Alternation in Interaction

M. Kiwi* MIT & U. Chile

C. Lund† AT&T

A. Russell ‡ MIT

D. Spielman§ MIT

R. Sundaram¶ MIT

November 27, 1995

Abstract

We study competing-prover one-round interactive proof systems. We show that one-round proof systems in which the first prover is trying to convince a verifier to accept and the second prover is trying to make the verifier reject recognize languages in NEXPTIME*, and, with restrictions on communication and randomness, languages in* NP*. We extended the restricted model to an alternating sequence of k competing provers, which we show characterizes Σ_{k-1}^{P}. Alternating oracle proof systems are also examined.*

*Dept. of Applied Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139. `mkiwi@math.mit.edu`. On leave of absence from Dept. de Ingeniería Matemática, U. de Chile. Supported by an AT&T Bell Laboratories PhD Scholarship. Part of this work was done while the author was at Bell Laboratories.

†AT&T Bell Laboratories, Room 2C324, 600 Mountain Avenue, P. O. Box 636, Murray Hill, NJ 07974-0636 USA, `lund@research.att.com`.

‡Dept. of Applied Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139. `acr@theory.lcs.mit.edu`. Partially supported by an NSF Graduate Fellowship and NSF 92-12184, AFOSR F49620-92-J-0125, and DARPA N0014-92-J-1799

§Dept. of Applied Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139. `spielman@math.mit.edu`. Partially supported by the Fannie and John Hertz Foundation, Air Force Contract F49620-92-J-0125 and NSF grant 9212184CCR. Part of this work was done while the author was at Bell Laboratories.

¶Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. `koods@theory.lcs.mit.edu`. Research supported by DARPA contract N0014-92-J-1799 and NSF 92-12184 CCR.

1. Introduction

Voter V is undecided about an important issue. A Republican wants to convince her to vote one way and a Democrat the other. What happens if the Republican has stolen the Democrat's briefing book and thus knows the Democrat's strategy? We show that if the voter can conduct private conversations with the Republican and the Democrat, then she can be convinced of how to vote on NEXPTIME issues, and, with suitable restrictions on communication and randomness, of issues in NP.

The framework of cooperating provers has received much attention. Babai, Fortnow and Lund [BFL91] showed that NEXPTIME is the set of languages that have two-cooperating-prover multi-round proof systems. This characterization was strengthened when [LS91, FL92] showed that NEXPTIME languages have two-cooperating-prover *one*-round proof systems. Recently, Feige and Kilian [FK94] have proved that NP is characterized by two-cooperating-prover one-round proof systems in which the verifier has access to only $O(\log n)$ random bits and the provers' responses are constant size. We show that two-*competing*-prover proof systems have similar power.

Stockmeyer [St77] used games between competing players to characterize languages in the polynomial-time hierarchy. Other uses of competing players to study complexity classes include [Reif84, PR79]. Feige, Shamir and Tennenholtz [FST88] proposed an interactive proof system in which the notion of competition is present. Recently, Condon, Feigenbaum, Lund and Shor [CFLS93a, CFLS93b] characterized PSPACE by systems in which a verifier with $O(\log n)$ random bits can read only a constant number of bits of a polynomial-round debate between two players. We show that Σ_k^P is the class of languages that can be recognized by a verifier with similar access to a k-round debate. We call this a system of k competing oracles.

We are naturally led to consider the scenario of k *competing* provers. In a competing-prover proof system two teams of competing provers $(P_i)_{i\in I}$ and $(P_i)_{i\in \bar{I}}$, $I \subseteq \{0,\ldots,k-1\}$, interact with a probabilistic polynomial-time verifier V. The first team of provers tries to convince V that ω is in L, for some pre-specified word ω and language L. The second team has the opposite objective, but V does not know which team to trust. Before the protocol begins, the provers fix their strategies in the order specified by their subindices. These strategies are deterministic. To model the situation of k competing provers, we propose and study the class k-APP of languages that have k-alternating-prover one-round proof systems.

Definition 1. A language L is said to have a k-APP system with parameters $(r(n), q(n), [Q_0,\ldots,Q_{k-1}])$, $Q_i \in \{\exists,\forall\}$, and error $(\epsilon_{acc},\epsilon_{rej})$ if there is a probabilistic polynomial-time non-adaptive verifier V that interacts with two teams of competing provers such that

if $\omega \in L$ then $Q_0 P_0,\ldots,Q_{k-1}P_{k-1}$ such that
$\mathrm{Prob}_r\,[(V \leftrightarrow P_0,\ldots,P_{k-1})(\omega,r)\ \mathit{accepts}] \geq 1-\epsilon_{acc}$,

if $\omega \notin L$ then $\bar{Q}_0 P_0,\ldots,\bar{Q}_{k-1}P_{k-1}$ such that
$\mathrm{Prob}_r\,[(V \leftrightarrow P_0,\ldots,P_{k-1})(\omega,r)\ \mathit{accepts}] \leq \epsilon_{rej}$,

where the verifier is allowed one round of communication with each prover and where the probabilities are taken over the random coin tosses of V. Furthermore, V uses $O(r(n))$ coin flips and the provers' responses are of size $O(q(n))$.

As in [FRS88], no prover has access to the communication generated by or directed to any other prover. We adopt the following conventions: when the parameters $r(n)$ and $q(n)$ are omitted, they are assumed to be $\log n$ and 1 respectively; when the quantifiers do not appear, they are assumed to alternate beginning with $\exists$; when $\epsilon_{acc} = \epsilon_{rej} = \epsilon$, we say the system has error ϵ. We define the class k-APP to be the set of languages that have a k-APP system with error $1/3$.

Proof systems related to the one given in Definition 1, but where the provers do not have access to each others' strategies, have been studied in [FST88, FKS93].

We define k-alternating-oracle proof systems analogously:

Definition 2. A language L is said to have a k-AOP system with parameters $(r(n), q(n), [Q_0,\ldots,Q_{k-1}])$, $Q_i \in \{\exists,\forall\}$, and error $(\epsilon_{acc},\epsilon_{rej})$ if there is a probabilistic polynomial-time non-adaptive verifier V that queries two teams of competing oracles such that

if $\omega \in L$ then $Q_0 O_0, \ldots, Q_{k-1} O_{k-1}$ such that
$\text{Prob}_r\,[(V \leftrightarrow O_0, \ldots, O_{k-1})(\omega, r)\ \textit{accepts}] \geq 1 - \epsilon_{acc}$,

if $\omega \notin L$ then $\bar{Q}_0 O_0, \ldots, \bar{Q}_{k-1} O_{k-1}$ such that
$\text{Prob}_r\,[(V \leftrightarrow O_0, \ldots, O_{k-1})(\omega, r)\ \textit{accepts}] \leq \epsilon_{rej}$,

where the probabilities are taken over the random coin tosses of V. Furthermore, V uses $O(r(n))$ coin flips and queries $O(q(n))$ bits.

The fundamental difference between an alternating-prover proof system and an alternating-oracle proof system is that provers are asked only one question, whereas oracles may be asked many questions but their answers may not depend on the order in which those questions are asked. This requirement was labeled in [FRS88] as the oracle requirement. It follows that a k-AOP system can be viewed as a k-APP system in which we have imposed the oracle requirement on the provers, and are allowed to ask many questions of each. The results of [LS91, FL92, FK94] show that, in many different scenarios, two-cooperating-prover one-round proof systems are equivalent in power to oracle proof systems. In this work we establish conditions under which alternating-prover proof systems are equivalent in power to alternating-oracle proof systems.

In Section 2, we introduce some of the techniques that we will use to prove our main theorem. Along the way, we show that the class of languages that have two-alternating-prover proof systems does not depend on the error parameter, that this class is equivalent to NP, and that two-alternating-prover proof systems lose power if the provers are allowed to choose randomized strategies. In Section 3, we show that k-alternating oracle systems characterize Σ_k^P. We use Section 4 to summarize work of Feige and Kilian [FK94] that we will need to prove our main theorem. In Section 5 we show that the class of languages that have a k-alternating-prover proof systems does not depend on the error parameter ϵ, and conclude that

Theorem 3. $\Sigma_{k-1}^P = k$-APP.

2. Two-Alternating Prover Proof Systems for NP

Suppose a language L can be recognized by a two-cooperating-prover one-round proof system with verifier $\widehat{V}$ and provers $\widehat{P}_0$ and $\widehat{P}_1$ quantified by $([\exists, \exists])$ in which $\widehat{V}$, on random string r, asks prover $\widehat{P}_0$ question $q^{(0)}(r)$, asks prover $\widehat{P}_1$ question $q^{(1)}(r)$, and uses the answers to the questions to decide whether or not to accept. We will transform this system into a two-alternating-prover proof system with verifier V and provers P_0, P_1 quantified by $([\exists, \forall])$. In this latter system, prover P_0 claims that there exist provers $\widehat{P}_0$ and $\widehat{P}_1$ that would convince $\widehat{V}$ to accept. Prover P_1 is trying to show that P_0 is wrong. The verifier V will simulate the verifier $\widehat{V}$ from the original system to generate the questions $q^{(0)}(r)$ and $q^{(1)}(r)$ that $\widehat{V}$ would have asked the cooperating provers. To justify his claim, P_0 will tell the verifier what $\widehat{P}_0$ or $\widehat{P}_1$ would have said in answer to any question. To test P_0's claim, V will pick one of the two questions $q^{(0)}(r)$ and $q^{(1)}(r)$ at random and ask P_0 to respond with what the corresponding prover would have said in answer to the question. Unfortunately, this does not enable V to get the answer to both questions. To solve this problem, we recall that P_1 knows what P_0 would answer to any question. Thus, the verifier V will send both questions to P_1, and request that P_1 respond by saying how P_0 would have answered the questions.

If $\omega \in L$, then P_0 is honest, and P_1 has to lie about what P_0 would say in order to get the verifier to reject. If P_1 lies about what P_0 said, he will be caught with probability at least $1/2$, because P_1 does not know which question V sent to P_0. Thus, the verifier will accept with probability at least $1/2$.

On the other hand, if $\omega \notin L$, then P_1 will honestly answer V by telling V what P_0 would have answered to both questions. In this case, V will accept only if the provers that P_0 represent would have caused $\widehat{V}$ to accept. Thus, we obtain a two-alternating-prover proof system with error $(1/2, \epsilon)$.

We will now show how we can balance these error probabilities by a parallelization of the above protocol with an unusual acceptance criteria. In our parallelized protocol, we have:

The Queries

- V generates m pairs of questions as $\widehat{V}$ would, *i.e.* V selects $r_1, \ldots, r_m \in \{0,1\}^{O(r(n))}$ and generates

$$\Big((q^{(0)}(r_1), q^{(1)}(r_1)), (q^{(0)}(r_2), q^{(1)}(r_2)),$$

$$\ldots, (q^{(0)}(r_m), q^{(1)}(r_m))\Big) .$$

V then picks one question from each pair, *i.e.* chooses $(j_1, \ldots, j_m) \in \{0,1\}^m$, and sends to P_0 the questions $q^{(j_1)}(r_1), q^{(j_2)}(r_2), \ldots, q^{(j_m)}(r_m)$ and the tuple $\vec{\jmath} = (j_1, \ldots, j_m)$.

- V sends all the questions

$$\Big((q^{(0)}(r_1), q^{(1)}(r_1)), (q^{(0)}(r_2), q^{(1)}(r_2)), \ldots, (q^{(0)}(r_m), q^{(1)}(r_m))\Big)$$

to P_1.

The Responses

- P_0 is supposed to respond with

$$\Big(\widehat{P}_{j_1}(q^{(j_1)}(r_1)), \widehat{P}_{j_2}(q^{(j_2)}(r_2)), \ldots, \widehat{P}_{j_m}(q^{(j_m)}(r_m))\Big) ,$$

which is how the provers $\widehat{P}_0$ and $\widehat{P}_1$ would have answered the questions.

- P_1 is supposed to respond with what P_0 would have said to $q^{(i_1)}(r_1), \ldots, q^{(i_m)}(r_m)$ and $\vec{\imath} = (i_1, \ldots, i_m)$, say $a_1^{\vec{\imath}}, \ldots, a_m^{\vec{\imath}}$, for every one of the 2^m possible assignments to $\vec{\imath}$ (we allow P_1 to answer this way because a cheating prover P_0 may vary its strategy according to the value of $\vec{\jmath}$ that it receives).

Acceptance Criteria

- V accepts if P_1 did not correctly represent what P_0 said on the tuple that P_0 was asked.
- V accepts if there exists a round k and index vector $\vec{\imath}$ such that $(\vec{\imath})_k = 0$ and $(\vec{\jmath})_k = 1$ (switch $\vec{\imath}$ and $\vec{\jmath}$ if necessary), and such that $\widehat{V}$ on random string r_k would accept if given answers $a_k^{\vec{\imath}}$ and $a_k^{\vec{\jmath}}$.

Lemma 4. *A two-cooperating-prover one-round proof system with error $(0, \epsilon)$ can be simulated by a two-alternating-prover one-round proof system with error $(2^{-m}, m2^{m-1}\epsilon)$, an m-fold increase in the verifier's randomness and an exponential in m increase in the length of the answers returned by the provers.*

Proof: Use the proof system described above. If $\omega \in L$, then P_0 is going to honestly answer the question he is asked in each round. We will see that this forces P_1 to lie in response to all but the set of indices $\vec{\jmath}$. If there is any m-tuple of questions indexed by $\vec{\imath}$, $\vec{\imath} \neq \vec{\jmath}$, for which P_1 honestly represents what P_0 would say, then there must be a k such that $(\vec{\imath})_k \neq (\vec{\jmath})_k$. Thus, if P_1 honestly represented P_0's answers on tuple $\vec{\imath}$, the verifier would accept. If $\omega \notin L$, then the probability that V accepts is at most ϵ for each k, and $\vec{\imath}$ such that $(\vec{\imath})_k \neq (\vec{\jmath})_k$, for a total error of $m2^{m-1}\epsilon$. $\square$

We will combine this Lemma with a Theorem of Feige and Kilian [FK94] which proves a weak version of the parallel repetition conjecture [FL92]. (For a formal statement of the parallel repetition conjecture see [FL92]).

Theorem 5 ([FK94]). *For any constant $\epsilon > 0$. A language L is in* NP *iff L has a* 2-APP *system with parameters $([\exists, \exists])$ and error $(0, \epsilon)$.*

Corollary 6. *For any constant ϵ, $0 < \epsilon < 1/2$. A language L is in* NP *iff L has a* 2-APP *system with error ϵ.*

Proof: The reverse direction is trivial, the other direction is a direct consequence of Theorem 5 and Lemma 4. □

Corollary 7. *A language L is in* NEXPTIME *iff L has a* 2-APP *system with parameters $(poly(n), poly(n), [\exists,\forall])$ and error $(1/poly(n), 1/exp(n))^1$.*

Proof: Again, the reverse implication is trivial. To prove the forward direction, observe that in [FL92] it is shown that a language in NEXPTIME has a 2-APP system with parameters $(poly(n), poly(n), [\exists, \exists])$ and error $(0, 1/exp(n))$. Hence the corollary follows again from Lemma 4. □

Remark 8. *In Section 5, we will want to use the following stronger version of Lemma 4: assume that the original two-cooperating-prover one-round proof system with verifier $\widehat{V}$ and honest provers $\widehat{P}_0$ and $\widehat{P}_1$ had two-sided error $(\epsilon_{acc}, \epsilon_{rej})$. We will say that $\widehat{V}$* accepts *in round k if there exists an index vector $\vec{\imath}$ such that $(\vec{\imath})_k = 0$ and $(\vec{\jmath})_k = 1$ (switch $\vec{\imath}$ and $\vec{\jmath}$ if necessary), and such that $\widehat{V}$ on random string r_k would accept if given answers $a_k^{\vec{\imath}}$ and $a_k^{\vec{\jmath}}$. After P_0 and P_1 have answered V's questions, we will allow player P_0 to arbitrarily choose some ξ fraction of the m rounds and we will say that V* accepts *in a round if either $\widehat{V}$ accepted in that round, or if that round was in the fraction that P_0 chose (Imagine that P_0 is able to alter the computation of V on those rounds). We will change the verifier's acceptance criteria to:*

- *V accepts if P_1 did not correctly represent what P_0 said on the tuple $\vec{\jmath}$ that P_1 was asked.*
- *V accepts if it accepts for a $(\xi+\beta\epsilon_{acc})$ fraction of the rounds (we will set β so that $\beta(1-\frac{1}{\beta}-\ln\frac{1}{\beta}) = -1$, because we will use a Chernoff bound to bound the probability of error).*

This two-alternating-prover proof system has error $(e^{-m\epsilon_{acc}} + 2^{-(1-2\xi-2\beta\epsilon_{acc})m},\ m2^{m-1}\epsilon_{rej})$.

In two-*cooperating*-prover proof systems, the power of the proof system is unchanged if the provers are allowed to choose randomized strategies. We end this section by observing that the situation for competing-prover proof systems differs.

Say language L has a *two-alternating randomized-prover proof system*, denoted 2-ARP, if it can be recognized by a 2-APP system where the provers are allowed to have randomized strategies, that is, before the protocol begins the provers now choose randomized strategies (instead of only deterministic strategies), in the order specified by their subindices. Equivalently, a 2-ARP system is a 2-APP system where the provers have access to private coins.

Lemma 9. *For any constant ϵ, $0 < \epsilon < 1/2$, a language L is in* P *iff L has a* 2-ARP *system with error ϵ.*

Proof: [Sketch] The forward direction is trivial. To prove the converse, observe first that if ρ_i, $i \in \{0,1\}$, is the probability distribution over deterministic strategies that prover P_i chooses, then the probability p_ω that the verifier V accepts input ω, $n = |\omega|$, depends on ρ_i through the probabilities $\rho_i(q_i, a_i) = \mathrm{Prob}_{P_i \leftarrow \rho_i}[P_i(q_i) = a_i]$, where q_i ranges over the set Q_i of possible questions to prover P_i, and a_i over the set A_i of possible responses of prover P_i.

Let,

$\vec{\rho}_i = (\rho_i(q_i, a_i))_{q_i \in Q_i, a_i \in A_i}$.

R be the set of random strings that the verifier may generate on input ω.

π_r be the probability that the verifier generates random string r.

$q_i(r)$ be the question that the verifier sends to prover P_i on random string r.

V_{r,a_0,a_1} be equal to 1 if on input ω, random string r, and provers' responses a_0 and a_1 the verifier accepts, and 0 otherwise.

[1] $poly(n)$ and $exp(n)$ refer to $O(n^c)$ and $O\left(2^{n^c}\right)$ respectively, for some positive constant c.

Applying the technique of [FL92], it follows that:

$$p_\omega = \max_{\vec{\rho}_0} C(\vec{\rho}_0), \qquad \text{subject to}$$
$$\forall q_0 \quad \sum_{a_0} \rho_0(q_0, a_0) = 1, \qquad \vec{\rho}_0 \geq 0,$$

where, $C(\vec{\rho}_0) =$

$$\min_{\vec{\rho}_1} \sum_{r,a_0,a_1} \pi_r V_{r,a_0,a_1}\, \rho_0(q_0(r), a_0)\, \rho_1(q_1(r), a_1)$$

subject to

$$\forall q_1 \quad \sum_{a_1} \rho_1(q_1, a_1) = 1, \qquad \vec{\rho}_1 \geq 0,$$

where r, q_0, q_1, a_0 and a_1 range over R, Q_0, Q_1, A_0 and A_1 respectively. By strong duality (see [Sc86]), $C(\vec{\rho}_0)$ can be expressed as the optimum of a linear program in max form. Thus, to compute p_ω it is enough to solve a linear program of $poly(n, |A_0|, |A_1|, |R|)$ size. Since in our case $|A_0|, |A_1| = O(1)$, $|R| = 2^{O(\log n)}$, and linear programming is polynomial-time solvable, the lemma follows. $\Box$

An analogous result for EXPTIME, for the only if part, was independently obtained in [FKS93].

3. AOP Systems for Σ_k^P

Feige and Kilian's proof of Theorem 5 uses an amplification of the main result of [AS92, ALMSS92], which states that NP = PCP($\log n, 1$). In our terminology, this says that NP languages have 1-AOP proof systems. In order to extend our results beyond NP, we will need analogous tools that we can apply to languages in Σ_k^P. We will begin by showing that languages in Σ_k^P have k-AOP proof systems.

We consider from now on only the case in which k is odd. Our results have analogous statements for even k.

The next theorem is implicit in [CFLS93a].

Theorem 10 (k odd). *For any constant $\epsilon > 0$. Every language L in Σ_k^P has a k-AOP system with error $(0, \epsilon)$.*

In the proof of this theorem, we will make use of some facts about Justesen codes. For a string x, let $E(x)$ denote the Justesen encoding of x [MS77]. The following facts are standard:

- The length of $E(x)$ is linear (for our purposes, polynomial would suffice) in the length of x.
- There is a polynomial time algorithm that on input x returns $E(x)$.
- There is a constant ϵ_J and a polynomial time algorithm C_J such that if y and $E(x)$ differ in at most an ϵ_J fraction of their bits, then $C_J(y) = x$ (in which case we say that x and y are *closer than* ϵ_J). Otherwise, $C_J(y)$ outputs "FAILURE" (in which case, we say that y is *farther than* ϵ_J from any codeword).

Proof: [of Theorem 10] Let L be a language in Σ_k^P. That is, there exists a polynomial-time Turing machine V such that $\omega \in L$ if and only if $\exists X_1, \forall X_2, \ldots, \exists X_k\ V(\omega, X_1, \ldots, X_k)$ accepts [CKS81]. As in [CFLS93a], we will view the acceptance condition as a game between an $\exists$ player and a $\forall$ player who take turns writing down polynomial-length strings X_i, with the $\exists$ player writing on the odd rounds.

In our k-AOP, the player who writes in round i purports to write down a Justesen encoding of X_i. In addition, in the k-th round, the $\exists$ player is to write down encodings of everything the $\forall$ player said and a $PCP(\log n, 1)$ proof that $V(\omega, X_1, \ldots, X_k)$ accepts. If each player actually wrote down codewords, then, using the techniques from [ALMSS92], the verifier would read a constant number of random bits from each oracle and a constant number of bits to check the $PCP(\log n, 1)$ proof and accept if $V(\omega, X_1, \ldots, X_k)$ would have accepted, or reject with high probability if V would have rejected.

In reality, one of the players will be trying to cheat and will have little incentive to write codewords. Let Y_i denote the oracle that is written in the i-th round. If a player writes an oracle Y_i that is within ϵ_J of a codeword, then the other player will proceed as if Y_i was that codeword. On the other hand, if a player writes an oracle Y_i that is farther than ϵ_J from a codeword, then with high probability the verifier will detect that player's perfidy.

In the last round, the $\exists$ player writes down strings Z_i which he claims are encodings of Y_i, for each even i. In addition, the $\exists$ player will, for each bit of each oracle Y_i, provide a $PCP(\log n, 1)$ proof that, when decoded, Z_i agrees with Y_i on that bit. For each even i, the verifier will test these $PCP(\log n, 1)$ proofs to check that $C_J(Z_i)$ agrees with Y_i on some constant number of randomly chosen bits. If any of these tests fails, then the verifier will know that the $\exists$ player has cheated and will reject accordingly. If the Z_i's pass all the tests, then the verifier will be confident that $C_J(Z_i)$ is close to Y_i, for all even i. The verifier then accepts if the last player's proof indicates either that,

- $C_J(Y_i) =$ "FAILURE" for some even i, or
- $V(\omega, X_1, C_J(Y_2), \ldots, C_J(Y_{k-1}), X_k)$ accepts (note that the $PCP(\log n, 1)$ proof refers to the X_i's for odd i through their encoding as Y_i, and to the Y_i's for i even through their encoding as Z_i).

To see why this protocol works, assume that $\omega \in L$. In this case, the $\exists$ player will always write codewords. Moreover, regardless of what oracle Y_i the $\forall$ player writes in turn i, the exist player will write $Z_i = E(Y_i)$. Thus, the Z_i's will always pass the consistency tests. If one of the $\forall$ player's oracles is farther than ϵ_J from a codeword, then the last player will include this fact in his proof, and the verifier will reject. On the other hand, if for each even i, Y_i is close to some codeword X_i, then the application of C_J to Y_i will result in Y_i being treated as the encoding of X_i in the $\exists$ player's $PCP(\log n, 1)$ proof.

If $x \notin L$, then the $\exists$ player will have to cheat in order to win. If the $\exists$ player writes a message Y_i that is not close to a unique codeword, then this will be detected with high probability when the verifier checks the validity of the $PCP(\log n, 1)$ proof supplied in the last round. If one of the Z_i's misrepresents the oracle, $Y_i = E(X_i)$, of a $\forall$ player, then either $C_J(Z_i)$ and Y_i will have to differ in at least an ϵ_J fraction of their bits, or the $\exists$ player will have to falsify the certification of the computation of C_J on $C_J(Z_i)$ so that it does not decode to X_i. In either case, the $\exists$ player will be caught with high probability. $\square$

We note that Theorem 10 exactly characterizes Σ_k^P because an alternating-Turing machine with k alternations can guess the k oracles and then compute the acceptance probability of the k-AOP system.

Using Theorem 10 and the standard technique of [FRS88] for simulating an oracle by a pair of provers, we can transform a k-alternating-oracle proof system into a $(k+1)$-alternating-prover proof system.

Corollary 11 (k odd). *$L \in \Sigma_k^P$ iff L has a $(k+1)$-APP system with parameters $([\exists, \forall, \exists, \ldots, \forall, \exists, \exists])$ and error $(0, 1 - \frac{1}{N})$, where N is a large constant depending on k.*

In order to prove Theorem 3, we have to reduce the error in Corollary 11 and show how to change the parameters $([\exists, \forall, \exists, \ldots, \forall, \exists, \exists])$ to $([\exists, \forall, \exists, \ldots, \forall, \exists, \forall])$. In the next two sections we present some of the necessary ideas that we need to achieve these goals.

4. Previous Work

In Section 5, we will prove an analogue of the theorem of Feige and Kilian [FK94] which applies to k competing provers. The techniques used in [FK94] provide a deep insight into how a few random variables influence the value of a multi-variate function. In order to prove our analogue of their theorem, we will need a better understanding of some of their results, which we will summarize in this section.

Consider a prover P to which we send m randomly suggested questions $q(r_1), \ldots, q(r_m)$ and to which P answers according to a fixed strategy $f = (f_1, \ldots, f_m)$, a function from m-tuples (the m-tuple of questions that P receives), to m-tuples (the m-tuple of answers that P responds with). We would like P to use a global strategy, f, in which each f_i is only a function of $q(r_i)$. We say that such a prover behaves *functionally*. Below, we state the consequences of a prover's failure to behave in this way.

Say that an *f-challenge*[2] $(\tilde{I},(q(r_i))_{i\in\tilde{I}})$ is *live* if $\exists(a_i)_{i\in\tilde{I}}$ such that

$$\operatorname*{Prob}_{(r_i)_{i\notin\tilde{I}}}\left[(f_i(q(r_1),\ldots,q(r_m)))_{i\in\tilde{I}}=(a_i)_{i\in\tilde{I}}\right]\geq\zeta,$$

where ζ is a parameter to be fixed later. If the above inequality holds, say that $(a_i)_{i\in\tilde{I}}$ is a *live answer set* for the challenge $(\tilde{I},(q(r_i))_{i\in\tilde{I}})$. Intuitively, if we know the questions $\tilde{Q}=(q(r_i))_{i\in\tilde{I}}$ that P receives on rounds $\tilde{I}$ and that $\tilde{A}=(a_i)_{i\in\tilde{I}}$ is not a live answer set for the challenge $(\tilde{I},\tilde{Q})$, we should not be willing to bet that P will answer $\tilde{A}$ on rounds $\tilde{I}$. If P acts functionally in each round, then every challenge $(\tilde{I},\tilde{Q})$ is live, since the questions that P is sent on a round completely determine his answer on that round.

For any choice of parameters such that $8\epsilon<\zeta^5$, $\frac{2}{\gamma}\zeta^{-5}<n$, $0<\gamma<1$, $\epsilon\geq\max\left(\frac{256\ln M}{M},\left(\frac{8\ln M}{M}\right)^{\frac{1}{3}}\right)$, where $M=m-n$, the following lemmas are implicit in the work of Feige and Kilian.

Lemma 12. *If $\tilde{A}=(a_i)_{i\in\tilde{I}}$ is not a live answer set for the challenge $(\tilde{I},(q(r_i))_{i\in\tilde{I}})$, then, with probability at least $1-n\epsilon$ (over the choices of $\{i_{j+1},\ldots,i_n\}$ and $(r_i)_{i\in I\setminus\tilde{I}}$, where $I=\{i_1,\ldots,i_n\}$ and $\tilde{I}=\{i_1,\ldots,i_j\}$) it holds that*

$$\operatorname*{Prob}_{(r_i)_{i\notin I}}\left[f_i(q(r_1),\ldots,q(r_m))=a_i,\ \textit{for all } i\in\tilde{I}\right]<\zeta+n\epsilon.$$

Intuitively, the preceding Lemma says that if $\tilde{A}$ is not a live answer set for the challenge $(\tilde{I},(q(r_i))_{i\in\tilde{I}})$, and in addition to knowing the questions that P receives in rounds $\tilde{I}$ we know the questions sent to P in rounds $I\supseteq\tilde{I}$, we usually should still not be willing to bet that P will answer $\tilde{A}$ in rounds $\tilde{I}$. Observe that knowledge of all the questions that P receives completely determines the answers in rounds $\tilde{I}$.

Lemma 13. *There is a good j for f, $j<\gamma n$, such that if more than a ζ fraction of the challenges $(\tilde{I},(q(r_i))_{i\in\tilde{I}})$ (over the choices of $\tilde{I}=\{i_1,\ldots,i_j\}$ and $(r_i)_{i\in\tilde{I}}$) are live, then for a $(1-\zeta)$ fraction of the live challenges $(\tilde{I},\tilde{Q}=(q(r_i))_{i\in\tilde{I}})$, for every $i\notin\tilde{I}$, and for each live answer set $\tilde{A}=(a_i)_{i\in\tilde{I}}$ for the challenge $(\tilde{I},\tilde{Q})$, it holds that there is a function $F_{f,i,\tilde{Q},\tilde{A}}$ such that*

$$\Pr\left[\frac{\left|\{i\in I\setminus\tilde{I}:f_i(q(r_1),\ldots,q(r_m))\neq F_{f,i,\tilde{Q},\tilde{A}}(q(r_i))\}\right|}{|I\setminus\tilde{I}|}\leq\frac{2\zeta}{\delta}\right]$$
$$\geq 1-\delta,$$

where the probability is taken over the choices of $(r_i)_{i\notin\tilde{I}}$ and $I\setminus\tilde{I}=\{i_{j+1},\ldots,i_n\}$.

In other words, to every prover P we can associate a j such that if a non-negligible fraction of the challenges $(\tilde{I},\tilde{Q})$ are live, where $|\tilde{I}|=j$, and if P answers the challenge $(\tilde{I},\tilde{Q})$ with live answer set $\tilde{A}$, then, P acts 'almost' functionally in a significant fraction of the rounds $I\setminus\tilde{I}$, for most of the live challenges $(\tilde{I},\tilde{Q})$.

5. APP systems for Σ_k^P

In Corollary 11 we characterized Σ_k^P in terms of alternating-prover proof systems with error $(0,1-\frac{1}{N})$, where N is a large constant, and where the quantifiers associated to the provers, *except* the last two, alternated. The goal of this Section is to again provide alternating-prover proof systems for Σ_k^P languages which in addition achieve:

- Low error rates, and
- The quantifiers associated to *all* the provers alternate.

[2] We omit f whenever it is clear from context.

To achieve these goals we follow an approach similar to the one taken in Section 2. First, we prove that languages that have a $(k+1)$-APP system with parameters $([\exists,\forall,\exists,\ldots,\forall,\exists,\exists])$ and error $(0,1-\frac{1}{N})$, can be recognized by a $(k+3)$-APP system with parameters $([\exists,\forall,\exists,\ldots,\forall,\forall,\exists,\exists,\exists])$ and error α, for any constant α, $0<\alpha<1/2$.[3] Notice that we achieve this without increasing the number of alternations of the quantifiers associated to the provers. This step can be viewed as an analogue of Theorem 5 which applies to competing-prover proof systems. We then show how to convert the proof systems obtained in the first step into $(k+1)$-APP systems in which all the quantifiers associated to the provers alternate. This latter step is performed without significantly increasing the error probability.

We describe below a protocol which achieves the first goal of this section. Consider a language that has a $(k+1)$-APP with parameters $([\exists,\forall,\exists,\ldots,\forall,\exists,\exists])$ and error $(0,1-\frac{1}{N})$ with verifier $\widehat{V}$ and provers $\widehat{P}_0,\ldots,\widehat{P}_k$. Let $q^{(l)}(r)$ denote the question that the verifier $\widehat{V}$, on random string r, asks the prover $\widehat{P}_l$. Consider now a verifier V interacting with provers $P_0,\ldots,P_{k+2}$ quantified by $([\exists,\forall,\exists,\ldots,\forall,\exists,\exists,\exists,\forall])$ respectively. The underlying idea of the protocol we are about to describe is the following: the verifier V tries to parallelize $\widehat{V}$'s protocol. In order to force cheating provers to behave functionally, a Consistency Test is implemented. This test requires the use of two (one for each competing team of provers) additional provers (P_{k+1} and P_{k+2}). If a team of provers fails the consistency test then their claim will be rejected. Honest provers will behave functionally in each round and thus they will always be able to PASS the Consistency Test. Nevertheless, it may be that cheating provers have a significant probability of passing the Consistency Test, but honest provers cannot determine the strategy which the cheating provers use to answer each round from the knowledge of their strategy alone. The protocol implemented by the verifier V will allow honest provers to make a set of 'educated' guesses regarding the strategies that the cheating provers might be using to answer each of the questions posed to them. Thus, either cheating provers will fail the Consistency Test, or the verifier V will end up (with high probability) with a set of rounds most of which can be treated as independent executions of $\widehat{V}$'s protocol.

In what follows we describe the questions that V makes, the format in which the provers are supposed to answer, and the acceptance criteria.

The Queries

- V chooses $I\subset\{1,\ldots,m\}$, $|I|=n$ at random, and selects for every $i\in I$ random string r_i as the old verifier $\widehat{V}$ would have.
- V selects $\forall j\in\{0,\ldots,k\}$, $\forall i\in\{1,\ldots,m\}\setminus I$, random string $r_i^{(j)}$ as the old verifier $\widehat{V}$ would have.
  For $i\in\{1,\ldots,m\}$, $j\in\{0,\ldots,k\}$ let
  $$\rho_i^{(j)}=\begin{cases} r_i & \text{if } i\in I,\\ r_i^{(j)} & \text{otherwise.}\end{cases}$$
- V chooses $I_l\subset I\setminus\bigcup_{0\le j<l}I_j$, $|I_l|=\gamma n$, at random together with a random ordering π_l of I_l for $l=0,\ldots,k$ (we will later set $\gamma=\frac{1}{2(k+1)}$).
- V sends to prover P_l, $l\in\{0,\ldots,k\}$, $(q^{(l)}(\rho_i^{(l)});\ i\in\{1,\ldots,m\}\setminus\bigcup_{0\le j<l}I_j)$, (when a question is sent to a prover the verifier indicates the round $i\in\{1,\ldots,m\}$ to which the question corresponds).
- V sends to prover P_{l+1}, $l\in\{0,\ldots,k-1\}$ all questions $(q^{(j)}(r_i))_{i\in I_j}$, the set of indices I_j, and the orderings π_j, for all $j\in\{0,\ldots,l\}$.
- V sends to provers P_{k+1} and P_{k+2} the questions $(q^{(l)}(r_i))_{i\in I_l}$, the set of indices I_l and the orderings π_l, for all $l\in\{0,\ldots,k\}$.

The Responses

[3] In fact, a $(k+2)$-APP with parameters $([\exists,\forall,\exists,\ldots,\forall,\forall,\exists,\exists])$ v and similar error rates suffices, but proving this would unnecessarily complicate our exposition.

The format of each prover's answer is a tree.

For a graph tree T (we will only consider rooted trees), we say that a node $v \in V(T)$ is at level i if its distance to the root of T is i. We say that an edge $e \in E(T)$ is at level i if it is rooted at a node of level i.

Definition 14. Let $J \subseteq \{0,\ldots,r\}$ be a set of indices. We say that a tree T is a J-tree if nodes at level $j \in J$ have only one child.

We will consider the nodes and edges of our trees as being labeled. Internal nodes will be labeled by sets of indices. Edges rooted at a node labeled J will be labeled by a tuple of strings $(a_j)_{j\in J}$. We denote the label of a node v (resp. edge e) by $\mathcal{L}_T(v)$ (resp. $\mathcal{L}_T(e)$).

Definition 15. Let v be a node of level l of tree T, and $v_0,\ldots,v_{l-1}$ the sequence of nodes along the path from the root of T to v. Define the history of v as follows:

$$\mathcal{H}^T_v = [\mathcal{L}_T(v_0), \mathcal{L}_T(v_0, v_1)] \ldots [\mathcal{L}_T(v_{l-1}), \mathcal{L}_T(v_{l-1}, v)]$$

P_l responds with a tree T_l.

- T_l, for $l \in \{0,\ldots,k\}$ is a tree of depth l labeled as follows:
  - An internal node v at level j is labeled by I_j.
  - Every internal node v at level j has an edge rooted at v for every possible set of answers of P_j to questions $(q^{(j)}(r_i))_{i\in I_j}$, and labeled by this tuple of answers.
  - A leaf is labeled "CONCEDE DEFEAT", "GARBAGE" or by the answers to the questions $\left(q^{(l)}(r_i); i \in \{1,\ldots,m\} \setminus \bigcup_{0\le j<l} I_j\right)$.
- T_{k+1} (resp. T_{k+2}) is a K-tree of depth $k+1$, where $K = \{0,2,\ldots,k-1,k\}$ (resp. $K = \{1,3,\ldots,k-2\}$), labeled as follows:
  - Nodes and edges at levels $j \notin K$ are labeled as in the trees described above.
  - The only edge of level $j \in K$ rooted at v is labeled by a tuple of answers $(a_i^{(j)})_{i\in I_j}$.

Assume now that $T_0,\ldots,T_k$ are in the proper format. Define the sequence of leaves $v_l \in V(T_l)$, $l \in \{0,\ldots,k\}$ as follows: v_0 is the only node of T_0, if $v_0,\ldots,v_l$ have been determined, and $\left(a_i^{(j)}; i \in \{1,\ldots,m\} \setminus \bigcup_{0\le s<j} I_s\right)$ is the label of v_j, then v_{l+1} is the leaf of tree T_{l+1} with history

$$[I_0, (a_i^{(0)})_{i\in I_0}] \ldots [I_l, (a_i^{(l)})_{i\in I_l}].$$

Define for $j \in \{0,\ldots,k\}$ the following *path*:

$$\mathcal{P}(T_0,\ldots,T_j) = [I_0, (a_i^{(0)})_{i\in I_0}] \ldots [I_j, (a_i^{(j)})_{i\in I_j}].$$

Note that there is only one leaf in T_{k+1} and T_{k+2} that have the same history. The history of both these leaves determines a common path of T_{k+1} and T_{k+2} which we denote by,

$$\mathcal{P}'(T_0,\ldots,T_j) = [I_0, ({a'}_i^{(0)})_{i\in I_0}] \ldots [I_j, ({a'}_i^{(j)})_{i\in I_j}].$$

Acceptance Criteria

The verifier V checks that

i. The responses of the provers have the proper format. If a prover does not comply with the format of the responses its claim is immediately rejected (honest provers will always answer in the proper format).

ii. The following Consistency Test PASSES.
   V compares $\mathcal{P} = \mathcal{P}(T_0, \ldots, T_k)$ and $\mathcal{P}' = \mathcal{P}'(T_{k+1}, T_{k+2})$. Two different situations may arise:
   (a) $\mathcal{P} = \mathcal{P}'$, in this case we say that the consistency test PASSES.
   (b) $\mathcal{P} \neq \mathcal{P}'$, in this case let l be the smallest j such that $(a_i^{(j)})_{i \in I_j} \neq (a'^{(j)}_i)_{i \in I_j}$. V then rejects P_l's claim.

V then accepts iff for more than a $1 - \frac{1}{cN}$ fraction of the rounds $i \in I \setminus \bigcup_{0 \leq j \leq k} I_j$ the old verifier $\widehat{V}$ on random string r_i would accept if given answers $a_i^{(0)}, \ldots, a_i^{(k)}$ from provers $\widehat{P}_0, \ldots, \widehat{P}_k$ respectively, (where c is a large constant whose value we will set later).

Lemma 16 (k odd). *For any constant α, $0 < \alpha < 1/2$. Every language L recognized by a $(k+1)$-APP with parameters $([\exists, \forall, \ldots, \forall, \exists, \exists])$ and error $(0, 1 - \frac{1}{N})$, where N is a large constant, has a $(k+3)$-APP with parameters $([\exists, \forall, \ldots, \forall, \forall, \exists, \exists, \exists])$ and error α.*

Proof: [Sketch] Use the protocol described above. First we show how honest provers respond. Assume P_l is honest, $l \leq k$, then P_l will generate a tree T_l of the proper format. We only have to show how the leaves on T_l are labeled. Let $u_0, \ldots, u_l$ be a path in T_l from the root u_0 to the leaf u_l. In labeling u_l, P_l considers the strategies $f_0, \ldots, f_{l-1}$ that $P_0, \ldots, P_{l-1}$ use in labeling the leaves (abusing notation) $u_0, \ldots, u_{l-1}$ of $T_0, \ldots, T_{l-1}$ with history $\mathcal{H}_{u_0}^{T_l}, \ldots, \mathcal{H}_{u_{l-1}}^{T_l}$ respectively. For $s \in \{0, \ldots, l-1\}$, let j_s be the smallest good j, as defined by Lemma 13, for f_s. Let $\tilde{I}_s \subseteq I_s$ be the set of the first j_s indices of I_s as determined by π_s. Let $\tilde{Q}_s = (q^{(s)}(r_i))_{i \in \tilde{I}_s}$ be the set of questions on rounds $\tilde{I}_s$, and $\tilde{A}_s = (\tilde{a}_i^{(s)})_{i \in \tilde{I}_s}$ be the set of answers in rounds $\tilde{I}_s$ induced by the label of the leaf u_s.

Define the following events:

Event E_1 : $\forall s \in \{0, \ldots, l-1\}$, the answer sets $\tilde{A}_s$ are live for the f_s-challenge $(\tilde{I}_s, \tilde{Q}_s)$.

Event E_2 : $\forall s \in \{0, \ldots, l-1\}$, more than a ζ fraction of the f_s-challenges $(\tilde{I}_s = \{i_1, \ldots, i_{j_s}\}, (q^{(s)}(r_i))_{i \in \tilde{I}_s})$ (over the choices of $\tilde{I}_s = \{i_1, \ldots, i_{j_s}\}$ and $(r_i)_{i \in \tilde{I}_s}$) are live.

For an appropriate choice of parameters we can distinguish two cases,

Events E_1 and E_2 occur: $\forall i \in \{1, \ldots, m\} \setminus \bigcup_{0 \leq s < l} I_s$, $\forall s \in \{0, \ldots, l-1\}$, P_l determines (if possible) the functions $F_{f_s, i, \tilde{Q}_s, \tilde{A}_s}$ (as in Lemma 13), and the optimal strategy for answering round i, say $P_i^{u_l}$, of an $(l+1)$-th prover of a k-APP of the form given in Corollary 11, where the first l provers are given by $F_{f_0, i, \tilde{Q}_0, \tilde{A}_0}, \ldots, F_{f_{l-1}, i, \tilde{Q}_{l-1}, \tilde{A}_{l-1}}$. If P_l is unable to determine some $F_{f_s, i, \tilde{Q}_s, \tilde{A}_s}$ he labels u_l with "CONCEDE DEFEAT", that is, P_l recognizes that cheating provers have outsmarted him. Otherwise, P_l answers round i of u_l according to $P_i^{u_l}$.

Otherwise: P_l labels u_l with "GARBAGE", that is, P_l expresses its confidence that the cheating provers will not PASS the consistency test.

It follows that honest provers never fail the consistency test, since in each leaf of their respective response tree honest provers answer each round functionally.

Consider now the leaf v_0 of T_0 and the leaves $v_1, \ldots, v_k$ of $T_1, \ldots, T_k$ with history $\mathcal{P}(T_0)$, $\ldots, \mathcal{P}(T_0, \ldots, T_{k-1})$, fix above $u_i = v_i$, for $i \in \{0, \ldots, k\}$, $l = k+1$, and define events E_1 and E_2 as before.

If E_1 does not occur: then, from Lemma 12, cheating provers will PASS the consistency test with probability at most $\zeta + 2(k+1)\gamma n \epsilon \leq 3\zeta$ (assuming $(k+1)\gamma n \epsilon \leq \zeta$).

If E_2 does not occur: then from the preceding paragraph, we conclude that cheating provers will PASS the consistency test with probability at most 4ζ.

If E_1 and E_2 occur: then, Lemma 13 implies that with probability at most $\frac{(k+2)}{2}\zeta$ the honest provers are outsmarted by the cheating provers; otherwise, with probability at least $1 - \frac{k+2}{2}\delta$ at least $(1 - (k+1)\gamma - \frac{(k+2)\zeta}{\delta})n$ of the rounds $i \in I \setminus \bigcup_{0 \leq j \leq k} I_j$ are answered according to $F_{f_0, i, \tilde{Q}_0, \tilde{A}_0}, \ldots, F_{f_k, i, \tilde{Q}_k, \tilde{A}_k}$.

Let $p = \frac{(k+2)}{2}(\delta + \zeta) + 7\zeta$, and $n' = (1 - (k+1)\gamma)n$. Now, choose $\gamma = \frac{1}{2(k+1)}$, $\delta = \sqrt{\zeta}$, ζ small enough so $\zeta \leq 1/(2cN(k+2))^2$ (hence $\mu = (1 - (k+1)\gamma - \frac{(k+2)\zeta}{\delta})\frac{n}{n'} \geq 1 - \frac{1}{cN}$) and $p = \frac{(k+2)}{2}(\sqrt{\zeta} + \zeta) + 7\zeta \leq \alpha/2$, n large enough so $n \geq 2N\log(2/\alpha)$ and $n > 2\zeta^{-5}/\gamma$, ϵ small enough so $8\epsilon < \zeta^5$ and $(k+1)\gamma n\epsilon \leq \zeta$, M large enough so if $M = m - n$, $\epsilon \geq \max(\frac{256\ln M}{M}, (\frac{8\ln M}{M})^{1/3})$. Remember that all the above mentioned parameters are constants.

If $\omega \in L$, with probability at least $1 - p \geq 1 - \alpha/2$, the old verifier will accept for a fraction of at least $\mu \geq 1 - \frac{1}{cN}$ of the rounds $i \in I \setminus \bigcup_{0 \leq j \leq k} I_j$, and thus V will accept.

If $\omega \notin L$, then the probability of V accepting is at most $p + 2^{-n'/N} \leq \alpha$, (where the bound on the probability of acceptance follows by choosing c to be a constant large enough so that $h(1-\frac{2}{cN}) - \frac{\log e}{N}(1-\frac{2}{cN}) \leq -1/N$, where $h(\cdot)$ is the binary entropy function).

Finally, note that the order of the provers might as well have been $P_0, \ldots, P_{k-2}, P_{k+2}, P_{k-1}, P_k, P_{k+1}$. □

Note that the protocol used to prove Lemma 16, is more generous with the cheating provers than actually required. We can modify the protocol by requiring that prover P_l respond with a tree T_l which is also an S_l-tree, where $s \in S_l$ if $s < l$ and prover P_s is in P_l's team. The labels of the edges of T_l at level s, $s \in S_l$, can now be required to be consistent with the labels that P_s assigns to the leaves of T_s. Combining this modified protocol, the protocol that Feige and Kilian [FK94] use in their proof of Theorem 5, and the protocol described in Remark 8, we can prove:

Lemma 17. *For any constant ϵ, $0 < \epsilon < 1/2$. Every language L recognized by a $(k+3)$-APP system with parameters $([\exists, \forall, \ldots, \forall, \forall, \exists, \exists, \exists])$ and error α for any constant α, can be recognized by a $(k+1)$-APP system with parameters $([\exists, \forall, \ldots, \forall, \exists, \forall])$ and error ϵ.*

Proof: Omitted. □

We can now prove our main theorem:

Theorem 3. *For any constant ϵ, $0 < \epsilon < 1/2$. A language L is in Σ_k^P if and only if it has a $(k+1)$-APP system with error ϵ.*

Proof: If L has a $(k+1)$-APP system with error $\epsilon < 1/2$, then a Σ_k^P Turing machine can guess the strategy of the first k provers, compute for each one of the polynomially many questions that the last prover can receive, the optimal constant-size response, and then calculate the acceptance probability of the verifier. The other direction follows from Theorem 10, Lemmas 16 and 17. □

We observe that Theorem 3 is best possible in the sense that unless the polynomial-time hierarchy collapses, all Σ_k^P languages do not have $(k+1)$-APP systems with one-sided error:

Remark 18. *For any constant $\epsilon > 0$. If language L has a $(k+1)$-APP system with error $(0, \epsilon)$ (resp. $(\epsilon, 0)$) then L is in Σ_{k-1}^P.*

Proof: Omitted, but not difficult. □

We hope that the theorems presented in this paper can be used to prove non-approximability results for *non-artificially constructed* problems.

6. Acknowledgments

This paper owes its existence to Joan Feigenbaum; we give her a special thanks. We would like to thank Uri Feige and Joe Kilian for explaining their early results to us and for providing us with an early draft of their paper. We would like to thank Mario Szegedy for suggesting that we generalize Corollary 6 and for atypical contributions. We would also like to thank Michel Goemans, Shafi Goldwasser and Mike Sipser for helpful discussions.

References

[ALMSS92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. "Proof verification and intractability of approximation problems". *Proc. of the 33rd IEEE FOCS*, pages 14–23, 1992.

[AS92] S. Arora, and S. Safra. "Probabilistic checking of proofs". In *Proc. of the 33rd IEEE FOCS*, pages 2–13, 1992.

[BFL91] L. Babai, L. Fortnow, and C. Lund. "Nondeterministic exponential time has two-prover interactive protocols". *Computational Complexity*, 1:3–40, 1991.

[CFLS93a] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. "Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions". In *Proc. of the 25th ACM STOC*, pages 305–314, 1993.

[CFLS93b] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. "Random debaters and the hardness of approximating stochastic functions". DIMACS TR 93-79, Rutgers University, Piscataway NJ, 1993.

[CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. "Alternation". *Journal of the ACM*, 28:114–133, 1981.

[FST88] U. Feige, A. Shamir, and M. Tennenholtz. "The Noisy Oracle Problem". *Proc. Crypto 88*, pages 284-296, 1988".

[FK94] U. Feige and J. Kilian, "Two prover protocols - Low error at affordable rates". *Proc. of the 26th ACM STOC*, pages 284-296, 1988".

[FKS93] J. Feigenbaum, D. Koller, and P. Shor. Private communication.

[FL92] U. Feige, and L. Lovász. "Two-provers one-round proof systems: their power and their problems". *Proc. of the 24th ACM STOC*, pages 733–744, 1992.

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

[LS91] D. Lapidot, and A. Shamir. "Fully parallelized multi prover protocols for NEXP-time". In *Proc. of the 32nd IEEE FOCS*, pages 13–18, 1991.

[MS77] F. J. MacWilliams, and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North Holland, Amsterdam, 1977.

[PR79] G. Peterson, and J. Reif. "Multiple-person alternation". In *Proc. of the 20th IEEE FOCS*, pages 348-363, 1979.

[Reif84] J. H. Reif. "The complexity of two-player games of incomplete information". *J. Comput. System Science,* 29, pages 274-301, 1984.

[Sc86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, Chicester, 1986.

[St77] L.J. Stockmeyer. "The polynomial-time hierarchy". In *Theoretical Computer Science* **3**, pages 1–22, 1977.