Almost-Everywhere Complexity Hierarchies for Nondeterministic Time*

Eric Allender†
Department of Computer Science, Rutgers University
New Brunswick, NJ, USA 08903

Richard Beigel‡
Department of Computer Science, Yale University
51 Prospect St.
P. O. Box 2158, Yale Station
New Haven, CT, USA 06520-2158

Ulrich Hertrampf
Institut für Informatik, Universität Würzburg
D-8700 Würzburg, Federal Republic of Germany

Steven Homer§
Department of Computer Science, Boston University
Boston, MA, USA 02215

*A preliminary version of this work appeared as [2].

†Supported in part by National Science Foundation grant CCR-9000045. Some of this research was performed while the author was a visiting professor at Institut für Informatik, Universität Würzburg, D-8700 Würzburg, Federal Republic of Germany.

‡Supported in part by the National Science Foundation grants CCR-8808949 and CCR-8958528. Work done in part while at the Johns Hopkins University.

§Supported in part by National Science Foundation grants MIP-8608137 and CCR-8814339, National Security Agency grant MDA904-87-H, and a Fulbright-Hays research fellowship. Some of this research was performed while the author was a Guest Professor at Mathematisches Institut, Universität Heidelberg.

SUMMARY

We present an a.e. complexity hierarchy for nondeterministic time, and show that it is essentially the best result of this sort that can be proved using relativizable proof techniques.

1 Introduction

The hierarchy theorems for time and space are among the oldest and most basic results in complexity theory. One of the very first papers in the field of complexity theory was [14], in which it was shown that if $t(n)^2 = o(T(n))$, then DTIME($T(n)$) properly contains DTIME($t(n)$), for any time-constructible[1] functions t and T. This result was later improved by Hennie and Stearns [15], who proved a similar result, with "$t(n)^2$" replaced by "$t(n) \log t(n)$". Still later, Cook and Reckhow [8] and Fürer [10] proved tighter results for RAM's and Turing machines with a fixed number of tapes, respectively.

The time hierarchy theorems proved in this series of papers may be described as "infinitely often" (or "i.o.") time hierarchies, because they assert the existence of a set $L \in$ DTIME($T(n)$) that requires more than time $t(n)$ for infinitely many inputs. Note, however, that it may be the case that for all n, membership in L for ninety-nine percent of the inputs of length n can be decided in linear time. That is, the i.o. time hierarchy theorems assert the existence of sets that are hard infinitely often, although it is possible that the "hard inputs" for these sets form an extremely sparse set.

However it turns out that much stronger time hierarchy theorems can be proved. One can show the existence of sets L in DTIME($T(n)$) that require more than time $t(n)$ for *all* large inputs. This notion is called "almost everywhere" (or a.e.) complexity, and has

[1] A function T is *time-constructible* if there is a deterministic Turing machine that, for all inputs of length n, runs for exactly $T(n)$ steps. For the purposes of this paper, if T is time-constructible, then $T(n) \geq n$ for all n.

been investigated in [11, 12, 25]. The following definitions make this precise.

Definitions:

- For any Turing machine M, the partial function $T_M : \Sigma^* \to \mathbf{N}$ is the *time function of M*. $T_M(x)$ is the number of steps that M uses on input x if this number exists; $T_M(x)$ is undefined if M does not halt on input x.
- For functions f and g (mapping either Σ^* or $\mathbf{N}$ to $\mathbf{N}$), we say that $f = \omega(g)$ iff, for all $r > 0$ and for all but finitely many x, if $f(x)$ is defined, then $f(x) > r\,g(x)$.
- We say that an infinite set L is a.e. complex for DTIME(t) iff for all deterministic Turing machines M, $L(M) = L \Longrightarrow T_M(x) = \omega(t(|x|))$.

One of the a.e. time hierarchy theorems of [12] can now be stated:

Theorem 1 [12] If T is a time-constructible function and $t(n) \log t(n) = o(T(n))$, then there is a set in DTIME($T(n)$) that is a.e. complex for DTIME($t(n)$).

Thus for deterministic time complexity classes, the a.e. hierarchy is just as tight as the i.o. hierarchy. However much less is known about hierarchies for probabilistic and nondeterministic time.

The best (i.o.) time hierarchy theorem for probabilistic time that has appeared in the literature is due to Karpinski and Verbeek [16]; however they are only able to show that BPTIME($t(n)$) is properly contained in BPTIME($T(n)$) when $T(n)$ grows very much more quickly than $t(n)$. On the other hand, a recent oracle result of Fortnow and Sipser [9] shows that it is not possible to prove a very tight time hierarchy theorem for probabilistic time using relativizable techniques; they present an oracle according to which BPP = BPTIME($O(n)$). Clearly the [9] result also shows that there is no tight a.e. time hierarchy for probabilistic time that holds for all oracles.

An early i.o. hierarchy theorem for nondeterministic time was given by Cook [7]. The strongest such theorem that is currently known is due to [23]. It is shown there that if t and T are time-constructible functions such that $t(n+1) = o(T(n))$, then there is a set L in NTIME$(T(n))$ $-$ NTIME$(t(n))$. (Furthermore, it is shown in [27, 17] that L can even be chosen to be a subset of 0^*). When t and T are bounded by polynomials, this result is even tighter than the best known results for deterministic time. However, when T and t are very large, the gap between $t(n)$ and $t(n+1)$ is also quite large, and thus the nondeterministic time hierarchy seems not to be very tight. On the other hand, Rackoff and Seiferas show in [21] that the i.o. time hierarchy theorem for nondeterministic time cannot be improved significantly using relativizable techniques. For example, they present an oracle A such that $\mathrm{NTIME}^A(2^{2^n}) = \mathrm{NTIME}^A(2^{2^{n+1}}/\log^* n)$. (Note on the other hand that for *all* oracles A, $\mathrm{DTIME}^A(2^{2^n}) \neq \mathrm{DTIME}^A(2^{2^{n+1}}/\log^* n)$.)

Surprisingly, it seems that no results concerning an a.e. hierarchy for nondeterministic time have appeared. Geske, Huynh, and Seiferas explicitly raise the question of an a.e. hierarchy for nondeterministic time as an open problem in [12].

This paper essentially settles this question. We present an a.e. hierarchy theorem for nondeterministic time, and we present an oracle relative to which the given theorem cannot be significantly improved.

It is necessary to discuss the interpretation that should be afforded our oracle constructions, in light of the fact that compelling examples of nonrelativizing proof techniques do exist [3, 19, 24]. We maintain that it is useful to know what sort of hierarchies hold relative to *all* oracles. Note that many important complexity classes are defined in terms of relativized nondeterministic computation; the levels of the polynomial hierarchy are the most obvious examples. For example, although it is a standard fact that there are sets in $\mathrm{DTIME}^{\mathrm{SAT}}(n^3)$ that are a.e. complex for $\mathrm{DTIME}^{\mathrm{SAT}}(n^2)$ (where the proof of this fact makes no use of any properties of SAT), the results of this paper show that there are oracles Y such that $\mathrm{NTIME}^{Y}(n^3)$ contains no sets a.e. complex for $\mathrm{NTIME}^{Y}(n^2)$. Thus any proof that $\mathrm{NTIME}^{\mathrm{SAT}}(n^3)$ contains sets a.e. complex for $\mathrm{NTIME}^{\mathrm{SAT}}(n^2)$ will have to make essential use of certain properties of the oracle SAT. A longer discussion along these lines may be found in [1].

In this paper we consider only time complexity. An investigation of almost-everywhere complexity hierarchies for nondeterministic space has been carried out by Geske and Kakihara [13]. They prove almost-everywhere hierarchy theorems for nondeterministic space that are quite similar to those for deterministic space.

The organization of the paper is as follows. In Section 2 we define precisely what we mean by almost everywhere complexity for nondeterministic time. In this section we also relate almost everywhere complexity to the concept of immunity. In Section 3 we present our main results. In section 4 we turn to the questions of bi-immunity and co-immunity, which also are relevant to almost everywhere complexity. The results we prove concerning immunity, co-immunity and bi-immunity are the best that can be proved using relativizable proof techniques.

2 A. E. Complexity and Immunity

In order to talk about an a.e. complexity hierarchy for nondeterministic time, we must first agree on the notion of running time for nondeterministic Turing machines. For deterministic Turing machines, it is clear how to define the running time. However there are at least two definitions that are commonly used in defining the running time of a nondeterministic Turing machine:

- NTM M runs in time t on input x if *every* computation path of M on input x has length $\leq t$.
- NTM M runs in time t on input x if [M accepts x implies there is an accepting computation path of length at most t].

Note that for time-constructible T, the class NTIME($T(n)$) is the same, no matter which choice is made in defining the running time of an NTM. That is, when one is concerned mainly with placing an *upper* bound on the running time of an NTM, it makes little difference how one defines the running time.

However, in defining an a.e. complexity hierarchy for nondeterministic time, it is necessary to talk about *lower* bounds on the running time of an NTM. We feel that an a.e. complexity hierarchy defined in terms of the first notion would be unsatisfactory, since using that definition, the running time of M can be large, even when there is a very short accepting computation of M on input x. If there is a short computation of M accepting x, then our intuition tells us that x is an easy input for M to accept. Therefore we define our a.e. complexity hierarchy using the second notion of running time. (Note that this definition does not depend on the behavior of M for strings $x \notin L$.)

Definition:

- Let M be a NTM. Then the (partial) function T_M from Σ^* into $\mathbf{N}$ is defined by: $T_M(x) = \min\{t : \text{there is an accepting path of } M \text{ on } x \text{ of length } t\}$.

- An infinite set L is a.e. complex for NTIME(t) iff for all nondeterministic Turing machines M, $L(M) = L \Longrightarrow T_M(x) = \omega(t(|x|))$.

A concept that is closely related to a.e. complexity is *immunity.* Let L be a language and let $\mathcal{C}$ be a class of languages. Then L is *immune to* $\mathcal{C}$ if L is infinite and L has no infinite subset in $\mathcal{C}$. L is *co-immune to* $\mathcal{C}$ if $\overline{L}$ is immune to $\mathcal{C}$. L is *bi-immune to* $\mathcal{C}$ if L is both immune and co-immune to $\mathcal{C}$.

In [4], Balcázar and Schöning noted the following relationship between almost-everywhere complexity and immunity for deterministic classes.

Theorem 2 [4] Let t be a time-constructible function. Then L is a.e. complex for DTIME($t(n)$) iff L is bi-immune to DTIME($t(n)$).

The forward implication in this theorem is in fact true for *all* functions t. However we note that time-constructibility is necessary for the reverse implication.

Theorem 3 There is a (non-time-constructible) function t and a set L that is bi-immune to DTIME($t(n)$) and not a.e. complex for DTIME($t(n)$).

Proof: (Sketch) Using diagonalization, one can construct a function t that oscillates back and forth between $t(n) = n^2$ and $t(n) = n^4$, such that every Turing machine that runs in time $t(n)$ actually runs in time n^2. Now using the a.e. complexity hierarchy for deterministic time, let L be a set in DTIME(n^4) that is a.e. complex for DTIME(n^2). It follows that L is bi-immune to DTIME($t(n)$), although L can be recognized in time n^4, which is less than or equal to $t(n)$ for infinitely many n. ∎

Although a.e. complexity for deterministic time is related to bi-immunity, we show below that a.e. complexity for nondeterministic time is related to immunity. This difference comes about because the definitions of running time and a.e. complexity for nondeterministic Turing machines are asymmetric, depending on the length of only the accepting computations. (In section 4 we return to this point again, and discuss bi-immunity in more depth.) The next proposition is immediate from the above definitions.

Proposition 4 Let t be a time-constructible function, and let L be a language. Then L is a.e. complex for NTIME($t(n)$) iff L is immune to NTIME($t(n)$).

3 Main Results

In this section we prove some results concerning immunity among nondeterministic time classes. Because of the results of the preceding section, these results may be interpreted in terms of a.e. complexity.

We will need the following notational conventions:

- For any function T and any natural number k, let $T^{(k)}$ denote T composed with itself k times.
- For any monotone nondecreasing unbounded function t defined on the natural numbers, let t^{-1} denote the function that maps y to the unique number x such that

$t(x) \leq y$ and $t(x+1) > y$, (or to -1 if $t(x) > y$ for all x). In particular, note that $t(t^{-1}(y)) \leq y$, and $t(t^{-1}(y)+1) > y$. (It is important to give a precise definition of the intuitive notion of inverse, because for very rapidly growing functions T, the function $T \circ t^{-1}$ is very sensitive to the particular way in which t^{-1} is defined.)

Our first main result is an a.e. complexity hierarchy theorem for nondeterministic time. Although this is proved using known translational methods (see, e.g. [22]), it appears to be a new result. Later on in the paper, we show that this hierarchy theorem cannot be improved substantially using relativizable techniques.

Our a.e. hierarchy theorem depends on the following lemma, which extends Theorem $2'$ of [23] in several small but technically necessary ways. We say that a function T is *ntime-constructible on* S if there is a nondeterministic Turing machine accepting S such that all accepting computations on any input $x \in S$ have length exactly $T(|x|)$. Also, given two functions T and t, we say that $T(n) \neq 2^{O(t(n))}$ *on* S if for all c there exist infinitely many $x \in S$ such that $T(|x|) > 2^{ct(|x|)}$.

Lemma 5 Let t be a time-constructible function, let T be a function that is ntime-constructible on some infinite set S, and assume that $T(n) \neq 2^{O(t(n))}$ on S. Then there is a subset of S that belongs to NTIME$(T(n))$ and is immune to NTIME$(t(n))$.

Proof: We construct the desired language L by a priority argument. Let $M_1, M_2, \ldots$ be an indexing of all nondeterministic Turing machines, clocked to run in time $t(n)$. The requirements are

$$\begin{array}{lcl} N_i & : & M_i \text{ accepts a finite set or a string that is not in } L, \\ P_i & : & L \text{ contains at least } i \text{ strings.} \end{array}$$

The priority order is $N_1, P_1, N_2, P_2, \ldots$. Initially, let $L = \emptyset$, and say that all requirements are unsatisfied.

Stage x:

Step 1: If $x \notin S$ then reject x. Let $n = |x|$.

Step 2: Deterministically, spend $T(n)$ time simulating the construction sequentially for short strings and verifying that certain requirements have been satisfied. Say that the remaining requirements need attention. In particular, let j be minimum such that P_j needs attention.

Step 3: Deterministically, for all i such that $1 \leq i \leq \log T(n)$, spend $2^{-i}T(n)$ steps checking if M_i accepts x (by simulating all computations of M_i on input x).

Step 4: Based on Steps 2 and 3, find the least i such that N_i needs attention and M_i accepts x. If i exists and $i \leq j$ then reject x; N_i is satisfied. Otherwise accept x; P_j is satisfied.

The correctness of the priority argument follows by standard techniques (see [23]). ∎

Theorem 6 Let T and t be monotone nondecreasing time-constructible functions such that, for some k, $(T \circ t^{-1})^{(k)}(n) \neq 2^{O(n)}$. Then there is a set in NTIME($T(n)$) that is immune to NTIME($t(n)$).

Before we present the proof of Theorem 6, let us present a few concrete examples of time bounds to which the theorem applies. For example, if $t(n) = n$, then $T(n)$ may be chosen to be any "nearly exponential" function such as 2^{n^ϵ} for some ϵ, or even $2^{n^{1/\log\log n}}$ or $2^{2^{\log^\epsilon n}}$, etc. (as well as functions that grow much more slowly, but cannot be represented quite so conveniently). In fact, for any of these choices of T, there are sets in NTIME($T(n)$) that are immune to NTIME($t(n)$), where t may be a function such as

$n^{\log^8 n}$ or $2^{2^{\log\log^{20} n}}$, etc. However, for these "nearly polynomial" functions t, Theorem 6 does not show the existence of sets in NTIME($t(n)$) that are immune to NTIME(n).

Proof: Note first that there is some l such that $T \circ (t^{-1} \circ T)^{(l)}(n) \neq 2^{O(t(n))}$.

Let k be the least number such that the hypothesis $T \circ (t^{-1} \circ T)^{(k)}(n) \neq 2^{O(t(n))}$ is true. Let S_0 equal the set of natural numbers. For $1 \leq i \leq k$, let

$$S_i = S_{i-1} \cap \{n : (t^{-1} \circ T)^{(i)}(n) > (t^{-1} \circ T)^{(i-1)}(n)\}.$$

Note that each S_i is infinite, for otherwise $(t^{-1} \circ T)^{(i)} \leq (t^{-1} \circ T)^{(i-1)}$ almost everywhere, and hence $(t^{-1} \circ T)^{(k)} \leq (t^{-1} \circ T)^{(k-1)}$ almost everywhere, so k could not be minimal. Let $\tilde{S}_i$ denote the class of languages containing only strings whose length belongs to the set S_i.

The proof proceeds by establishing the following two claims, which clearly suffice to prove the theorem.

<u>Claim 1:</u> There is a set in $\tilde{S}_k \cap$NTIME($T \circ (t^{-1} \circ T)^{(k)}(n)$) that is immune to NTIME($t(n)$).

<u>Claim 2:</u> If every infinite set in NTIME($T(n)$) has an infinite subset in NTIME($t(n)$), then for all $i \leq k$, every infinite set in $\tilde{S}_i \cap$ NTIME($T \circ (t^{-1} \circ T)^{(i)}(n)$) has an infinite subset in NTIME($t(n)$).

<u>Proof of Claim 1:</u> By Lemma 5 it suffices to show that the set of all strings with lengths in S_k belongs to NTIME($T \circ (t^{-1} \circ T)^{(k)}(n)$) and that $T \circ (t^{-1} \circ T)^{(k)}$ is ntime-constructible on that set. One can see by induction that for all $n \in S_k$, there is a simple nondeterministic strategy for guessing and then verifying the value of $(t^{-1} \circ T)^{(i)}(n)$ that can be accomplished in time $T \circ (t^{-1} \circ T)^{(i)}(n)$. (This involves guessing the value of $t^{-1}(T(m))$ for various m, and then verifying in time $T(m)$ that this guess is correct.)

This strategy suffices for testing membership in S_k and for computing $T \circ (t^{-1} \circ T)^{(k)}(n)$.

Proof of Claim 2: The proof of this claim proceeds by induction on i. For $i = 0$ it is true by assumption. Assume therefore that every infinite set in $\tilde{S}_i \cap \mathrm{NTIME}(T \circ (t^{-1} \circ T)^{(i)}(n))$ has an infinite subset in $\mathrm{NTIME}(t(n))$, and let L be an infinite set in $\tilde{S}_{i+1} \cap \mathrm{NTIME}(T \circ (t^{-1} \circ T)^{(i+1)}(n))$.

Let $L' = \{x10^j : x \in L \text{ and } |x10^j| = t^{-1}(T(|x|))\}$. Then L' is an infinite set in $\tilde{S}_i$. To test membership of $x10^j$ in L' it suffices to

- test that $t(|x10^j|) \leq T(|x|)$ and $t(|x10^j|+1) > T(|x|)$, which takes time $\leq T(|x|) \leq T \circ (t^{-1} \circ T)^{(i+1)}(|x|) = T \circ (t^{-1} \circ T)^{(i)}(|x10^j|)$, and
- test if $x \in L$, in time $T \circ (t^{-1} \circ T)^{(i+1)}(|x|) = T \circ (t^{-1} \circ T)^{(i)}(t^{-1}(T(|x|))) = T \circ (t^{-1} \circ T)^{(i)}(|x10^j|)$.

Thus $L' \in \mathrm{NTIME}(T \circ (t^{-1} \circ T)^{(i)}(n))$.

By the inductive hypothesis, L' has an infinite subset $A \in \mathrm{NTIME}(t(n))$. Let $A' = \{x : x10^j \in A, \text{ where } |x10^j| = t^{-1}(T(|x|))\}$. Note that A' is an infinite subset of L.

To test if $x \in A'$ it suffices to

- guess the value of $t^{-1}(T(|x|))$, which can be checked in time $T(|x|)$; compute j such that $|x10^j| = t^{-1}(T(|x|))$.
- check that $x10^j \in A$, which can be done in time $t(|x10^j|) = t(t^{-1}(T(|x|))) \leq T(|x|)$.

Thus $A' \in \mathrm{NTIME}(T(n))$, and by assumption A' (and hence L) has an infinite subset in $\mathrm{NTIME}(t(n))$. ∎

Corollary 7 Let T be a monotone nondecreasing time-constructible function such that, for some k and almost all n, $T^{(k)}(n) \geq 2^n$. Then there is a set in NTIME($T(n)$) that is a.e. complex for NTIME(n).

Remark: The proof of Theorem 6 makes use of nondeterminism only in order to compute the function t^{-1} in linear time. Because the inverse is unique, "NTIME" in that theorem could just as well be replaced by "UTIME," "$\oplus$TIME," or various other counting classes. If we assume outright that t^{-1} is computable in linear time, then essentially the same proof yields analogous hierarchy theorems with immunity for classes like "BPTIME" and "RTIME" as well. Furthermore, it should be noted that if t is *any* time-constructible function, then $t^{-1}(n)$ can be computed in time $n \log n$ via a naïve binary search strategy. For essentially all "natural" time-constructible functions, there is enough additional structure available to allow t^{-1} to be computable in linear time; thus the condition that t^{-1} be computable in linear time is not very restrictive.

In particular, Corollary 8 below strengthens the hierarchy theorem for probabilistic computation classes proved by Karpinski and Verbeek [16, Theorem 2] in two ways: it presents a set in BPTIME($T(n)$) that is immune for BPTIME($t(n)$), and it allows the difference in the growth rates of T and t to be smaller[2].

Corollary 8 Let T and t be monotone nondecreasing time-constructible functions such that t^{-1} is computable in linear time. Assume that, for some k, $(T \circ t^{-1})^{(k)}(n) \neq 2^{O(n)}$. Then there is a set in BPTIME($T(n)$) that is immune to BPTIME($t(n)$).

For the particular case of NTIME, the next result and its corollaries show that the

[2] We leave it as an exercise to show that if T and t are monotone increasing functions satisfying the conditions of Theorem 2 in [16], then $(T \circ t^{-1})^{(3)}(n) \neq 2^{O(n)}$.

almost-everywhere complexity hierarchy theorem given by Theorem 6 cannot be improved by any relativizable proof technique.

Theorem 9 Let T be a monotone nondecreasing function such that for every k, $T^{(k)}(n) = o(2^n)$. Then there is an oracle A relative to which every infinite set in NTIME($T(n)$) has an infinite subset in NTIME($O(n)$).

Corollary 10 Let T and t be monotone nondecreasing functions such that for every k, $(T \circ t^{-1})^{(k)}(n) = o(2^n)$. Then there is an oracle A relative to which every infinite set in NTIME($T(n)$) has an infinite subset in NTIME($O(t(n))$).

Proof of Corollary 10: Let T and t be given, and note that $T \circ t^{-1}$ satisfies the conditions of Theorem 9. Let A be the oracle guaranteed by Theorem 9, and let L be any infinite set in NTIME($T(n)$), and let $L' = \{x10^i : x \in L \text{ and } |x10^i| = t(|x|)\}$. L' can be recognized relative to A in time $T(|x|) = T(t^{-1}(|x10^i|))$, and thus by the choice of A there is an infinite set $B \subseteq L'$ in $\mathrm{NTIME}^A(O(n))$. Thus the set $B' = \{x : \exists i \; x10^i \in B\}$ is in $\mathrm{NTIME}^A(O(t(n)))$. ∎

Proof of Theorem: Let $\{M_1, M_2, \ldots\}$ be an indexing of nondeterministic oracle Turing machines.

We will build A so that, for every machine M_i, if M_i^A accepts infinitely many strings in time $T(n)$, then there are infinitely many strings of the form $\langle i, x, y\rangle$ in A with $|x| = |y|$. Furthermore, we guarantee that, for all i, x, and y, if $\langle i, x, y\rangle \in A$, then $|x| = |y|$ and M_i accepts x with oracle A in time $T(|x|)$. Define $L_i = \{x : \exists y(|y| = |x| \text{ and } \langle i, x, y\rangle \in A)\}$. If A satisfies the properties above and $L(M_i^A)$ is infinite then clearly L_i is an infinite subset of $L(M_i^A)$ in $\mathrm{NTIME}^A(O(n))$.

Here is a brief outline of the proof strategy, which is intended to make the proof itself easier to follow. The oracle A is constructed by an initial segment argument. During stage $s+1$ we attempt to add one element to each of $L_1, L_2, ..., L_s$. Of course this may not be possible, since for example it may be the case that one of the machines M_i (for $i < s$) accepts no strings at all, which makes it impossible to add any strings to L_i subject to the conditions outlined above. Suppose that j is the smallest index such that it is possible to add an element to L_j during stage s. When such a string $\langle j, x, y\rangle$ is added to A, this change in the oracle set may make it possible to add an element to some other L_k for $k < j$. (That is, M_k may accept some string with the new oracle that was not accepted relative to the old oracle.) In order to add elements to as many different L_i as possible at stage $s+1$, we repeat our attempts up to s times during that stage.

Our assumptions about T imply that for every k, there exists a number $N[k]$ such that, for all $n \geq N[k], (k+1)T^{(k+1)}(n) < 2^n$. The construction defines an increasing sequence of integers n_s with the property that all elements of A of length $\leq n_s$ are determined by the end of stage s. A set S is used to keep track of which L_i have been augmented in the current stage; k will denote the cardinality of S; $d_0, d_1, \ldots$ is a nondecreasing sequence of numbers denoting the lengths of strings that have been considered during the current stage. A is a global variable in the following construction.

Stage 0: Set $n_0 = 0$ and set A to $\emptyset$.

Stage $s+1$: Set $S = \emptyset$, $k = 0$, and $d_0 = 1 + \max(T^{(s)}(N[s]), T^{(s+1)}(n_s))$.

LOOP

Among all $i \leq s, i \notin S$ satisfying

M_i^A accepts some string u with $T^{(k)}(|u|) \geq d_k$ in time $T(|u|)$, (*)

choose i such that $i + |L_i|$ is minimized, and denote it by i_k. (If no such i exists then exit the LOOP. If more than one such i exists then it does not matter which one we choose, so choose the least such i.)

Then choose the lexicographically least string u satisfying (*), and call this string u_k. Select an accepting path of $M_{i_k}^A(u_k)$. Reserve for $\overline{A}$ all strings queried negatively in this path.

Choose a string v_k such that $|v_k| = |u_k|$ and $\langle i_k, u_k, v_k \rangle$ is not reserved for $\overline{A}$, and add $\langle i_k, u_k, v_k \rangle$ to A. (We will prove shortly that such a string v_k exists.)

Add i_k to S, set $d_{k+1} = \max(|u_k|, d_k)$ and set k to $k+1$.

END OF LOOP

Set n_{s+1} = the length of the longest string so far put into A or reserved for $\overline{A}$. Reserve for $\overline{A}$ all strings of length $\leq n_{s+1}$ that have not been placed into A.

End of Stage $s+1$.

We first prove a simple claim relating $|u_k|$ and d_{k+1}.

Claim 1: In stage s, if u_k is defined, then $T^{(k)}(|u_k|) \geq d_{k+1}$.

Proof: If u_k is defined, it is because it satisfies (*), and hence $T^{(k)}(|u_k|) \geq d_k$. Now either $d_{k+1} = d_k$ or $d_{k+1} = |u_k|$. In either case $T^{(k)}(|u_k|) \geq d_{k+1}$. This completes the proof of Claim 1.

We next prove that, in the loop of the construction, when a string u_k is found, it is possible to find a triple $\langle i_k, u_k, v_k \rangle$ to add to A. This follows from the following claim.

Claim 2: When a string u_k is found satisfying (*) during stage $s+1$, the number of strings of length greater than n_s reserved for $\overline{A}$ is $< 2^{|u_k|}$.

Proof: Note that for $j < k, |u_j| \leq d_k$. Thus the number of strings of length $> n_s$ reserved for $\overline{A}$ through the part of stage $s+1$ where u_k is found and acted upon is $\leq \sum_{j=0}^{k} T(|u_j|) \leq (k+1)T(d_k) \leq (k+1)T(T^{(k)}(|u_k|)) \leq (s+1)T^{(s+1)}(|u_k|) < 2^{|u_k|}$.

(To verify this last inequality note that $d_k > T^{(s)}(N[s])$ and $T^{(s)}(|u_k|) \geq d_k$, hence $T^{(s)}(|u_k|) > T^{(s)}(N[s])$, which implies $|u_k| > N[s]$. The inequality now follows, by the properties of $N[s]$.) This completes the proof of Claim 2.

It is clear from the construction that for all i, x, and y, if $\langle i, x, y \rangle \in A$, then M_i accepts x with oracle A and $|x| = |y|$. It remains only to show that if M_i^A accepts infinitely many strings in time $T(n)$, then L_i is infinite.

For the sake of a contradiction, assume that this is not the case, and let i be the least index such that M_i^A accepts infinitely many strings in time $T(n)$, but L_i is finite. Let t be a stage by which, for every $j \leq i + |L_i|$

- if L_j is finite, then every string of the form $\langle j, x, y \rangle$ in A is in A by stage t, and
- if L_j is infinite, then there are more than $i + |L_i| - j$ elements in L_j by stage t.

Note in particular that no element is added to L_i after stage t.

Let m be the value of d_0 at the beginning of stage $t+1$. Since M_i^A accepts infinitely many strings, let x be the lexicographically least string of length $\geq m$ accepted by M_i^A in time $T(|x|)$. Let A_t denote the partial oracle constructed by the end of stage t. From the construction of A and the choice of stage t, one can see that if $M_i^{A_t}$ accepted x, then

x would be placed in L_i in stage $t+1$, contrary to our choice of t. Thus, oracle machine M_i, on input x with oracle A, queries a string in $A - A_t$ along every accepting path (since it accepts x with oracle A but rejects with oracle A_t). Choose one such path and let z be the string that is added to A last among all the strings queried along that path.

Let s be the stage in which z is put into A. Let B_0 denote the partial oracle that has been constructed by the beginning of stage s. In stage s, we build a sequence of partial oracles $B_1, B_2, \ldots$, where for all $k = 1, 2, \ldots$ there exists a string $\langle i_k, u_k, v_k \rangle$, such that $B_k = B_{k-1} \cup \{\langle i_{k-1}, u_{k-1}, v_{k-1} \rangle\}$.

Note that $z = \langle i_{k-1}, u_{k-1}, v_{k-1} \rangle$ for some k; that is, there is some k such that $B_{k-1} \cup \{z\} = B_k$. Since M_i, on input x, queries z, it is clear that $T(|x|) \geq |z| \geq |u_{k-1}|$. By Claim 1, $T^{(k-1)}(|u_{k-1}|) \geq d_k$. But now it follows that $M_i^{B_k}$ accepts the string x with $T^{(k)}(|x|) \geq T^{(k-1)}(|u_{k-1}|) \geq d_k$, and thus the next time through the LOOP in stage s we add x to L_i, in contradiction to our assumption that L_i has reached its final value. ∎

The argument given in the proof of Theorem 9 is nonconstructive. It is possible to construct a recursive oracle A with the desired properties, but that construction is more complicated. We also note that the bound "2^n" in the statement of Corollary 10 comes only from the fact that we consider only oracles over the alphabet $\{0, 1\}$ and thus there are 2^n possible queries of length n. If we were to consider oracles encoded over arbitrary alphabets, then the condition on t and T could be weakened to "there exists a c such that for all k, $(T \circ t^{-1})^{(k)}(n) = o(2^{cn})$," which comes closer to matching the negation of the hypothesis of Theorem 6.

Note that in this oracle construction, we actually prove that there are oracles relative to which every infinite set in NTIME($T(n)$) has an infinite subset in UTIME($t(n)$). Similarly, as Rutger Verbeek [26] has suggested, the same arguments could be used to produce an infinite subset in RTIME($t(n)$).

On the other hand, it should be mentioned that there are oracles relative to which NTIME($O(t)$) = DTIME($O(t)$) for all time-constructible t, and thus relative to these oracles, the a.e. hierarchy theorem for deterministic time carries over to nondeterministic time as well.

4 Consistent Nondeterministic Turing Machines

Although we showed above that immunity is the natural notion to consider when defining an almost-everywhere complexity notion for nondeterministic time complexity, it must be admitted that there is something unsatisfying about our definition of a.e. NTIME complexity. For example, Theorem 6 shows the existence of a set in NTIME($2^{\sqrt{n}}$) that has no infinite subset in NTIME(n). This set is a subset of $Z = \{x10^j : |x10^j| = 2^{\sqrt{|x|}}\}$. Thus every input that is *not* in Z can be rejected immediately. It would be more satisfactory if our notion of almost-everywhere complexity precluded the possibility that infinitely many inputs could be *rejected* easily. This section introduces consistent nondeterministic Turing machines, in order to form a more acceptable notion of a.e. complexity for nondeterministic time.

Consistent NTMs were considered by Buntrock [6]. We are unaware of any earlier mention of this notion in the literature. Consistent Turing machines are very similar in some respects to the strong nondeterministic Turing machines that were defined by Long [18]. Among other uses, strong nondeterministic Turing machines provide a nice characterization of NP $\cap$ coNP.

Definition: A nondeterministic Turing machine M is *consistent* if, on every input x, either all halting computation paths are accepting or all halting computation paths are rejecting.

Note that Long's *strong* nondeterministic Turing machines [18] are simply consistent Turing machines such that there is a halting path for each input.

Given a consistent nondeterministic Turing machine M, we define the "consistent" running time of M on input x as follows: $CT_M(x) = \min\{t :$ there is a halting path of M on input x of length $t\}$.

We say that L is a.e. complex for consistent NTIME($T(n)$) iff [$L(M) = L$ and M a consistent NTM implies $CT_M(x) = \omega(T(|x|))$]. The next theorem follows immediately from these definitions.

Theorem 11 Let T be a time-constructible function. Then L has consistent a.e. NTIME complexity T iff L is bi-immune with respect to NTIME($T(n)$).

Proposition 12 If $T(n) = 2^{\omega(t(n))}$, then there is a set in NTIME($T(n)$) that is bi-immune to NTIME($t(n)$).

Proof: By diagonalization. ∎

The following theorem shows that this bi-immunity result cannot be improved using relativizable proof techniques.

Theorem 13 Let T and t be time-constructible functions such that for all large n, $T(n) < 2^{t(n)}$. Then there is an oracle A relative to which, for every infinite set L in NTIME($T(n)$), either L has an infinite subset in NTIME($t(n)$), or $\overline{L}$ has an infinite subset in DTIME(n).

Corollary 14 Let T and t be time-constructible functions such that for all large n, $T(n) < 2^{t(n)}$. Then there is an oracle A relative to which no set in NTIME($T(n)$) is bi-immune to NTIME($t(n)$).

Proof of Theorem: The oracle is built via a stage construction. At stage 0, $A = \emptyset$.

Stage $\langle i, l \rangle$. Let A be the finite oracle constructed so far, and let n be the length of the longest string in A. If there exists any string x of length longer than n so that M_i rejects x with oracle $A \cup \{\langle i, x \rangle\}$, then let A be extended to $A \cup \{\langle i, x \rangle\}$, and reserve all strings shorter than $\langle i, x \rangle$ for $\overline{A}$.

Else, if no such string x exists, then for all large strings x, there is an accepting computation of M_i on x with oracle $A \cup \{\langle i, x \rangle\}$. "Reserve" one such accepting computation. Note that along this path, at most $T(|x|) < 2^{t(|x|)}$ strings are queried. Let y be a string of length $t(|x|)$ such that $\langle i, x, y \rangle$ is not queried along this path, and extend A to the new oracle $A \cup \{\langle i, x \rangle, \langle i, x, y \rangle\}$.

That completes the construction.

Let L be an infinite set in NTIME$^A(T(n))$, where L is accepted by M_i with oracle A. If for all large l, case 1 is used in stage $\langle i, l \rangle$, then the following algorithm accepts an infinite set that contains only finitely many elements of L: On input x, accept iff $\langle i, x \rangle$ $\in A$.

If this is not the case, then for infinitely many l, case 2 is used in stage $\langle i, l \rangle$. In this case, the following algorithm accepts infinitely many elements of L: on input x, accept iff $\langle i, x \rangle \in A$ and there exists a string y of length $t(|x|)$ such that $\langle i, x, y \rangle \in A$. ∎

Remark: Note that the complement of the set we constructed in Theorem 6 contains an infinite subset that is very easy to recognize, because of the way we used padding in the translational argument. Theorem 13 is evidence that this is no accident, and it suggests that any proof of our a.e. hierarchy theorem may actually require translational techniques in some fundamental way. To see what we mean, consider for example the case when $T(n) = 2^{2^{n-1}}$ and $t(n) = 2^n$. The translational methods used in proving Theorem 6 show that there is a set $L \in \mathrm{NTIME}(T(n))$ that is immune to $\mathrm{NTIME}(t(n))$, but as we observed at the start of Section 4, any string that is not "padded" is trivially in $\overline{L}$, and thus $\overline{L}$ has an infinite subset in deterministic linear time. Furthermore, Theorem 13 shows that *any* set in $\mathrm{NTIME}(T(n))$ that can be shown to be immune to $\mathrm{NTIME}(t(n))$ via relativizable proof techniques *must* have an infinite subset of its complement in deterministic linear time.

Having settled the questions of immunity and bi-immunity, it now remains only to consider co-immunity.

Theorem 15 Let T and t be time-constructible functions with T monotone and $t(n) = o(T(n))$. Then there is a set in $\mathrm{NTIME}(T(n))$ that is co-immune with respect to $\mathrm{NTIME}(t(n))$.

Proof: What is needed is to construct a set $L \in \mathrm{NTIME}(T(n))$ that is co-infinite and intersects every infinite set in $\mathrm{NTIME}(t(n))$.

Choose an unbounded nondecreasing function r such that $r(n)$ is computable in time $T(n)$, and such that $r(n)^2 2^{T(\log r(n))} \leq T(n)$. (For example $r(n)$ can be defined to be the largest $i \leq \log n$ such that $2^{T(\log i)} \leq T(n)/\log^2 n$.)

Let $M_1, M_2 \ldots$ be an enumeration of nondeterministic two-tape Turing machines that run in time $t(n)$. By [5], for every set $L \in \mathrm{NTIME}(t(n))$, there is a machine M_i in this enumeration such that $L = L(M_i)$.

Consider the following program P, which has a parameter M:

On input z of length n:

PART 1: Run M on inputs $x \in \{1, \dots, r(n)\}$, and let $\mathrm{OUT}(z) =$ the number of these inputs that M does not accept within $T(|x|)$ time. (This can be determined using exhaustive search of the computation tree. Note that x is viewed as a number written in binary.)

Let LIST $= \{1, 2, \dots, \mathrm{OUT}(z)\}$

For all $i \in$ LIST

For all $y \in \{1, 2, \dots, r(n)\}$, using exhaustive search of the computation trees, determine if $y \in L(M_i) \cap L(M)$. (Here, we consider y to be in $L(M)$ only if there is an accepting computation of M on y of length at most $T(|y|)$.) If so, remove i from LIST.

Let LIST(z) denote the current contents of LIST.

PART 2: Nondeterministically guess $i \in \mathrm{LIST}(z)$. Use $T(n)$ time to simulate M_i on input z.

For any fixed Turing machine M, the running time of this (nondeterministic) program P is $(r(n))2^{T(\log r(n))} + (r(n))^2 2^{T(\log r(n))} + T(n) = O(T(n))$. Also, if P is run on a three-tape nondeterministic Turing machine, then for each i, for all large inputs z, $T(|z|)$ time is sufficient to simulate $t(|z|)$ steps of M_i on input z in PART 2.

By the time-bounded version of the recursion theorem (see e.g. Theorem 6.1.8 in

[20]), there is a nondeterministic Turing machine $\mathcal{M}$ that executes P with parameter $\mathcal{M}$, having time complexity $O(T(n))$. We claim that $L(\mathcal{M})$ is co-immune with respect to NTIME$(t(n))$. That is, we need to show that $\mathcal{M}$ rejects infinitely many strings, and that $L(\mathcal{M})$ intersects every set in NTIME$(t(n))$.

In order to do this, first we argue that OUT(z) is a monotone nondecreasing, unbounded function. It is obvious that OUT(z) is monotone nondecreasing. Assume for the sake of a contradiction that OUT$(z) < k$ for all z. Then for all z, LIST(z) is always a subset of $\{1, 2, \ldots, k\}$. Note also that $x > z \Longrightarrow$ LIST$(x) \subseteq$ LIST(z). Thus there is some set $S \subseteq \{1, 2, \ldots, k\}$ such that for all large z, LIST$(z) = S$.

We now claim that $i \in S \Longrightarrow L(M_i)$ is finite. This is the case, since if (1) $i \in$ LIST(z) and (2) M_i accepts z and (3) the computation of M_i on z can be simulated in $T(|z|)$ steps, then $z \in L(\mathcal{M})$. Thus on inputs of length n such that $\log \log n > |z|$, it will be discovered that $z \in L(M_i) \cap L(\mathcal{M})$, and thus i will be removed from S, contrary to our choice of S. This establishes our claim that $i \in S \Longrightarrow L(M_i)$ is finite.

Thus for all large inputs z, none of the simulations carried out in PART 2 lead to acceptance, and thus $L(\mathcal{M})$ is finite. But then for all large z there are more than k strings of length $r(|z|)$ that are rejected by $\mathcal{M}$, and thus OUT$(z) > k$ for all large z. Thus we have proved that OUT(z) is an unbounded function. (And thus $\mathcal{M}$ rejects infinitely many strings.)

Now let L be any infinite set in NTIME$(t(n))$. Then $L = L(M_i)$ for some i. Let z be a string in L such that OUT$(z) > i$ and such that a simulation of M_i on input z can be carried out in time $T(|z|)$. It is clear from the construction that either $z \in L(\mathcal{M})$, or there is some $x < z$ such that $x \in L(M_i) \cap L(\mathcal{M})$. The theorem follows. ∎

5 Conclusions

We have considered the question of whether one can prove a tight a.e. hierarchy theorem for nondeterministic time. We have considered different ways in which one might define what is meant by a.e. complexity for nondeterministic Turing machines, and in all cases we have presented hierarchies that are close to the best that can be proved using relativizable proof techniques.

One way to view these results is as an exploration of the strengths and weaknesses of translational methods. We have proved essentially the strongest immunity results that can be proved using translational methods, and we have also shown that any relativizing proof showing the existence of immune sets in NTIME classes has a characteristic of a "padding" argument. (See the remarks after the proof of Theorem 13.)

The i.o. time hierarchy for nondeterministic time is also proved by translational methods, but in that setting a more sophisticated argument allows a tighter hierarchy to be proved. Up to now, the best i.o. time hierarchy for probabilistic time classes is the same as the a.e. time hierarchy mentioned after the proof of Theorem 6. It would be interesting to know if the oracle construction of [9] can be improved to show that the (i.o.) probabilistic time hierarchy given by Theorem 6 is optimal.

6 Acknowledgments

We thank the authors of [12] for sharing this paper with us, and John Geske, Joel Seiferas, Alan Selman and Jie Wang for helpful discussions.

References

[1] E. Allender, Oracles vs Proof techniques that do not relativize, in: *Proc. SIGAL International Symposium on Algorithms*, Lecture Notes in Computer Science **450** (Springer, Berlin, 1990) 39–52.

[2] E. Allender, R. Beigel, U. Hertrampf, and S. Homer, A note on the almost-everywhere hierarchy for nondeterministic time, in: *Proc. 7th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science **415** (Springer, Berlin, 1990) 1–11.

[3] L. Babai, L. Fortnow, and C. Lund, Non-deterministic exponential time has two-prover interactive protocols, in: *Proc. 31st IEEE Symposium on Foundations of Computer Science* (1990) 16–25.

[4] J. Balcázar and U. Schöning, Bi-immune sets for complexity classes, *Mathematical Systems Theory* **18** (1985) 1–10.

[5] R. Book, S. Greibach, and B. Wegbreit, Time- and tape-bounded Turing acceptors and AFL's, *J. Computer and System Sciences* **4** (1970) 606–621.

[6] G. Buntrock, *Logarithmisch platzbeschränkte Simulationen* (Doctoral Dissertation, Technische Universität Berlin, 1989).

[7] S. Cook, A hierarchy for nondeterministic time complexity, *Journal of Computer and System Sciences* **7** (1973) 343–353.

[8] S. Cook and R. Reckhow, Time-bounded random access machines, *Journal of Computer and System Sciences* **7** (1973) 354–375.

[9] L. Fortnow and M. Sipser, Probabilistic computation and linear time, in: *Proc. 21st IEEE Symposium on Foundations of Computer Science* (1989) 148–156.

[10] M. Fürer, Data structures for distributed counting, *Journal of Computer and System Sciences* **28** (1984) 231–243.

[11] J. Geske, D. Huynh, and A. Selman, A hierarchy theorem for almost everywhere complex sets with application to polynomial complexity degrees, in: *Proc. 4th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science **247** (Springer, Berlin, 1987) 125–135.

[12] J. Geske, D. Huynh, and J. Seiferas, A note on almost-everywhere-complex sets and separating deterministic-time-complexity classes, *Information and Computation* **92** (1991) 97–104.

[13] J. Geske and D. Kakihara, Almost-everywhere complexity, bi-immunity, and nondeterministic space, in: *Advances in Computing and Information – ICCI '90*, Lecture Notes in Computer Science **468**, (Springer, Berlin, 1990) 44–51.

[14] J. Hartmanis and R. Stearns, On the computational complexity of algorithms, *Transactions of the AMS* **117** (1965) 285–306.

[15] F. Hennie and R. Stearns, Two-tape simulation of multitape Turing machines, *J. ACM* **13** (1966) 533–546.

[16] M. Karpinski and R. Verbeek, Randomness, provability, and the separation of Monte Carlo time and space, in: E. Börger, ed, *Computation Theory and Logic*, Lecture Notes in Computer Science **270** (Springer, Berlin, 1987) 189–207.

[17] Ming Li, *Lower Bounds in Computational Complexity*, (Doctoral Dissertation, Cornell University, 1985).

[18] T. Long, Strong nondeterministic polynomial-time reductions, *Theoretical Computer Science* **21** (1982) 1–25.

[19] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, Algebraic methods for interactive proof systems, in: *Proc. 31st IEEE Symposium on Foundations of Computer Science* (1990) 2–10.

[20] M. Machtey and P. Young, *An Introduction to the General Theory of Algorithms* (Elsevier North-Holland, Inc., New York, 1978).

[21] C. Rackoff and J. Seiferas, Limitations on separating nondeterministic complexity classes, *SIAM J. Comp.* **10** (1981) 742–745.

[22] S. Ruby and P. Fischer, Translational methods and computational complexity, in: *Proc. 6th IEEE Symposium on Switching Circuit Theory and Logical Design* (1965) 173–178.

[23] J. Seiferas, M. Fischer, and A. Meyer, Separating nondeterministic time complexity classes, *J. ACM* **25** (1978) 146–167.

[24] A. Shamir, IP = PSPACE, in: *Proc. 31st IEEE Symposium on Foundations of Computer Science* (1990) 11–15.

[25] C. Smith, A note on arbitrarily complex recursive functions, *Notre Dame Journal of Formal Logic* **29** (1988) 198–207.

[26] R. Verbeek, personal communication.

[27] S. Zák, A Turing machine hierarchy, *Theoretical Computer Science* **26** (1983) 327–333.