by $D=\bigcup_{s=0}^{\infty} D(s)$. We show that D is a desired oracle set.

Claim 1. When an odd stage $2s+1$ is executed, there is such a string u. Note that even though u had been queried in some computation for the case (b) at an earlier even stage $2s'$ $(<2s+1)$ the case (b) remains true after adding u to D. [Proof. By (i) we do not have to consider any earlier odd stage. So, we evaluate the number of strings reserved at earlier even stages. Let $2s'<2s+1$. At stage $2s'$, for each string $y=0^i1x10^n$ with $|y|=s'$ the number of strings reserved at the computation on x is less than $(s')^{k+1}$. So the number of strings reserved with respect to all such y's is less than

$$(s')^{k+1}\cdot(\text{the number of such } x\text{'s})<(s')^{k+1}\cdot 2^{s'}<2^{2(s'-2)}$$

by (1). Hence the entire number of strings reserved at all earlier even stages is less than $\sum_{s'=s_0}^{s} 2^{2(s'-2)}<2^{2s}$, where s_0 is the least t such that $t^{k+1}2^t<2^{2(t-2)}$ holds. The number of strings queried at the stage $2s+1$ is less than $(2s+1)^{k+1}$. So by (iii), their sum is less than $2^{(2s+1)^k}$. Consequently, there is such a string u.]

By using Claim 1 we can show $\mathrm{P}(D, k)\neq\mathrm{NP}(D, k)$ as before. Finally we show $\mathrm{P}(D, k+1)=\mathrm{NP}(D, k+1)$. Let L be an arbitrary language in $\mathrm{NP}(D, k+1)$ and let i be such that $L=T(NP^{D}_{\langle k+1,i\rangle})$. We define a deterministic

$O(n^{k+1})$-time-bounded OTM M with oracle D as follows: Given an input x, M produces a string y such that $y=0^i1x10^n$, where $n=|x|$. This is done in fewer than $O(|x|)$ steps. Put $s=|y|$. So we are considering (2). We execute stage $2s$. Let the above y be the j-th member in the enumeration (3): $y=y_j$. M produces a string $y0^m$, where $m=s^{k+1}$. This is done in fewer than $O(|x|^{k+1})$ steps. Then, M asks the oracle D if $y0^m$ is in D. If the answer is yes, then M accepts x. Otherwise M rejects x. By (4), x is in L iff $NP^{D_j(2s)}_{\langle k+1,i\rangle}$ accepts x iff $y0^m$ is in D. Therefore M accepts L. Clearly M is deterministic $O(n^{k+1})$-time-bounded OTM. So, L is in $\mathrm{P}(D, k+1)$ and hence $\mathrm{NP}(D, k+1) \subseteq \mathrm{P}(D, k+1)$. □

We can easily extend Theorems 1, 2, and 3 to the following form:

Theorem 4. *There is an oracle set E such that for all $k>0$,* $\mathrm{DEXT}(E, k) \subsetneq \mathrm{NEXT}(E, k) \subsetneq \mathrm{DEXT}(E, k+1)$.

Theorem 5. *There is an oracle set F such that for all $k>0$,* $\mathrm{DEXT}(F, k) = \mathrm{NEXT}(F, k)$.

Theorem 6. *For each $k>0$, there is an oracle set G_k such that* $\mathrm{DEXT}(G_k, k) \neq \mathrm{NEXT}(G_k, k)$ *but* $\mathrm{DEXT}(G_k, k+1) = \mathrm{NEXT}(G_k, k+1)$.

Proof. In the proof of Theorem 3, we used the equality $\mathrm{DTIME}(X, n^k) = \cup\{\mathrm{DTIME}(X, cn^k): c>0\}$ for $k>1$. This time we do not have

$$\mathrm{DTIME}(X, 2^{n^k}) = \cup\{\mathrm{DTIME}(X, 2^{cn^k}): c>0\}.$$

So we must modify the proof of Theorem 3 a little. Let $\mathscr{F}_k=\{2^{cn^k}: c>0\}$. Let $DEP_{\langle k,i\rangle}$ [$NEP_{\langle k,i\rangle}$] be the i-th deterministic [resp. nondeterministic] $\mathscr{F}_k$-time-bounded OTM with its time-bound $h_{\langle k,i\rangle}$, where $h_{\langle k,i\rangle}(n)=2^{c'n^k}$ with $c'=c_{\langle k,i\rangle}$. Let $k\geq 1$ be fixed. We may assume that $c^{1/(k+1)}_{\langle k+1,i\rangle}$ is an integer. Let

$$L_{EX_k}(X)=\{0^m: \exists u \in X[|u|=2^{m^k}]\}.$$

As before, we construct $G=G_k$ in stages. Let $G(0)=\emptyset$, $q_{-1}=0$ and $h_{\langle k,-1\rangle}(0)=0$.

Stage $2s$. Consider the strings y such that

$$y=0^i1x10^n, \quad \text{where } |y|=s.$$

Let $y_1,\ldots, y_r$ be an enumeration of all such y's. Let $G_0(2s)=G(2s)$ and consider $y=y_j$ $(1\leq j\leq r)$. This time, run $NEP^{G_{j-1}(2s)}_{\langle k+1,i\rangle}$ on x in fewer than $2^{n^{k+1}}$ steps. [Note. If the machine freely runs on x, then some strings of length about

$\exp(2, c_{\langle k+1,i\rangle}\cdot|x|^{k+1})$ may be queried in the computation. $c_{\langle k+1,i\rangle}$ may be very large (e.g., $2^{2^{\cdot^{\cdot^{2^i}}}}$) and can not be bounded by an appropriate number, e.g., by s^{k+1}. Then computation with oracle $G(2s+1)$ may not coincide with one with oracle G.] Subsequent argument is mutatis mutandis the same as in the proof of Theorem 3. Claim. DEPT$(G, k+1)=$NEPT$(G, k+1)$. For, let $L=T(NEP^G_{\langle k+1,i\rangle})$. We define a deterministic $2^{O(|x|^{k+1})}$-time-bounded OTM M^G which simulates $NEP^G_{\langle k+1,i\rangle}$, as follows: Given x, M first produces the string

$$y=0^i1x10^n, \quad \text{where} \quad n=c^{1/(k+1)}_{\langle k+1,i\rangle}\cdot|x|.$$

This is done by $O(|x|)$ steps. Subsequent argument is mutatis mutandis the same as in the proof of Theorem 3. □

§5. 2^{lin} Versus 2^{poly}.

In this section, we consider a higher-level analog to a theorem of Book, Wilson and Mei-Rui [3]. Let DE_i [NE_i] be the i-th deterministic [resp. nondeterministic] 2^{lin}-time-bounded OTM with its time-bound g_i, where $g_i(n)=2^{d_i n}$, for a positive integer d_i. Further, let DEP_i [NEP_i] be the i-th deterministic [resp. nondeterministic] 2^{poly}-time-bounded OTM with its time-bound $2^{p_i(n)}$, where p_i is a polynomial. The following proposition is due to Book [2]:

Proposition. (*a*) P$=$NP *implies* DEXT$=$NEXT, *and* (*b*) DEXT$=$NEXT *implies* DEPT$=$NEPT.

Proof. We shall state a proof of the relativized version of (b): For all oracle X, if DEXT$(X)=$NEXT(X), then DEPT$(X)=$NEPT(X). Let

$$K(X)=\{0^i1x10^n\text{: some computation of } NEP^X_i \text{ accepts } x \text{ in fewer than } 2^n \text{ steps}\}.$$

Clearly $K(X)$ is in NEXT(X) and hence NEPT$(X)\subseteq$P$(K(X))$ as in [1; Proof of Lemma on p. 433]. Suppose DEXT$(X)=$NEXT(X). Then $K(X)$ is in NEXT(X) and so there is an index i such that DE^X_i accepts $K(X)$. To show NEPT$(X)\subseteq$DEPT(X), let L be an arbitrary langugage in NEPT(X). Then there is an index j for which $\mathrm{P}^{K(X)}_j$ accepts L. Define a deterministic 2^{poly}-time-bounded OTM M with oracle X which accepts L: Given an input x, M first simulates the computation of $P^{K(X)}_j$ on x. If a string w is queried in the computation, then using oracle X M simulates the computation of DE^X_i on w

and decides whether w is in $K(X)$. The latter simulation can be done in fewer than $2^{d_i \cdot |w|}$ steps. M accepts x iff $P_j^{K(X)}$ does. So, M accepts L. Length of such a string w is less than $p_j(|x|)$ and the number of queried strings is less than $p_j(|x|)$. Therefore the entire steps of the computation of M on x is bounded by $2^{p(|x|)}$ for some polynomial p. Hence M is a deterministic 2^{poly}-time-bounded OTM. So, L is in DEPT(X). □

In contrast with (a), Book, Wilson and Mei–Rui [3] have shown that there is an oracle set W such that $\mathrm{P}(W) \neq \mathrm{NP}(W)$ but $\mathrm{DEXT}(W) = \mathrm{NEXT}(W)$. Here we show a counterpart for (b):

Theorem 7. *There is an oracle set H such that* $\mathrm{DEXT}(H) \neq \mathrm{NEXT}(H)$ *but* $\mathrm{DEPT}(H) = \mathrm{NEPT}(H)$.

Proof. As before, H will be constructed by stages. Proof is less complicated than that of Theorem 1 or 3. So we directly state the definition of H. Let $H(s)$ be the set of all strings placed into H before stage s and let $H(0) = \emptyset$. At some stages we reserve some strings for $\bar{H}$; and the index e of each DE_e is cancelled at some stage q_e when we ensure that DE_e^H does not accept the language

$$L_{EX}(H) = \{0^n : \exists u \in H[|u| = 2^n]\}.$$

Clearly $L_{EX}(H)$ is in NEXT(H).

Stage $2s$. Consider strings y such that

(1) $y = 0^i 1 x 1 0^n$ and $2s = |y| = 2^{p_i(|x|)}$ for some $i, n \in \omega$ and $x \in \Sigma^*$.

Run $NEP_i^{H(2s)}$ on x. If it accepts x, then we take a string yw such that

(2) $|w| = p_i(|x|)^2 + \varepsilon$, where $\varepsilon = 0$ or 1,

(3) $|yw|$ is odd and

(4) yw is not reserved for $\bar{H}$, yet.

And we add yw to H. Such a string yw exists. (See Claim 1 below.) If $NEP_i^{H(2s)}$ rejects x we do nothing. Let $H'(2s+1)$ be the set of all strings placed into H at this stage after performing the above procedure for every y for which (1) holds, and set $H(2s+1) = H(2s) \cup H'(2s+1)$. If there is no such y let $H(2s+1) = H(2s)$.

Stage $2s+1$. Let e be the first uncancelled index at the beginning of this stage. Suppose the following 4 conditions are satisfied:

(i) $2s+1 > g_{e-1}(\log q_{e-1})$ (If $e=0$ put the right-hand-side $=0$).

(ii) $\log(2s+2)$ is an even number,

(iii) There is no string of length larger than $2s+1$ which is reserved for $\bar{H}$

before this stage,

(iv) $g_e(\log(2s+2))<2^{(\log(2s+2))^2}<2^{2s+2}$.

Then run $DE_e^{H(2s+1)}$ on the string $z_{2s+1}=0^{\log(2s+2)}$. We reserve for $\bar{H}$ all strings of lengths larger than $2s+1$ negatively queried in the computation. If $DE_e^{H(2s+1)}$ rejects z_{2s+1}, then we choose a string u of length $2s+2$ not queried in the computation and add it to $H(2s+1)$ to make $H(2s+2)$. By (iv), such a string u exists. If $DE_e^{H(2s+1)}$ accepts z_{2s+1} let $H(2s+2)=H(2s+1)$. In either case we cancel the index e and let $q_e=2s+1$. If at least one of (i)–(iv) is not satisfied, then we do nothing and put $H(2s+2)=H(2s+1)$. Clearly every index e will eventually be cancelled.

Let $H=\bigcup_{s=0}^{\infty} H(s)$. H is a desired oracle set. To prove this, we note the following: When an odd stage $2s+1$ is executed, the following condition holds:

(v) For any ε, y, i, x and n, if $\varepsilon=0$ or 1 and $y=0^i1x10^n$ and $2s+1\leq|y|\leq 2^{p_i(|x|)}$, then there is a string w such that $|w|=p_i(|x|)^2+\varepsilon$ and $|yw|$ is odd and $DE_e^{H(2s+1)}$ does not query yw in the computation on z_{2s+1}.

This is easily proved by using (iv). Next,

Claim 1. When an even stage $2s$ is executed, such a yw exists and this string is not queried in any computation performed at any earlier stage. [Proof. Let $y=0^i1x10^n$ with $2s=|y|=2^{p_i(|x|)}$ be any string taken at stage $2s$. Let $2s'+1$ be the last odd stage executed before $2s$, and let e be the cancelled index at this $2s'+1$. By (v) with $s=s'$, there is a string w such that $|w|=p_i(|x|)^2+\varepsilon$, $|yw|$ is odd and $DE_e^{H(2s'+1)}$ does not query yw in the computation on $z_{2s'+1}$. Since only strings of lengths less than $2s'+1$ are queried at odd stages $<2s'+1$ (because of (i) with $s=s'$), yw is not reserved for $\bar{H}$ yet. Moreover, only strings of lengths less than $2s$ are queried at even stages $<2s$. So yw is not queried at any earlier stage.]

And, when an odd stage $2s+1$ is executed, the chosen string u is not queried at any earlier stage. This is because of (iii). So, as before, we see that $L_{EX}(H)$ is not in DEXT(H). Finally, we show NEPT(H) is contained in DEPT(H). Let L be in NEPT(H) and let i be such that $L=T(NEP_i^H)$. We define a deterministic 2^{poly}-time-bounded OTM M with oracle H which accepts L. Given x, M first produces a string y such that

$$y=0^i1x10^n \& |y|=2^{p_i(|x|)}. \tag{5}$$

Let $|y|=2s$. Then M produces a string w such that $|w|=p_i(|x|)^2+\varepsilon$ ($\varepsilon=0$ or 1) and such that $|yw|$ is odd. M accepts x iff yw is in H. So, by Claim 1, M

accepts x iff $NEP_i^{H(2s)}$ accepts x iff NEP_i^H accepts x iff x is in L. Hence L is accepted by M^H. Guessing such a string w can deterministically be done in fewer than $c \cdot 2^{p_i(|x|)^2+1}$ steps for some constant c. So, by using oracle H M can deterministically decide whether it accepts x in fewer than $2^{p(|x|)}$ setps for some polynomial p. Hence L is in DEPT(H). Consequently NEPT(H) is contained in DEPT(H). □

A language on a one-letter alphabet is called a tally language. It is known, by Book, that if DEXT = NEXT then P can not be separated from NP by any tally language. Constrast with this, we have:

Corollary. *There is an oracle set H such that* DEPT(H) = NEPT(H) *but* DEXT(H) *is separated from* NEXT(H) *by a tally language.*

References

[1] T. Baker, J. Gill and R. Solovay, Relativizations of P = ? NP quesetion, *SIAM J. Comput.*, 4 (1975), 431–442.
[2] R. Book, Compairing complexity classes, *J. Comput. Syst. Sci.*, 9 (1974), 213–229.
[3] R. Book, C. Wilson and Xu Mei-Rui, Relativizing time, space, and time-space, *SIAM J. Comput.*, 11 (1982), 571–581.
[4] C. Kintala and P. Fischer, Refining nondeterminism in relativized polynomial-time bounded computations, *Ibid.* 9 (1980), 46–53.
[5] W. J. Paul, N. Pippenger, E. Szemerédi and W. T. Trotter, On determinism versus non-determinism and related problems, *Proc. 24th Annual Symosium on Foundations of Computer Science*, (1983), 429–438.