Theoretical Computer Science 75 (1990) 335-346
North-Holland

NOTE

DOWNWARD TRANSLATIONS OF EQUALITY

Eric ALLENDER*
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, U.S.A.

Christopher WILSON**
Department of Computer and Information Science, University of Oregon, Eugene, OR 97403, U.S.A.

Communicated by J. Diaz
Received April 1989
Revised October 1989

Abstract. In this paper we construct oracles relative to which DTIME($T(n)$) equals NTIME($T(n)$) and DTIME($t(n)$) does not equal NTIME($t(n)$), for $t(n)$ sufficiently smaller than $T(n)$. A stronger result than this is also obtained, though for fewer $T(n)$, expressed in two parts. For $T(n) \leq 2^n$, there is an oracle relative to which DTIME($T(n)$) equals NTIME($T(n)$) and NTIME($2n$) contains a set not in DTIME($t(n)$) for any $t(n)$ growing more slowly than $T(n)$. For $T(n) \leq 2^{2^{n+O(1)}}$, there is an oracle relative to which DTIME($T(n)$) equals NTIME($T(n)$) and NTIME(log $T(n)$) contains a set not in DTIME($t(n)$) for $t(n)$ growing more slowly than $T(n)$. These results expand on those obtained by Dekhtyar (1976), Book et al. (1982), and Allender (1989).

1. Introduction

A common feature of complexity classes is that equality of two classes generally implies equality of classes at some corresponding higher level. For example, if P = NP, then E = NE follows directly from a padding argument. A natural question, then, is the converse. If we know that E = NE, does this necessarily imply that P = NP? If not, what can be concluded?

There have been various approaches to this problem. In [3] it was shown that E = NE if and only if NP − P contain no tally sets; [6] shows that E = NE if and only if NP − P contains no sparse sets.

Another approach to the issue has been the use of relativization. In [5] and [4], there are exhibited oracles relative to which P ≠ NP and E = NE. Incidentally, [9] constructs an oracle relative to which P ≠ NP and NP − P contains no sparse sets.

* Supported in part by National Science Foundation Research Initiation Grant number CCR-8810467.
** Supported in part by National Science Foundation Grant number CCR-8810051.

Since the methods of [6] relativize, this is seen to be equivalent to the results presented in [5] and [4]. Related oracle constructions can be found in [11].

The techniques in the papers above have not been applicable to complexity classes above the exponential level. In [1], however, there is an oracle relative to which $E \neq NE$ and $EE = NEE$, where EE refers to doubly exponential time. Thus, we have two relativized situations where there is equality at a high level and inequality at a lower one. These results imply that any proof of $E = NE \Rightarrow P = NP$ or $EE = NEE \Rightarrow E = NE$ will require a proof technique which will not relativize, as most standard techniques do.

The results of this paper address the extent to which the oracle constructions mentioned above can be generalized and strengthened. The main result of this paper is that if $t(n)$ is sufficiently smaller than $T(n)$, then there is an oracle relative to which $\mathrm{DTIME}(t(n)) \neq \mathrm{NTIME}(t(n))$ and $\mathrm{DTIME}(T(n)) = \mathrm{NTIME}(T(n))$. It is interesting that two different constructions are required, depending on the growth rate of $T(n)$. If $T(n) = O(2^n)$, then we can generalize the construction of [4]. However, if $T(n) = \omega(2^n)$, then we adapt the construction of [1]. Neither technique alone appears appropriate for the entire range of time bounds $T(n)$.

We are able to obtain stronger results as well. Suppose $\mathrm{DTIME}(T(n))$ equals $\mathrm{NTIME}(T(n))$ and $t(n)$ is smaller than $T(n)$ (the precise notions of smaller growth rates will be formalized later). Clearly $\mathrm{NTIME}(\log t(n)) \subseteq \mathrm{DTIME}(t(n))$, but it is not a priori obvious that $\mathrm{NTIME}(\log T(n)) \subseteq \mathrm{DTIME}(t(n))$. We construct oracles which show that such containment need not be the case. If $T(n) \leqslant 2^n$, then there is an oracle for which $\mathrm{DTIME}(T(n))$ equals $\mathrm{NTIME}(T(n))$ and $\mathrm{NTIME}(2n) - \mathrm{DTIME}(t(n))$ is not empty. If $T(n) \geqslant 2^{2^{n+O(1)}}$, then there is an oracle relative to which $\mathrm{DTIME}(T(n))$ equals $\mathrm{NTIME}(T(n))$ and $\mathrm{NTIME}(\log T(n))$ contains a set not in $\mathrm{DTIME}(t(n))$.

An interesting gap has appeared in the range of $T(n)$ for which we can show the stronger result. Our techniques do not hold for $2^n < T(n) \leqslant 2^{2^n}$. For example, it is open whether there is an oracle where $\mathrm{DTIME}(2^{n^2}) = \mathrm{NTIME}(2^{n^2})$ and $\mathrm{NTIME}(n^2) - E$ is not empty. We conjecture that such an oracle exists. However, at this point, it is still conceivable that conventional (relativizable) techniques could show that $\mathrm{DTIME}(2^{n^2}) = \mathrm{NTIME}(2^{n^2})$ implies that $\mathrm{NTIME}(n^2) \subseteq E$.

2. Preliminaries

It is expected that the reader will be familiar with basic concepts from complexity theory, such as Turing machines, oracle Turing machines, and complexity classes such as P, NP, etc. For background and definitions, see e.g. [7, 16, 2]. We will use E and NE to refer to $\mathrm{DTIME}(2^{O(n)})$ and $\mathrm{NTIME}(2^{O(n)})$, respectively. EE and NEE denote $\mathrm{DTIME}(2^{O(2^n)})$ and $\mathrm{NTIME}(2^{O(2^n)})$, respectively.

For any string x, the length of x is denoted by $|x|$. For any set S, $\|S\|$ denotes the cardinality of S. All languages considered in this paper are subsets of $\{0, 1\}^*$.

Given any sets A and B, the set $\{1x: x \in A\} \cup \{0x: x \in B\}$ is denoted by $A \oplus B$.

We make frequent use of "Big Oh" notation, and follow the conventions of [8]. Thus, for example, $f(n) = \omega(g(n)) \Leftrightarrow \forall c\, [f(n) > cg(n)$ for all large $n]$, and $f(n) = o(g(n)) \Leftrightarrow \forall c\, [cf(n) < g(n)$ for almost all $n]$.

Some of the proofs in this paper make use of some simple notions from Kolmogorov complexity theory. The definitions and facts relating to Kolmogorov complexity which are used in this paper are standard and elementary; background information can be obtained e.g. from the introduction to [13]. More extensive background and an excellent history of the area can be found in [10]. We will let $K(x)$ denote the Kolmogorov complexity of x (i.e., the length of its shortest description on some universal Turing machine).

Although the model of oracle Turing machine which we use is standard [7, 16, 2], there is one peculiarity of this formalism which needs to be pointed out. Namely, it is well known that the Linear Speed-Up Theorem (see e.g., [7, 2]) does not hold for this model [12, 15]. The Linear Speed-Up Theorem requires the tape alphabet to grow in order to achieve speed-up. However, there is no way to compress the oracle tape. Thus, for instance, it is easy to show that $\{x: x0^{3|x|} \in A\}$ is in $\mathrm{DTIME}(4n) - \mathrm{DTIME}(3n)$ relative to almost all oracles A. In [12] there is presented an oracle access mechanism under which the Linear Speed-Up Theorem does hold.

This quality of the oracle Turing machine model makes it somewhat awkward to talk about relativized time classes $\mathrm{DTIME}^A(T(n))$, since, for instance, it is quite possible that there are sets recognizable in time $T(n)+1$ relative to oracle A, but which cannot be recognized in time $T(n)$. In order to simplify the statement of our results, we find it useful to define relativized time-complexity classes in the following slightly nonstandard way.

Definition 2.1. For any oracle A and any integer function T, $\mathrm{DTIME}^A(T(n))$ denotes the class of languages L for which there is a deterministic oracle Turing machine M and a constant c such that $L = L(M^A)$ and on all inputs x of length n, M runs for at most $T(n)+c$ steps on input x with oracle A. $\mathrm{NTIME}^A(T(n))$ is defined similarly.

3. Main results

Theorem 3.1. *If $t(n)$ is a function and $T(n)$ is a time constructible monotone function such that $8n \leq T(n) \leq 2^n$ and $T(\frac{1}{4}n - \frac{1}{2}) - t(n) = \omega(1)$, then there exists an oracle A such that $\mathrm{NTIME}^A(2n) - \mathrm{DTIME}^A(t(n)) \neq \emptyset$ and $\mathrm{DTIME}^A(T(n)) = \mathrm{NTIME}^A(T(n))$.*

Proof. The oracle A will be constructed in stages. At various points, strings will be reserved for (i.e., placed into) A or $\bar{A}$. Once a string is reserved, its status will never change. Occasionally we will simulate a machine that will query the partially

constructed oracle, and if an unreserved string is queried, the oracle will answer *no* to that string.

An index $l = 10^i 10^c$ will be cancelled when we ensure that the deterministic oracle Turing machine P_i^A does not accept $L(A)$ in time $t(n)+c$. Initially, all indices are uncancelled. We also let M_i be an enumeration of all nondeterministic oracle Turing machines.

In the proof we use a diagonal set $L(A)$ and a complete set $S(A)$, both based on the oracle A. They are defined as follows:

$$L(A) = \{x: \exists y\, |y| = |x|, xy \in A\},$$

$$S(A) = \{x10^i10^c: M_i^A \text{ accepts } x \text{ in} \leqslant T(|x|)+c \text{ steps}\}.$$

We will separate the stages of the construction into odd and even stages. At
- *even stages* we ensure that $L(A) \notin \mathrm{DTIME}^A(t(n))$, and at
- *odd stages* we ensure that $y = x10^i10^c \in S(A) \Leftrightarrow y10^{T(|x|)-|x|} \in A$.

Since $L(A) \in \mathrm{NTIME}^A(2n)$ for any A, the even stages ensure that

$$\mathrm{NTIME}^A(2n) - \mathrm{DTIME}^A(t(n)) \neq \emptyset.$$

The odd stages guarantee that

$$\mathrm{DTIME}^A(T(n)) = \mathrm{NTIME}^A(T(n))$$

via the following argument. Suppose that $L \in \mathrm{NTIME}^A(T(n))$. Then there are i and c so that M_i^A accepts L within $T(n)+c$ steps. By construction, $x \in L$ if and only if $x10^i10^c10^{T(|x|)-|x|} \in A$. This query can be made in deterministic time $T(|x|)+i+c+3$ since T is constructible. Since i and c are constant, $L \in \mathrm{DTIME}^A(T(n))$.

The stages of the construction are numbered $n = 0, 1, 2, \ldots$ Stage 0 is an initialization stage where we set A to be $\emptyset$. We perform only those odd stages of the form $n = 4k+1$. At even stages $n = 4k+2$ something is done only if certain preconditions are met. Nothing will be done at a stage $n = 4k$ or $4k+3$.

Construction of A

Stage 0: $A \leftarrow \emptyset$.

Even Stage $n = 4k+2$: Find the least uncancelled index $l = 10^i10^c$. In order to cancel l, it will suffice to consider the run-time of P_i to be $t(n)+c$. If the following four preconditions are satisfied, then we will cancel index l.

(C1) $t(n)+c$ is less than $T(\frac{1}{4}(n+2)-1)+1$. (Note that $T(\frac{1}{4}(n+2)-1)+1$ is the length of the shortest string of the form $y10^*$ which may have to be left unreserved for the next odd stage.)

(C2) $2n < T(\frac{1}{4}(n+2)-1)+1$.

(C3) $2n$ is greater than the length of the longest string queried or reserved at any earlier even stage.

(C4) $t(n)+c < 2^n$

To cancel index l, find an $x(|x|=n)$ such that for no $y(|y|=|x|)$ has xy been reserved. Run P_i^A on x for $t(n)+c$ steps and place all unreserved strings queried into $\bar{A}$. If P_i^A accepts x, then put all xy $(|y|=|x|)$ into $\bar{A}$ (thus $x \notin L(A)$). Otherwise, put some unqueried xy $(|y|=|x|)$ into A (thus $x \in L(A)$).

Odd Stage $n=4k+1$: For each string y of the form $x10^i10^c$ satisfying $T(k-1)< T(|x|)+i+c+3 \leq T(k)$, simulate M_i^A on x as described below. Handle these strings y in order of increasing $T(|x|)+i+c+3$.

Run M_i^A on x for $T(|x|)+c$ steps. If it accepts, then put $y10^{T(|x|)-|x|}$ into A and reserve for $\bar{A}$ all unreserved strings queried on one accepting path.

If it rejects, then see if adding at most $T(|x|)+c<T(k)$ strings to A will force M_i^A to accept x. If so, then reserve those strings for A, reserve $y10^{T(|x|)-|x|}$ for A, and reserve the remaining unreserved strings on some accepting path for $\bar{A}$. Otherwise, the behavior of M_i^A is independent of later changes to the oracle set, so reserve $y10^{T(|x|)-|x|}$ for $\bar{A}$.

end construction

We now have to prove that the construction is possible.

Point 1: *An odd stage does not impede the construction at the next odd stage.* At odd stage $4k+1$, only strings of length at most $T(k)$ are queried and reserved by the simulation; similarly only strings of length at most $T(k)$ are reserved by the encoding. At the next odd stage $4(k+1)+1$ performed, only strings of length greater than $T(k)$ will be considered for membership in A. None of these strings will have been reserved in stage $4k+1$.

Point 2: *During any odd stage* $n=4k+1$, *fewer than* 2^n *strings are reserved.* Any $y=x10^i10^c$ causes at most $T(|x|)+c<T(k)$ strings to be reserved. For any j, there are at most 2^j strings x of length j. There are at most $(T(k))^2$ possible strings 0^i10^c. An upper bound on the number of strings reserved at this stage is thus

$$\sum_{j=0}^{k} T(k)2^j(T(k))^2<(T(k))^3 2^{k+1} \leq (2^k)^3 2^{k+1}=2^{4k+1}=2^n.$$

Point 3. *At an even numbered stage* $n=4k+2$, *fewer than* 2^n *strings have been reserved by previous odd stages.* This follows from Point 2 and noticing that

$$\sum_{i=0}^{k} 2^{4i+1} < \sum_{j=0}^{4k+1} 2^j = 2^{4k+2}-1=2^n-1.$$

Point 4: *An even stage does not reserve any* $y0^{T(|x|)-|x|}$ *which will be examined (and possibly reserved) at the next odd stage.* This follows from preconditions C1 and C2.

Point 5: *The construction at the odd stages is possible.* Because of Points 1 and 4. Notice also that the strings are processed in order of nondecreasing $T(|x|)+i+c+3$. This prevents the simulation initiated by $y=x10^c10^i$ from reserving $\hat{y}10^{T(|\hat{x}|)-|\hat{x}|}$, for a different $\hat{y}$, which may need to be reserved for A later during *this* stage. This is since

$$T(|x|)+c<T(|x|)+i+c+3 \leq T(|\hat{x}|)+\hat{i}+\hat{c}+3$$

for any $\hat{y}=\hat{x}10^{\hat{i}10^{\hat{c}}}$ dealt with later. Further, the simulation initiated by y cannot reserve $y10^{T(|x|)-|x|}$.

Point 6. An even stage does not affect the construction at the next even stage. By precondition C3.

It remains to prove that the construction at the even stages is possible. Now only the odd stages cause concern, by Point 6. So we must show that at stage $n=4k+2$, we can always find some x such that for no y has xy been reserved. There are 2^n different sets $H(x)=\{xy: |x|=|y|=n\}$. By Point 3, the number of strings reserved by all previous even stages is less than 2^n. Thus there must exist an x of length n for which no member of $H(x)$ is reserved.

Once such an x is found, we may still need to find some xy not queried at the end of the stage (this xy may be placed in A). The number of strings reserved at this stage is, for some c, at most $t(n)+c<2^n=\|H(x)\|$, by precondition C4. So there will always exist a y for which xy is not reserved.

Finally, note that for each index i, the preconditions C1-C4 will be satisfied infinitely often, so every index will eventually be cancelled. The eventual satisfaction of C1 and C4 follow from the assumptions $T(\frac{1}{4}n-\frac{1}{2})-t(n)=\omega(1)$ and $T(n)\leqslant 2^n$. C2 follows since $T(n)\geqslant 8n$.

Notes on the proof. (1) The conditions on $T(n)$, such as $T(n)\leqslant 2^n$, need only hold asymptotically. That is, we need only that for some N and all $n\geqslant N$, $T(n)\leqslant 2^n$. The construction would then start at stage $n=4N+1$. The proof that $\mathrm{DTIME}^A(T(n))=\mathrm{NTIME}^A(T(n))$ would involve a patching lemma whereby a deterministic machine simulating a non-deterministic one would use a table look-up for short strings (length $\leqslant N$).

(2) The separation of NTIME($2n$) from a deterministic class can be shown for a wider class of time bounds than those of the form $t(n)+O(1)$. In particular, let F be a countable class of functions such that $\forall t\in F$, $T(\frac{1}{4}n-\frac{1}{2})-t(n)=\omega(1)$. Then we can build A so that $\mathrm{NTIME}^A(2n)$ contains a set not in $\bigcup_{t\in F}\mathrm{DTIME}^A(t(n))$. If F is a class of recursive functions and F is recursively enumerable (as a set of indices), then A is recursive. □

Corollary 3.2. *There is an oracle A such that*

$$\mathrm{E}^A=\mathrm{NE}^A \quad \textit{and} \quad \mathrm{NTIME}^A(2n)-\mathrm{DTIME}^A(2^{n/5})\neq\emptyset.$$

Proof. Follows from Theorem 3.1 with $T(n)=2^n$. It follows by a straightforward padding argument that $\mathrm{DTIME}^A(2^n)=\mathrm{NTIME}^A(2^n)$ implies that $\mathrm{E}^A=\mathrm{NE}^A$. □

Corollary 3.3 (Dekhtyar [5], Book et al. [4]). *There is an oracle A such that $\mathrm{P}^A\neq\mathrm{NP}^A$ and $\mathrm{E}^A=\mathrm{NE}^A$.*

Corollary 3.4. *There is an oracle A such that*

$$\mathrm{DTIME}^A(2^{\text{poly-log}})\neq\mathrm{NTIME}^A(2^{\text{poly-log}}) \quad \textit{and} \quad \mathrm{E}^A=\mathrm{NE}^A.$$

Corollary 3.5. *There is an oracle A such that*

$$\mathrm{DTIME}^A(\mathrm{O}(n^1)) \neq \mathrm{NTIME}^A(\mathrm{O}(n)) \quad \textit{and} \quad \mathrm{DTIME}^A(n^2) = \mathrm{NTIME}^A(n^2).$$

Corollary 3.6. *There is an oracle A such that*

$$\mathrm{DTIME}^A(2n) \neq \mathrm{NTIME}^A(2n) \quad \textit{and} \quad \mathrm{DTIME}^A(9n) = \mathrm{NTIME}^A(9n).$$

In [14] there is proof of the fact that in the unrelativized case DTIME(O(n))) does not equal NTIME(O(n)). Since it is not too hard to construct an oracle for which these two classes are equal, the result in [14] provides an excellent example of a proof technique which does not relativize. Corollary 3.5 indicates that the inequality in [14] does not easily translate upwards. Thus, it will require another nonrelativizing technique to prove that deterministic and nondeterministic quadratic time differ.

The proof of the previous theorem alternates between encoding stages and diagonalization stages. The encoding stages ensure that the higher classes are equal, while the diagonalization guarantees the separation at the lower level. If $T(n)$ were larger than 2^n, the stages would interfere with each other. To handle the larger time bounds, the encoding is done all at once, and the diagonalization is against all possible encodings.

Theorem 3.7. *Let $T(n)$ and $t(n)$ be nondecreasing time-constructible functions such that $T(n) = \omega(2^n)$ and $\forall k\ 3n \leq t(n) = \mathrm{o}(T(n-k))$. Then there exists an oracle B relative to which* DTIME($T(n)$) = NTIME($T(n)$) *and* NTIME($t(n)$) *is not contained in* DTIME(O($t(n)$)).

Proof. First, we claim that, under the conditions given above, there is a slowly growing function s satisfying certain properties.

Claim 3.8. *There is a monotone nondecreasing, unbounded function s such that*

$$\forall j \quad [T(n - s(j \cdot t(n))) > j \cdot t(n) \textit{ for all large } n], \tag{1}$$

$$\forall j \quad [(s(jn))^2 2^{T^{-1}(jn)} \leq \tfrac{1}{2}n \textit{ for all large } n]. \tag{2}$$

(where by "T^{-1}" we mean the function which maps n to the largest m such that $T(m) \leq n$).

Although Claim 3.8 is very elementary, its proof is lengthy, and is given in the Appendix. It should be mentioned that the function s need not be recursive.

Let $r(n) = \min(m\colon s(m) \geq n\}$; note that r is essentially s^{-1}. Let $M_1, M_2, \ldots$ be an enumeration of nondeterministic Turing machines, and let $P_1, P_2, \ldots$ be an enumeration of deterministic Turing machines such that P_j runs in time $j \cdot t(n)$.

For any set $A \subseteq \Sigma^*$, define

$$Q(A) = \{x10^{r(i)}1^{r(c)}0^{T(|x|)-|x|}: M_i^{A \oplus Q(A)} \text{ accepts } x \text{ in } \leq T(|x|)+c \text{ steps}\}.$$

Note that $Q(A)$ is well-defined, since for any y of the form $x10^{r(i)}1^{r(c)}0^{T(|x|)-|x|}$ it follows that $|y| > T(|x|)+c$, and thus M_j cannot query y in $T(|x|)+c$ steps.

We claim that for every set A, $\text{DTIME}^{A \oplus Q(A)}(T(n)) = \text{NTIME}^{A \oplus Q(A)}(T(n))$. To see this, let L be accepted by some machine M_i in time $T(n)+c$ with oracle $A \oplus Q(A)$. Then $x \in L \Leftrightarrow x10^{r(i)}1^{r(c)}0^{T(|x|)-|x|} \in Q(A)$. This query can be asked in time $T(|x|)+r(i)+r(c)+2$. Since i and c are constants depending only on L, it follows that L can be recognized in time $T(n)+O(1)$, and thus is in $\text{DTIME}^{A \oplus Q(A)}(T(n))$.

Let $L(A) = \{x: \exists y \ [xy \in A \text{ and } |y| = t(|x|) - |x|]\}$. For all sets A, $L(A) \in \text{NTIME}^{A \oplus Q(A)}(t(n))$. It will thus suffice to build a set A such that for all j, the set accepted by P_j with oracle $A \oplus Q(A)$ differs from $L(A)$. For each j, we will build a finite oracle A_j so that $A_{j-1} \subseteq A_j$, and we will choose a number $n = n_j$ so that

- there is a set $C \subseteq \Sigma^{t(n)}$ such that $A_j = A_{j-1} \cup C$.
- The language accepted by P_j with oracle $A_j \oplus Q(A_j)$ differs from $L(A_j)$ on some string of length n.

In addition, n_{j+1} will be chosen to be larger than $j \cdot t(n_j)$, so that all queries by P_j to the oracle $A_{j+1} \oplus Q(A_{j+1})$ on inputs of length n will be answered the same as they are with oracle $A_j \oplus Q(A_j)$. Thus the diagonalization performed during stage j is not damaged by later stages of the construction. The oracle A will be $\bigcup_j A_j$.

Let κ be a large constant. At a certain point in the proof, we will use the fact that κ is larger than the description of some simple Turing machines.

Construction of A

Stage 0: $A_0 \leftarrow \emptyset$.

Stage j: To construct A_j ($j \geq 1$), first choose $n = n_j$ so that

$$j \cdot t(n) \leq T(n-1-2\log s(j \cdot t(n))), \tag{3}$$

$$s(j \cdot t(n))^2 2^{T^{-1}(j \cdot t(n))} \geq \tfrac{1}{2}t(n), \tag{4}$$

$$j \cdot t(n_{j-1}) < n, \tag{5}$$

$$\tfrac{1}{2}t(n) > (6+2s(j \cdot t(n)))\log(j \cdot t(n)) + n + 2\log n + 4 \cdot 2^{t(n_{j-1})} + 2\log j + 2\kappa. \tag{6}$$

(Note that all large n satisfy all of these criteria, by (1) and (2), and by the fact that $t(n) \geq 3n$.) Define

$$Q = \{z: |z| \leq j \cdot t(n) \text{ and } z \text{ is of the form } y10^{r(i)}1^{r(c)}0^{T(|y|)-|y|} \text{ for some } y, i \text{ and } c\}.$$

When P_j computes on inputs of length n, all queries to the $Q(A)$ part of the oracle will be answered negatively for queries not in Q. That is, knowing membership in

$Q(A)$ for all elements of Q is sufficient to enable us to answer all queries to the $Q(A)$ part of the oracle. Note that $\|Q\| \leq s(j \cdot t(n))^2 2^{T^{-1}(j \cdot t(n))}$. By (3), $\|Q\| \leq 2^{n-1}$. By (4), $\|Q\| \leq \frac{1}{2}t(n)$.

An element $q \in \{0,1\}^{\|Q\|}$ will be called a Q-vector. We will write $P_j^{A \oplus q}(x)$ to denote the outcome of the computation of P_j on input x, where all queries to the $Q(A)$ part of the oracle are answered according to the vector q. That is, if P_j asks the $Q(A)$ part of its oracle about a string not in Q, the oracle responds negatively, and if P_j asks about the ith element of Q, then the oracle responds with the ith bit of q.

Let $W = \{w0^{n-|w|}: |w| = \lceil \log \|Q\| \rceil + 1\}$. Since $\|Q\| \leq 2^{n-1}$, this definition makes sense. Since elements of W all end with a long string of trailing zeros, every element of W has a fairly short description. Later, we will make use of the fact that W is a set of strings of low Kolmogorov complexity; we will carry out our diagonalization by making P_j fail on W.

Given any Q-vector q, we associate with it a set $F(q) \subseteq W$ as follows:

$$F(q) = W \cap \{x: P_j^{A_{j-1} \oplus q}(x) = 1\}.$$

Since there are only $2^{\|Q\|} < 2^{\|W\|}$ Q-vectors, there must be some $B \subseteq W$ such that for all $q \in Q$, $F(q) \neq B$. Choose one such B, and let z be a string of length $t(n) - n$ of maximal Kolmogorov complexity (i.e., $K(z) \geq |z|$).

Let $A_j = A_{j-1} \cup \{wz: w \in B\}$, and let q be the Q-vector encoding $Q(A_j)$. Note that $W \cap L(A_j) = B \neq F(q) = (W \cap \{x: P_j^{A_{j-1} \oplus 1}(x) = 1\})$. Thus in order to show that P_j with oracle $A_j \oplus Q(A_j)$ does not accept $L(A_j)$, it will suffice to show that for all $x \in W$, $P_j^{A_{j-1} \oplus q}(x) = P_j^{A_j \oplus q}(x)$. If this can be shown, then for all $x \in W$, x is accepted by P_j with oracle $A_j \oplus Q(A_j) \Leftrightarrow x \in F(q)$, and thus the set accepted by P_j with oracle $A_j \oplus Q(A_j)$ differs from $L(A_j)$ on at least one element of W.

Claim 3.9. $\forall x \in W, P_j^{A_{j-1} \oplus q}(x) = P_j^{A_j \oplus q}(x)$.

Proof of claim. Since $A_{j-1} \oplus q$ and $A_j \oplus q$ differ only on the strings of the form wz for some strings w in W, the outcome of $P_j^{A_{j-1} \oplus q}(x)$ and $P_j^{A_j \oplus 1}(x)$ can only differ if some such string wz is queried on each of these computations. We will show that no such string can be queried.

Note that $K(z) \leq K(wz) + 2 \log n + \kappa$, so $K(wz) \geq (t(n) - n) - 2 \log n - \kappa$. Note also that any query asked during the computation of $P_j^{A_{j-1} \oplus q}(x)$ can be reconstructed from

- j;
- a description of A_{j-1} (which can be given by a bit vector of length $2^{t(n,j-1)+1} - 1$);
- q (which has length $\|Q\| \leq \frac{1}{2}t(n)$);
- a number between 1 and $j \cdot t(n)$, indicating the step at which the query is made;
- a description of x (which need be no longer than $2 \log n + \lceil \log \|Q\| \rceil + 1 < 3 \log t(n)$, since $x \in W$ and thus x is of the form $u0^{n-|u|}$ for some u of length $\|Q\| + 1$);

- a description of the values of $r(1), r(2), \ldots, r(s(j \cdot t(n)))$ (this sequence of $s(j \cdot t(n))$ values, each no greater than $j \cdot t(n)$, can be encoded using $2s(j \cdot t(n)) \log(j \cdot t(n))$ bits).

We can now conclude that the Kolmogorov complexity of any query is at most $\frac{1}{2}t(n)+2s(j \cdot t(n)) \log(j \cdot t(n))+6 \log(j \cdot t(n))+\kappa+2 \log j+4 \cdot 2^{t(n_{l-1})}$. By (6), this is less than $(t(n)-n)-2 \log n-\kappa \leq K(wz)$. Thus no such string wz is queried. □

Using an essentially identical proof, one can also prove the following result.

Theorem 3.10. *Let $T(n)$ and $t(n)$ be nondecreasing time-constructible functions such that $\forall k\,[t(n)<T(\log \log T(n)-k)$ a.e.], and $\exists \delta>1\ [t(n)^{\delta}<T(n)]$. Then there exists an oracle B relative to which* $\mathrm{DTIME}(T(n))=\mathrm{NTIME}(T(n))$ *and* $\mathrm{NTIME}(\log T(n))$ *is not contained in* $\mathrm{DTIME}(t(n))$.

Corollary 3.11. *Let $T(n)$ and $t(n)$ be nondecreasing time-constructible functions such that $\forall k\ [T(n)>2^{2^{n+k}}$ a.e.] and $\exists \delta>1\ [t(n)^{\delta}<T(n)]$. Then there exists an oracle B relative to which* $\mathrm{DTIME}(T(n))=\mathrm{NTIME}(T(n))$ *and* $\mathrm{NTIME}(\log T(n))$ *is not contained in* $\mathrm{DTIME}(t(n))$.

4. Conclusion

In the results above, we have shown that in the relativized case, it can be true that equality of certain complexity classes does not imply equality of lower classes. Even stronger, we have exhibited an oracle relative to which $\mathrm{DTIME}(T(n))$ equals $\mathrm{NTIME}(T(n))$ while $\mathrm{NTIME}(t(n))$ contains a set not in $\mathrm{DTIME}(2n)$ so long as $t(n)$ is sufficiently less than $T(n)$ and $T(n) \leq 2^n$. Similarly, we have an oracle for which $\mathrm{DTIME}(t(n))$ equals $\mathrm{NTIME}(T(n))$ while $\mathrm{NTIME}(t(n))$ contains a set not in $\mathrm{DTIME}(\log T(n))$ so long as $t(n)$ is sufficiently less than $T(n)$ and $T(n) \geq 2^{2^{n+O(1)}}$. These results tell us that in almost all cases, a very strong statement of inequality involving complexity classes of small time bounds will not yield an inequality of complexity classes of large time bounds, at least not without using a proof technique immune to relativization.

The main open question is to close the gap in the bounds on $T(n)$ for the stronger result, when $2^n<T(n) \leq 2^{2^n}$. For example, is there an oracle for which $\mathrm{DTIME}(2^{n^2})=\mathrm{NTIME}(2^{n^2})$ and $\mathrm{NTIME}(n^2)-\mathrm{E}$ is not empty?

Appendix

Here, we present the proof of Claim 3.8. It is convenient to break this into two parts:

(1) Let $T(n)=\omega(2^n)$. Then there is a monotone, unbounded function f such that, for all j, $(f(jn)2^{T^{-1}(jn)} \leq \frac{1}{2}n$ for all large $n)$.

(2) Assume $\forall k\, T(n-k)=\omega(t(n))$. Then there is a monotone, unbounded function g such that, for all j, $(T(n-g(j\cdot t(n)))\geq j\cdot t(n)$ for all large n).
The claim follows by taking $s(n)=\min\{\sqrt{f(n)}, g(n)\}$.

We prove the first claim first. By assumption, for all c, $T(n)>c2^n$ for all large n. Thus $T^{-1}(n)\leq \log(n/c)$ for all large n, so we have that for all j and c, $2^{T^{-1}(jn)}\leq (j/c)n$ for all large n.

Define $f_j(n)=\max\{0\}\cup\{b\colon \forall m\geq n\, b2^{T^{-1}(jm)}\leq \frac{1}{2}m\}$. Clearly, each f_j is nondecreasing and unbounded, and $j_1<j_2\Rightarrow f_{j_1}\geq f_{j_2}$. Define f and i recursively as follows:

$f(0)=0$, $i(0)=0$.
for $n\geq 1$ (let $j=i(n-1)$).
  if $f_{j+1}(n/(j+1))>f(n-1)$
    then $i(n):=j+1$, and $f(n):=f_{j+1}(n/(j+1))$
    else $i(n):=j$ and $f(n):=f(n-1)$.

It is easily verified that f is nondecreasing and unbounded, and that for all $j, f(j\cdot n)\leq f_j(n)$ for all large n. This proves the first claim.

The second claim is proved in a very similar manner. By assumption we have that for all k and j, $T(n-k)>j(t(n))$ for all large n. Let $g_j(n)=\max\{0\}\cup\{k\colon \forall m\geq n\ T(m-k)\geq jt(m)\}$. Clearly, each g_j is nondecreasing and unbounded, and $j_1<j_2\Rightarrow g_{j_1}\geq g_{j_2}$. Define g and i recursively as follows:

$g(0)=0$, $i(0)=0$.
for $n\geq 1$, (let $j=i(n-1)$)
  if $g_{j+1}(t^{-1}(n/(j+1)))>g(n-1)$
    then $i(n):=j+1$, and $g(n):=g_{j+1}(t^{-1}(n/(j+1)))$
    else $i(n):=j$ and $g(n):=g(n-1)$.

As above, g is nondecreasing and unbounded and for all j, $g(j\cdot t(n))\leq g_j(n)$ for all large n. This proves the second claim. □

References

[1] E. Allender, Limitations of the upward separation technique (preliminary version), in: *Proc. 16th Internat. Colloquium on Automata, Languages, and Programming*, Lecture Notes in Computer Science **372** (Springer, Berlin, 1989) 18-30.

[2] J. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity I* (Springer, Berlin, 1988).

[3] R. Book, Talley languages and complexity classes, *Inform. and Control* **26** (1974) 186-193.

[4] R.V. Book, C.B. Wilson and M.R. Xu, Relativizing time, space, and time-space, *SIAM J. Comput.* **11** (1982) 571-581.

[5] M. Dekhtyar, On the relativization of deterministic and nondeterministic complexity classes, in: *Proc. 5th Symp. on Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science **45** (Springer, Berlin, 1976) 255-259.

[6] J. Hartmanis, N. Immerman and V. Sewelson, Sparse sets in NP-P: EXPTIME versus NEXPTIME, *Inform. and Control* **65** (1985) 158-181.

[7] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation* (Addison-Wesley, Reading, MA, 1979).

[8] D.E. Knuth, Big omicron and big omega and bit theta, *SIGACT News* **18**(2) (1976) 18-24.

[9] S. Kurtz, Sparse sets in NP−P: relativizations, *SIAM J. Comput.* **14** (1985) 113-119.

[10] M. Li and Paul Vitányi, Two decades of applied Kolmogorov complexity, in: *Proc. 3rd Structure in Complexity Theory Conf.* (IEEE, Washington, DC, 1988) 80-101.

[11] G. Lischke, Oracle-constructions to prove all possible relationships between relativizations of P, NP, EL, NEL, EP and NEP, *Z. Math. Logik Grundlag. Math.* **32** (1986) 257-270.

[12] S. Moran, Some results on relativized deterministic and nondeterministic time hierarchies, *J. Comput. System Sci.* **22** (1981) 1-8.

[13] W.J. Paul, On-line simulation of $k+1$ tapes by k tapes requires nonlinear time, *Inform. and Control* **53** (1982) 1-8.

[14] W.J. Paul, N. Pippenger, E. Szemeredi and W.T. Trotter, On nondeterminism versus determinism and related problems, in: *Proc. 24th Symp. on Foundations of Computer Science* (IEEE, Washington, DC, 1983) 429-438.

[15] C. Rackoff and J. Seiferas, Limitations on separating nondeterministic complexity classes, *SIAM J. Comput.* **10** (1981) 742-745.

[16] U. Schöning, *Complexity and Structure*, Lecture Notes in Computer Science **211** (Springer, Berlin, 1986).