Classes of bounded nondeterminism

J. Díaz
J. Torán

Report LSI–88–6

CLASSES OF BOUNDED NONDETERMINISM

Josep Díaz* and Jacobo Torán
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028 Barcelona

Abstract

We study certain language classes located between P and NP that are defined by polynomial time machines with a bounded amount of nondeterminism. We observe that these classes have complete problems and find a characterization of the classes using robust machines with bounded access to the oracle, obtaining some other results in this direction. We also study questions related with the existence of complete tally sets in these classes and closure of the classes under different types of polynomial time reducibilities.

1. Introduction and basic definitions

If $P \neq NP$, are there classes with "natural" complete problems between P and the class of NP-complete problems (NP-C)? It is well known, that if $P \neq NP$ there are sets of intermediate complexity, lying between P and NP-complete ([La 75]; see also chapter 7 of [Ba,Di,Ga 88]). As a matter of fact, several classes of sets have been proposed, which include the class P and do not seem to be complete for NP; among these classes we could mention the class UP of NP languages that can be accepted by nondeterministic polynomial time machines with unique accepting paths [Va 76], the class FewP of languages recognized by nondeterministic Turing machines with a bounded number of accepting paths [Al 85], [Ca,He 89], [Kö,Sc,To,To 89], the class R of languages accepted by polynomially clocked probabilistic machines which have zero error probability for inputs not in the language and error probability bounded by $\varepsilon < \frac{1}{2}$ for strings in the language [Gi 77], and the class ZPP of languages which are recognized in polynomial time by Las Vegas type algorithms [Gi 77] (see also chapter 6 of [Ba,Di,Ga 88]). A common characteristic of these

* The research of this author was supported by CIRIT grant EE87/2

classes is the fact that the existence of complete problems for any of them is not known [Ha,He 86].

In this work we study complexity classes that have "natural" complete problems and appear to be strictly between P and NP-complete. These classes arise considering problems that can be solved in polynomial time using a small amount of nondeterminism:

Definition: For any function $f : \mathbb{N} \longrightarrow \mathbb{N}$ let

$$\beta_f = \{L \subseteq \Sigma^* \mid \exists A \in P, \exists c \in \mathbb{N}, \forall x \in \Sigma^* : \ x \in L \Leftrightarrow (\exists y, |y| \leq c \cdot f(|x|) \text{ and } \langle x, y\rangle \in A)\}.$$

Recall that from the classical characterization of NP in terms of quantifiers we have:

$$NP = \{L \subseteq \Sigma^* \mid \exists A \in P, \exists k \in \mathbb{N}, \ \forall x \in \Sigma^* : x \in L \Leftrightarrow (\exists y, |y| \leq |x|^k \text{ and} \langle x, y\rangle \in A)\}.$$

The class β_f is therefore the class of problems which can be recognized in polynomial time by a nondeterministic machine which for any input w uses at most an "$O(f(|w|))$ amount of nondeterminism".

From the definition it follows immediately that $P = \beta_{\log(n)}$ and $NP = \bigcup_{k \geq 1} \beta_{n^k}$.

Although every "reasonable" function f defines a complexity class β_f, in this work we will only consider the classes defined by the polylogarithmic functions $f(n) = \log^k(n)$. These classes satisfy certain basic closure properties (for example closure under polynomial time many-one reducibility), and define a hierarchy of complexity classes that seems to be strict. For succintness, we abbreviate $\beta_{\log^k(n)}$ as β_k for $k \geq 1$. We also define β_{polylog} as the union of these classes, i.e., $\beta_{\text{polylog}} = \bigcup_{k \geq 1} \beta_k$.

We have then

$$P = \beta_1 \subseteq \beta_2 \subseteq \ldots \subseteq \beta_k \subseteq \ldots \subseteq \beta_{\text{polylog}} \subseteq \ldots \subseteq NP.$$

The classes of bounded nondeterminism are not new. The classes in β_{polylog} appeared in [Ki,Fi 84] illustrating the issue of relativizations separating P from NP. In the article by Kintala and Fisher, some oracles separating every class in β_{polylog} are found. Also these classes are briefly mentioned in [Ba 84], [Sc,Bo 84], and [Xu,Do,Bo 83]. We should remark that none of the mentioned papers goes into a deep study of the classes themselves or the relations between them and some other structural classes placed in the "no man's land" between P and NP-complete.

From the results in [Ki,Fi 84] the hierarchy of β_k-classes seems to be strict although the result is hard to prove since it implies P$\neq$NP. Using padding arguments we can see that a collapse in the hierarchy has consequences on other complexity classes. This result has been pointed to us by R. Beigel.

Theorem 1: Let j and k be two natural numbers with $j \leq k$. If $\beta_j = \beta_k$ then

(i) $SAT \in DTIME(2^{n^{j/k}})$

(ii) $NTIME(n) \subseteq DTIME(2^{n^{j/k}})$.

Proof: We show (i). (ii) is analogous. Let $L = \{0^m\#x \mid x \in SAT \text{ and } |x| = \log^k(m)\}$. Then L is in β_k, and by the hypothesis L is in β_j. Simulating the nondeterminism we have $L \in DTIME(2^{\log^j(n)})$ and $SAT \in \beta_{n^{j/k}}$, which implies $SAT \in DTIME(2^{n^{j/k}})$. □

In this article we show that all the β_k classes have natural complete problems, which are obtained by a simple modification of certain complete problems for P, i.e., by adding nondeterminism. It is interesting to observe that by the complementary process of restricting the nondeterminism in natural NP-complete problems, we do not seem to obtain complete problems for the β-classes.

Although the complete problems for β_k are simple variations of complete problems for P, the classes seem to be much closer to NP than to P and many of the results obtained for the class NP can be translated to the classes β_k. Nevertheless, not all of the results that hold for NP seem to be true for the β-classes; for example, for every k, β_k is closed under disjunctive polynomial time reducibility but does not seem to be closed under conjunctive polynomial time reducibility. (In section 3 we present a relativization under which β_k is not closed under conjunctive reducibility).

We also obtain (section 4) a characterization of the classes in β_{polylog} in terms of one-way robust Turing machines, showing that the small amount of nondeterminism defining the classes can be simulated by bounding the number of questions to the oracle of a deterministic robust machine. We observe an unusual phenomenon related with the β-classes and the robust machines, presenting a relativized world in which β_k helps (in the sense of [Ko 87]) a class strictly bigger than β_k.

2. Complete problems in β_k

Let us start by showing the existence of complete problems in β_k. In order to enumerate the machines recognizing languages in β_k, we define a machine model for this class.

Definition: For every k, a β_k-machine is a nondeterministic polynomial time machine equiped with a clock, that for a certain constant $c > 0$ works as follows:

> Given input x, the machine first writes nondeterministically $c\log^k(|x|)$ symbols on one of its working tapes. Thereafter, the machine works deterministically, and if it has to take a nondeterministic decision, the machine follows the path given by the written nondeterministic string.

It should be clear that these machines recognize the languages in β_k, and that for every k there exists a recursive enumeration of the β_k machines. We can suppose that machine M_i with an input of length n can make $i \cdot \log^k(n)$ nondeterministic choices. From this enumeration we define in a canonical way the following β_k-complete language

$$L_k = \{\langle i, x, 0^n\rangle \mid M_i \text{ accepts } x \text{ in } n \text{ or less steps } \},$$

where, as we have said, M_i is a nondeterministic machine using $i{\cdot}\log^k(|x|)$ nondeterministic choices. Observe that we can also enumerate all the machines computing languages in β_{polylog} and obtain in the same way complete languages for this class.

We define now more "natural" complete problems for β-classes. These problems are found introducing a certain amount of "controlled nondeterminism" into P-complete problems. Consider the following variation of the circuit value problem from [La 75], which is β_k-complete.

β_k-CVP:

Input: A pair $\langle x, y\rangle$ such that $x \in \{0,1\}^\star$ and y encodes a boolean circuit with $|x| + \lceil\log(|x|)\rceil$ input gates.

Question: Is there a string $z \in \{0,1\}^\star$ of length $\lceil\log(|x|)\rceil$ such that the circuit encoded by y with input xz outputs a 1 ?

Theorem 2: For every $k \geq 1$, β_k-CVP is β_k-complete under logarithmic space many-one reducibility.

The hardness proof is essentially the same as the one given by Ladner in [La 75], but using the β_k-machine model defined in the previous section. The second stage of the β_k-machine (when all the nondeterministic bits have been written) can be simulated by a boolean circuit, and the nondeterministic bits of the machine correspond to the input bits that are not fixed in the circuit.

It is not hard to introduce nondeterminism in some other P-complete problems, like, for example, the generator problem from [Jo,La 76], making it β_k-complete.

β_k-GEN:

Input: A set X of elements; a binary operation $\circ$ defined on X and explicitly given by a table and defined in such a way that in $\log^k(|X|)$ cases the operation $\circ$ can have two possible values; a subset $S \subset X$; and a distinguished element x of X.

Question: Is there a way to define the $\circ$ operation unambiguously such that x is contained in the smallest subset of X which contains S and is closed under the given operation $\circ$?

Observe that the two problems above as well as a small modification of L_k, are self-reducible, a fact that will be used in next sections. Another interesting point is that by restricting the amount of nondeterminism in NP-complete problems, we do not seem to obtain complete problems for β_k. For example, the satisfiability problem for formulas with only a polylogarithmic amount of variables is in $ATIME(polylog)$ [Bu 87], and therefore it may not even be hard for P. In spite of this fact, as we will see in the next sections, the β-classes share many of the properties of the class NP.

3. Closure under polynomial time reducibilities

The β-classes present interesting properties concerning their closure under polynomial time reducibilities. It is easy to see that they are closed under many-one polynomial time reducibility, and they do not seem to be closed under polynomial time Turing reducibility (it is not hard to find a relativization in which these classes are not closed under this reducibility; in fact the relativization presented in the previous section can be used for this purpose). So far these closure properties are the same as for the class NP; the interesting aspect arises when we consider polynomial time truth table reducibility, and more specifically disjunctive and conjunctive reducibilities. Recall that a set A is conjunctive reducible to a set B, if there is a function $f \in$ PF such that for every string x, $f(x)$ computes a list of strings $y_1, \ldots, y_{p(|x|)}$, and $x \in A$ if and only if $y_i \in B$ for each i, $1 \leq i \leq p(|x|)$. Analogously, A is disjunctive reducible to B if there is a function $f \in$ PF computing a list $y_1, \ldots, y_{p(|x|)}$, and $x \in A$ if and only if $y_i \in B$ for some i, $1 \leq i \leq p(|x|)$. The β-classes are closed under disjunctive reducibility but do not seem to be closed under conjunctive reducibility. Again, this last remark implies P$\neq$NP, and therefore we are only able to prove it in a relativized world.

Theorem 3: For every k, the class β_k is closed under polynomial time disjunctive reducibility.

Proof: Let B be a set in β_k, and A a set disjunctive reducible to B via a function $f \in$ PF bounded by a polynomial q. To decide if a string x belongs to A, compute the list of strings $y_1, \ldots, y_{p(|x|)}$ given by $f(x)$, choose one of the strings y_i ($\log(p(|x|))$ nondeterministic bits) and then guess a string of length $\log^k(q(|y_i|))$ to witness the membership of y_i in B.

The algorithm runs in polynomial time and uses a $O(\log^k(n))$ amount of nondeterminism. Therefore A is in β_k. □

Theorem 4: For every $k \geq 2$ there is a relativization in which the class β_k is not closed under polynomial time conjunctive reducibility. This also holds for β_{polylog}.

Proof: We will find an oracle B and a language L such that L is conjunctive reducible to β_2^B, but L is not in β_{polylog}^B.

For every set B, consider the test language

$$L_B = \{0^n \mid \forall u, |u| = \lceil \log(n) \rceil, \exists v, |v| = \lceil \log^2(n) \rceil \text{ and } uv \in B\}.$$

It is clear that for every B, L_B is conjunctive reducible to a set in $\beta_2(B)$. As we will see, using counting arguments, a set B can be found such that $L_B \notin \beta_{\text{polylog}}^B$.

Let $M_1, M_2 \ldots$ be an enumeration of β_{polylog} oracle machines, with the computation time of M_s bounded by polynomial p_s and its nondeterminism bounded by $s\log^s$. We will construct set B in stages:

stage 0: $B_0 = \emptyset$; $n_0 = 0$.

stage s: Let n_s be the smallest m such that m is a power of 2, $\log^2(m) > p_i(n_i)$ for $i < s$, and

$$\frac{2^{\log^2(m)m}}{\binom{p_s(m)}{m}} > 2^{s\log^s(m)}.$$

Let $\{x_1, \ldots, x_{n_s}\}$ be the set of all strings of length $\log(n_s)$.
If $0^{n_s} \in L(M_s, B_{s-1})$ then $B_s = B_{s-1}$
else

1. If there are n_s strings $w_1, \ldots, w_{n_s}$ (not neccessarily different) of length $\log^2(n_s)$ such that $0^{n_s} \notin L(M_s, B_{s-1} \cup \{x_1w_1, \ldots, x_{n_s}w_{n_s}\})$ then let $B_s = B_{s-1} \cup \{x_1w_1, \ldots, x_{n_s}w_{n_s}\}$.
2. If the condition in 1 is not true then there is an integer r, $(r < n_s)$, and r strings of length $\log^2(n_s)$, $w_1, \ldots, w_r$ (not neccessarily different), such that $0^{n_s} \in L(M_s, B_{s-1} \cup \{x_1w_1, \ldots, x_rw_r\})$. Let $B_s = B_{s-1} \cup \{x_1w_1, \ldots, x_rw_r\}$.

end (of stage s)

Let $B = \bigcup_s B_s$. Following similar arguments as in [Ba,Gi,So 75] it is not hard to check that $L_B \notin \beta_{\text{polylog}}^B$. It is only left to show that the assertion in 2 is true.

Claim: For every n_s, if for every sequence of n_s strings $\{w_1, \ldots, w_{n_s}\}$ (not neccessarily different) of length $\log^2(n_s)$, $0^{n_s} \in L(M_s, B_{s-1} \cup \{x_1w_1, \ldots, x_sw_{n_s}\})$ then there is an integer r, $r < n_s$, and r strings $\{w_1, \ldots, w_r\}$ (not neccessarily different) of length $\log^2(n)$, such that $0^{n_s} \in L(M_s, B_{s-1} \cup \{x_1w_1, \ldots, x_rw_{n_r}\})$.

Proof of the claim: Given n_s, we will call a string of $s\log^s(n_s)$ bits a computation path of M_s, since it forces the computation of M_s (for fixed input and a fixed orace). Observe that there are at the most $2^{s\log^s(n_s)}$ computation paths, and that this is independent of the oracle we use.

Suppose that the claim is not true. Then for every tuple of n_s strings $\{w_1,\ldots,w_{n_s}\}$ of length $\log^2(n_s)$, every accepting path of machine M_s with input 0^{n_s} and oracle $B_{s-1}\cup\{x_1w_1,\ldots,x_{n_s}w_{n_s}\}$ must query all the strings $x_1w_1,\ldots,x_{n_s}w_{n_s}$ and all the answers must be "yes".

We will say that a tuple of n_s strings $\{w_1,\ldots,w_{n_s}\}$ "is included" in a computation path if every element from the tuple $\{x_1w_1,\ldots,x_sw_{n_s}\}$ is queried by M_s on that computation path with input 0^{n_s} and oracle $B_{s-1}\cup\{x_1w_1,\ldots,x_sw_{n_s}\}$. Observe that although in a computation path the range of oracle queries that can be made could be very big (the machines are non-adaptive), the number of tuples of n_s strings "included" in a computation path is at the most $\binom{p_s(n_s)}{n_s}$. The explanation for this is that there are at the most as many tuples included in a computation path as there are sequences of $p_s(n_s)$ oracle answers including exactly n_s "yesses".

But then for every tuple $\{w_1,\ldots,w_{n_s}\}$ there is an accepting path that contains exactly the mentioned strings answered "yes".

From the above two facts it follows that the number of computation paths of M_s that accept 0^n for some oracle $B_{s-1}\cup\{x_1w_1,\ldots,x_sw_{n_s}\}$ is greater or equal than the number of different tuples $w_1,\ldots,w_{n_s}$ (considering the order and with repetitions allowed) divided by the number of tuples that can be "included" in a computation path. But this number is

$$\frac{2^{\log^2(n_s)n_s}}{\binom{p_s(n_s)}{n_s}}$$

which is strictly greater than $2^{s\log^s(n_s)}$, the number of possible computation paths of M_s with input 0^{n_s}. This is a contradiction, and therefore assertion 2 has to be true. □

4. Robust machines and the β-classes

In this section we give a characterization of the β-classes using robust machines i.e. oracle machines that recognize the same language independently from the oracle they use. This kind of machines were first introduced in [Sc 85], and also studied in [Ko 87], and [Ha,He 87].

Definition: For any language L, $L\in P_{1\text{-help}}$ if and only if there is a deterministic polynomial time bounded machine M and an oracle A such that:

i/ For every $x\in L$, $x\in L(M,A)$ and

ii/ For every oracle B and every $x\notin L$, $x\notin L(M,B)$.

For a language class K, a language L is in $P^K_{\text{1-help}}$ if $L \in P_{\text{1-help}}$ and the oracle A satisfying condition i/ in the definition above is in the class K.

By bounding the access to an NP oracle, we can characterize the β-classes in terms of robust machines. Between square brackets, we indicate the amount of oracle queries allowed.

Theorem 5: For every $k \geq 2$,
i/ $P_{1-\text{help}}^{NP\,[\log^k]} = P_{1-\text{help}}^{\beta_k\,[\log^k]} = \beta_k$
ii/ $\beta_k \subseteq P^{\beta_k}_{1-\text{help}}$.

Proof: (Sketch) i/ The inclusions from left to right follow from the fact that an accepting computation path from a deterministic machine that makes $\log^k$ questions to the oracle can be found guessing $\log^k$ bits; since the machine is robust, if a certain input is not in the language recognized by the machine, none of the possible paths that can be guessed for this input will accept.

The first inclusion from right to left is straightforward. For the other inclusion, given a language $L \in \beta_k$ which for a certain polynomial time predicate P satisfies:

$$L = \{x \mid \exists y, |y| \leq \log^k(|x|) \text{ and } \langle x,y\rangle \in P\} \quad \text{let}$$

$$\text{Pref}(L) = \{\langle x,y\rangle \mid \exists y',\ |y| + |y'| = \log^k(|x|) \text{ and } \langle x, yy'\rangle \in P\}.$$

Clearly Pref(L) is also in β_k. Any set L in β_k can be semi-decided following the usual methods by a deterministic robust machine doing binary search in oracle Pref(L). This proves also result ii/. □

Result ii/ only seems to hold in one direction; consider a language L in β_k and the language $L' = \{\langle a_1, a_2, \ldots, a_n\rangle \mid a_i \in L \text{ for } i = 1 \ldots n\}$. Clearly L' is in $P^{\beta_k}_{1-\text{help}}$ since in order to check that a given string w is in L', the robust machine only needs to ask the oracle for the witnesses of all the substrings in w, and then check that they are correct. This can be done in polynomial time with a "good" oracle in β_k. On the other hand, L' does not seem to be in β_k since in order to check the validity of an input, $n \log^k$ nondeterministic bits are needed.

This is a strange phenomenon since all the classes that have appeared in the literature helping some other class in the sense of one-way robust machines, can only help themselves or some of their subclasses, ($P^{NP}_{\text{1-help}} = NP$, $P^{P}_{\text{1-help}} = P$, $P^{BPP}_{\text{1-help}} \subseteq R$, [Sc 85], [Ko 87]), and β_k seems to help a class strictly greater than itself. It would be very hard to prove that β_k is strictly contained in $P^{\beta_k}_{\text{1-help}}$ since it would immediately imply $P \neq NP$, but we still can prove the result in a relativized world, in fact, we can use for this the relativization from theorem 4.

In order to do this we first have to define relativized robust machines.

Definition: For any language L and any oracle B, $L \in (P_{\text{1-help}})^B$ if and only if there is a deterministic polynomial time bounded machine M and an oracle A such that:
i/ For every $x \in L$, $x \in L(M, A \oplus B)$ and
ii/ For every oracle D and every $x \notin L$, $x \notin L(M, D \oplus B)$.

Intuitively a relativized robust machine is just a deterministic machine with an oracle which is the join of two sets. Questions to one of the sets in the join are always answered correctly, but the questions to the other set have to be verified by the machine. It is not hard to see following the same techniques as in [Sc 85], [Ko 87] that for any oracle B, $(P_{\text{1-help}})^B = NP^B$.

As in the unrelativized case, we can define consider the case in which the helping oracle is in a certain complexity class.

Definition: Let K be a language class that can be relativized. For any language L and any oracle B, $L \in (P^K_{\text{1-help}})^B$ if and only if there is a deterministic polynomial time bounded machine M and an oracle $A \in K^B$ such that:
i/ For every $x \in L$, $x \in L(M, A \oplus B)$ and
ii/ For every oracle D and every $x \notin L$, $x \notin L(M, D \oplus B)$.

Observe that in the above definition we also relativize the helping class. The motivation for this is that if we give more power to the robust machine by letting it access "two" oracles, we also have to give more power to the helping class or it will not be able to help the machine. It is also easy to prove that for any oracle B, $(P^{NP}_{\text{1-help}})^B = NP^B$.

Theorem 6: For any $k \geq 2$, there is an oracle B such that $(P^{\beta_k}_{\text{1-help}})^B \not\subseteq \beta_k^B$.

Proof: For every set B, consider the test language

$$L_B = \{0^n \mid \forall u, |u| = \lceil \log(n) \rceil, \exists v, |v| = \lceil \log^2(n) \rceil \text{ and } uv \in B\}.$$

In theorem 4 we showed that there is a set B such that for every k, $L_B \notin \beta_k$, therefore we will just show that for every set B, $L_B \in (P^{\beta_2}_{\text{1-help}})^B$. (This obviously implies $L_B \in (P^{\beta_k}_{\text{1-help}})^B$ for every $k \geq 2$).

Consider the set

$$B' = \{0^n \# x \# u \mid |x| = \lceil \log(n) \rceil \text{ and } \exists v \ |u| + |v| = \lceil \log^2(n) \rceil \text{ and } xuv \in B\}.$$

Clearly, $B' \in \beta_2^B$ since in order to check that a given input is in the set, we only need to guess $\log^2$ bits, at the most, and then make one query to B.

Informally we describe now a deterministic one-way robust machine that recognizes L_B using oracle B': With input 0^n, cycle over all strings x of length $\lceil \log(n) \rceil$, and for each one of them, doing binary search in the oracle, find a string w of length $\lceil \log^2(n) \rceil$ in B. Finally check that the string obtained, xw is in B by directly asking oracle B. If this can be done for every string x of length $\lceil \log(n) \rceil$ then accept. □

5. Similarities between β and NP

Although we have seen that β_k-complete problems can be obtained with small modifications from P-complete problems, we will see in this section several results which indicate that the classes β_k share many of the properties of the class NP. We will begin by re-stating in terms of β_k an old result of Adleman [Ad 79] which says that the difference between P and NP is the ability of the NP-machines to manufacture randomness. In the next theorem, we prove that this difference also exists between P and β_k, which in a certain sense means that the ability to manufacture randomness is independent of the amount of nondeterminism used by the machine. Our proof follows directly a similar one in [He,We 89]. We introduce first two definitions.

Definition: Given a β_k machine M and an input x, a certificate of M on x is an accepting path of $M(x)$.

Definition: Let M_U be a universal Turing machine, $z \in \Sigma^\star$, and $f, g : \mathbb{N} \longrightarrow \mathbb{N}$. Define the Kolmogorov complexity relative to string z as

$$K[f(n), g(n) \mid z] = \{x \mid \exists y,\ |y| \leq f(|x|) \wedge M_u(y\#z) \text{ prints } x \text{ within } g(x) \text{ steps}\}.$$

We say that a string x is Kolgomorov simple relative to z if there is a constant c such that $x \in K[c \log n, n^c | z]$.

Theorem 7: $P = \beta_k$ iff β_k has Kolmogorov simple certificates relative to the input.

Proof. If $P = \beta_k$, by using the self-reducibility of β_k-CVP we can find certificates for any instance in the language. In the other direction, there are only polynomially many simple strings relative to the input and we can produce all of them in polynomial time, and check if at least one of them is a true certificate. □

Besides this similarity between the sets accepted by machines using a polynomial amount of nondeterminism and a $\log^k$ amount of nondeterminism, there are other common characteristics. One of them is that β_k cannot have complete tally sets unless $P = \beta_k$. This result is similar to the well known result of Berman about the non-existence of NP-complete tally sets if $P \neq NP$. The result is given in the following theorem, whose proof is practically the same as the one given by Berman [Ber 78], but using the set β_k-CVP, shown previously to be complete in β_k, and self-reducible. The details of the proof are left to the reader.

Theorem 8: If there exists a co-sparse and β_k-complete set, then $P = \beta_k$.

As a corollary to this last theorem we can state

Corollary: If there exists a tally β_k-complete set, then $P = \beta_k$.

It is an open question whether the result holds also for sparse sets. Mahaney's proof for sets in NP [Ma 80], cannot be carried over to β since it needs a massive use of nondeterminism.

It is also interesting to compare the ability of β_k-machines to compress queries to an oracle in NP with the ability of NP-machines to do the same thing. It is well known that an NP-machine with access to an orace in NP can be simulated by another machine that queries the oracle just once ($NP^{NP} = NP^{NP[1]}$). We present results in this line for the class β_k. Recall that between the square brackets we indicate the number of oracle queries allowed.

Theorem 9: For every k, and $j \geq k$,
i/ $\beta_k^{NP[\log^j(n)]} \subseteq \beta_j^{NP[2]}$.
ii/ $\beta_k^{NP[\log^k(n)]} = \beta_k^{NP[2]}$

Proof: We sketch i/. ii/ is a corollary. A β_k-machine M making $\log^j$ queries to an oracle in NP can be simulated by another machine M' making just 2 queries to another NP oracle in the following way: M' guesses a computation path of M and a list of oracle answers from the oracle ($O(\log^j(n))$ bits). With this information M' computes the list of oracle queries corresponding to the guessed answers and the guessed computation path, and needs just two queries to the new oracle to check that all the queries corresponding to a (guessed) positive answer are in the old oracle, and all the queries corresponding to a negative answer are not in the old oracle. Since NP is closed under disjunctive and conjunctive truth table reducibilities, the new oracle is also in NP. □

Final remarks.

We have presented a hierarchy of β-classes, which are located between P and NP-complete and which have complete problems. We have characterized these classes in terms of robust machines, introducing the concept of relativized robust machine. Under the hypothesis $P \neq \beta_k$, we obtained a negative results about the existence of complete tally sets in the β-classes.

The relationship between the β-classes and other complexity classes that lie between P and NP, like R or ZPP, is an interesting open question since any result in this direction will clarify the relationship between nondeterminisim and probabilism. We do not know either whether the β-classes lie whithin Schöning's Low or High hierarchies (see [Sc 86]), but we conjecture that the classes are not included in either of them.

Acknowledgements: The authors sincerely thank Richard Beigel for pointing to us theorem 1, and Ron Book and the anonymous referee for very helpful comments on the paper.

References

[Ad 79] L. Adleman: Time, Space and Randomness. Techn. Report MIT/LCS/TM-131, MIT 1979.

[Al 85] E. Allender: *Invertible functions.* Ph.D. dissertation, Georgia Institute of Technology, 1985.

[Ba,Gi,So 75] I. Baker, J. Gill, R. Solovay: Relativization of the P=?NP question. *SIAM J. Comput.*, 4, 1975, 431–442.

[Ba 84] J.L. Balcázar: *En torno a oraculos que ciertas maquinas consultan, y las espantables consecuencias a que ello da lugar.* Ph.D. dissertation, FIB. 1984.

[Ba,Di,Ga 88] J.L. Balcázar, J.Diaz, J.Gabarró: *Structural Complexity*, (Vol.1) 1988, Springer-Verlag.

[Be 78] P. Berman: Relationships between density and deterministic complexity of NP-complete languages. *Proceedings 5th ICALP*, 1978, 63–72.

[Bo 74] R. Book: Tally languages and complexity classes. *Information and Control* , 1974, 186–193.

[Bu 87] S. Buss: The boolean formula value problem is in ALOGTIME. *19th STOC* 1987, 123–131.

[Ca,He 89] J. Cai, L. Hemachandra: On the power of parity. *Symp. Theor. Aspects of Comput. Sci.* 1989, 229–240.

[Gi 77] J. Gill: Computational complexity of probabilistic Turing machines. *SIAM J. Comput.*, **6**, 1977, 675–695.

[Ha,He 86] J. Hartmanis, L. Hemachandra: Complexity classes without machines: on complete languages for UP. *13th. ICALP*, 1986, 123–135.

[Ha,He 87] J. Hartmanis, L. Hemachandra: One-way functions, robustness, and the non-isomorphism of NP-complete sets. *2nd Structure in Complexity Theory Conference*, 1987, 160–175.

[He,We 89] L. Hemachandra, G. Wechsung: Using randomness to characterize the complexity of computation. To appear in *IFIP* 89.

[Jo,La 76] N. Jones, W. Laaser: Complete problems for deterministic polynomial time. *Theoretical Computer Science*, **3**, 1976, 105–118.

[Ki,Fi 84] C. Kintala, P .Fisher: Refining nondeterminism in relativized complexity classes. *SIAM J. Comput.*, **13**, 1984, 329–337.

[Ko 87] K. Ko: On helping by robust oracle machines. *Theoretical Computer Science*, **52**, 1987, 15–36.

[Kö,Sc,To,To 89] J. Köbler, U. Schöning, S. Toda, J. Torán: Turing machines with few accepting computations and low sets for PP. To appear in *4th Structure in Complexity Theory Conference*, 1989.

[La 75] R. Ladner: On the structure of polynomial time reducibility. *J.ACM*, **22** 1975, 155–171.

[Ma 80] S. Mahaney: Sparse complete sets for NP: solution of a conjecture of Berman and Hartmanis. In *Proceedings IEEE Symposium on Foundations of Computer Science*, 1980, 54–60.

[Sc 85] U. Schöning: Robust algorithms: a different approach to oracles. *Theoretical Computer Science*, **40**, 1985, 57–66.

[Sc 86] U. Schöning: *Complexity and Structure.* Lecture Notes in Computer Science, 1986, Springer-Verlag.

[Sc,Bo 84] U. Schöning, R. Book: Immunity, relativizations, and nondeterminism. *SIAM J. Comput.*, **9**, 1984, 46–53.

[Va 76] L. Valiant: Relative complexity of checking and evaluating. *Information Processing Letters*, **5**, 1976, 20–23.

[Xu,Do,Bo 83] M. Xu, J. Doner, R. Book: Refining nondeterminism in relativizations of complexity classes. *JACM* **30**, 1983, 677-685.