On Helping and Interactive Proof Systems

V. Arvind*
Department of Computer Science
Institute of Mathematical Sciences
Madras, India

J. Köbler, R. Schuler
Theoretische Informatik
Universität Ulm
D-89069 Ulm, Germany

Abstract

We investigate the complexity of honest provers in interactive proof systems. This corresponds precisely to the complexity of oracles helping the computation of robust probabilistic oracle machines. We obtain upper bounds for languages in FewEXP and for sparse sets in NP. Further, interactive protocols with provers that are reducible to sets of low information content are considered. Specifically, if the verifier communicates only with provers in P/poly, then the accepted language is low for Σ_2^p. In the case that the provers are polynomial-time reducible to $\log^*$-sparse sets or to sets in strong-P/log then the protocol can be simulated by the verifier even without the help of provers. As a consequence we obtain new collapse results under the assumption that intractable sets reduce to sets with low information content.

1 Introduction and overview of results

Two extensions of the concept of NP (as the class of languages with efficient proofs of membership [Coo71]), namely the class IP of languages that have single prover interactive proof protocols and the class MIP of languages with multi-prover interactive proof protocols (formal definitions are given in the next section), have been shown to characterize the complexity classes PSPACE and NEXP respectively [Sh90, BFL91].

In the interactive proof system model of computation [GMR89] one or more provers try to convince a probabilistic polynomial time verifier that a given input is a member of the considered language. The protocol has the following behavior: if an input is in the language then there are prover(s) that convince the verifier to accept with high probability. On the other hand, if the input is not in the language then no set of provers can convince the verifier with more than negligible probability. Arthur Merlin games, introduced in [Bab85], are special cases of single-prover interactive proof systems in which the random choices of the verifier (called Arthur) are public and therefore visible to Merlin, the prover. In [Bab85] it is shown that for any constant k, a k-round interaction AM[k] can be substituted by a two round protocol AM = AM[2] and that AM is contained in Π_2^p the second level of the polynomial time hierarchy. Here AM[k] denotes a k round interaction of messages between Arthur and Merlin, where the first message is from Arthur. The class MA of languages

*Part of the work was done while visiting Universität Ulm. Supported in part by an Alexander von Humboldt research fellowship.

accepted by a two round protocol starting with a message from Merlin is an apparently smaller class than AM.

It turned out that the Arthur Merlin games model of computation is as powerful as the general interactive proof systems. In fact any interactive proof system can be simulated by an Arthur Merlin game [GS86, BM88] without increasing the number of rounds. In general the notation AM is used to denote single-prover interactive proof systems with a constant number of interaction, whereas IP is used if there is no restriction. Following the advances made in [LFKN92], it was finally shown in [Sh90] that PSPACE = IP. Subsequently, it was shown that multi-prover interactive proof systems can accept all of NEXP [BFL91]. Intuitively, the multi-prover model is more powerful since the verifier can use one prover to check that the answers of another prover do not depend on previous queries made to him. This is made precise in the proof of Babai et al. (of the above mentioned result) where it is shown that a polynomial number of provers can be substituted by two provers. One prover answers all the queries of the original protocol, and the second is only used to verify that the answers do not depend on the history of interaction.

An interesting issue that arises in interactive proof systems – which is also the main concern of this paper – is bounding the complexity of the honest prover(s) that make the verifier accept a given language. For example, it is known ([Fe86, Sh90]) that each language in PSPACE has an IP protocol with prover of complexity PSPACE. The result of Lund et al. [LFKN92] implies that P^{PP} has IP protocols with prover complexity PP. Similarly, EXP has MIP protocols of prover complexity EXP [BFL91]. An important aspect of bounding prover complexity in interactive protocols is its connection to program checking. Here one considers MIP protocols where the honest prover(s) are restricted to answer only questions about the language they are asked to prove (these MIP protocols defined in [BFL91] are essentially the function-restricted IP class of [BK89]). If both a language and its complement have such MIP protocols, then it turns out that the language has a program checker in the sense of Blum and Kannan [BK89].

In this paper, we examine a different application of bounding prover complexity: deriving new collapse consequences under the assumption that intractable complexity classes are reducible to subclasses of P/poly. Such applications were first discovered in [BFL91], e.g. Corollary 1.2 below.

Let IP[$\mathcal{C}$] and MIP[$\mathcal{C}$] denote respectively the class of languages with prover complexity bounded by FP($\mathcal{C}$) (i.e., the class of functions computable by polynomial-time bounded transducers with access to an oracle from the class $\mathcal{C}$). In this notation we summarize some of the known results on upper bounds for prover complexity.

Theorem 1.1

1. [Fe86, Sh90] PSPACE = IP[PSPACE].

2. [LFKN92] $\mathrm{P}^{\mathrm{PP}} \subseteq$ IP[PP].

3. [BF91] $\oplus$P $\subseteq$ IP[$\oplus$P].

4. [BFL91] NEXP = MIP[$\mathrm{EXP}^{\mathrm{NP}}$].

5. [BFL91] EXP = MIP[EXP].

A surprising application of Theorem 1.1 to classical structural complexity is the following corollary.

Corollary 1.2 [LFKN92, BFL91] *For* $\mathcal{K} \in \{\mathrm{PP}, \mathrm{PSPACE}, \mathrm{EXP}\}$*, if* $\mathcal{K} \subseteq \mathrm{P/poly}$ *then* $\mathcal{K} = \mathrm{MA}$*.*

The above result improves previously known collapses to Σ_2^p under the same assumption, obtained by Meyer, Karp, and Lipton [KL80]. This is particularly interesting since it is known that the collapse of PH to Σ_2^p under the assumption $\mathrm{NP} \subseteq \mathrm{P/poly}$ [KL80] cannot be improved to Δ_2^p with relativizable proof methods [Hel86], and the LFKN protocol is known to be not relativizable [FS88, FRS88].

The crux of the proof of Corollary 1.2 is as follows: if $\mathcal{K}$ is contained in P/poly, then by Theorem 1.1, every language in $\mathcal{K}$ has an MIP protocol in which the provers are polynomial size circuits. This protocol can be simulated by an MA protocol where Merlin simply sends the circuits for the provers to Arthur in the first round [LFKN92, BFL91].

We observe that the MA protocol in the above proof-sketch can be seen as a single prover protocol with prover complexity in P/poly. Thus we can identify the classes MIP[P/poly] and IP[P/poly] of interactive proof systems in which the provers are polynomial size circuits: $\mathrm{MIP[P/poly]} = \mathrm{IP[P/poly]} \subseteq \mathrm{MA}$.

On the other hand, in the case of NEXP the $\mathrm{NEXP} = \mathrm{MIP}[\mathrm{EXP}^{\mathrm{NP}}]$ characterization does not appear to directly imply any significant collapse of NEXP under the assumption $\mathrm{NEXP} \subseteq \mathrm{P/poly}$. It is due to the very high upper bound on the prover complexity. A related result has been proved by Buhrman and Homer [BH92] who showed that under a stronger assumption $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathrm{P/poly}$ it holds that $\mathrm{EXP}^{\mathrm{NP}} = \Sigma_2^p$.

An important motivation of the present paper is to explore further applications of characterizations of complexity classes using interactive proof systems, in order to derive new collapse consequences assuming that the considered class has polynomial size circuits.

We note that in Babai et al. [BFL91] it is an open question as to whether the upper bound of $\mathrm{EXP}^{\mathrm{NP}}$ for the prover complexity for NEXP can be improved. The reason for the apparently weak upper bound is that all provers must have access to the same tableau of an accepting NEXP computation. To wit, the high prover complexity is the cost of disambiguating down to the lexicographically least consistent NEXP computation.

As a first step, we show that FewEXP languages (accepted by NEXP machines that have at most exponentially many accepting paths) are contained in $\mathrm{MIP}[\mathrm{NP}^{\mathrm{FewEXP}}]$.

Theorem 3.4 $\mathrm{FewEXP} \subseteq \mathrm{MIP}[\mathrm{NP}^{\mathrm{FewEXP}}]$.

It is to be noted that $\mathrm{NP}^{\mathrm{FewEXP}}$ is a substantially smaller class of provers than $\mathrm{EXP}^{\mathrm{NP}}$. Indeed, since $\mathrm{NP}^{\mathrm{FewEXP}}$ is known to be contained in $\mathrm{P}^{\mathrm{NEXP}}$ [Hem89], it follows that $\mathrm{FewEXP} \subseteq \mathrm{MIP[NEXP]}$. As a consequence of Theorem 3.4 we can extend Corollary 1.2 to the class $\mathcal{K} = \mathrm{FewEXP}$.

Corollary 3.5 If $\mathrm{FewEXP} \subseteq \mathrm{P/poly}$, then $\mathrm{FewEXP} = \mathrm{MA} = \mathrm{MIP[P/poly]}$.

As observed by Schöning [Sch88], there is a close relationship between the complexity of provers in interactive proof systems and the complexity of oracles helping the computation of *robust* oracle machines. A machine M is called robust if $L(M, \emptyset) = L(M, B)$ for all oracles B, and an oracle A (one-sided) helps the computation of M if M with oracle A runs in polynomial time on all inputs $x \in \Sigma^*$ (resp., $x \in L(M, \emptyset)$). The notion of helping by robust deterministic oracle machines was introduced by Schöning [Sch85] and was extended to one-sided helping by Ko [Ko87]. In a sense, the notion of helping by robust machines was a forerunner to the concept of interactive proof systems. Indeed, it follows from the probabilistic oracle machine characterization of MIP in [FRS88] that the class $\mathrm{MIP}[\mathcal{C}]$ coincides with the class $\mathrm{BPP}_{1-\mathrm{help}}(\mathcal{C})$ of languages accepted by probabilistic robust machines with the one-sided help of an oracle from $\mathcal{C}$. With this interpretation, Theorem 3.4 shows that $\mathrm{NP}^{\mathrm{FewEXP}}$ oracles are able to give one-sided help to the computation of languages in FewEXP. On the other hand, the inclusion $\mathrm{MIP}[\mathrm{P/poly}] \subseteq \mathrm{MA}$ can be interpreted as stating that oracles in P/poly cannot help the computation of languages outside MA.

An interesting open question is whether P/poly provers are useful at all, i.e., whether MIP[P/poly] contains sets that are not in BPP. As follows from the next theorem settling this question is likely to be very hard since a negative answer would imply that all sparse NP sets are already in BPP, which in turn implies that NE is contained in BPE known to be false in relativized worlds.

Theorem 4.2 All sparse NP sets are contained in $\mathrm{P}_{1-\mathrm{help}}(\mathrm{P/poly})$.

In [Ko87] it is shown that every set A accepted by a robust deterministic oracle machine with the one-sided help of an oracle from P/poly, i.e., $A \in \mathrm{P}_{1-\mathrm{help}}(\mathrm{P/poly})$, is low for Σ_2^p. Since $\mathrm{P}_{1-\mathrm{help}}(\mathrm{P/poly}) \subseteq \mathrm{MIP}[\mathrm{P/poly}]$, the following theorem extends both the above-mentioned result of Ko and the lowness of BPP for Σ_2^p [Sip83, ZH86, Sch89].

Theorem 4.7 MIP[P/poly] is low for Σ_2^p.

Considering MIP protocols with provers reducible to sets of very low information content (e.g. to sets in strong-P/log or to log^*-sparse sets) we show that such provers cannot help the verifier to accept languages outside of BPP.

Theorem 4.10 Let A be a log^*-sparse set and let B be a set in strong-P/log. Then $\mathrm{MIP}[A] = \mathrm{MIP}[B] = \mathrm{BPP}$.

These results extend the work of Ko [Ko87] showing that every log^*-sparse set A and every set B in strong-P/log are no-1-helpers, i.e., $\mathrm{P}_{1-\mathrm{help}}(A) = \mathrm{P}_{1-\mathrm{help}}(B) = \mathrm{P}$. While Theorem 4.10 is its parallel, the fact that the verifier is probabilistic introduces new difficulties in the case of log^*-sparse sets which are handled in the proof. Theorem 4.10 has again certain collapse consequences.

Corollary 4.11 Let A be a log^*-sparse set or a set in strong-P/log.

1. $\mathrm{NP} \subseteq \mathrm{BPP}^A$ implies $\mathrm{PH} = \mathrm{BPP}$.
2. For any class $\mathcal{K} \in \{\mathrm{PP}, \mathrm{PSPACE}, \mathrm{EXP}\}$, $\mathcal{K} \subseteq \mathrm{BPP}^A$ implies $\mathcal{K} = \mathrm{BPP}$.

2 Preliminaries

In this section we give preliminary definitions and notation used in the paper. We fix the alphabet for languages as $\Sigma = \{0,1\}$. For a set $A \subseteq \Sigma^*$, $A^{=n}$ ($A^{\leq n}$) denotes the set of all strings in A of length n (up to length n, respectively). The cardinality of a set A is denoted by $|A|$. The length of a string $x \in \Sigma^*$ is also denoted $|x|$, by abuse of notation. We use lower case letters to denote strings and upper case letters to denote sets. A set S is called sparse if $|S^{\leq n}|$ is bounded above by a polynomial.

To encode pairs (or tuples) of strings we use a standard polynomial-time computable pairing function denoted by $\langle \cdot, \cdot \rangle$ whose inverses are also computable in polynomial time.

Following standard notation [BDG], we denote the complexity classes DTIME($2^{n^{O(1)}}$) (NTIME($2^{n^{O(1)}}$)) of languages accepted by (non)deterministic Turing machines in time 2^{poly} by EXP (resp. NEXP). The corresponding 2^{linear} time bounded classes are denoted by E and NE. Further, we denote the class of functions computable by deterministic polynomial-time Turing machine transducers with natural access to an oracle A by FP(A).

Definition 2.1 *A language L is in* FewEXP *if there are polynomials p and q such that there is a nondeterministic machine M accepting A in time $2^{p(n)}$ with the property that on any input of size n M has at most $2^{q(n)}$ accepting paths.*

The class FewEXP is just the FewP analogue for nondeterministic exponential time.

Definition 2.2 [KL80] *A set L is in* P/poly *if there exists a set $B \in$ P, a polynomial q, and a sequence $(a_n)_{n\geq 0}$ of strings such that $|a_n| \leq q(n)$, and $L^{=n} = \{x \in \Sigma^n \mid \langle x, a_n\rangle \in B\}$.*

It is well-known that P/poly coincides with the class of languages which can be recognized by a non-uniform family of polynomial-size circuits.

The definition of multiprover interactive proof systems first appeared in Ben-Or et al. [BGKW88] and Babai et al. [BFL91].

Definition 2.3 *Let p be a polynomial and V be a probabilistic polynomial time machine. There is a multiprover interactive (i.e. MIP) protocol for a language L, if for every n there are provers $P_1, \cdots, P_k$, $k \leq p(n)$, such that for every $x \in \Sigma^n$:*

$$x \in L \quad \to \quad \mathrm{Prob}[P_1, \cdots, P_k \textit{ make } V \textit{ accept}\,] > 3/4,$$

$$x \notin L \quad \to \quad \forall P_1', \cdots, P_k' : \mathrm{Prob}[P_1', \cdots, P_k' \textit{ make } V \textit{ accept}\,] < 1/4,$$

where the set of provers $P_1, \cdots, P_k$ are machines of unlimited computational power which share the input tape with V, and V shares a separate communication tape with each prover P_i.

The class of languages accepted by an MIP protocol is denoted MIP and the class of languages with an IP protocol (where $k = 1$ in the above definition) is denoted IP. We denote by MIP[$\mathcal{C}$] and IP[$\mathcal{C}$] the respective language classes where the prover complexity is bounded by FP($\mathcal{C}$).

We shall also use in this paper Schöning's notion of robust probabilistic machines [Sch88] which coincides with the probabilistic oracle machine characterization of MIP in [FRS88]. The notion of robust deterministic machines was earlier introduced by Schöning in the deterministic setting [Sch85] and is further studied in [Ko87].

Definition 2.4 [Sch85, Ko87] *A language L is in $\mathrm{P}_{1-\mathrm{help}}(A)$, if there exist a deterministic oracle Turing machine M and a polynomial p such that the following robustness and helping properties hold:*

- *for every oracle B, $L(M,B) = L$,* *(robustness)*
- *for all inputs $x \in L$, $M^A(x)$ halts in $p(|x|)$ steps.* *(one-sided-helping)*

$\mathrm{P}_{1-\mathrm{help}}(\mathcal{C})$ denotes the class of languages L such that $L \in \mathrm{P}_{1-\mathrm{help}}(A)$ for some set $A \in \mathcal{C}$.

Definition 2.5 [FRS88, Sch88] *A language L is in $\mathrm{BPP}_{1-\mathrm{help}}(A)$, if there exists a probabilistic polynomial time oracle Turing machine M such that the following probabilistic helping and robustness properties hold:*

- *for all inputs $x \in L$, $\mathrm{Prob}[M^A(x) = 1] \geq 3/4$,* *(one-sided-helping)*
- *for all inputs $x \notin L$ and all oracles B, $\mathrm{Prob}[M^B(x) = 1] < 1/4$.* *(robustness)*

For other standard definitions used in the paper we refer the reader to a standard book on structural complexity theory [BDG].

The next theorem states that $\mathrm{BPP}_{1-\mathrm{help}}$ exactly characterizes MIP, where the complexity of the helping oracle corresponds to the prover complexity in the MIP model.

Theorem 2.6 [FRS88] *For all sets A:* $\mathrm{BPP}_{1-\mathrm{help}}(A) = \mathrm{MIP}[A]$.

The following proposition shows that, interestingly, for any complexity class $\mathcal{C}$, both $\mathcal{C}$ and $\mathrm{BPP}^{\mathcal{C}}$ have the same helping power. Moreover, the classes IP and MIP are closed under application of the BP operator.

Proposition 2.7 *For every class $\mathcal{C}$:*

1. $\mathrm{BP}\cdot\mathrm{IP}[\mathcal{C}] = \mathrm{IP}[\mathrm{BPP}^{\mathcal{C}}] = \mathrm{IP}[\mathcal{C}]$,
2. $\mathrm{BP}\cdot\mathrm{MIP}[\mathcal{C}] = \mathrm{MIP}[\mathrm{BPP}^{\mathcal{C}}] = \mathrm{MIP}[\mathcal{C}]$.

Lozano and Torán [LT92] proved that if A and $\overline{A}$ have both MIP protocols with provers in FP^A then $A \in \mathrm{P/poly}$ implies that A is low for MA. By a technical extension of their proof, we get the following theorem.

Theorem 2.8 $\mathrm{MIP}[\mathrm{P/poly}] \cap \text{co-}\mathrm{MIP}[\mathrm{P/poly}]$ *is low for* $\mathrm{MIP}[\mathrm{P/poly}]$.

Corollary 2.9 *If* $\text{co-NP} \subseteq \mathrm{MIP}[\mathrm{P/poly}]$, *then* $\mathrm{PH} = \mathrm{MIP}[\mathrm{P/poly}]$.

Proof: Assume that $\text{co-NP} \subseteq \mathrm{MIP}[\mathrm{P/poly}]$. Since NP is contained in $\mathrm{P}_{1-\mathrm{help}}(\mathrm{NP})$ [Ko87] and since $\mathrm{MIP}[\mathrm{P/poly}] \subseteq \mathrm{P/poly}$ it follows that $\mathrm{NP} \subseteq \mathrm{P}_{1-\mathrm{help}}(\mathrm{NP}) \subseteq \mathrm{MIP}[\mathrm{NP}] \subseteq \mathrm{MIP}[\mathrm{P/poly}]$. By Theorem 2.8, this implies the collapse of PH to $\mathrm{MIP}[\mathrm{P/poly}]$. ■

An interesting open question is whether co-NP is contained in MIP[PH]. A positive answer would imply that the assumption of Corollary 2.9 could be weakened to $\mathrm{NP} \subseteq \mathrm{P/poly}$. The best known upper bound for the prover complexity in the case of co-NP languages is $\oplus\mathrm{P}$, i.e., $\text{co-NP} \subseteq \mathrm{MIP}[\oplus\mathrm{P}]$. Since $\text{co-NP} \subseteq \mathrm{BP}\cdot\oplus\mathrm{P}$ [VaVa86, To91] this follows by combining the result of Babai and Fortnow [BF91], that $\oplus\mathrm{P} \subseteq \mathrm{IP}[\oplus\mathrm{P}]$ with the above Proposition 2.7. Moreover, together with the result of Lund et al. [LFKN92] that $\mathrm{P}^{\mathrm{PP}} \subseteq \mathrm{IP}[\mathrm{PP}]$, we get the following corollary to Proposition 2.7.

Corollary 2.10

1. $\mathrm{IP}[\oplus\mathrm{P}] = \mathrm{MIP}[\oplus\mathrm{P}] = \mathrm{BPP}^{\oplus\mathrm{P}}$.

2. $\mathrm{IP}[\mathrm{PP}] = \mathrm{MIP}[\mathrm{PP}] = \mathrm{BPP}^{\mathrm{PP}}$.

As shown in [LT92], the graph isomorphism problem GI is low for MA if it is contained in P/poly. This follows from the result of Lozano and Torán mentioned above and the fact [GMW85, Schn76] that GI and its complement are both contained in IP[GI]. Using Theorem 2.8 we get the following improvement.

Corollary 2.11 *If* GI *is contained in* P/poly *then* GI *is low for* MIP[P/poly].

Finally, we observe that MIP[GI] = IP[GI]. In general, if both L and $\overline{L}$ have an IP[L] protocol, then every MIP[L] protocol can be simulated by an IP[L] protocol in the following way: substitute every query to one of the provers by (1) a query to the single prover and, depending on the answer, (2) a simulation of the IP protocol of either L or of $\overline{L}$ to verify that the answer of the prover is correct.

Theorem 2.12 *If L and $\overline{L}$ are in* IP[L], *then* MIP[L] = IP[L].

Notice that the assumption of the above theorem implies that L is checkable in the sense of [BK89] (meaning that L and $\overline{L}$ are in MIP[L]).

3 Few exponential time computations and MIP protocols

Although it is known that MIP = NEXP [BFL91], it is open whether provers of complexity NEXP are sufficient. Following the proof of MIP = NEXP it appears that all the honest provers must have access to the same tableau of an NEXP machine. For the cost of disambiguating down to one consistent tableau, the provers need $\mathrm{EXP}^{\mathrm{NP}}$ computational power, so that the provers can compute bit by bit a lexicographically first tableau for the NEXP language being verified [BFL91].

In this section we prove as the main result that if we have a FewEXP language A then the complexity of honest provers in the MIP protocol of Babai et al. [BFL91] can be bounded by $\mathrm{NP}^{\mathrm{FewEXP}}$. In the proof we use the notion of branch points.

Definition 3.1 *Let T be the computation tree of a nondeterministic Turing machine M on some input x. A node t in T is called a* branch point *if there are accepting computations along both descendants of t.*

The idea for the provers to access a consistent tableau is as follows: Every accepting computation w on some input $x \in L(M)$ is uniquely determined by the nondeterministic choices at the branch points. In fact, it suffices to know the levels of the branch points together with the nondeterministic choices. Therefore every accepting path going through k branch points at levels $i_1, \cdots, i_k$ can be succinctly described by the list $\langle i_1, \cdots, i_k, b_1, \cdots, b_k \rangle$ where b_j is the nondeterministic choice at level i_j. This idea is made precise by the following lemma.

Lemma 3.2 *For every* FewEXP *machine M, there exists a function f computable in* $\mathrm{FP}(\mathrm{NP}^{\mathrm{FewEXP}})$ *which for every $x \in L(M)$ computes a tuple $\langle i_1, \cdots, i_k, b_1, \cdots, b_k \rangle$, such that there exists exactly one accepting path of M on input x that has bit b_j at position i_j, for $j = 1, \ldots, k$.*

Proof: For $x \in L(M)$, we define $f(x)$ as the lexicographically smallest string $\langle i_1, \cdots, i_k, b_1, \cdots, b_k \rangle$ such that there is exactly one accepting path of M on input x which branches at level i_j according to bit b_j, for $j = 1, \ldots, k$. (For $x \notin L(M)$, we let $f(x)$ undefined.)

Since the number of accepting paths on input $x \in L(M)$ is at most $2^{p(|x|)}$, for some polynomial p, there exists an accepting path $w = w_1 w_2 \cdots w_r$ that goes through at most $p(|x|)$ many branch points. Let $i_1, \cdots, i_k$ be the respective levels of the branch points on w. Then the tuple $\langle i_1, \cdots, i_k, w_{i_1}, \cdots, w_{i_k} \rangle$ is of polynomial length in $|x|$, and uniquely describes w. This shows that the length of $f(x)$ is polynomially bounded in $|x|$. To show that f is in $\mathrm{FP}(\mathrm{NP}^{\mathrm{FewEXP}})$, consider the language B defined as

$$B = \{\langle x, i_1, \cdots, i_k, b_1, \cdots, b_k \rangle \mid k \le p(|x|) \text{ and there exists exactly one accepting path of } M \text{ on input } x \text{ that has bit } b_j \text{ at level } i_j, \text{ for } j = 1, \ldots, k\}.$$

Using the fact that M is a FewEXP machine, it is immediate that $B \in \mathrm{P}^{\mathrm{FewEXP}}$. Thus, for an input string x, $f(x)$ can be computed by prefix search by querying an appropriate prefix version of the language B which is in $\mathrm{NP}^{\mathrm{FewEXP}}$. ■

Lemma 3.3 *Let M be a* FewEXP *machine. Then for every $x \in L(M)$, an accepting path of M on input x can be computed in exponential time with a* FewEXP *oracle, where all oracle queries are of length polynomial in $|x|$.*

Proof: Consider the function f in Lemma 3.2. For every $x \in L(M)$, $f(x)$ is a tuple that uniquely identifies one accepting computation path of M on input x. Once the value of $f(x)$ is computed every bit of the path can be determined by a single query of polynomial length to the FewEXP language

$$T = \{\langle x, i_1, \cdots, i_k, b_1, \cdots, b_k, i_0, b_{i_0} \rangle \mid k \le p(|x|) \text{ and there is an accepting path for } M(x) \text{ with bits } b_1, \cdots, b_k \text{ at levels } i_1, \cdots, i_k, \text{ and with the bit } b_{i_0} \text{ at level } i_0\}.$$

Since f is in $\mathrm{FP}(\mathrm{NP}^{\mathrm{FewEXP}})$ it is computable in exponential time with a FewEXP oracle, where all queries are of polynomial length. ■

Theorem 3.4 $\mathrm{FewEXP} \subseteq \mathrm{MIP}[\mathrm{NP}^{\mathrm{FewEXP}}]$.

Proof: The proof uses the fact shown in [BFL91] that any NEXP language B has a multiprover protocol with prover complexity bounded by EXP provided that for a given input $x \in B$ all honest provers have access to the same tableau of an accepting computation of the underlying NEXP machine on input x. Let A be a language in FewEXP accepted by a

FewEXP machine M. By Lemma 3.2, the honest provers can compute the description $f(x)$ of an accepting path w_x in polynomial time by asking a suitable $\mathrm{NP}^{\mathrm{FewEXP}}$ oracle. Having computed $f(x)$ they can generate the same answers as the honest provers in the BFL-protocol in polynomial time by feeding a suitable FewEXP oracle F with the description $f(x)$ as part of the queries. The FewEXP machine accepting F uses the description $f(x)$ to guess the corresponding accepting path of M on input x and then simulates the honest provers of the BFL-protocol. ■

Corollary 3.5

1. *If* FewEXP $\subseteq$ EXP/poly, *then* FewEXP = EXP.

2. *If* FewEXP $\subseteq$ P/poly, *then* FewEXP = MA = MIP[P/poly].

Proof: By Lemma 3.3, an accepting path of a FewEXP machine M can be computed in $\mathrm{EXP}^{\mathrm{FewEXP}}$ where all queries are of polynomial length. Thus, if FewEXP $\subseteq$ EXP/poly, then an accepting path can be found (and verified) in EXP, by cycling through all advice strings of polynomial length. This proves 1.

If FewEXP $\subseteq$ P/poly, then, by part 1, FewEXP = EXP and hence it follows by Corollary 1.2 [BFL91] that FewEXP = MA = MIP[P/poly]. ■

Using the following proposition, containment of any of the classes PP, PSPACE, EXP, and FewEXP in MA (or MIP[P/poly]) actually implies lowness of the class for MA (respectively MIP[P/poly]).

Proposition 3.6 *For every class $\mathcal{K}$ containing* co-NP, *if $\mathcal{K}$ is contained in* MA *(or* MIP[P/poly]*) then $\mathcal{K}$ is also low for* MA *(respectively* MIP[P/poly]*).*

Proof: Assume that $\mathcal{K} \subseteq$ MA. Since co-NP $\subseteq$ AM implies PH $\subseteq$ AM [BHZ87] and since co-NP $\subseteq$ MA implies $\Sigma_2^p \subseteq$ MA it follows that $\mathrm{MA}^{\mathcal{K}} \subseteq \mathrm{MA}^{\mathrm{MA}} \subseteq \mathrm{PH} \subseteq \mathrm{MA}$.

If $\mathcal{K} \subseteq$ MIP[P/poly] the lowness of $\mathcal{K}$ for MIP[P/poly] follows from Corollary 2.9 by a similar argument as above. ■

4 MIP protocols with small circuits as provers

As already mentioned in the introduction, this section is motivated by Corollary 1.2, which gives a collapse of any class $\mathcal{K} \in \{\mathrm{PSPACE}, \mathrm{PP}, \mathrm{EXP}\}$ to MA under the assumption that $\mathcal{K} \in$ P/poly. In fact, the collapse is actually to the possibly smaller class MIP[P/poly]. Naturally one is led to investigate the complexity of this new class MIP[P/poly]. We recall the following proposition.

Proposition 4.1 BPP $\subseteq$ MIP[P/poly] $\subseteq$ P/poly*.*

Next we give an argument showing that it is unlikely that MIP[P/poly] coincides with BPP.

Theorem 4.2 *All sparse sets in* NP *are contained in* $\mathrm{P}_{1-\mathrm{help}}(\mathrm{P/poly})$.

Proof: Let S be a sparse set in NP. Then we can construct a deterministic robust machine M that uses the oracle to perform a prefix search for a witness of the given input x. If a witness is found then M accepts. Otherwise M starts an exhaustive search. It is easy to see that there exists an oracle in P/poly helping the computation of M on all inputs $x \in S$. ∎

Since $\mathrm{P}_{1-\mathrm{help}}(\mathrm{P/poly})$ is contained in MIP[P/poly], the existence of sparse sets in NP \ BPP implies that BPP is a proper subclass of MIP[P/poly]. Using the well-known downward separation technique of Book [Bo74] we get the following corollary.

Corollary 4.3 MIP[P/poly] *is not contained in* BPP *unless* NE *is a subclass of* BPE.

Next we show that MIP[P/poly] is actually low for Σ_2^p. This extends the lowness for Σ_2^p of $\mathrm{P}_{1-\mathrm{help}}(\mathrm{P/poly})$ [Ko87] and of BPP [Sip83, ZH86, Sch89]. The proof is by an application of universal hash functions.

A linear hash function h from Σ^p to Σ^m is given by a Boolean (m, p)-matrix (a_{ij}) and maps any string $x = x_1 \dots x_p \in \Sigma^p$ to some string $y = y_1 \dots y_m$ where $y_i = \bigoplus_{j=1}^{p}(a_{ij} \wedge x_j)$. A family $H = \{h_1, \dots, h_s\}$ of linear hash functions from Σ^p to Σ^m is said to *hash* a set $X \subseteq \Sigma^p$ if every $x \in X$ can be mapped to Σ^m by some hash function $h_k \in H$ which avoids collisions between x and other strings in X:

$$\forall\, x \in X\, \exists\, k\, (1 \leq k \leq s)\, \forall\, y \in X : x \neq y \Rightarrow h_k(x) \neq h_k(y).$$

We state the actual form of the hashing lemmas required before we proceed to the proof of Theorem 4.7. The first two of these lemmas are from Sipser [Sip83]. The third is from Gavaldà [Gav92] and extends Sipser's second lemma to exponentially many sets.

Lemma 4.4 *Let* $A \subseteq \Sigma^p$ *such that there exists a hash family* $H = \{h_1, \cdots, h_s\} : \Sigma^p \to \Sigma^k$ *that hashes* A. *Then* $|A| \leq s \cdot 2^k$.

Lemma 4.5 *Let* $A \subseteq \Sigma^p$ *such that* $|A| \leq 2^{k-1}$. *Then there exists a hash family* $H = \{h_1, \cdots, h_k\} : \Sigma^p \to \Sigma^k$ *that hashes* A.

Lemma 4.6 *Let* $A_1, A_2, \cdots, A_{2^l} \subseteq \Sigma^p$ *such that* $|A_i| \leq 2^{k-1}$ *for every* $i, 1 \leq i \leq 2^l$. *Then there exists a hash family* $H = \{h_1, \cdots, h_{k(l+1)}\} : \Sigma^p \to \Sigma^k$ *that hashes each* A_i.

Theorem 4.7 MIP[P/poly] *is low for* Σ_2^p.

Proof: Let $A \in$ MIP[P/poly]. Then, by definition, there is a multiprover interactive protocol (we can assume that it is a two-prover protocol [BFL91]). Since $A \in$ MIP[P/poly], the honest provers for A are in FP/poly and hence there is a polynomial p such that for every length m, there is a corresponding advice string $c_m \in \Sigma^{p(m)}$ encoding the polynomial-size circuits for both provers at length m.

We say that $V(y,w,c)$ accepts if on input y the provers corresponding to the circuits encoded by c make V accept on path w. Then, the above protocol can be rewritten after amplifying the error probability as

$$y \in A \quad \to \quad \mathrm{Prob}[V(y,w,c_{|y|}) \text{ accepts}] \geq 1 - 2^{-|y|},$$

$$y \notin A \quad \to \quad \forall c \in \Sigma^{p(|y|)} : \mathrm{Prob}[V(y,w,c) \text{ accepts}] \leq 2^{-|y|},$$

where w is chosen uniformly at random from $\Sigma^{s(|y|)}$ for a suitable polynomial s.

For each instance y and each c let $Acc(y,c)$ ($Rej(y,c)$) denote the set of $w \in \Sigma^{s(|y|)}$ such that $V(y,w,c)$ accepts (respectively, rejects). Then we have for all y of length m,

- if $y \in A$ then for the correct advice c_m, $|Rej(y,c_m)| \leq 2^{s(m)-m}$,
- if $y \notin A$ then for any advice c, $|Acc(y,c)| \leq 2^{s(m)-m}$.

Let $k(m) = s(m) - m + 1$ and $l(m) = m + p(m)$. We say that an advice string c of length $p(m)$ and a hash family $H = \{h_1, \cdots, h_{k(m)(l(m)+1)}\} : \Sigma^{s(m)} \to \Sigma^{k(m)}$ are good for length m if the following co-NP predicate holds:

For all $y \in \Sigma^m$ and $c' \in \Sigma^{p(m)}$: H hashes $Rej(y,c)$ or $Acc(y,c')$.

Claim 1 *For m large enough it holds that for every collection of $2^{m+p(m)}$ many sets of cardinality at most $2^{s(m)-m}$ there exists a hash family $H = \{h_1, \cdots, h_{k(m)(l(m)+1)}\} : \Sigma^{s(m)} \to \Sigma^{k(m)}$ that simultaneously hashes all sets in the collection, but no such hash family can hash a set of cardinality $2^{s(m)} - 2^{s(m)-m}$.*

Proof of Claim 1. The first part follows immediately from Lemma 4.6. To prove the second part, assume to the contrary that there exist arbitrary large m such that some hash family $H = \{h_1, \cdots, h_{k(m)(l(m)+1)}\} : \Sigma^{s(m)} \to \Sigma^{k(m)}$ hashes a set of cardinality $2^{s(m)} - 2^{s(m)-m}$. By Lemma 4.4, it follows that

$$2^{s(m)} - 2^{s(m)-m} \leq k(m)(l(m)+1)2^{k(m)} = k(m)(l(m)+1)2^{s(m)-m+1}$$

implying that $2^m - 1 \leq 2k(m)(l(m)+1)$ for arbitrary large m; a contradiction. □ *Proof of Claim 1.*

Thus, for every length m there exists a good pair c, H, and for every such pair it holds that for all y of length m, H hashes $Acc(y,c)$ iff $y \notin A$ iff H does not hash $Rej(y,c)$. As a consequence there is a nondeterministic polynomial time Turing machine M', that on input $\langle y,c,H\rangle$, where c, H are good for length $|y|$, strongly (in the sense of [Lo82]) decides membership of y in A:

input $\langle y,c,H\rangle$;
guess $w, w' \in \Sigma^{s(|y|)}$;
if w, w' witness that H does not hash $Acc(y,c)$ **then accept**;
if w, w' witness that H does not hash $Rej(y,c)$ **then reject**;
halt in "don't know" state.

Now, let M be a Σ_2^p oracle machine with oracle A accepting some language L. We describe an equivalent Σ_2^p machine M'' for L. We first note that on input x the size of the queries that M^A asks is bounded above by $r(|x|)$ for some polynomial r. Now, our strategy to simulate $M^A(x)$ by a Σ_2^p computation is as follows:

> First the new machine $M''(x)$ guesses the correct advice strings c_m for all lengths up to $r(|x|)$ with which the provers can be replaced. Then it guesses polynomial size hash families H_m, $1 \leq m \leq r(|x|)$, and verifies that all the pairs c_m, H_m are good. Then machine $M''(x)$ simulates $M^A(x)$ using the pairs c_m, H_m to answer oracle queries y of length m, by simulating the strong computation of the machine M' on input $\langle y, c_m, H_m \rangle$.

This completes the proof. ■

Now we show that sets of very low information content are powerless as provers. This extends the work of Ko in [Ko87] where he proved that certain oracles do not help the computation of deterministic robust oracle Turing machines. In particular he showed that oracles in LOG-INF do not help deterministic robust oracle machines where LOG-INF is a class that contains all log*-sparse sets and all sets in strong-P/log.

Definition 4.8 [Ko87]

1. *A set A is in* strong-P/log *if there exist a set $B \in$* P *and a constant c such that for all n there exists a string w, $|w| \leq c \log n$ such that for all x, $|x| \leq n$: $x \in A \Leftrightarrow \langle x, w \rangle \in B$.*
2. *A language L is in* log*-sparse *if there exists a constant k such that for all n it holds that $|L \cap \{x \mid n \leq |x| < 2^n\}| \leq k$.*

The class strong-P/log (which is a restriction of P/*log* in that there exists one advice of length ($O(\log n)$) for all strings of length *up to n* rather than for all strings of length *exactly n*) is closed under Turing reductions. We show that if the provers are restricted to be Turing reducible to strong-P/log or to log*-sparse, then the accepted languages are in BPP. The proof idea is to cycle through all oracle answer sequences which can possibly be generated by oracles of this type. Since the verifier is probabilistic, we first fix the random choices of the computation (after amplifying the probability). In the case of oracles in strong-P/log the simulating verifier accepts if an oracle answer sequence leading to an accepting computation is found. However in the case of log*-sparse oracles the number of oracles that contribute to the acceptance probability of the input string could be exponentially large. Thus, in order to avoid acceptance of strings not in the language, the simulating verifier has to perform an additional test verifying that the induced oracle indeed leads to acceptance with high probability.

Theorem 4.9 MIP[strong-P/log] = BPP.

Proof: The basic idea is to cycle through all possible advice strings to find a helping oracle. Assume $L \in \mathrm{BPP}_{1-\mathrm{help}}(A)$ for a strong-P/log set A and a $\mathrm{BPP}_{1-\mathrm{help}}$ machine M whose running time is bounded by some polynomial p and which has an exponentially small error probability $\epsilon < 2^{-n}$. Let $B \in$ P and c be a constant witnessing that $A \in$ strong-P/log. Now consider the following procedure.

input x, $|x| = n$
guess randomly steps for a computation of M
for all w, $|w| \leq c \log p(n)$
  simulate the computation of M where every oracle query z is answered according to $\langle z, w\rangle \in B$.
  if M accepts **then accept**
reject

Intuitively speaking, the above procedure considers only polynomially many oracles, and at least one will be the helping oracle A. Therefore, if $x \in L$, the probability that the procedure accepts is larger than $1 - \epsilon > 3/4$. If $x \notin L$ this probability is smaller than $2^{c \log p(n)+1} \cdot \epsilon < 1/4$, for n large enough. ■

Theorem 4.10 *Let A be a* $\log^*$-sparse *set, then* $\mathrm{MIP}[A] = \mathrm{BPP}$.

Proof: Assume $L \in \mathrm{BPP}_{1-\mathrm{help}}(A)$, for some $\log^*$-sparse set A and a $\mathrm{BPP}_{1-\mathrm{help}}$ machine M_L which is time bounded by some polynomial p and has an exponentially small error probability ϵ. For every n, let $\Sigma^{\leq p(n)}$ be divided into 2 parts. A first part contains all strings of length smaller than $\log\log n$, and the second part all strings x, $\log\log n \leq |x| \leq p(n)$. Then the first part contains at most $\log n$ many elements and the number of strings in the other part that are in A is bounded by some constant k. Now consider the following machine M (for $v = v_1 \ldots v_m$, v_y denotes the bit v_i where y is the ith string in lexicographic order).

input x, $|x| = n$
guess randomly steps for a computation of M_L
for all v, $|v| \leq \log n$
  for all w, $|w| \leq p(n)$ and $|\{i \mid w_i = 1\}| \leq k$
    simulate M_L and use the pair v, w to answer the oracle queries: if y is the i-th query and $|y| \leq \log\log n$ answer yes iff $v_y = 1$. If $|y| > \log\log n$ then answer yes iff $w_i = 1$. Let $\{y_1 \ldots y_q\}$ be the positive queries.
    if M_L accepts **then**
      guess randomly 2^{k+5} additional computations of M_L and accept if at least one of the 2^{k+5} computations accepts where the set $Y = \{y_1 \ldots y_q\} \cup \{y \mid |y| \leq \log\log n \text{ and } v_y = 1\}$ is used as oracle.
reject

Assume $x \in L$, $|x| = n$. Then the probability that M_L accepts x with oracle A is larger than $7/8$. We show that M accepts with probability larger than $1/2 + 1/8$. The accepting computations of M_L can be divided into $d = 2^k$ sets S_i, depending on the positive queries of length at least $\log\log n$ and smaller than $p(n)$. Let p_i be the probability that a randomly guessed path is in the set S_i and let $I = \{i \mid p_i \geq 1/(8d)\}$. If we consider only sets S_i where $i \in I$, then the probability that a path is not in S_i is smaller than $1 - 1/(8d)$. Therefore the probability that all $2^{k+5} = 4 \cdot 8d$ paths guessed by M are not in S_i, is smaller than $1/8$. Since $\sum_{i \in I} p_i \geq 6/8$, the sum of the probabilities that the first computation is in the same set S_i as at least one of the additional 2^{k+5} computations is larger than $\sum_{i \in I}(7/8)p_i = (7/8)(6/8) > 1/2 + 1/8$.

Now assume that $x \notin L$, then the probability that M_L accepts x is smaller than ϵ for every oracle. Therefore the probability that for any fixed set Y one of the 2^{k+5} randomly guessed computation paths accepts is smaller than $1 - (1-\epsilon)^{2^{k+5}}$. Observe that for every initially guessed computation of M_L (in the above procedure) we sum up the acceptance probability of $n \cdot k \cdot n^k$ strings vw. Consequently, for every such initial computation the probability is smaller than $n \cdot k \cdot n^k(1 - (1-\epsilon)^{2^{k+5}}) < 1/4$, for n large enough. ∎

Corollary 4.11 *Let A be a log*-sparse set or a set in* strong-P/log.

1. NP $\subseteq$ BPP^A *implies* PH = BPP.
2. *Let* $\mathcal{K} \in \{\text{PP}, \text{PSPACE}, \text{EXP}\}$, *then* $\mathcal{K} \subseteq \text{BPP}^A$ *implies* $\mathcal{K} = \text{BPP}$.

Acknowledgments. The authors thank Uwe Schöning for useful discussions.

References

[Bab85] L. Babai. Trading group theory for randomness. *17th ACM Symp. Theory of Computing 1985*, 421-429.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences* **36**:254-276, 1988.

[BGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, A. Wigderson. Multiprover interactive proofs: How to remove the intractability assumptions. *Proc. 20th Ann. ACM Symposium Theory of Computing*, 113-131, 1988.

[BF91] L. Babai, L. Fortnow. Arithmetization: A new method in structural complexity. Computational Complexity **1**:41-66, 1991.

[BFL91] L. Babai, L. Fortnow, C. Lund. Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity, **1**:1-40, 1991.

[BDG] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I,II.* EATCS Monographs on Theoretical Computer Science, Springer Verlag, 1988.

[BK89] M. Blum, S. Kannan. Designing programs that check their work. *Proc. 21st ACM Symposium on Theory of Computing*, 86-97, 1989.

[Bo74] R. Book. Tally languages and complexity classes. *Information and Control* **26**:186-193, 1974.

[BHZ87] R.B. Boppana, J. Hastad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters* **25**:27-32, 1987.

[BH92] H. Buhrman, S. Homer. Superpolynomial circuits, almost sparse oracles, and the exponential hierarchy. *Proc. 12th Foundations of Software Technology and Theoretical Computer Science*, 116-127, 1992.

[Coo71] S.A. Cook. The complexity of theorem proving procedures. *Proc. 3rd Ann. ACM Symposium Theory of Computing*, 151-158, 1971.

[Fe86] P. Feldman. The optimal prover lives in PSPACE. manuscript, 1986.

[FRS88] L. Fortnow, J. Rompel, and M. Sipser. On the power of multiprover interactive protocols. *Proc. 3rd Conference on Structure in Complexity Theory*, 156-161, 1988.

[FS88] L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages. *Information Processing Letters* **24**:249-251, 1988.

[Gav92] R. Gavalda. Bounding the complexity of advice functions. *Proc. 7th Structure in Complexity Theory Conference*, IEEE Computer Society Press, 222-238, 1992.

[GMW85] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th Symposium on Foundations of Computer Science* 174-187, 1986.

[GMR89] S. Goldwasser, S. Micali, C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* **18**:186-208, 1989.

[GS86] S. Goldwasser, M. Sipser. Private coins versus public coins in interactive proof systems. In: S. Micali (ed.): *Randomness and Computation*, Vol. 5 of *Advances in Computing Research*, JAI Press, 1989,

[Hel86] H. Heller. On relativized exponential and probabilistic complexity classes. *Information and Control* **71**:231-243, 1986.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences***39**:299-322, 1989.

[KL80] R.M. Karp and R.J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proc. 12th ACM Symposium Theory of Computer Science*, 302-309, 1980.

[Ko87] K.I. Ko. On helping by robust oracle machines. *Theoretical Computer Science* **52**:15-36, 1987.

[Lo82] T.J. Long. Strong nondeterministic polynomial-time reducibilities. *Theoretical Computer Science* **21**:1-25, 1982.

[LT92] A. Lozano, J. Torán. On the non-uniform complexity of the graph isomorphism problem. In *Complexity Theory, Current Research*, Cambridge University Press, 245-273, 1993.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, N. Nisan. Algebraic methods for interactive proof systems. Journal of ACM **39**(4):859-868, 1992.

[Schn76] C. P. Schnorr. Optimal algorithms for self-reducible problems. In *3rd Int. Colloq. Automata, Languages and Programming, Edinburgh, University Press*, 322-337.

[Sch85] U. Schöning. Robust algorithms: a different approach to oracles. *Theoretical Computer Science*, **40**:57-66, 1985.

[Sch88] U. Schöning. Robust oracle machines. *Proc. 13th Mathematical Foundations of Computer Science*, 2-8, 1988.

[Sch89] U. Schöning. Probabilistic complexity classes and lowness. *Journal of Computer and System Sciences* **39**:84-100, 1989.

[Sh90] A. Shamir. IP=PSPACE. *Journal of ACM* **39**(4):869-877, 1992.

[Sip83] M. Sipser. A complexity theory approach to randomness. *Proc. 15th ACM Symposium Theory of Computing*, 330-335, 1983.

[To91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing* **20**:865-877, 1991.

[VaVa86] L.G. Valiant and V.V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science* **47**:85-93, 1986.

[ZH86] S. Zachos, H. Heller. A decisive characterization of BPP. *Information and Control* **69**:125-135, 1986.