DOWNWARD SEPARATION FAILS CATASTROPHICALLY FOR LIMITED NONDETERMINISM CLASSES

R. BEIGEL† AND J. GOLDSMITH‡

Abstract. The β hierarchy consists of classes $\beta_k = \mathrm{NP}[\log^k n] \subseteq \mathrm{NP}$. Unlike collapses in the polynomial hierarchy and the Boolean hierarchy, collapses in the β hierarchy do not seem to translate up, nor does closure under complement seem to cause the hierarchy to collapse. For any consistent set of collapses and separations of levels of the hierarchy that respects $\mathrm{P} = \beta_1 \subseteq \beta_2 \subseteq \cdots \subseteq \mathrm{NP}$, we can construct an oracle relative to which those collapses and separations hold; at the same time we can make distinct levels of the hierarchy closed under computation or not, as we wish. To give two relatively tame examples: For any $k \geq 1$, we construct an oracle relative to which

$$\mathrm{P} = \beta_k \neq \beta_{k+1} \neq \beta_{k+2} \neq \cdots$$

and another oracle relative to which

$$\mathrm{P} = \beta_k \neq \beta_{k+1} = \mathrm{PSPACE}.$$

We also construct an oracle relative to which $\beta_{2k} = \beta_{2k+1} \neq \beta_{2k+2}$ for all k.

Key words. structural complexity theory, limited nondeterminism, hierarchies, oracles

AMS subject classifications. 68Q15

1. Introduction.

Although standard nondeterministic algorithms solve many NP-complete problems with $O(n)$ nondeterministic moves, there are other problems that seem to require very different amounts of nondeterminism. For instance, clique can be solved with only $O(\sqrt{n})$ nondeterministic moves, and Pratt's algorithm [16] solves primality, which is not believed to be NP-complete, with $O(n^2)$ nondeterministic moves. Motivated by the different amounts of nondeterminism apparently needed to solve problems in NP, Kintala and Fischer [9, 10, 11] defined limited nondeterminism classes within NP, including the classes we now call the β hierarchy. The structural properties of the β classes were studied further by Àlvarez, Diaz and Toran [1, 6]. These classes arose yet again in the work of Papadimitriou and Yannakakis [15] on particular problems inside NP (e.g., quasigroup isomorphism can be solved with $O(\log^2 n)$ nondeterministic moves).

Kintala and Fischer [11] defined $\mathcal{P}_{f(n)}$ to be the class of languages accepted by a nondeterministic polynomial-time bounded Turing machine that makes at most $f(n)$ c-ary nondeterministic moves (equivalently, $O(f(n))$ binary nondeterministic moves) on inputs of length n. Being mostly interested in polylogarithmic amounts of nondeterminism, they defined $\mathcal{P}L_k = \mathcal{P}_{\log^k n}$.

Diaz and Toran [6] wrote $\beta_{f(n)}$ to denote Kintala and Fischer's $\mathcal{P}_{f(n)}$ and β_k to denote $\mathcal{P}L_k$. Papadimitriou and Yannakakis [15] wrote $\mathrm{NP}[f(n)]$ to denote $\mathcal{P}_{f(n)}$. (Their work is surveyed in [7].) We will adopt the $\mathrm{NP}[f(n)]$ notation of Papadimitriou and Yannakakis, as well as the β_k notation of Diaz and Toran. To reiterate:

DEFINITION 1.1.

- *A language L belongs to* $\mathrm{NP}[f(n)]$ *if there exists a polynomial-time bounded nondeterministic Turing machine that accepts L and makes $O(f(n))$ nondeterministic choices on inputs of length n. (Note:* $\mathrm{NP}[f(n)] \subseteq \mathrm{DTIME}(2^{O(f(n))})$*.)*
- $\beta_k = \mathrm{NP}[\log^k n]$.
- *The β hierarchy consists of* $\beta_1, \beta_2, \ldots$.
- $\beta = \bigcup_k \beta_k$.

†Yale University, Dept. of Computer Science, P.O. Box 208285, New Haven, CT 06520-8285. email: beigel-richard@cs.yale.edu. Research supported in part by the United States National Science Foundation under grant CCR-8958528 and by the Netherlands Organization for Scientific Research (NWO) under Visitors Grant B 62-403.

‡University of Kentucky, Dept. of Computer Science, Lexington, KY 40506. email: goldsmit@cs.engr.uky.edu. Research supported in part by the National Science Foundation under grant CCR-9315354

Kintala and Fischer [11] constructed oracles that make the β hierarchy collapse to any desired level. That is, there is an oracle relative to which

$$\beta_1 \neq \beta_2 \neq \cdots$$

and, for every $k \geq 1$, there is an oracle relative to which

$$\beta_1 \neq \beta_2 \neq \cdots \neq \beta_k = \mathrm{NP}.$$

Oracles can also make the polynomial hierarchy and the Boolean hierarchy collapse to any desired level [12, 4]. The polynomial and Boolean hierarchies have a very nice property: collapses translate upward. I.e., if the kth and $(k+1)$st levels are equal, then all levels are contained in the kth [5, 4]. This is also reflected in the nondeterminism hierarchy, now known as the b hierarchy, studied by Buss and Goldsmith [3]. The classes in the b hierarchy are defined by two parameters: the exponent of the polynomial time bound (ignoring log factors), and the constant factor for $k \log n$ bits of nondeterminism. This hierarchy exhibits upward collapse for both time and k. All attempts to prove an analogous translational property for the β hierarchy have failed. In fact the obvious technique extends a collapse by only a constant factor in the number of nondeterministic bits, giving one of the aforementioned upward collapses of the b hierarchy.

Hemachandra and Jha [8] attempted to explain this failure by examining the tally sets in the β hierarchy. For each k, they constructed an oracle that makes $\beta_j \cap \mathrm{TALLY} = \beta_{j+1} \cap \mathrm{TALLY}$ iff $j \leq k$. We find this explanation unsatisfactory because it considers only tally sets.

The known behavior of relativized β hierarchies is that $\beta_0^A = \beta_1^A$ and that $\beta_i^A \subseteq \beta_j^A$ for all $i < j$ and all oracles A. A *collapse* is a statement of the form $\beta_i^A = \beta_j^A$. A *separation* is a statement of the form $\beta_i^A \neq \beta_j^A$. A *closure* is a statement of the form $\beta_i^A = \text{co-}\beta_i^A$. A *nonclosure* is a statement of the form $\beta_i^A \neq \text{co-}\beta_i^A$. A *requirement* is a collapse, separation, closure, or nonclosure. We call a set of requirements *consistent* if it is consistent with the known behavior of relativized β hierarchies, as stated above, and the standard axioms for $=$ and $\subseteq$.

Given a set S of requirements let X be the union of $[0, 1]$ and all intervals $[i, j]$ such that $i < j$ and the collapse $\beta_i^A = \beta_j^A$ belongs to S. It is easy to see that S is consistent iff the following conditions hold for all a and b such that $[a, b] \subseteq X$ or $[b, a] \subseteq X$: (1) the separation $\beta_a^A \neq \beta_b^A$ does not belong to S; (2) the closure $\beta_a^A = \text{co-}\beta_a^A$ and the nonclosure $\beta_b^A \neq \text{co-}\beta_b^A$ do not both belong to S.

For any consistent set of requirements, we construct an oracle A such that the β hierarchy relative to A satisfies them. For example, for each $k \geq 0$, there is an oracle that makes $\beta_j = \beta_{j+1}$ iff $j \leq k$. Another oracle makes $\beta_j = \beta_{j+1}$ iff $j = 0$ or j is prime. We can also make distinct levels in the hierarchy be closed under complementation or not, as long as this is consistent with the collapses (if $\beta_i^A = \beta_j^A$ then we cannot have $\beta_i^A = \text{co-}\beta_i^A$ and $\beta_j^A \neq \text{co-}\beta_j^A$).

We prove two initial results for every k:

- *There is an oracle that makes the first k levels of the β hierarchy coincide, but makes the remaining levels all distinct (Theorem 2.3).*
- *There is an oracle that makes the first k levels of the β hierarchy coincide, the $(k+1)$st level different from the kth, and the remaining levels all equal (Theorem 2.5).*

The techniques from these two constructions can be combined to get any consistent finite or infinite set of collapses and separations. To collapse β_k into β_j (for

$k > j$), we code a complete set for β_k into β_j. The same coding techniques can also code $\overline{\beta}_i$ into β_i, for any i's we wish, as long as this doesn't violate any collapses. (If β_k is collapsed to β_j, then either both or neither will be closed under complement.) Finally, in Section 3, we extend our results to β_r for real $r \geq 0$.

One theme in complexity theory is to ask whether NP − P contains any easy sets (assuming P $\neq$ NP). The answer to the question above depends on the definition of "easy." Ladner [14] showed that if P $\neq$ NP then NP − P contains an incomplete set. On the other hand, there are oracles relative to which P $\neq$ NP, but NP − P contains (a) no tally sets [13] or (b) no sets in co-NP [2]. It is unknown whether the assumption P $\neq$ NP implies that NP − P contains a set in DTIME(n^{polylog}); a positive answer would improve many constructions in the literature. As a step toward understanding that question, we construct an oracle relative to which P $\neq$ NP but NP − P contains no set in the β hierarchy (Corollary 2.2).

2. Limited Nondeterminism Hierarchies. The construction below gives almost all the techniques used in subsequent theorems.

THEOREM 2.1. *Let g_0 and g_1 be polynomial-time computable monotone increasing functions such that $\log n \in o(g_1(n))$ and $g_0(n) \in n^{O(1)}$. If $g_0(n^{O(1)}) \subseteq o(g_1(n))$, then there exists an oracle A such that $\mathrm{P}^A = (\mathrm{NP}[g_0(n)])^A \neq (\mathrm{NP}[g_1(n)])^A$ (and in fact there is a tally set in $(\mathrm{NP}[g_1(n)])^A - (\mathrm{NP}[g_0(n)])^A$).*

Proof. Let $C^A = \{(e, x, 0^s)$: oracle NTM e accepts x within s steps with oracle A, making at most $g_0(|x|)$ nondeterministic choices$\}$. Then C^A is $\leq_m^p$-complete for $(\mathrm{NP}[g_0(n)])^A$ for every A. Let $p(n)$ be the polynomial time bound for some $\mathrm{NP}[g_0(n)]$ oracle Turing machine recognizing $C^{()}$.

Let $D^A = \{x \in 1^* : \exists y[|y| = g_1(|x|) \wedge 0xy \in A]\}$. Note that $D^A \in \mathrm{NP}[g_1(n)]^A$.

The construction consists of coding C^A into A in a polynomial-time recoverable manner, making $(\mathrm{NP}[g_0(n)])^A \subseteq \mathrm{P}^A$, while diagonalizing, i.e, guaranteeing that no P^A machine recognizes the set D^A, so $\mathrm{P}^A \neq (\mathrm{NP}[g_1(n)])^A$.

At the end of the construction, we will have

$$w \in C^A \iff 1^{p(|w|)^2}0w \in A.$$

We refer to all strings beginning with 1 as *coding strings.* We refer to all strings beginning with 0 as *diagonalizing strings.*

Assume that $\mathrm{P}^{()}$ is enumerated by Turing machines $P_1^{()}, P_2^{()}, \ldots$ so that $P_i^{()}$ runs in time bounded by n^i for all i, and for sufficiently large n.

The construction proceeds in stages. At the end of stage s, A is decided for all strings of length up to n_s, and D^A is extended so that P_s^A does not recognize D^A. The stage consists of one diagonalization, which determines n_s, and continued encoding of C^A.

At stage s, choose $n > n_{s-1}$ such that n is a power of 2, the running time of $P_s^{()}$ on inputs of length n is at most n^s, and n satisfies an inequality to be specified below that is to be specified below that is true for almost all n. Let $x = 1^n$. The value of $D^A(1^n)$ depends only on oracle strings of length $\ell = 1 + n + g_1(n)$. Code C^A for all coding strings of length $\leq \ell - 1$ (i.e., for all w such that $p(|w|)^2 + 1 + |w| \leq \ell - 1$, let $1^{p(|w|)^2}0w \in A \iff w \in C^A$) and freeze A up to length $\ell - 1$.

In order to diagonalize, $P_s^A(x)$ must be calculated. But that computation may query coding strings that code computations of C^A that are not yet decided, because those computations in turn rely on strings for which A is not yet decided. Those strings in turn may depend on other coding strings. Any diagonalizing strings that do not already belong to A and are queried by $P_s^A(x)$, or by the computation corresponding to a coding string that $P_s^A(x)$ queries, or in the computation corresponding to a coding string that one of those computations queries, or so on, are restrained from A. We claim that there are more potential witnesses for x to be in D^A than there are possible queries in such a cascade of queries, so deciding $P_s^A(x)$ does not restrict our decision about $D^A(x)$.

Because of the encoding of C^A, a coding string z codes a computation that depends only on strings of length bounded by $\sqrt{|z|}$. $C^A(w)$ directly depends on at most $p(|w|)2^{g_0(|w|)}$ of these shorter strings.

A computation of $P_s^A(x)$ may query no more than n^s strings, each of length bounded by n^s. Each of these strings may code a computation on a string of length at most $n^{s/2}$. Each of these computations depends on at most $p(n^{s/2})2^{g_0(n^{s/2})}$ strings, each of which depends on at most $p(n^{s/4})2^{g_0(n^{s/4})}$ strings, etc.

This recursion can be cut off at strings of length $\ell - 1$, because A is already determined up to that length. The total number of queries needed to decide $P_s^A(x)$ is bounded by n^s times the product of all the terms above of the form $p(n^{s/2^i})2^{g_0(n^{s/2^i})}$. There are at most $\log\log n^s - \log\log(\ell-1) \le \log\log n^s - \log\log n = \log s$ such terms, and each of them is bounded by $p(n^s)2^{g_0(n^s)} = 2^{O(g_0(n^s))}$. Therefore the total number of queries on which $P_s^A(x)$ depends is $n^s 2^{O(\log(s) g_0(n^s))}$, which is less than $2^{g_1(n)}$ for sufficiently large n. (The inequality that n must satisfy is $n^s(p(n^s)2^{g_0(n^s)})^{\log(s)} < 2^{g_1(n)}$.)

Thus there remains an unrestrained diagonalizing string of length ℓ, which we put into A if $P_s^A(x)$ rejects x. That is, we set $D^A(x) = 1 - P_s^A(x)$, adding a string $0xy$ to A if necessary. Thus, for each s, we can guarantee that P_s^A does not accept D^A, so $D^A \notin \mathrm{P}^A$. Since $D^A \in (\mathrm{NP}[g_1(n)])^A$, this shows that $(\mathrm{NP}[g_1(n)])^A \ne \mathrm{P}^A$. Since $C^A \in \mathrm{P}^A$ is complete for $(\mathrm{NP}[g_0(n)])^A$, this shows that $(\mathrm{NP}[g_0(n)])^A \subseteq \mathrm{P}^A$.

We complete stage s by letting $n_s = \max(\ell, n^s)$, and finishing the coding of any $C^A(w)$ that was begun or queried in this stage. □

The preceding theorem is tight because if $\mathrm{P} = \mathrm{NP}[g_0(n)]$ then $\mathrm{P} = \mathrm{NP}[g_0(n^{O(1)})]$ (even if we restrict to binary nondeterministic moves) via a relativizable proof. (Previously, Sanchis [17] had observed that if $\mathrm{P} = \mathrm{NP}[n^\epsilon]$ for some $\epsilon > 0$, then $\mathrm{P} = \mathrm{NP}$).

Because the classes are separated by tally sets, we also separate the exponential-time versions of these classes (see [8] for elaboration of this).

COROLLARY 2.2. *There is an oracle relative to which* $\mathrm{P} = \beta \ne \mathrm{NP}$.

Proof. Let $g_0(n) = 2^{(\log\log n)^2}$ and $g_1(n) = n$ in the previous theorem. Then for all k, $\beta_k^A \subseteq (\mathrm{NP}[2^{(\log\log n)^2}])^A = \mathrm{P}^A \ne \mathrm{NP}^A$. □

THEOREM 2.3. *Let* $g(\cdot,\cdot)$ *be a polynomial-time computable, monotone increasing (in both variables) function with* $\log n \in o(g(n,i))$ *and* $g(n,i) \in n^{O(1)}$ *for all* $i \ge 0$. *If* $g(n^{O(1)}, 2i) \subseteq o(g(n, 2i+1))$ *for all* $i \ge 1$, *then there exists an oracle* A *such that* $\mathrm{P}^A = (\mathrm{NP}[g(n,0)])^A$ *and* $(\mathrm{NP}[g(n,2i)])^A \ne (\mathrm{NP}[g(n,2i+1)])^A$ *for all* $i \ge 0$ *(and in fact there is a tally set in* $(\mathrm{NP}[g(n,2i+1)])^A - (\mathrm{NP}[g(n,2i)])^A$ *for each* i*).*

(In this theorem, we ignore the relationship between $\mathrm{NP}[g(n,2i-1)]$ and $\mathrm{NP}[g(n,2i)]$. We will take that up in the next theorem.) The only difference between this construction and the previous one is that there are infinitely many diagonalizations going on. At stage $s = \langle e, i\rangle$, we guarantee that the eth machine for $(\mathrm{NP}[g(n,2i)])^A$ does not accept the diagonal set $D_i^A \in (\mathrm{NP}[g(n,2i+1)])^A$. Thus, $(\mathrm{NP}[g(n,2i+1)])^A \ne (\mathrm{NP}[g(n,2i)])^A$. The counting argument for this construction is identical to that in the proof of Theorem 2.1.

COROLLARY 2.4. *For any k, there is an oracle relative to which*

$$\mathrm{P} = \beta_k \neq \beta_{k+1} \neq \beta_{k+2} \neq \cdots.$$

Proof. Let $g(n,i) = \log^{k+\lfloor i/2 \rfloor} n$ in the preceding theorem. □

THEOREM 2.5. *Let g_0 and g_1 be polynomial-time computable monotone increasing functions such that $\log n \in o(g_1(n))$ and $g_0(n) \in n^{O(1)}$. If $g_0(n^{O(1)}) \subseteq o(g_1(n))$ then there exists an oracle A such that*

$$\mathrm{P}^A = (\mathrm{NP}[g_0(n)])^A \neq (\mathrm{NP}[g_1(n)])^A = \mathrm{PSPACE}^A$$

(and in fact there is a tally set in $(\mathrm{NP}[g_1(n)])^A - (\mathrm{NP}[g_0(n)])^A$).

Sketch. In this construction, we do two encodings and one diagonalization. In addition to coding C_k^A into P, we also code E^A, a generic $\leq_m^p$-complete set for PSPACE, into A. ($E^A = \{(e, x, 0^s)$: oracle DTM e accepts x using at most s tape squares with oracle $A\}$, where we also count the space used on the oracle tape.) At the end of the construction, we have

$$x \in E^A \iff (\exists y)[|y| = g_1(n) \wedge 2xy \in A]$$

(If one prefers binary oracles, one may code 0, 1, and 2 as 00, 01, and 10.) When we are doing a diagonalization to make $P_s^A(x) \neq D^A(x)$, if a coding string for $C_k^A(w)$ is queried, we proceed as before; if a coding string for $E^A(w)$ is queried, where $|w| \geq |x|$, then we simply restrain that coding string from the oracle. This will not restrain all the coding strings for $E^A(w)$, since there are $2^{g_1(|w|)}$ coding strings for $E^A(w)$; if $|w| \geq |x|$, then $2^{g_1(|w|)} > (p(|x|^s)2^{g_0(|x|^s)})^{\log(s)}$, where $(p(|x|^s)2^{g_0(|x|^s)})^{\log(s)}$ is the upper bound on the total number of queries generated by the computation of $P_s^A(x)$, as in the proof of Theorem 2.1. Therefore, restraining any such coding strings queried in the computation of $P_s^A(x)$ or in its cascade of queries can not restrain all such coding strings, and thus can not decide $E^A(w)$. At the end of each stage, we complete all codings begun or queried in that stage, so that it will not be changed in any subsequent stage.
□

COROLLARY 2.6. *For every k, there is an oracle relative to which*

$$\mathrm{P} = \beta_k \neq \beta_{k+1} = \mathrm{PSPACE}.$$

With only a slight modification of this technique, we get far more bizarre collapses.

THEOREM 2.7. *Let $g(\cdot,\cdot)$ be a polynomial-time computable, monotone increasing (in both variables) function with $\log n \in o(g(n,i))$ and $g(n,i) \in n^{O(1)}$ for all $i \geq 1$. If $g(n^{O(1)}, 2i) \subseteq o(g(n, 2i+1))$ for all $i \geq 1$, then there exists an oracle A such that $\mathrm{P}^A = (\mathrm{NP}[g(n,0)])^A$ and $(\mathrm{NP}[g(n,2i)])^A \neq (\mathrm{NP}[g(n,2i+1)])^A = (\mathrm{NP}[g(n,2i+2)])^A$ for all $i \geq 0$ (and in fact there is a tally set in $(\mathrm{NP}[g(n,2i+1)])^A - (\mathrm{NP}[g(n,2i)])^A$ for each i).*

We include the full proof of this result, although it uses techniques mentioned before, since this shows how all the pieces fit together.

Proof. Let $C_i^A = \{(e, x, 0^s)$: oracle NTM e accepts x within s steps with oracle A, making at most $g(|x|, i)$ nondeterministic choices$\}$. Then C_i^A is $\leq_m^p$-complete for $(\mathrm{NP}[g(n,i)])^A$ for any A. Let $p(n,i)$ be the nondeterministic time bound for some

NP$[g(n,i)]$ oracle Turing machine recognizing $C_i^{()}$. Without loss of generality, assume that for all i and almost all n, $p(n,i) \leq p(n,i+1)$.

Let $D_i^A = \{x : \exists y[|y| = g(|x|,i) \wedge 0xy \in A]\}$.

For convenience, define $g(n,-1) = 0$ for all n.

The construction consists of coding C_{2i}^A into $(\mathrm{NP}[g(n,2i-1)])^A$, for each $i \geq 0$, so $(\mathrm{NP}[g(n,0)])^A \subseteq \mathrm{P}^A$ and $(\mathrm{NP}[g(n,2i+2)])^A \subseteq (\mathrm{NP}[g(n,2i+1)])^A$ for all i; and diagonalizing, i.e, guaranteeing that no $(\mathrm{NP}[g(n,2i)])^A$ machine recognizes the set D_{2i+1}^A, for any i, so $(\mathrm{NP}[g(n,2i+1)])^A \not\subseteq (\mathrm{NP}[g(n,2i)])^A$ for any i.

At the end of the construction, we will have $x \in C_{2i}^A \iff \exists y[|y| = g(|x|,2i-1) \wedge 10^{2i}1^{p(|x|,2i-1)^2}0xy \in A]$.

Assume that $(\mathrm{NP}[g(n,i)])^A$ is enumerated by oracle NTMs $M_{1,i}, M_{2,i}, \ldots$, where $M_{e,i}$ runs in time bounded by n^e for sufficiently large n.

The construction proceeds in stages. Stage $s = \langle e,i \rangle$ consists of some encodings and one diagonalization, which determines n_s. At the end of stage s, A is decided for all strings of length $\leq n_s$ (and some further coding strings), and A has been extended so that $M_{e,2i}^A$ does not recognize D_{2i+1}^A.

At stage s, let $\langle e,i \rangle = s$, and then choose $n > n_{s-1}$ such that n is a power of 2, $M_{e,2i}$ runs in time bounded by n^e on inputs of length n, and n satisfies an inequality to be specified below that is true for almost all n. Let $n_s = n$. Let $x = 0^n$. The value of $D_{2i+1}^A(x)$ depends only on strings of length $\ell = 1 + n + g(n,2i+1)$. Do all coding involving witnesses of length less than ℓ, and then freeze A through length $\ell - 1$.

As before, in order to diagonalize, we will need to calculate $M_{e,2i}^A(x)$, which may generate a cascade of queries. Any diagonalizing strings that do not already belong to A and are queried in this cascade are restrained from A. But coding strings may be queried as well. (Because we are coding nondeterministically, coding strings can be thought of as potential witnesses to membership.) If $M_{e,2i}^A(x)$ queries a potential witness that $w \in C_{2j}^A$, where $2j > 2i$ and $C_{2j}^A(w)$ has not yet been decided, that potential witness is restrained from A. If $2j \leq 2i$ and $C_{2j}^A(w)$ has not yet been decided, then we compute $C_{2j}^A(w)$ recursively. We will show below that the number of queries generated by such a cascade of queries is smaller than both of the following bounds: (1) the number of potential witnesses for $w \in C_{2j}^A$, (2) the number of potential witnesses for $x \in D_{2i+1}^A$. In fact, bound (1) implies bound (2) as follows. The number of witnesses for $D_{2i+1}^A(x)$ is $2^{g(|x|,2i+1)}$, and the number of witnesses for $C_{2j}^A(w)$ is $2^{g(|w|,2j-1)}$. If a witness of $C_{2j}^A(w)$ is restrained, then $|w| \geq |x|$ and $2j > 2I$. Thus $2j-1 \geq 2i+1$, so, by monotonicity of g, $g(|w|,2j-1) \geq g(|x|,2j-1) \geq g(|x|,2i+1)$.

Thus, restraining potential witnesses as described does not impede any encodings or restrict our decision about $D_{2i+1}(x)$, or those $C_{2j}^A(w)$ for which we restrict coding strings. (We don't have to worry about what happens to potential witnesses for $w \in C_{2j}^A$ at a later stage, because any affected codings, i.e., $C_{2j}^A(w)$, will be completed at this stage; later diagonalizations will not affect them.)

Now we show that there are more potential witnesses for $x \in D_{2i+1}^A$ than there are possible queries in such a cascade of queries. Because of how we encode C_k^A, a coding string z codes a computation that depends on strings of length bounded by $\sqrt{|z|}$. For $k \leq 2i+1$, $C_k^A(w)$ depends on at most $p(|w|,k)2^{g(|w|,k)} \leq p(|w|,2i+1)2^{g(|w|,2i+1)}$ of these shorter strings.

$M_{e,2i}^A(x)$ has at most $2^{g(n,2i)}$ computations, and each of those computations may query no more than n^e strings, each of length bounded by n^e. Each such string may code a computation $C_{2j}^A(w)$ where $|w| \leq \sqrt{n^e}$, but we only need to expand that computation if $2j \le 2i$. Each of these computations depends on at most $p(n^{e/2}, 2i)2^{g(n^{e/2},2i)}$ strings, each of which depends on at most $p(n^{e/4}, 2i)2^{g(n^{e/4},2i)}$ strings, etc. As before, the total number of queries needed to decide $M^A_{e,i}(x)$ is bounded by the product of $\log e \le \log s$ terms, each of which is $2^{o(g(n,2i+1))}$. Therefore the total number of queries on which $M^A_{e,2i}(x)$ depends is $2^{o(g(n,2i+1))}$, which is less than $2^{g(n,2i+1)}$ for sufficiently large n.

Thus there remains an unrestrained diagonalizing string of length ℓ, which we put into A if and only if $M^A_{e,2i}(x)$ rejects x. That is, we set $D^A_{2i+1}(x) = 1 - M^A_{e,2i}(x)$, adding a string $0xy$ to A if necessary. Thus, for each s, we can guarantee that $M^A_{e,2i}$ does not accept D^A_{2i+1}, so $D^A_{2i+1} \notin (\mathrm{NP}[g(n,2i)])^A$. Since $D^A_i \in (\mathrm{NP}[g(n,2i+1)])^A$, this shows that $(\mathrm{NP}[g(n,2i+1)])^A \not\subseteq (\mathrm{NP}[g(n,2i)])^A$ for all $i \ge 0$.

Since C^A_{2i} is complete for $(\mathrm{NP}[g(n,2i)])^A$, our encoding guarantees that $(\mathrm{NP}[g(n,2i)])^A \subseteq (\mathrm{NP}[g(n,2i-1)])^A$ for all $i \ge 0$. □

COROLLARY 2.8. *There is an oracle relative to which, for each k, $\beta_{2k} = \beta_{2k+1} \neq \beta_{2k+2}$.*

COROLLARY 2.9. *For any consistent pattern of collapses and separations of the β_k's, there is an oracle relative to which that pattern holds.*

Notice that if the set of collapses is not recursive, then the oracle will also be non-recursive.

In addition to collapsing or separating β_j and β_k, we can code co-β_k into β_k — or separate the two. This involves some additional argument.

THEOREM 2.10. *Let $g(\cdot,\cdot)$ be a polynomial-time computable, monotone increasing (in both variables) function with $\log n \in o(g(n,i))$ and $g(n,i) \in n^{O(1)}$ for all $i \ge 1$. If $g(n^{O(1)}, 2i) \subseteq o(g(n,2i+1))$ for all $i \ge 1$, then there exists an oracle A such that $\mathrm{P}^A = (\mathrm{NP}[g(n,0)])^A$ and $(\mathrm{NP}[g(n,2i)])^A \neq (\mathrm{NP}[g(n,2i+1)])^A = (\mathrm{NP}[g(n,2i+2)])^A \neq (\text{co-NP}[g(n,2i+2)])^A$ for all $i \ge 0$ (and in fact there are tally sets in $(\mathrm{NP}[g(n,2i+1)])^A - (\mathrm{NP}[g(n,2i)])^A$ and in $(\mathrm{NP}[g(n,2i+2)])^A - (\text{co-NP}[g(n,2i+2)])^A$ for each i).*

Sketch. For convenience, we will separate $(\mathrm{NP}[g(n,2i+1)])^A$ from $(\text{co-NP}[g(n,2i+1)])^A$ rather than $(\mathrm{NP}[g(n,2i+2)])^A$ from $(\text{co-NP}[g(n,2i+2)])^A$; given the other requirements, this is equivalent. In order to separate $(\mathrm{NP}[g(n,2i+1)])^A$ from $(\text{co-NP}[g(n,2i+1)])^A$, we use the set D^A_{2i+1} in $(\mathrm{NP}[g(n,2i+1)])^A$, and guarantee that $D^A_{2i+1} \notin (\text{co-NP}[g(n,2i+1)])^A$. Most of this construction is identical to that of Theorem 2.7, except that we interleave an extra diagonalization into the construction; the codings and diagonalizations are analogous to earlier constructions, and the counting argument is identical.

We code complete sets C^A_{2i} for $(\mathrm{NP}[g(n,2i)])^A$ into $(\mathrm{NP}[g(n,2i-1)])^A$, and diagonalize so that no $(\mathrm{NP}[g(n,2i)])^A$ machine recognizes $D^A_{2i+1} \in (\mathrm{NP}[g(n,2i+1)])^A$. (Thus D^A_{2i+1} does double duty: during even stages, it diagonalizes against $(\mathrm{NP}[g(n,2i+2)])^A$, and during odd stages, against $(\text{co-NP}[g(n,2i+1)])^A$.)

To guarantee that D^A_{2i+1} is not in $(\text{co-NP}[g(n,2i+1)])^A$, we make sure that, for each e, the e^{th} machine for $(\mathrm{NP}[g(n,2i+1)])^A$ does not recognize $\overline{D^A_{2i+1}}$. This holds if and only if there is some x such that $D^A_{2i+1}(x) = M^A_{e,2i+1}(x)$. This diagonalization differs from earlier ones only when $M^A_{e,2i+1}(x)$ queries a witness for $x \in D^A_{2i+1}$. As before, if $M^A_{e,2i+1}(x)$ queries a coding string for some computation of $C^A_{2j}(w)$ where $(2j > 2i)$ or $(2j = 2i$ and $w \neq x)$, then we can safely restrain the coding string. (If $2j = 2i$ and $w \neq x$, this may exclude w from D^A_{2i+1}, but that doesn't matter. As long as $D^A_{2i+1}(x) = M^A_{e,2i+1}(x)$, we don't care what happens to D^A_{2i+1} for other strings of lengths between n_{s-1} and n_s, where $s = \langle e, i \rangle$.) If $2j < 2i$, then we retrace the computation, as before.

If $M^A_{e,2i+1}(x)$ queries a witness for $x \in D^A_{2i+1}$, we first restrain all such witnesses, and continue. If this leads to a rejecting computation of $M^A_{e,2i+1}(x)$, then $M^A_{e,2i+1}(x) = D^A_{2i+1}(x)$, and the diagonalization is successful. If it leads to an accepting computation, we preserve the lexicographically least accepting path for that computation, and all of its cascade of queries. As before, the computation of $M^A_{e,2i+1}(x)$ restrains at most $2^{o(g(n,2i+1))}$ strings, so this will not restrain all the witnesses for $x \in D^A_{2i+1}$. Thus we can find an unrestrained witness and add it to A, so $D^A_{2i+1}(x) = M^A_{e,2i+1}(x)$, as desired.

Therefore, this additional set of diagonalization requirements can be interleaved with the previously-described diagonalizations and collapses. $\square$

THEOREM 2.11. *Let $g(\cdot,\cdot)$ be a polynomial-time computable, monotone increasing (in both variables) function with $\log n \in o(g(n,i))$ and $g(n,i) \in n^{O(1)}$ for all $i \geq 1$. If $g(n^{O(1)}, 2i) \subseteq o(g(n, 2i+1))$ for all $i \geq 1$, then there exists an oracle A such that $\mathrm{P}^A = (\mathrm{NP}[g(n,0)])^A$ and $(\mathrm{NP}[g(n,2i)])^A \neq (\mathrm{NP}[g(n,2i+1)])^A = (\mathrm{NP}[g(n,2i+2)])^A = (\text{co-NP}[g(n,2i+2)])^A$ for all $i \geq 0$ (and in fact there are tally sets in $(\mathrm{NP}[g(n,2i+1)])^A - (\mathrm{NP}[g(n,2i)])^A$ for each i).*

Sketch. As before, we construct A so that no $(\mathrm{NP}[g(n,2i)])^A$ machine recognizes the set D^A_{2i+1}, and so that $C^A_{2i+2} \in (\mathrm{NP}[g(n,2i+1)])^A$. In addition, in order to make $(\mathrm{NP}[g(n,2i+1)])^A = (\text{co-NP}[g(n,2i+1)])^A$, we code $\overline{C^A_{2i+1}}$ into $(\mathrm{NP}[g(n,2i+1)])^A$ as follows: $x \notin C^A_{2i+1} \iff \exists y[|y| = g(|x|, 2i+1) \wedge 21^{p(|x|,2i+1)^2}0x \in A]$.

For each i, let N^A_i be an $(\mathrm{NP}[g(n,i)])^A$ machine recognizing C^A_i in nondeterministic time bounded by $p(n,i)$ (regardless of the oracle). By the form of the encoding, $N^A_i(x)$ cannot query any of its own coding strings. If a witness string for $\overline{C^A_{2j+1}(x)}$ is queried in the course of a diagonalization $(\mathrm{NP}[g(n,2i)])^A \neq (\mathrm{NP}[g(n,2i+1)])^A$, and $2j+1 < 2i+1$, then we can retrace the computation. If $2j+1 \geq 2i+1$, then we can restrain the queried witness string (for $|x|$ sufficiently large) without deciding $\overline{C^A_{2j+1}(x)}$, by the same counting argument as in previous proofs.

Thus, we can add this extra encoding, without interfering with the other collapses and codings. $\square$

This gives us the following stronger version of Hemachandra and Jha's oracle [8].

COROLLARY 2.12. *For each k, there is an oracle relative to which for all j, $\beta_j = \text{co-}\beta_j$, and*

$$\mathrm{P} = \beta_k \neq \beta_{k+1} \neq \beta_{k+2} \neq \cdots$$

(and the separations are witnessed by tally sets).

Combining the results (and techniques) of Theorems 2.7, 2.10, and 2.11, we get the following very strong result.

COROLLARY 2.13. *For any consistent set of requirements, there is an oracle relative to which the β hierarchy satisfies those requirements.*

In constructing such an oracle, one must be careful in closing classes under complement. In particular, if we close one class under complement, and separate another from its complement, we cannot then make the two classes equal.

Corollary 2.9 implies that there are uncountably many different patterns of collapse that can be realized in relativized worlds. If the set of requirements is recursive, then the oracle can be made recursive, but certainly some of those patterns are realized by only nonrecursive oracles.

3. Dense β Hierarchies. Previously we considered β_r only when r is a natural number. But the class $(\mathrm{NP}[\log^r n])^A$ is meaningful whenever r is a nonnegative real number (regardless of whether r is computable). Even when we allow r to be real, we can make the β hierarchy obey any consistent set of requirements. For example, we can make the β hierarchy look like a Cantor set.

THEOREM 3.1. *Let X be any subset of $[1,\infty)$. There exists an oracle A such that, for all $s,t \geq 1$, $\beta_s^A = \beta_t^A$ if and only if $[s,t] \subseteq X$.*

Note that there may be uncountably many distinct β_ts. Because there are two rationals between any two reals, we need only separate the distinct β_qs where q is rational.

Proof. Without loss of generality, assume that X is a union of intervals, each containing more than one point. Every interval in X contains a rational point; therefore X contains countably many intervals.

We will satisfy the following requirements for each maximal interval in X, depending on its type:
$[s,t]$: $\mathrm{NP}[\log^s n] = \mathrm{NP}[\log^t n]$
$[s,t)$: $\mathrm{NP}[\log^s n] = \mathrm{NP}[\log^t n/\log\log n]$
$(s,t]$: $\mathrm{NP}[\log^s n\log\log n] = \mathrm{NP}[\log^t n]$
(s,t) : $\mathrm{NP}[\log^s n\log\log n] = \mathrm{NP}[\log^t n/\log\log n]$
$[s,\infty)$: $\mathrm{NP}[\log^s n] = \mathrm{NP}[n]$
(s,∞) : $\mathrm{NP}[\log^s n\log\log n] = \mathrm{NP}[n]$
In addition, for each rational number q in $(1,\infty) - X$, we will make

$$\mathrm{NP}[\log^q n/\log\log n] \neq \mathrm{NP}[\log^q n] \neq \mathrm{NP}[\log^q n\log\log n].$$

If $1 \notin X$, then we make $\mathrm{P} \neq \mathrm{NP}[\log n\log\log n]$ as well.

The construction is a slight modification of that in the proof of Theorem 2.7. We perform the diagonalizations in some well-founded order, while maintaining the codings as we go along. The only significant difference here is that the diagonalizations are not performed in increasing order. Suppose that at some stage we are making $\mathrm{NP}[a(n)] \subset \mathrm{NP}[b(n)]$ and a coding string for some $\mathrm{NP}[c(n)]$ computation is queried; we restrain that string if and only if $(\exists n)[c(n) > b(n)]$ if and only if $(\forall n)[c(n) \geq b(n)]$. The counting argument is the same as before. □

Note: we could also close each distinct β_r under complement or not, as we wish, in the theorem above.

4. Open Problems. The class β_k is contained in $\mathrm{NP} \cap \mathrm{DTIME}(2^{\log^k n})$. Our work shows that there is no relativizing proof that $\mathrm{P} = \beta_2 \Rightarrow \mathrm{P} = \mathrm{NP}$. We would like to know whether $\mathrm{P} = \mathrm{NP} \cap \mathrm{DTIME}(n^{\log n})$ implies $\mathrm{P} = \mathrm{NP}$, i.e., if $\mathrm{P} \neq \mathrm{NP}$ are there any easy languages in $\mathrm{NP} - \mathrm{P}$? The best we can show is that if $\mathrm{P} = \mathrm{NP} \cap \mathrm{DTIME}(f(n))$ for a well-behaved function f, then $\mathrm{P} = \mathrm{NP} \cap \mathrm{DTIME}(f(f(n)))$. Is there an oracle relative to which this is the best possible translation of the collapse? Does $\mathrm{P} = \mathrm{NP} \cap \mathrm{DTIME}(2^{\log^k n})$ imply $\mathrm{P} = \mathrm{NP}$?

Acknowledgments. We are grateful to Leen Torenvliet, Andrew Klapper, and Martin Kummer for helpful discussions, and Andrew Klapper, Bill Gasarch, and Martin Kummer for careful proofreading of earlier drafts.

REFERENCES

[1] C. Àlvarez, J. Díaz and J. Torán, "Complexity Classes With Complete Problems Between P and NP-Complete," in *Foundations of Computation Theory,* Lecture Notes in Computer Science 380, Springer-Verlag, 1989, pp. 13-24.

[2] T. Baker, J. Gill and R. Solovay. Relativizations of the P = NP question. *SICOMP* 4:431-442, 1975.

[3] J. F. Buss and J. Goldsmith, "Nondeterminism within P," *SIAM Journal of Computing* **22** (1993), 560–572.

[4] J. Cai, T. Gundermann, J. Hartmanis, L. A. Hemachandra, V. Sewelson, K. W. Wagner, and G. Wechsung. The Boolean hierarchy I: structural properties. *SICOMP*, 17(6):1232–1252, Dec. 1988.

[5] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28:114–133, 1981.

[6] J. Díaz and J. Torán. Classes of bounded nondeterminism. *MST*, 23:21–32, 1990.

[7] J. Goldsmith, M. Levy, and M. Mundhenk, Limited nondeterminism *SIGACT News,* pages 20–29, June 1996.

[8] L. Hemachandra and S. Jha. Defying upward and downward separation. In *10th Annual Symposium on Theoretical Aspects of Computer Science* pages 185–195. Springer-Verlag, 1993. LNCS 665.

[9] C. M. R. Kintala. *Computations with a restricted number of nondeterministic steps.* PhD thesis, Pennsylvania State University, University Park, PA, 1977.

[10] C. M. R. Kintala and P. C. Fischer. Computations with a restricted number of nondeterministic steps. In *9th Annual Symp. Theory of Comput.* pages 178–185, 1977.

[11] C. M. R. Kintala and P. C. Fischer. Refining nondeterminism in relativized polynomial-time bounded computations. *SICOMP*, 9(1):46–53, Feb. 1980.

[12] K.-I. Ko. Relativized polynomial hierarchies having exactly k levels. *SICOMP*, 18:392–408, 1989.

[13] S. Kurtz. Sparse sets in NP − P: relativizations. *SICOMP*, 14:113–119.

[14] R. E. Ladner. On the structure of polynomial time reducibility. *JACM,* 22(1):155-171, Jan. 1975.

[15] C. H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the V–C dimension. In *8th Ann. Conf. Structure in Complexity Theory* pages 12–18, 1993.

[16] V. R. Pratt. Every prime has a succinct certificate. *SICOMP,* 4(3):214–220, 1975.

[17] L. Sanchis, Constructing language instances based on partial information. *International Jour. Found. Comp. Sci.,* 5(2):209–229. 1994.