The Complexity of Space Bounded Interactive Proof Systems

ANNE CONDON

Computer Science Department, University of Wisconsin-Madison

1 INTRODUCTION

Some of the most exciting developments in complexity theory in recent years concern the complexity of interactive proof systems, defined by Goldwasser, Micali and Rackoff (1985) and independently by Babai (1985). In this paper, we survey results on the complexity of space bounded interactive proof systems and their applications.

An early motivation for the study of interactive proof systems was to extend the notion of NP as the class of problems with efficient "proofs of membership". Informally, a prover can convince a verifier in polynomial time that a string is in an NP language, by presenting a witness of that fact to the verifier. Suppose that the power of the verifier is extended so that it can flip coins and can interact with the prover during the course of a proof. In this way, a verifier can gather statistical evidence that an input is in a language.

As we will see, the interactive proof system model precisely captures this interaction between a prover P and a verifier V. In the model, the computation of V is probabilistic, but is typically restricted in time or space. A language is accepted by the interactive proof system if, for all inputs in the language, V accepts with high probability, based on the communication with the "honest" prover P. However, on inputs not in the language, V rejects with high probability, even when communicating with a "dishonest" prover. In the general model, V can keep its coin flips secret from the prover. An important restriction is obtained by requiring that the verifier communicate all its coin flips to the prover as it flips them. Such interactive proof systems were first studied by

Babai (1985), who labeled them Arthur-Merlin games. They are also known as interactive proof systems with public coins, as opposed to the more general interactive proof systems with private coins.

There have been major breakthroughs in understanding the complexity of interactive proof systems. These breakthroughs have also had profound applications in diverse areas of computer science, including cryptography (zero-knowledge interactive proofs), program checking, formal language theory, group theory, stochastic processes, and in proving non-approximability results for NP-complete problems.

Our goal in this paper is to provide a survey of results for space bounded interactive proof systems - where space, rather than time, is the primary restricted resource of the verifier. We present bounds on the resulting complexity classes, and describe applications of these results to computational problems in areas such as formal language theory, stochastic processes and non-approximability of NP-complete problems. Unlike time bounded interactive proof systems, many fundamental problems on space bounded interactive proof systems still remain unsolved. Our goal is to describe these problems in a unified context, and to give the reader some insight into techniques that may be applicable to solving them.

Our decision to focus on space bounded interactive proof systems reflects our own biases and experience, and keeps our task within reasonable bounds. As a result, we do not describe in depth the remarkable results on time bounded interactive proof systems, which led to a complete characterization of both single and multi-prover, polynomial time bounded interactive proof systems (see for example Lund, Fortnow, Karloff and Nisan (1990), Shamir (1990) Babai, Fortnow and Lund (1990)). For completeness, we do compare these results with what is known about space bounded interactive proof systems. We also omit discussion of results on other related models, such as the games against nature of Papadimitriou (1985) (which can be thought of as interactive proof systems with unbounded error) and the private alternating Turing machines of Peterson and Reif (1979).

The interactive proof system model is defined in detail in Section 2. Results on space bounded interactive proof systems and selected proofs are described in the remaining sections. For convenience, all results are summarized in Figures 1 and 2 at the end of the paper. In the rest of this section, we highlight some of these results and their applications.

We begin by describing informally a very simple interactive proof system (P, V), due to Dwork and Stockmeyer (1992), in order to make more concrete the model of an interactive proof system when the verifier uses limited space. This interactive proof system accepts the language Pal $= \{x \in \{a, b\}^* \mid x = x^R\}$, where x^R denotes the string x written backwards.

On input x, the prover P repeatedly sends x to the verifier V. V performs the following computation each time it receives a string, say w, from a prover. First, V flips a coin. If the outcome is heads, V checks that the string w matches the input x, by scanning the input from left to right while receiving w. If the outcome of the coin flip is tails, V checks that w matches x^R, by scanning the input from right to left. If the check succeeds on all iterations, the verifier accepts the input.

Note that the verifier uses $O(1)$ space in this interactive proof system. We call such a verifier a 2pfa, since it is essentially a probabilistic finite state automaton with a 2-way input head. If the input $x \in$ Pal, then (P, V) accepts with probability 1, whereas if $x \notin$ Pal, then on each iteration the verifier finds a mismatch with probability at least $1/2$, no matter what string the prover sends. This is true because the verifier keeps its coin flips hidden from the prover. Thus, (P, V) accepts all strings in Pal with probability 1, whereas the probability that (P^*, V) accepts a string not in Pal is at most $1/2^k$, if there are k iterations of the above protocol.

This example does not illustrate the full power of $O(1)$ space bounded interactive proof systems. We will see that in fact, any language in DTIME$(2^{O(n)})$ has an interactive proof which is $O(1)$ space bounded. Furthermore, any language in DTIME$(2^{poly(n)})$ has an interactive proof system which is log space bounded. We denote the classes of languages accepted by interactive proof systems which are $O(1)$ and log space bounded by IP(2pfa) and IP(log-space), respectively. The best known upper and lower bounds on these classes are as follows.

$$\text{DTIME}(2^{poly(n)}) \subseteq \text{IP(log-space)} \subseteq \text{ATIME}(2^{2^{poly(n)}}) \text{ and}$$

$$\text{DTIME}(2^{O(n)}) \subseteq \text{IP(2pfa)} \subseteq \text{ATIME}(2^{2^{O(n)}}),$$

where DTIME and ATIME refer to deterministic and alternating time bounded classes, respectively, and $poly(n)$ denotes $n^{O(1)}$. Note that there is a large gap between the upper and lower bounds here.

The ability of the verifier to keep its coin flips hidden from the prover is crucial in both the Pal example and in the above bounds. If the interactive proof

system is an Arthur-Merlin game, its power is considerably weaker. We denote by AM(2pfa) and AM(log-space) the classes of languages accepted by public coin interactive proof systems with $O(1)$ and log space, respectively. Also, we denote by 2PFA the class of languages accepted by 2-way probabilistic finite state automata with bounded error. Then,

$$\text{2PFA} \subset \text{AM(2pfa)} \subset \text{AM(log-space)} = \text{P}.$$

An example of a language separating 2PFA from AM(2pfa) is Center, the set of strings over the alphabet $\{a, b\}$ which have a b in the center. An example of a language separating AM(2pfa) from P is Pal.

The expected time needed by a $O(1)$ space bounded interactive proof system to recognize a language in DTIME($2^{O(n)}$) may be double exponential in the size of the input. It is therefore useful to consider complexity classes where the time, as well as the space, used by the interactive proof system is limited. In the statement of the following results, the restriction poly-time is added to denote complexity classes where, in addition to a space bound, the number of steps taken by the verifier is bounded by a polynomial.

$$\text{IP(log-space, poly-time)} = \text{IP(poly-time)} = \text{PSPACE and}$$

$$\text{NC} \subseteq \text{AM(log-space,poly-time)} \subseteq \text{P} \subseteq \text{AM}(o(\log^2 n)\text{-space, poly-time}).$$

NC denotes the class ASPACE,TIME($\log n, \log^{O(1)} n$). In the case of $O(1)$ space bounded interactive proof systems, the class IP(2pfa,poly-time) contains an NP-complete language, and properly contains AM(2pfa). Again, Pal separates the two classes.

Yet another possible restriction on the verifier is that it uses few random bits. We denote the complexity classes of interactive proof systems in which the verifier uses log random bits, by adding the notation log-random-bits. Note that the complexity classes IP(log-space,log-random-bits) and AM(log-space,log-random-bits) are contained in IP(log-space,poly-time) and AM(log-space,poly-time), respectively. This is because if only $O(\log n)$ random bits are used, the verifier can flip them all at the start and behave deterministicially thereafter; and a $O(\log n)$ space bounded computation that halts must run in polynomial time. The following additional results indicate that the containments may be strict.

$$\text{IP(log-space,log-random-bits)} = \text{NP and}$$

$$\text{NLOG} \subseteq \text{AM(log-space,log-random-bits)} \subseteq \text{LOGCFL}.$$

Here, NLOG is nondeterministic log space and LOGCFL is the class of languages that are log-space reducible to context free language recognition (see Sudborough (1978)).

So far, we have described results on space bounded interactive proof systems with additional restrictions on the time or amount of randomness used by the verifier. We next consider more fundamental variations of the model. The first is the multiple-prover model, where the verifier can interact with two or more provers. The provers cannot communicate with each other during the proof. Intuitively, it is potentially more powerful than the single prover model because the verifier can ask overlapping sets of questions of each prover, and use the consistency of the provers on the common questions to verify that both are honest. We denote the class of languages accepted by interactive proof systems with two provers and a verifier which is a 1-way probabilistic finite state automaton, or pfa, by 2IP(pfa). This model is extremely powerful.

$$\text{2IP(pfa)} = \text{Recursive languages and}$$

$$\text{2IP(pfa, poly-time)} = \text{NTIME}(2^{poly(n)}).$$

The other variation of the model that we consider is zero knowledge interactive proof systems. Roughly, an interactive proof system is zero knowledge if on all accepted inputs, the verifier can learn nothing from the proof other than the fact that the input is in the language. For a reasonable formalization of this notion for space bounded interactive proof systems, the following results are known. We denote by ZKIP(2pfa) and ZKIP(log-space,poly-time) the classes of languages which have zero knowledge interactive proof systems when the verifier is a 2pfa or is simultaneously log space and polynomial time bounded, respectively.

$$\text{ZKIP(2pfa)} \subset \text{IP(2pfa)}.$$

In fact, Pal is an example of a language in IP(2pfa) but not in ZKIP(2pfa).

$$\text{IP(log-space,poly-time)} = \text{ZKIP(log-space,poly-time)}.$$

To conclude this section, we give some examples of applications of results on space bounded interactive proof systems to other computational problems. Most of these applications are discussed in more detail in future sections.

We first cite two examples of undecidability results that follow from results on $O(1)$ space bounded interactive proof systems. Condon and Lipton (1989)

applied a result on "weak" interactive proof systems, which are $O(1)$ space bounded, to show that the emptiness problem for 1-way probabilistic finite state automata with unbounded error probability is undecidable, a problem that had been open since the late 60's (see Theorem 3). Feige and Shamir (1989) applied results on space bounded multiple prover interactive proof systems to prove that a game-theoretic problem proposed by Reif (1984) is undecidable. Roughly the problem is to decide if, in a game from a certain class of "reasonable" 2-player games of incomplete information, a given player has a strategy which is expected to win.

The first application of interactive proof systems to non-approximability results for NP-complete problems arose from the study of space bounded interactive proof systems. Condon (1991) showed that a variation of the word problem for matrices, called the "max word problem for matrices" is NP-complete and furthermore, that the corresponding optimization problem cannot be approximated within any constant factor, unless P = NP. The result also has applications to the emptiness problem for 1-way probabilistic finite state automata with unbounded error.

A nice application to problems in automata theory is due to Dwork and Stockmeyer (1989). They showed that 2pfa's that are restricted to run in expected polynomial time and have bounded error accept exactly the regular languages. The techniques used to prove this were derived from their work on $O(1)$ space bounded verifiers.

A final application is in the area of bounding the rate of convergence of stochastic processes. In studying space bounded interactive proof systems, Condon and Lipton (1989) obtained tight bounds on the rate of convergence of certain classes of discrete time-varying Markov chains to their absorbing states. A time-varying Markov chain is a sequence of random variables over a finite state space, with the following property. For all positive integers i, a transition matrix P_i determines the value of the $(i+1)st$ random variable, given the value of the ith random variable. Let $\mathcal{M}$ be the family of n-state time-varying Markov chains such that the matrices P_k are all from some finite set of stochastic matrices, say $\{A, B\}$. We assume that all the entries of A and B are rational, of the form p/q where p and q are integers, $p \leq q \leq 2^n$. A special case of the results of Condon and Lipton on time-varying Markov chains can be stated simply as follows. Suppose that for all chains M in $\mathcal{M}$, n is a halting state which is eventually reached from the initial state with probability 1. Then the expected time to reach the halting state n is $2^{2^{\Theta(n)}}$. A well known result for

stationary Markov chains under similar conditions is that the expected time to reach a halting state n is $2^{\Theta(n)}$.

The rest of the paper is organized as follows. We first define precisely the model of an interactive proof system and related complexity classes. In Sections 3 and 4, we present results on log space bounded interactive proof systems with private and public coins, respectively. We consider $O(1)$ space bounded interactive proof systems in Section 5. We give an overview of some of the proofs of these results. We note that the results of Sections 3 and 4 can be extended to other space bounds $s(n) = \Omega(\log n)$ and the results of Section 5 can be extended to sublogarithmic space bounds. Finally, two variations on the model – multiple prover interactive proof systems and zero knowledge interactive proof systems are considered in Section 6, and known results are stated without proof. Some open problems are discussed in the concluding section.

2 DEFINITIONS

In this section, we describe the interactive proof system model. The definitions we use here are probably closest to those of Dwork and Stockmeyer (1992), although many alternative, equivalent definitions can be found in the literature (Babai and Moran (1988), Condon (1989), Goldwasser, Micali, Rackoff (1985)).

An interactive proof system consists of a prover P and a verifier V. The verifier is a probabilistic Turing machine with a 2-way, read-only input tape, a read-write work tape and a source of random bits. The states of the verifier are partitioned into reading and communication states. In addition, the Turing machine has a special communication cell that allows the verifier and prover to communicate.

A transition function describes the one-step transitions of the verifier. Whenever the verifier is in a reading state, the transition function of the verifier determines the next configuration of the verifier, based on the symbol under the tape heads, the state and possibly the outcome of an unbiased coin toss. Whenever the verifier is in a communication state, the next configuration is determined as follows. Associated with each communication state is a symbol; without loss of generality we assume that the set of such symbols is $\{0, 1\}$. When in communication state c, the verifier writes the symbol associated with c in the communication cell and in response, the prover writes a symbol in the cell. Based on the state and the symbol written by the prover, the verifier's transition function defines the next state of the verifier.

The prover P is specified by a prover transition function. This function determines what communication symbol is written by the prover in response to a symbol of the verifier, based on the input and the sequence of all past communication symbols written by the verifier. Without loss of generality we assume that all symbols written by the prover in the communication cell are from the set $\{a, b\}$ and that the input alphabet is Σ. Thus the prover's transition function is a mapping from $\Sigma^* \times \{0,1\}^*$ to $\{a, b\}$.

The probability that (P, V) accepts (rejects) x is the limit as $k \to \infty$ of the probability, (taken over all coin tosses of the verifier), that (P, V) reaches the accepting (rejecting) state on x in k steps. The probability that (P, V) halts is defined to be the probability that (P, V) accepts or rejects. The prover-verifier pair (P, V) is an interactive proof system for L with error probability $\epsilon < 1/2$ if

1. for all $x \in L$, the probability that (P, V) accepts x is $> 1 - \epsilon$,

2. for all $x \notin L$, and all provers P^*, the probability that (P^*, V) rejects x is $> 1 - \epsilon$.

In the paper, we assume that $\epsilon = 1/4$, unless otherwise specified. In most of the results of this paper (except for those in which the number of random bits is limited), the constant ϵ can be replaced by any function of the form $1/2^{O(n)}$.

A different, weaker, definition of language acceptance for space bounded interactive proof systems obtained by replacing condition 2 above by the following.

2′. for all $x \notin L$ and all provers P^*, the probability that (P^*, V) accepts x is $\leq \epsilon$.

For most complexity classes, the definitions are equivalent. A notable exception is the class of languages accepted by interactive proof systems which are $O(1)$ space bounded. In this case, we use the notation weak-IP(2pfa) to refer to the class of languages accepted by interactive proof systems with respect to the weaker definition, that is, with condition 2′ instead of condition 2. Condon and Lipton (1989) showed that the class weak-IP(2pfa) contains all the recursively enumerable languages. We discuss this class further in Section 4.

In many of the interactive proof systems that we describe, the roles of the prover and verifier are typically to send strings to each other, and informally

we say that "the verifier sends a string w to the prover", or "the verifier receives a string w from the prover". This can be made precise in our model of a single communication cell, as follows. Suppose the verifier wishes to send a string $w = w_1 w_2 \ldots w_k$ to the prover, where for all i, $w_i \in \{0, 1\}$. To do this, the verifier can write $w_1 1 w_2 \ldots 1 w_k 0$ in the communication cell. The prover can recognize the end of the string by the appearance of a 0 at an even numbered position. In a similar fashion, the prover can send a string over $\{a, b\}^*$ to the verifier.

Just as for Turing machines, a *configuration* of the verifier of an interactive proof system for a fixed input is a tuple containing an encoding of the work tapes, the positions of the tape heads on the input and work tapes of the verifier, and the state and the contents of the communication cell. A configuration that contains a communication state is called a communication configuration, and one which contains a reading state is called a reading configuration.

Two well-studied special cases of the general definition of interactive proof systems represent two extremes; one in which the verifier sends the prover complete information about its current configuration and the other in which the verifier sends the prover no information. In an *Arthur-Merlin game*, whenever the verifier flips a coin, the outcome is written in the communication cell. From this, a prover has complete information about the computation of the verifier. In this case, we can make certain simplifying assumptions about the prover P, namely that the response of P depends only on the input x and current configuration of V, and not on the complete sequence of symbols written by V in the communication cell (Condon, 1989 shows why such an assumption can be made without loss of generality). We say in this case that the prover P uses a *Markov strategy*. At the other extreme is a *oneway* interactive proof system, where the verifier sends no information to the prover. In this model, the verifier simply writes the same symbol in the communication cell whenever it needs another symbol from the prover. In this case, the prover can be simply represented as an infinite string over the alphabet $\{a, b\}$, where the ith symbol of the string is the symbol written by the prover in the communication cell the ith time that the verifier enters a communication state.

The notion of a single-prover interactive proof system was generalized to two and more provers by Ben-Or, Goldwasser, Kilian and Wigderson (1988). In a 2-prover interactive proof system, two provers interact with the verifier, but the provers cannot communicate with each other during the proof. We discuss this model in Section 6.

We next describe the resource bounds considered in the paper. An interactive proof system (P, V) is $s(n)$ *space bounded* if on any input of length n, for all P^*, the number of work tape cells read by V is at most $s(n)$, during the computation of (P^*, V). If the number of work tape cells used by the verifier is $O(1)$, the verifier is a probabilistic 2-way finite state automaton, or 2pfa. We say that (P, V) is $t(n)$ *time bounded* if on any input of length n, for all P^*, the *expected* number of steps taken by the verifier during the computation of (P^*, V) is at most $t(n)$. We consider expected time, rather than worst case time, because we will consider interactive proof systems which are $O(1)$ space bounded, and expected time is a more natural definition in this case. A third bound we consider is a limit on the number of random bits used by the verifier. We say that (P, V) uses $r(n)$ *random bits* if on any input of length n, for all provers P^*, the number of random bits used by V during the computation of (P^*, V) is at most $r(n)$.

2.1 Notation

As we stated earlier, we use IP and AM to refer to interactive proof systems with private or public coins, respectively. We use oneway-IP to denote oneway interactive proof systems. Thus, IP(<restrictions>) is the class of languages which have interactive proof systems with restrictions denoted by <restrictions>. The most common restrictions we consider are: (i) log-space, (ii) 2pfa, (iii) poly-time and (iv) log-random-bits, which mean that the interactive proof system (i) is $O(\log n)$ space bounded (ii) is $O(1)$ space bounded, (iii) is polynomial time bounded and (iv) uses $O(\log n)$ random bits, respectively. Thus, IP(log-space,poly-time) is the class of languages accepted by interactive proof systems with private coins, which are simultaneously $O(\log n)$ space bounded and polynomial time bounded. Also, oneway-IP(log-space, log-random-bits) is the class of languages accepted by oneway interactive proof systems which are simultaneously $O(\log n)$ space bounded, and use $O(\log n)$ random bits. In Section 6, we also consider interactive proof systems which are $O(1)$ space bounded, and in addition, the verifier can read its input only in one direction. In this case, we say that the verifier is a pfa. Then, AM(pfa) is the class of languages accepted by Arthur-Merlin games, or public coin interactive proof systems, where the verifier is a pfa.

2.2 Other Complexity Classes and Related Work

We refer to the alternating Turing machines of Chandra, Kozen and Stockmeyer (1981) widely in the paper, and so we give a brief description here. An alternating Turing machine is a generalization of a nondeterministic Turing machine, with both existential and universal states. The roles of these states with

respect to language acceptance is specified using a computation tree as follows. The nodes of a computation tree of an alternating Turing machine on an input w are labeled by configurations. The root is labeled by the initial configuration of the machine on x, and leaves are labeled by halting configurations. Each internal node labeled by a universal configuration has one child labeled by each possible configuration that is reachable in one step. Each internal node labeled by an existential configuration has exactly one child, labeled by some configuration reachable in one step. The tree is accepting if the tree is finite and all leaves are labeled with accepting configurations. The input w is accepted by the machine if there is an accepting computation tree corresponding to w.

An alternating Turing machine is $t(n)$ time bounded, or $s(n)$ space bounded, if on all accepted inputs of length n, there is an accepting computation tree of height $\leq t(n)$, or whose nodes are labeled by configurations of length $\leq s(n)$ respectively. We assume that the input can be accessed by writing an address on a special index tape, so that sublinear time bounds give rise to meaningful complexity classes.

We let ATIME($t(n)$) and ASPACE($s(n)$) denote the class of problems which are accepted by $O(t(n))$ time bounded and $O(s(n))$ space bounded alternating Turing machines, respectively. Also, the class of languages accepted by alternating Turing machines which are simultaneously $O(s(n))$ space bounded and $O(t(n))$ time bounded is denoted by ASPACE,TIME($s(n), t(n)$). Chandra, Kozen and Stockmeyer (1981) showed that ATIME(poly(n)) = PSPACE and ASPACE($\log n$) = P. We will also consider the complexity class NC (Cook, 1979) of problems which have $\log^{O(1)} n$ time algorithms on a PRAM (parallel random access machine) with a polynomial number of processors. Ruzzo (1981) showed that NC = ASPACE,TIME($\log n$, $\log^{O(1)} n$).

For completeness, we next discuss results on polynomial time bounded, single prover interactive proof systems. Although a detailed treatment of these results are beyond the scope of this paper, we will compare them with the results on space bounded interactive proof systems presented in this paper (see Figure 1). We denote by IP(poly-time) and AM(poly-time) the classes of languages accepted by polynomial time bounded interactive proof systems with private and public coins, respectively.

It is not too hard to see from the definitions that NP $\subseteq$ AM(poly-time) $\subseteq$ IP(poly-time) $\subseteq$ PSPACE (see Condon, 1989). Goldwasser and Sipser (1989) showed that IP(poly-time) = AM(poly-time). Lund, Fortnow, Karloff and

Nisan (1990) found an interactive proof system for the permanent function, which is hard for the class $\#P$ of Valiant (1979) and thus hard for the polynomial time hierarchy, PH, by a result of Toda (1991). Thus they showed that any language in PH is in IP(poly-time). Their proof uses a result of Lipton (1991) that the permanent of square matrices over a finite field is random self-reducible. Lipton's proof is based on the construction of Beaver and Feigenbaum (1990) of "instance hiding schemes" for arbitrary Boolean functions. Building on this work, Shamir (1990) showed that all languages in PSPACE have interactive proof systems. Similar techniques have been used to obtain results on the power of multiple prover interactive proof systems, which we discuss in Section 6, most notably the result of Babai, Fortnow and Lund (1991) that any language in nondeterministic exponential time is accepted by a polynomial time bounded, two-prover, interactive proof system.

There has been much other work on polynomial time bounded interactive proof systems with various restrictions on the prover and verifier, other than space. We mention two examples here, which are closely related to results in this paper. Condon and Ladner (1992) introduce a variation of the model of interactive proof system, in which the power of the prover is restricted and study resulting complexity classes, including a model in which the verifier is log space bounded. Arora and Safra (1992) and Arora, Lund, Motwani, Sudan and Szegedy (1992) considered a model in which the verifier can only use a limited number of symbols received from the prover in its computation. Their techniques show that any language in NP can be accepted by a polynomial time bounded interactive proof system, which uses $O(\log n)$ random bits and in which the verifier uses only $O(1)$ (randomly chosen) symbols received from the prover in its computation. This result was used to show that, unless P=NP, the problem of approximating the size of the largest clique in a graph is NP-complete. Earlier results on the hardness of approximating the clique number were proved by Feige, Goldwasser, Lovasz, Safra and Szegedy (1991). The proofs of Arora et al. also build on previous work of Babai, Fortnow, Levin and Szegedy (1991) on checking computations and of Babai, Fortnow and Lund (1991) on multiple prover interactive proof systems.

3 LOG SPACE; PRIVATE COINS

In this section, we consider the power of interactive proof systems with no time bounds, in which the verifier is $O(\log n)$ space bounded. The main results of this section show that

$$\text{DTIME}(2^{poly(n)}) \subseteq \text{IP(log-space)} \subseteq \text{ATIME}(2^{2^{poly(n)}}).$$

The lower bound on IP(log-space) was proved by Condon (1991b) and independently by Dwork and Stockmeyer (1989). The proof is based on the following idea: a computation of a polynomial space bounded alternating machine is repeatedly sent by the prover to the verifier and the verifier probabilistically checks for errors in the computation. If none are found, and all computations end in an accepting state, the verifier accepts the input. It is interesting to note in the proof that the interactive proof system requires double exponential expected time on accepted inputs, so it appears that even with $O(\log n)$ space bounded verifiers, interactive proof systems can run "usefully" for double exponential time.

The upper bound was proved by Condon and Lipton (1989) and requires a proof that interactive proof systems must halt with high probability in double exponential time. The proof of this exploits a close relationship between space bounded interactive proof systems and families of time varying Markov chains, and in fact has applications in the theory of time-varying Markov chains.

The lower and upper bounds on IP(log-space) are presented in Theorems 1 and 2. Following this, we briefly discuss the complexity of log space bounded interactive proof systems, when there are additional resource bounds on the verifier, such as time and randomness. Applications of these results to the non-approximability of NP-complete problems are discussed at the end of this section.

Theorem 1 *DTIME($2^{poly(n)}$) $\subseteq$ IP(log-space).*

Proof: To build up to the proof, we first describe why NP is contained in IP(log-space). Suppose that L is accepted by a nondeterministic Turing machine M in polynomial time. On input x of length n, the prover sends a computation, or sequence of configurations of M on x, to the verifier. The verifier checks that the computation ends in an accepting computation. The verifier also checks that the computation is *valid*, namely that it starts in the initial configuration and that the $(i+1)$st configuration follows from the ith configuration according to the rules of M. However, this check cannot be done deterministically in $O(\log n)$ space. Instead, the verifier chooses one symbol, say the jth symbol, from each configuration, uniformly at random. (Without loss of generality, we can assume that the length of a configuration is a power of 2, to make this possible.) Then V checks that symbol j is correct. Using standard encodings of configurations, this can be done using $O(\log n)$ space, by storing only the index j and a constant number of symbols from the ith

configuration. If the prover sends an invalid computation, the verifier detects that it is invalid with probability at least $1/poly(n)$. To reduce the error, the verifier repeats the above protocol polynomially many times and accepts if and only if no computation is found to be invalid. Note that the above argument actually shows that NP $\subseteq$ oneway-IP(log-space,poly-time), since the verifier never sends information to the prover.

This idea can be extended to show that DTIME($2^{2^{poly(n)}}$) is contained in IP(log-space). The following argument is essentially due to Dwork and Stockmeyer (1992). Recall that ASPACE(poly(n)) = DTIME($2^{poly(n)}$); thus it is sufficient to show that ASPACE(poly(n)) is contained in IP(log-space). In this case, the verifier must determine that on input x, there is an accepting subtree of some alternating Turing machine M on input x. We assume that every leaf of the tree has depth exponential in the input length, and thus the tree has $2^{2^{poly(n)}}$leaves. The prover sends the verifier computations, corresponding to paths in the computation tree, and the verifier tests just as above that each computation is valid and that the final configuration is accepting. The verifier sends the prover random coins, to direct the choice of the path taken by the prover at universal nodes of the tree. The verifier accepts if and only if all computations are valid and end in an accepting configuration. $2^{2^{p(n)}}$computations must be sent by the prover to the verifier, for some polynomial $p(n)$, in order to ensure that with high probability, all paths of the computation subtree are checked.

However, V cannot count to $2^{2^{p(n)}}$in log space. Thus, we extend the description of V, to ensures that the expected number of computations V receives from P is $2^{2^{p(n)}}$, for some polynomial $p(n)$. To do this, V performs a "halt test" each time it receives a computation from P. V flips a polynomial number of coins for each configuration received from P, and halts in an accepting state at the end of the computation if all coin flips are heads. Since there are exponentially many configurations in a computation, this ensures that the probability that V halts at the end of a given configuration is $1/2^{2^{poly(n)}}$.

This completes our informal description of (P, V). To prove that (P, V) accepts L, it must be shown that given an input $x \notin L$, (P^*, V), rejects x with high probability, where P^* is any prover. To see why this is true, note that if the prover P^* sends V an infinite computation, there must be infinitely many i such that the $(i+1)$st configuration does not follow from the ith configuration. In this case, V will reject with probability 1. Also, if P^* sends V an invalid computation, V is more likely to find an error than to halt as a result of the

halt test. □

Theorem 2 *(i) IP(log-space) $\subseteq$ ATIME($2^{2^{poly(n)}}$) and*

(ii) oneway-IP(log-space) $\subseteq$ NTIME($2^{2^{poly(n)}}$).

Proof: We describe only the proof that oneway-IP(log-space) is contained in NTIME($2^{2^{poly(n)}}$). The proof that IP(log-space) $\subseteq$ ATIME($2^{2^{poly(n)}}$) is a generalization using similar techniques. Suppose that L is a language in the class oneway-IP(log-space), where L is accepted by (P, V). To further simplify the presentation here, we assume that on inputs not in L, (P^*, V) halts with probability 1 for all provers P^*. (We do not know if this assumption can be made without loss of generality, but the proof is similar in the case that (P^*, V) halts with probability $> 1/2$.)

The key to the proof is to show that if an input x of length n is not in L, then for all P^*, (P^*, V) reaches a rejecting state with high probability in time $2^{2^{poly(n)}}$. To prove this, we need the following notation. Let m be the number of communication configurations of V on x, that is, the number of configurations in which the verifier is in a communication state. Number them $1, \ldots, m$, and without loss of generality assume that 1 is the initial configuration and that m is a unique rejecting configuration. For communication configurations i and j, let $p(i, j, a)$ be the probability that from configuration i, if the prover's response is a, the next communication configuration reached (eventually) is j. Define $p(i, j, b)$ similarly, replacing a by b. Note that these probabilities are completely determined by x, i, j, a, b and the transition function of V, and hence can be computed in polynomial time. In fact, these probabilities are rational numbers of the form p/q where $p \leq q \leq 2^{m+1}$. The proof of this is very similar to a proof of Gill (1977) on the transition probabilities of $\log n$ space bounded probabilistic Turing machines. Finally let A and B be the $m \times m$ matrices whose ijth entries are $p(i, j, a)$ and $p(i, j, b)$, respectively.

Since the interactive proof system is oneway, each prover P^* corresponds to an infinite string $\sigma_1\sigma_2 \ldots, \sigma_i \ldots$ where each $\sigma_i \in \{a, b\}$. Suppose that at some time t, the computation of (P^*, V) has reached communication configuration I. We claim that with probability > 0, a halting (accepting or rejecting) configuration is reached within the next 2^m communication configurations. Suppose not. For $0 \leq l \leq 2^m$, let S_l be the set of communication configurations reachable from I after receiving l further symbols from the prover. By the pigeon-hole principle, since each $S_i \subseteq \{1, \ldots, m\}$, $S_j = S_k$ for some $0 \leq j < k \leq 2^m$. Then, if

P^{**} is the prover corresponding to the sequence $\sigma_1 \ldots \sigma_{t+j}(\sigma_{t+j+1} \ldots \sigma_{t+k})^*$, the computation of (P^{**}, V) does not halt with probability 1, contradicting our assumption that on all provers, the computation halts with probability 1.

Thus, with probability > 0, a halting configuration is reached within $2^{poly(n)}$ steps, if it has not been reached already. In fact, since the probabilities $p(i,j,a)$ and $p(i,j,b)$ are bounded below by $1/2^{poly(n)}$, the probability of halting is at least $1/2^{2^{poly(n)}}$. It is straightforward to conclude from this that with probability $> 1/2$, a halting configuration is reached in $t(n) = 2^{2^{poly(n)}}$steps. Since $x \notin L$, then with probability $> 1/4$, the rejecting configuration has been reached.

We can now describe a simple nondeterministic algorithm for L, that runs in $2^{2^{poly(n)}}$time. On an input of length n, nondeterministically guess a string $\sigma_1 \ldots \sigma_{t(n)}$. This string represents the first $t(n)$ symbols of the strategy of a prover of the interactive proof system. Next, compute the product $M_1 \ldots M_{t(n)}$, where M_i is A or B if σ_i is a or b, respectively. Let p be the $(1,m)$th entry of this product. Recall that the mth configuration is the rejecting configuration. Hence, p is the probability that (P^*, V) rejects after V has received $t(n)$ symbols from P^*, where P^* is the prover corresponding to the string $\sigma_1 \ldots \sigma_{t(n)}$. If $p > 1/4$, reject, else accept. $\square$

The previous theorem should be contrasted with the following result of Lipton, which shows that with respect to the weak definition, interactive proof systems are extremely powerful.

Theorem 3 *Any recursively enumerable language L is in weak-IP(2pfa).*

The proof of this theorem generalizes a result of Frievalds (1981), who showed that the emptiness problem for 2-way probabilistic finite state machines is undecidable. As an application of this result, Lipton showed also that the emptiness problem for 1-way probabilistic finite state machines is undecidable.

We now turn to complexity classes where the time as well as the space is limited. If a log space bounded interactive proof system has the additional restriction that the time is polynomially bounded, the following results are known.

Theorem 4 *(i) PSPACE = IP(log-space,poly-time) and*

(ii) NP = oneway-IP(log-space, poly-time).

Proof: The proof of (i) follows from the fact that PSPACE = IP(poly-time) (1991) and that IP(poly-time) = IP(log-space,poly-time). The latter can be proved using essentially the same techniques that were developed in Theorem 1; the proof can be found in Condon (1991b) and was independently proved by Rompel.

We give a brief description of the proof of (ii). Note that we actually showed in Theorem 1 that NP $\subseteq$ oneway-IP(log-space, poly-time). To prove the other direction of (ii), we reduce the problem of deciding if a string is in L, where L is accepted by a oneway interactive proof system (P, V), which is log space bounded, to the following problem, which is easily seen to be in NP.

The *max word problem for matrices* is: given a tuple (S, v, w, k, c), where S is a set of $m \times m$ matrices, v and w are m-vectors, k is an integer and c is a constant, is there a way to select a sequence of k matrices $M_1, \ldots, M_k$ (not necessarily distinct) from S in such a way that the product $vM_1 \ldots M_k w^T$ is greater than c? All entries of the matrices and the vectors, as well as the bound c, are rational numbers expressed in binary and k is an integer, expressed in unary notation.

Given x, we reduce the problem of deciding if $x \in L$ to the max word problem for matrices as follows. In the reduction, k is polynomially bounded in $|x|$ and S consists of two matrices A and B are the matrices defined in Theorem 2. Thus, the entries of A and B are the transition probabilities of the V between communication configurations, with respect to the two possible symbols a and b that V can receive from the prover. (Recall that these matrices can be constructed in polynomial time). The constant $c = 1/2$, and the vectors v and w are such that all entries are 0, except that entry 1 of v is 1, where again 1 is the number of the initial configuration, and entry $m - 1$ of w is 1, where $m - 1$ is the number of the unique accepting configuration. Then, the value $vM_1 \ldots M_k w^T$ is the probability that V accepts when the prover sends the string $\sigma_1 \ldots \sigma_k$, where $\sigma_i = a$ if $M_i = A$ and $\sigma_i = b$ if $M_i = B$. $\square$

As a consequence of this proof that the max word problem for matrices is NP-complete, we can conclude that in fact, the problem cannot be approximated by any constant factor, unless NP=P. To our knowledge, this is the first example of the use of interactive proof systems to prove a non-approximability result for an NP-complete problem. Also, this result on the complexity of the max word problem has applications in the theory of probabilistic finite state automata, rational series and k-regular sequences. We describe one of these applications

briefly. We consider probabilistic finite state automata (*pfa*'s) with rational transition probabilities, as defined in Paz (1971). Suppose we define the *k-emptiness problem* for pfa's as follows. Given a pfa and a number k, expressed in unary notation, does the pfa reject every string of length $\leq k$? By a simple reduction from the max word problem, we prove that the k-emptiness problem for pfa's is complete for co-NP. Moreover, unless NP=P, it is not possible to approximate the maximum probability that a string of length n is accepted by a pfa.

In the last result of this section, we consider the complexity classes resulting when the verifier is restricted to log space and log random bits.

Theorem 5 *NP = oneway-IP(log-space, log-random-bits) = IP(log-space, log-random-bits).*

The proof that NP $\subseteq$ oneway-IP(log-space, log-random-bits) can be found in Condon and Ladner (1988). The main technique of the proof is due to Lipton (1991). Briefly, it is possible to efficiently reduce the problem of testing if a Boolean formula is satisfiable to that of testing if two multi-sets are equal. Lipton described a test for equality of two multisets using very few random bits and limited space, by comparing a short *fingerprint* of each multiset. The proof that IP(log-space,log-random-bits) $\subseteq$ NP is fairly straightforward.

4 LOG SPACE; PUBLIC COINS

The results of this section show that if space is restricted, interactive proof systems with public coins are significantly less powerful than those with private coins.

The first main theorem of this section, Theorem 6, due to Condon (1989), shows that in fact, AM(log-space) = P. One direction of the proof is based on the fact that when coins are public, a space bounded interactive proof system can be modeled as a Markov decision process, and the probability of reaching an absorbing state of such a process can be computed using linear programming.

We then consider the class AM(log-space,poly-time). From Theorem 6, it immediately follows that NLOG $\subseteq$ AM(log-space,poly-time) $\subseteq$ P. By cleverly adapting the techniques of Lund et al. (1990) and Shamir (1990) to space bounded interactive proof systems, Fortnow and Lund (1991) improved the lower bound of NLOG to show that NC $\subseteq$ AM(log-space,poly-time). Even more, they showed that P $\subseteq$ AM($\frac{\log^2 n}{\log \log n}$-space, poly-time). We present their

techniques in Theorem 7. Thus, "slightly more" than log space is sufficient for a public coin, polynomial time bounded interactive proof system to recognize all languages in P. It is an intriguing open question whether in fact AM(log-space,poly-time) = P.

Theorem 6 *AM(log-space) = P.*

Proof: We show that AM(log-space) $\subseteq$ P. Let $L \in$ AM(log-space), and suppose that L is accepted by (P, V). We can assume without loss of generality that for all provers P^*, (P^*, V) halts with probability 1 on all inputs. (Roughly, this is because the verifier V can be modified to flip $p(n)$ coins after each step for some polynomial p and reject if all are heads. In this way, with exponentially small probability at each original step, the computation halts, and so eventually halts with probability 1. If the polynomial p is sufficiently large, the probability of halting because of this coin-flipping test is so small that it does not significantly affect the error probability).

Fix an input x, and number the configurations of V on x so that the initial configuration is numbered 1. For each i, define $p(i)$ to be the maximum probability of reaching an accepting configuration from configuration numbered i, maximized over all provers which use a Markov strategy. (Recall from Section 2 that it is sufficient to consider Markov strategies in a public coins interactive proof system). To determine whether x is accepted by (P, V) for some P, it is sufficient to determine whether $p(1) > 1/2$. If $p(1) < 1/2$, then on all Markov provers P^*, the probability that (P^*, V) accepts x is $< 1/2$, and so x is not in L. If $p(1) > 1/2$, then x must be in L.

Thus, to prove that AM(log-space) is in P, it is sufficient to show that the values $p(i)$ for all i can be computed in polynomial time. The proof relies on the fact that the probabilities $p(i)$ satisfy the following equations. If i is an accepting or rejecting configuration, then $p(i)$ is 1 or 0, respectively. Otherwise, suppose that j and k (j may equal k) are the configurations reachable from i in one step of the verifier. Then if i is a reading configuration, $p(i) = 1/2(p(j) + p(k))$ and if i is a communication configuration, $p(i) = \max\{p(j), p(k)\}$. Intuitively, this is because, at a reading configuration, where a random move is made, the probability of eventually accepting is the average of the probabilities of accepting from the configurations reachable in one step, whereas at a communication configuration, the probability of eventually accepting is the maximum of the probabilities of accepting, taken over the two possible responses of the prover.

A precise justification can be found in Condon (1989), and is based on results of Howard (1960) on Markov decision processes.

Derman (1972) showed that the values $p(i)$ which satisfy the above equations are the unique solution to the following linear programming problem. Let m be the number of configurations of v on x. Minimize $\sum_{l=1}^{m} p(l)$, subject to the constraints

$p(i) \geq p(j)$,	if i is a communication configuration from which j and k are reachable in one step
$p(i) \geq 1/2(p(j) + p(k))$,	if i is a reading configuration from which j and k are reachable in one step
$p(i) = 0$,	if i is a rejecting configuration
$p(i) = 1$,	if i is an accepting configuration
$p(i) \geq 0$,	$1 \leq i \leq m$.

Since the linear programming problem is in P (Khachiyan (1979)), the values $p(i)$ can be computed in polynomial time, completing the proof.

The other direction, that P $\subseteq$ AM(log-space), is proved by simulating an alternating machine which is log space bounded by an Arthur-Merlin game which is log space bounded. The Arthur-Merlin game has exponential expected running time. $\square$

4.1 Log Space and Polynomial Time

From Theorem 6, it follows that AM(log-space,poly-time) $\subseteq$ P. This was also shown by Fortnow (1989), who also showed that NLOG $\subseteq$ AM(log-space,poly-time). However, it is open whether AM(log-space, poly-time) = P. In Theorem 8, we describe the techniques of Fortnow and Lund (1991), which shed light on this question. They show how an alternating machine can be simulated by an Arthur Merlin game, with only a modest increase in the time and space used.

$$\text{ASPACE,TIME}(s(n),\, t(n)) \subseteq$$

$$\cap_{\epsilon>0}\ \text{AM}(\tfrac{s(n)\log t(n)}{\log s(n)}\text{-space},\ (s^2(n)t(n) + n\log n)s^{\epsilon}(n)\log^2 t(n)\text{-time}).$$

The proof illustrates how the techniques of "arithmetizing" Boolean formulas can be applied to gain insight to space bounded interactive proof systems. From this and well known relationships between alternating machine classes and P and NC, the following results are obtained.

Theorem 7 *(i) NC $\subseteq$ AM(log-space, poly-time), and*

(ii) $P \subseteq AM(\frac{\log^2 n}{\log \log n}$-space, poly-time).

We will describe the proof of a slightly different result, which is simpler to present than the above result of Fortnow and Lund, but which illustrates all of the important techniques.

Theorem 8

$$ASPACE,TIME(s(n),t(n)) \subseteq$$

$$AM(s(n)\log t(n)\text{-space},(s^2(n)t(n)+n\log n)\log^2 t(n)\text{-time}).$$

Proof: Lund and Fortnow show that if $L \in ASPACE,TIME(s(n),t(n))$, then L is accepted by an alternating Turing machine M that uses $O(s(n))$ space, $O(t(n))$ time and has the following additional properties. M uses one tape, and alternates between existential and universal states at each step, with two possible transitions at each step.

Let $\phi_i(I,x)$ be a Boolean predicate which is true if and only if on input x, there is an accepting subtree of M on x which is rooted at I and has depth $2i$. Input x is accepted by M if and only if $\phi_N(I_0,x)$ is true, where I_0 is the initial configuration and $2N$ is the running time of M on x. Because of the properties of M, $\phi_i(I,x)$ has a nice inductive definition.

$$\phi_i(I,x) = \begin{cases} g(I), & \text{if } i=0, \\ \exists z_1 \in \{0,1\} \forall z_2 \in \{0,1\} \exists I' \in \{0,1\}^{k-2} : & \\ \qquad f(I,z_1,z_2,I') \wedge \phi_{i-1}(I',x), & \text{otherwise.} \end{cases}$$

Here, $k-2$ is the length of a binary encoding of a configuration of M on x, and $f(I,z_1,z_2,I')$, is a predicate over $\{\wedge,\vee,^-\}$ which is true if and only if M on input x moves from existential configuration I to existential configuration I', when the existential and universal choices in the next two moves are z_1 and z_2, respectively. Similarly, $g(I)$ is a predicate over $\{\wedge,\vee,^-\}$ which is true if and only if I is an accepting configuration.

The next step is to define an arithmetic formula $A_i(I,x)$ such that for all i, $\phi_i(I,x)$ is true if and only if $A_i(I,x)=1$ and $\phi_i(I,x)$ is false if and only if $A_i(I,x)=0$. $A_i(I,x)$ has the form

$$A_i(I,x) = \begin{cases} G(I), & \text{if } i=0, \\ \coprod_{z_1\in\{0,1\}} \prod_{z_2\in\{0,1\}} \sum_{I'\in\{0,1\}^{k-2}} & \\ \qquad F(I,z_1,z_2,I') \cdot A_{i-1}(I',x), & \text{otherwise.} \end{cases}$$

Here, $\coprod_{a\in\{0,1\}} \sigma(a) = \sigma(0) + \sigma(1) - \sigma(0)\sigma(1)$.

The functions F and G are arithmetic formulas obtained from f and g with the following properties. $F(I, z_1, z_2, I')$ is either 0 or 1 and is 1 if and only if $f(I, z_1, z_2, I')$ is true. Similarly, $G(I)$ is either 0 or 1 and is 1 if and only if $g(I)$ is true. F and G are obtained from the Boolean functions f and g by the following inductive rules. If a is a variable, then the corresponding function A is a. If A and B are the arithmetic functions obtained from Boolean expressions a and b, then AB, $1-(1-A)(1-B)$ and $1-A$ are the arithmetic functions obtained from $a\wedge b$, $a\vee b$ and $\bar{a}$, respectively. If f and g are suitably expressed as Boolean functions (details omitted), then F has size $O(s^2(n))$, depth $O(\log s(n))$ and constant degree, and G has size $O(n \log n)$, depth $O(\log n)$ and has constant degree.

The verifier checks that $A_N(I_0, x) = 1$ inductively, by reducing the problem of verifying that $A_i(I, x) = \beta$ to the problem of verifying that $A_{i-1}(I', x) = \beta'$. When $i = 0$, $A_i(I, x)$ is evaluated by computing $G(I)$. All calculations are done over the field $\mathbf{Z}/p\mathbf{Z}$, for some prime p.

To describe this reduction, we introduce the following notation. Suppose that

$$A_i(I, x) = Q^{(1)}_{z_1\in\{0,1\}} Q^{(2)}_{z_2\in\{0,1\}} \cdots Q^{(k)}_{z_k\in\{0,1\}} F(I, z_1, \ldots, z_k) A_{i-1}(z_3, \ldots, z_k, x).$$

Here, the vector $(z_3, \ldots, z_k)$ corresponds to I' and each $Q^{(j)}$ is from the set $\{\prod, \coprod, \sum\}$. Given numbers $r_1, \ldots, r_k \in \mathbf{Z}/p\mathbf{Z}$, for $1 \le j \le k$ let $P_j(z_j)$ be the function

$$Q^{(j+1)}_{z_{j+1}\in\{0,1\}} \cdots Q^{(k)}_{z_k\in\{0,1\}} F(I, r_1, \ldots, r_{j-1}, z_j, \ldots, z_k) \\ A_{i-1}(r_3, \ldots, r_{j-1}, z_j, \ldots, z_k, x).$$

Note that $P_j(z_j)$ is a univariate polynomial in z_j which has constant degree. Then the following protocol of the verifier V reduces the problem of verifying that $A_i(I, x) = \beta$ to the problem of verifying that $A_{i-1}(I', x) = \beta'$.

1. Choose a sequence $r_1, \ldots, r_k$ of values in $\mathbf{Z}/p\mathbf{Z}$ independently and uniformly at random. Let $j = 1$.

2. Receive from the prover a polynomial $f_j(z_j) \in \mathbf{Z}/p\mathbf{Z}[z_j]$. (The prover P is defined so that $f_j(z_j) = P_j(z_j) \bmod p$.)

3. If $j > 1$, check that $Q^{(j)}_{z_j\in\{0,1\}} f_j(z_j) = f_{j-1}(r_{j-1}) \bmod p$ and if $j = 1$, that $Q^{(1)}_{z_1\in\{0,1\}} f_1(z_1) = \beta \bmod p$.

4. If the check fails, halt and reject. Otherwise if $j < k$, send r_j to the prover, set $j = j + 1$, and repeat the protocol from step 2. If $j = k$, let $\beta' = f_k(r_k)/F(I, z_1, \ldots, z_k) \bmod p$ and let $I' = r_3, \ldots, r_k$.

The protocol is such that if $A_i(I, x) = \beta \bmod p$ then $A_{i-1}(I', x) = \beta' \bmod p$. However, if $A_i(I, x) \neq \beta \bmod p$ then the probability that $A_{i-1}(I', x) \neq \beta'$ is high, regardless of what the prover sends to V.

The key to the proof of this second property is that for all $j \geq 1$, if the polynomial $f_j(z_j)$ sent by the prover to the verifier is not equal to $P_j(z_j)$, then with high probability, $f_j(r_j) \neq P_j(r_j)$. To see this, note that if two polynomials of constant degree are not equal, they agree at at most a constant number of points. Hence, the probability that they agree at a point r_j chosen randomly and uniformly from $\mathbf{Z}/p\mathbf{Z}$ is $O(1/p)$. There are $k = \Theta(s(n))$ iterations of the above protocol for a given i, and the protocol is repeated $\Theta(t(n))$ times, since $N = \Theta(t(n))$. Hence if $p = \Theta(s(n)t(n))$, the error probability of the protocol is small.

The space needed is dominated by the space to store $k = O(s(n))$ elements $r_1, \ldots, r_k$ of $\mathbf{Z}/p\mathbf{Z}$, each of which has length $O(\log p) = O(\log t(n))$. Hence the space is $O(s(n) \log t(n))$. We next consider the time needed by (P, V). The time to execute the above protocol is dominated by the time to execute step 4, where an arithmetic formula F of size $O(s^2(n))$ must be evaluated. This requires $O(s^2(n))$ additions and multiplications over the field $\mathbf{Z}/p\mathbf{Z}$, and each of these operations can be done in $O(\log^2 p)$ steps. The protocol is executed $N = O(t(n))$ times; hence the total time required for the protocol is $O(t(n)s^2(n) \log^2 p)$. Once the protocol has been executed N times, G is evaluated, which takes time $O(n \log n \log^2 p)$. Since $p = \Theta(s(n)t(n))$, it follows that $\log p = O(\log t(n))$, the total time is $O((t(n)s^2(n) + n \log n) \log^2 t(n))$. □

Finally, we consider public coin interactive proof systems with log space and log random bits. The following relationships were proved by Condon and Ladner (1988).

Theorem 9 *NLOG ⊆ AM(log-space, log-random-bits) ⊆ LOGCFL.*

The left containment is immediate.

The proof that AM(log-space, log-random-bits) ⊆ LOGCFL, uses the equivalence of LOGCFL and the class of languages accepted by nondeterministic

pushdown automata which have $O(\log n)$ auxiliary storage and run in polynomial time (1978), and describes how a *computation tree*, which represents all of the computations of an interactive proof system (P, V), can be traversed using the pushdown store. This is possible with only $O(\log n)$ auxiliary space since the number of branching points in the tree is $O(\log n)$, due to the limited number of random bits used by the verifier. When traversing a path, only the direction at the branching points taken so far, and the current configuration need be stored. For more details, see Condon and Ladner (1992).

5 CONSTANT SPACE

Dwork and Stockmeyer (1992) have proved a number of strong results about the power of interactive proof systems with constant space bounded verifiers. In this restricted setting, they have obtained separation results that are not possible for polynomial time, or even log space bounded interactive proof systems. For example, we will see in Section 5.1 that AM(2pfa,poly-time) is properly contained in IP(2pfa, poly-time). The techniques used to obtain these results laid the foundations for new results on the power of probabilistic finite state automata. Dwork and Stockmeyer (1989) showed that 2-way probabilistic finite state automata with bounded error, that run in polynomial expected time, accept exactly the regular languages.

We describe the results on constant space bounded interactive proof systems in two sections, one for private coins and the other for public coins. In the constant space bounded model, we assume that the input is presented on a finite tape, with endmarkers # at both ends of the input.

5.1 Private Coins

In the introduction, we saw that the language Pal, of strings that read the same backwards as forwards, is in the class IP(2pfa,poly-time). The following theorem shows that languages much more complex than Pal have interactive proof systems which are $O(1)$ space bounded, if more than polynomial time is allowed.

Theorem 10 *$DTIME(2^{O(n)}) \subseteq IP(2pfa) \subseteq ATIME(2^{2^{O(n)}})$.*

The lower bound is due to Dwork and Stockmeyer (1992) and the upper bound to Condon and Lipton (1989). The proof of the lower bound is similar to that of Theorem 1. In this case, on input x, the prover sends the verifier the computation of a linear space bounded alternating Turing machine, and the verifier must check that the computation is valid. One of the main differences

between this proof and that of Theorem 1 is in the way that the verifier checks that the jth symbol of the $(i+1)$st configuration of the computation follows correctly from the ith configuration. In Theorem 1, V stores j, but this is not possible here since the verifier has only $O(1)$ space. Instead, when receiving the computation from the prover, the verifier uses the input as a "ruler", in order to locate the jth symbol of the $(i+1)$st configuration in the string obtained from the prover, starting from the j symbol of the ith configuration. This is possible since the length of a configuration is linear in the length of the input. Another difference is that it is no longer possible to choose j uniformly at random. It turns out that an alternative simple method, which favors symbols early in the computation, suffices for the correctness of the protocol.

The next theorem, proved using similar techniques, shows that IP(2pfa,poly-time) is also quite powerful.

Theorem 11 *AM($O(n)$-space,poly-time) $\subseteq$ IP(2pfa,poly-time).*

Thus, IP(2pfa,poly-time) contains an NP-complete language. Also, AM(2pfa) $\subseteq$ AM($O(n)$-space,poly-time), and hence in IP(2pfa,poly-time). The proof that AM(2pfa) $\subseteq$ AM(O(n)-space,poly-time) is similar to the proof of Theorem 6, except instead of using linear programming to compute the values $p(i)$, they are sent by the prover to the verifier.

5.2 Public Coins

We now consider Arthur-Merlin games which are $O(1)$ space bounded. In Theorem 12, we describe the result of Dwork and Stockmeyer that this class does not contain Pal. It follows that AM(2pfa) is properly contained in P and also that AM(2pfa) is properly contained in IP(2pfa, poly-time). In addition, we state other results of Dwork and Stockmeyer, which show that AM(2pfa) contains languages that are neither in AM(2pfa, poly-time) nor in 2PFA.

Theorem 12 *Pal $\notin$ AM(2pfa).*

Proof: Suppose to the contrary that (P, V) is a public coin interactive proof system which is $O(1)$ space bounded and accepts Pal. To simplify the proof, we assume that for all P^*, (P^*, V) halts with probability 1 and that the computation ends with the input head at the right end of the tape. Only slight modifications to the following proof are necessary when these conditions are not satisfied. There are three main steps to the proof: (i) we define a notion of "closeness" of two strings; (ii) we argue that for sufficiently large m, there

are two distinct strings w_i and w_j of length m which are close and (iii) we show that for some P^*, (P^*, V) accepts $w_j w_i^R$ with probability greater than $1/2$, achieving a contradiction.

To define closeness, we need the following notation, which describes the possible conditions in which a sub-computation of (P, V) can start or end on the string w, when the input is ww^R. Define a *starting condition* to be a pair (q, η), where q is a state of M and $\eta \in \{\text{Left,Right}\}$; intuitively it means that the computation of (P, V) is started in state q at the end of w denoted by η. Similarly, a pair (q, η) denotes a *stopping condition*, intuitively that the head of V falls off the η end of w with V in state q. Let $p(w, a, b)$ be the probability that on the computation of (P, V) on input ww^R, the stopping condition is b, given that the starting condition is a. Note that $p(w, a, b)$ depends on P and hence indirectly on the fact that w is followed by w^R on the input tape.

We say two numbers x and y are β-*close* for $\beta > 1$ if (a) $x = 0$ if and only if $y = 0$ and (b) if $x > 0$ and $y > 0$, then $1/\beta \leq x/y \leq \beta$. Also, two strings w_i and w_j are β-close if for all a, b, $p(w_i, a, b)$ and $p(w_j, a, b)$ are β-close. This completes (i), the definition of closeness.

We next outline a proof of (ii), that given any constant $\beta > 1$, for sufficiently large m, there is a pair of strings w_i, w_j, both of length m, which are β-close. Let d be the number of pairs (a, b) where a is a starting condition and b is a stopping condition. Note that there is a set S of strings of length m such that $|S| \geq 2^{m-d}$, and for all pairs (a, b), either $p(w, a, b) > 0$ for all $w \in S$, or $p(w, a, b) = 0$ for all $w \in S$. Moreover, if w is of length m and $p(w, a, b) \neq 0$, then it can be shown that $p(w, a, b) \geq 1/c^m$ for some constant c. Hence the range in which $p(w, a, b)$ lies is $[1/c^m, 1]$. From this, a pigeon-hole argument can be used to show that for sufficiently large m, two of the 2^{m-d} strings of S are β-close. Let these be w_i and w_j.

We finally describe the proof of (iii), that there is a prover P^* such that (P^*, V) accepts $w_j w_i^R$ with probability $> 1/2$. P^* is the prover which, when V's head is in the string $\#w_j$, responds to V as if the input is $w_j w_j^R$, and when V's head is in the string $w_i^R\#$, responds to V as if the input is $w_i w_i^R$. It remains to show that the probability that (P^*, V) accepts $w_j w_i^R$ is at least $1/2$. To do this, we show that the probability that (P^*, V) accepts $w_j w_i^R$ is within some small constant of the probability that (P, V) accepts $w_i w_i^R$. The computations of (P^*, V) on $w_i w_i^R$ and on $w_j w_i^R$ are modeled as Markov chains H_i and H_j, respectively, which have the same number of states. Both chains have special

initial, accept and reject states, and in addition, have two states $(q, l), l = 1, 2$ for each state q of V. State $(q, 1)$ of H_i means that the verifier is in state q with the head at the right end of w_i, and state $(q, 2)$ means that the verifier is in state q with the head at the left end of w_i^R. The initial state of H_i means that the verifier is in its initial state with the head under the left endmarker #, and the accept and reject states mean that the verifier has halted in an accepting or rejecting state, respectively. The probability transitions between these states model the computation of (P, V) on $w_i w_i^R$. H_j is defined similarly, except that the state $(q, 1)$ of H_j means that the verifier is in state q with the head at the right end of w_j. Because w_i and w_j are β-close, and because P and P^* are the same on the right half of the strings $w_i w_i^R$ and $w_j w_i^R$, these Markov chains satisfy the property that the transition probabilities between any two states are β-close.

A result of Leighton and Rivest (1986) shows that if two Markov chains satisfy this property, then the probabilities of reaching the accept states of both chains are β^{2s}-close, where s is the number of states in the Markov chains. Thus, the probability that $w_i w_j^R$ is accepted is at least β^{-2s} times the probability that $w_i w_i^R$ is accepted, which is at least $\beta^{-2s} 3/4$. For sufficiently large m, we can choose β so that this is greater than $1/2$, completing the proof of (iii). □

The argument in the above theorem was generalized by Dwork and Stockmeyer to prove the following theorem, which can be used to identify other languages that are not in AM(2pfa).

Theorem 13 *Let $L \subseteq \Sigma^*$. Suppose there is an infinite set I of positive integers and, for each $m \in I$, sets $W_m = \{w_1, w_2, \ldots, w_{N(m)}\}$, $U_m = \{u_1, u_2, \ldots, u_{N(m)}\}$, and $V_m = \{v_1, v_2, \ldots, v_{N(m)}\}$ of words such that*

1. *$|w| \leq m$ for all $w \in W_m$,*
2. *for every integer k there is an m_k such that $N(m) \geq m^k$ for all $m \in I$ with $m \geq m_k$, and*
3. *for all $1 \leq i, j \leq N(m)$, $u_j w_i v_j \in L$ if and only if $i = j$.*

Then $L \notin$ AM(2pfa).

Similar techniques can be used to separate the classes 2PFA and AM(2pfa,poly-time) from AM(2pfa), leading to the following results.

Theorem 14 *(i) 2PFA $\subset$ AM(2pfa),*

(ii) AM(2pfa,poly-time) $\subset$ AM(2pfa) and

(iii) 2PFA $\not\subset$ AM(2pfa,poly-time).

The language Upal = $\{a^n b^n \mid n \geq 0\}$ is an example of a language in 2PFA but not in AM(2pfa,poly-time). Frievalds (1981) showed that Upal is in the class 2PFA. The language Center = $\{wbx \mid w, x \in \{a, b\}^*$ and $|w| = |x|\}$ is an example of a language in the class AM(2pfa) but is not in either of the classes AM(2pfa,poly-time) or 2PFA. The proof that Center is in AM(2pfa) is a simple generalization of Frievalds' proof that Upal is in the class 2PFA. (In this case, the prover is needed to direct the verifier to the center of the string.)

The proofs that Center and Upal are not in AM(2pfa,poly-time) and that Center is not in 2PFA are refinements of the techniques used to prove Theorems 12 and 13.

6 VARIATIONS ON THE MODEL

Applications in cryptography, distributed computation and other fields have prompted studies of variations of the model of interactive proof systems which we have considered so far. We describe two of these variations here, and state without proof known results for the models. We consider multiple provers in Section 6.1. Feige and Shamir (1989) and independently Condon and Lipton (1989) showed that, even when the verifier is a probabilistic finite state automaton with a 1-way input head, there is a 2-prover interactive proof system which can accept any recursive language.

Zero knowledge interactive proof systems are described in Section 6.2 and both positive and negative results about the existence of zero knowledge interactive proof systems for certain languages are presented. Unlike many results on polynomial time bounded interactive proof systems, these results are not based on any unproven assumptions. Results on log space bounded zero knowledge interactive proof systems are due to Kilian (1988) and on constant space bounded interactive proof systems are due to Dwork and Stockmeyer (1992).

6.1 Multiple Provers

The multiple prover model, in which the verifier interacts with $k \geq 2$ provers, $P_1, \ldots, P_k$, was first introduced by Ben-Or, Goldwasser, Kilian and Wigderson (1988). It is natural to ask whether multiple provers increases the power of

the model. The answer appears to be "yes" when the interactive proof system is polynomial time bounded, since Babai, Fortnow and Lund (1991) showed that polynomial time bounded, 2-prover interactive proof systems accept all languages in nondeterministic exponential time. We will see in this section that, also in the case of log space bounded verifiers, the answer is a resounding "yes".

In the multiple prover model, the verifier has k communication cells, and the communication states of the verifier are partitioned into k groups. Whenever the verifier's state is a communication state in the ith group, the next configuration is determined by communicating with the ith prover via the ith communication cell, as in the single-prover model. Each prover P_i is specified by a prover transition function from $\Sigma^* \times \{0,1\}^*$ to $\{a,b\}$. $P_i(x, b_1 \ldots b_j)$ is the response of prover P_i on input x, when the verifier has just sent b_j to the prover, and $b_1, \ldots, b_{j-1}$ is the sequence of all past communication symbols written by the verifier in the ith communication cell. Language acceptance is defined in a similar manner to language acceptance for single prover interactive proof systems, with $(P_1, \ldots, P_k, V)$ replacing (P, V) and $(P_1^*, \ldots, P_k^*, V)$ replacing (P^*, V).

In what follows, we restrict our attention to 2-prover interactive proof systems. The results below are also true for k-prover interactive proof systems, for any constant k. We denote by 2IP(<restrictions>) the class of languages which have 2-prover interactive proof systems with the the restrictions denoted by <restrictions>.

The first result states that the class of languages accepted by multi-prover interactive proof systems is exactly the recursive languages, even when the verifier is a pfa, or a 1-way probabilistic finite state automaton.

Theorem 15 *2IP(pfa) is exactly the set of recursive languages.*

This result was proved by Feige and Shamir (1989) and independently by Condon and Lipton (1989). The proof of Feige and Shamir actually proves something stronger - that the result is true even if the verifier acts *synchronously*, which means that the verifier communicates with each prover at regular intervals. This can be formalized by requiring that the verifier communicates with prover P_i at step t if and only if $t = i \bmod k$. The main technique introduced in their proof is a method whereby the verifier can simulate a Turing machine computation, using the provers to "store" the contents of the tape. Roughly,

this is done by giving each prover random information about the tape contents. This information is meaningless to a single prover, but by combining the information of both provers, the verifier can reconstruct the tape cells whenever necessary.

Feige and Shamir extended their technique to show that when the verifier is a pfa, and is simultaneously polynomially time bounded, then 2IP(pfa,poly-time) = 2IP(poly-time). Combining this with the result of Babai, Fortnow and Lund (1991) that 2IP(poly-time) = NTIME($2^{poly(n)}$), the following theorem is obtained.

Theorem 16 *2IP(pfa,poly-time) = NTIME($2^{poly(n)}$).*

A variation of the multi-prover model, called the noisy oracle model, was proposed by Feige, Shamir and Tennenholtz (1988). The paper of Feige and Shamir (1989) also contains results on the complexity of the noisy oracle model when space is restricted.

6.2 Zero Knowledge

Informally, a zero knowledge interactive proof system for a language L is one in which, on input $x \in L$, a verifier can learn nothing from the prover P, other than the fact that $x \in L$. The notion of a zero knowledge interactive proof system was first introduced by Goldwasser, Micali and Rackoff (1985) for polynomial time bounded interactive proof systems. The notion has been central to much recent work in cryptography, and also has interesting applications in distributed computing (see for example Feige, Fiat and Shamir (1987) and Goldreich, Micali and Wigderson (1986).

However, many of the results on polynomial time zero-knowledge interactive proof systems are based on unproven assumptions. Because of this, Dwork and Stockmeyer (1992) and later Kilian (1988) studied space bounded zero knowledge interactive proof systems, with the goal of proving, without using any unproven assumptions, both positive and negative results on the existence of zero knowledge interactive proof systems for certain languages.

Formalizing the intuitive notion of zero knowledge, especially that of "learning nothing" from the prover, is no easy task. The example of the language Pal = $\{x \mid x = x^R\}$, introduced in Section 1, gives some insight to the difficulty. We saw there an interactive proof system (P, V) for Pal in which the prover P simply sends the input repeatedly to the verifier. This seems to be a zero

knowledge interactive proof system, since the verifier is obtaining nothing from the prover that it does not already "know". However, Dwork and Stockmeyer argue intuitively that it is not a zero knowledge interactive proof system, as follows. Let A be the set of all strings that are double palindromes, that is, a string x is in A if and only if $x = ww^R$ and w is also a palindrome. Let $B = \text{Pal} - A$. Techniques similar to those of Theorem 12 show that no 2pfa can separate A from B. However, there is a 2pfa V^* such that (P, V^*) can separate A from B. In order to check if the second half of the input is a palindrome, the verifier can perform the following computation, as it receives the input from the prover. Starting at the left end of the tape, V^* moves its head two steps to the right for every symbol it receives from P, until the right end of the tape is reached. At this point, if the input is ww^R, P has finished sending w and is ready to send w^R to V^*. Now, V^* can test if $w = w^R$.

Dwork and Stockmeyer proposed a definition of zero knowledge, based on the following notion of separating sets. We say (P, V) *separates* sets $A, B \subseteq \Sigma^*$ with $A \cap B = \emptyset$ if for all $x \in A$, the probability that (P, V) accepts x is at least $3/4$ and for all $x \in B$, the probability that (P, V) accepts x is at most $3/4$. Similarly, we can define what it means for a 2pfa, M to separate two sets. Let $\mathcal{V}$ be a class of verifier machines and let $(\emptyset, \mathcal{V})$ be the subset of machines in $\mathcal{V}$ that do not communicate with the prover. Let (P, V) be an interactive proof system for the language L where $V \in \mathcal{V}$. Then (P, V) is a recognition zero knowledge interactive proof system for L with $\mathcal{V}$ verifiers if, for any $V^* \in \mathcal{V}$ and any $A, B \subseteq L$ with $A \cap B = 0$ such that (P, V^*) separates A and B, there is an $M_{V^*} \in (\emptyset, \mathcal{V})$ such that M_{V^*} separates A and B. We denote by ZKIP(<restrictions>) the class of languages which have recognition zero knowledge interactive proof systems with the restrictions denoted by <restrictions>. With this definition, Dwork and Stockmeyer proved the following results.

Theorem 17 *Pal and the graph isomorphism problem are not in ZKIP(2pfa).*

We have already seen that interactive proof systems do exist for these languages. Dwork and Stockmeyer also prove a positive result for a restricted class of verifiers. A *sweeping-2pfa* is a 2pfa which, on any input, only switches the direction of head movement when at the left or right end of the input.

Theorem 18 *The language Upal* $= \{a^n b^n \mid n \geq 0\}$ *is in ZKIP(sweeping-2pfa, poly-time).*

The proof of this result is quite intricate, and uses deep results from the theory of Markov chains. The theorem is not vacuous, as Greenberg and Weiss (1986) showed that Upal is not accepted by any automaton which runs in polynomial expected time.

Kilian (1988) studied zero knowledge interactive proof systems with verifiers which are simultaneously log space bounded and polynomially time bounded. He proposed a stronger definition of zero knowledge than that of Dwork and Stockmeyer, and proved the following result.

Theorem 19 *IP(log-space, poly-time) = ZKIP(log-space,poly-time).*

The techniques of Kilian are quite different from those of Dwork and Stockmeyer, and involve tools from communication complexity and cryptography.

7 OPEN PROBLEMS

In the preceding sections, we have mentioned numerous unsolved problems on the complexity of space bounded interactive proof systems. We discuss some of these further here.

• There is a large gap between the best known upper and lower bounds for *IP(2pfa)*. The best known lower bound is deterministic exponential time and the best known upper bound is alternating double exponential time. Can either of these bounds be improved?

One way to improve these bounds is suggested by the structure of interactive proof systems for languages in DTIME($2^{poly(n)}$), as described in Theorem 1. All known interactive proof systems with a log space bounded verifier have the property that on any input, the verifier and prover iterate a double exponential number of times a protocol that takes only exponential time. We call such an interactive proof system *periodic*. It would be interesting to find a non-periodic interactive proof system that accepts languages not known to be in DTIME($2^{poly(n)}$). Alternatively, a proof that all languages accepted by log space interactive proof systems are also accepted by periodic interactive proof systems would imply better bounds for IP(log-space).

• Dwork and Stockmeyer (1992) asked the following question. Is AM(2pfa,poly-time) equal to the class of regular languages?

Dwork and Stockmeyer show that there is a pair of sets A and B which can be

separated by an Arthur-Merlin game which is polynomial time bounded and $O(1)$ space bounded, but which cannot be separated by 2pfa. The sets are defined as follows. Let U be the set of words of the form $x\#^k$ where $k = 2^{|x|}$. Then, A is the set of $x\#^k \in U$ such that $x \in$ Center and B is the set of $x\#^k \in U$ such that $x \notin$ Center. However, the set U is not in AM(2pfa,poly-time), which makes it difficult to extend this separation problem to show that AM(2pfa,poly-time) accepts a non-regular language.

• The work of Fortnow and Lund (1991) motivates the question: is AM(log-space, poly-time) = P? Their result that P is contained in AM($o(\log^2 n)$-space, poly-time) leads one to conjecture that perhaps the answer to the above open question is yes. However, it is not clear that their techniques can be extended to prove this. A positive answer to the question might imply that Markov decision processes can be evaluated by new polynomial time algorithms that do not involve linear programming techniques.

• From Section 6, it appears that questions on the complexity of space bounded multiple prover interactive proof systems are completely resolved. However, one intriguing question remains. What languages are in the class oneway-2IP(2pfa)? The best bounds we know are (i) any language in nondeterministic linear space is in oneway-2IP(2pfa) and (ii) oneway-2IP(2pfa) is contained in the class of recursive languages. This is a rather large gap, and one of these bounds can surely be improved.

• Finally, can other nonapproximability results, such as the result on the max word problem presented in Section 3, be obtained from further study of space bounded interactive proof systems?

8 ACKNOWLEDGEMENTS

We thank an anonymous referee for many helpful comments on the presentation. This work was supported in part by NSF grants, numbers CCR-9100886 and CCR-9257241.

9 REFERENCES

S. Arora and S. Safra. Probabilistic checking of proofs, Proceedings ofthe 33rd Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1992, 2-13.

S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Hardness of Approximation Problems, Proceedings of the 33rd Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1992, 14-23.

L. Babai. Trading group theory for randomness. Proceedings of the 17th Annual ACM Symposium on the Theory of Computing, May 1985, 421-429.

L. Babai, L. Fortnow and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity 1* 1991, 3-40.

L. Babai, L. Fortnow, L. Levin and M. Szegedy. Checking computations in polylogarithmic time, Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing, May, 1991, 21-31.

L. Babai and S. Moran. Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. System Sci.*, 36 1988, 254-276.

D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science, Springer Verlag Lecture Notes in Computer Science 415, February 1990, 37-48.

M. Ben-Or, S. Goldwasser, J. Kilian and A. Wigderson. Multi-prover interactive proofs: how to remove intractability, Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, May, 1988, 113-131.

A. K. Chandra, D. C. Kozen and L. J. Stockmeyer. Alternation, *J. ACM*, 28(1), 1981, 114-133.

A. Condon. *Computational Models of Games*, MIT Press, July 1989.

A. Condon. The complexity of the max word problem and the power of one-way interactive proof systems, Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science, Springer Verlag Lecture Notes in Computer Science 480, February 1991, 456-465.

A. Condon. Space bounded computational games, *J. ACM*, 38(2), April 1991b, 472-494.

A. Condon. The complexity of stochastic games, *Information and Computation*, 96(2), February 1992, 203-224.

A. Condon and R. Ladner. Probabilistic game automata, *J. Comput. System Sci.* 36(3), June 1988, 452-489.

A. Condon and R. Ladner. Interactive proof systems with polynomially bounded strategies, Proceedings of the Seventh Annual Conference on Structure in Complexity Theory, IEEE Computer Society Press, June 1992, pages 282-294.

A. Condon and R. J. Lipton. On the complexity of space bounded interactive proof systems, Proceedings of the 30th Annual Symposium on the Foundations of Computer Science, IEEE Computer Society Press, October 1989, 462-467. See also University of Wisconsin, Madison Technical Report Number 841, 1989.

S. A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space, Proceedings of the 11th Annual ACM Symposium on Theory of Computing, 1979, 338-345.

C. Derman. *Finite State Markov Decision Processes*, Academic Press, 1972.

C. Dwork and L. Stockmeyer. Interactive proof systems with finite state verifiers, Tech. Report RJ 6262, IBM Research Division, Almaden Research Center, San Jose, CA, 1988. See also "Finite state verifiers I: the power of interaction", *J. ACM*, 39(4) October 1992, 800-828 and "Finite state verifiers II: zero knowledge, *J. ACM*, 39(4) October 1992, 829-858.

C. Dwork and L. Stockmeyer. On the power of 2-way probabilistic finite state automata, Proceedings of the 30th Annual Symposium on the Foundations of Computer Science, IEEE Computer Society Press, October 1989, 480-485. See also "A time complexity gap for two-way probabilistic finite state automata", SIAM J. Comput. 19, 1990, 1011-1023.

U. Feige, A. Fiat and A. Shamir. Zero knowledge proofs of identity, Proceedings of the 19th Annual ACM Symposium on Theory of Computing, May 1987, 210-217.

U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete, Proceedings of the 32nd Annual Symposium on the Foundations of Computer Science, IEEE Computer Society Press, October 1991, 2-12.

U. Feige and A. Shamir. Multi-oracle interactive protocols with space bounded verifiers, Proceedings of the Fourth Annual Conference on Structure in Com-

plexity Theory, IEEE Computer Society Press, June 1989, 158-164.

U. Feige, A. Shamir and M. Tennenholtz. The noisy oracle problem, Proceedings of CRYPTO 1988.

L. J. Fortnow. Complexity-theoretic aspects of interactive proof systems, Ph. D. Thesis, Technical Report Number TR-447, Laboratory for Computer Science, MIT.

L. Fortnow and C. Lund. Interactive proof systems and alternating time-space complexity, Proceedings of 8th Annual Symposium on Theoretical Aspects of Computer Science, February, 1991, 263-274.

L. Fortnow, J. Rompel and M. Sipser. On the power of multi-prover interactive protocols, Proceedings of the Third Annual Conference on Structure in Complexity Theory, IEEE Computer Society Press, 1988, 156-161.

R. Frievalds. Probabilistic two-way machines, Proceedings of the International Symposium on Mathematical Foundations of Computer Science,
Springer-Verlag Lecture Notes in Computer Science 118, 1981, 33-45.

J. Gill. Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* 6(4), 1977, 675-695.

O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity and a method of cryptographic protocol design, Proceedings of the 27th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, October 1986, 218-229.

S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems, Proceedings of 17th Annual ACM Symposium on the Theory of Computing, May 1985, 291-304. See also *SIAM J. Comput.* 18, 1989, 186-208.

S. Goldwasser and M. Sipser. Public coins vs. private coins in interactive proof systems, *Randomness and Computation*, Volume 5 of Advances in Computing Research, JAI Press, Greenwich, 1989 73-90.

A. G. Greenberg and A. Weiss. A lower bound for probabilistic algorithms for finite state machines, *J. Comput. System Sci.* 33, 1986, 88-105.

R. A. Howard. *Dynamic Programming and Markov Processes*, M.I.T. Press, 1960.

L. G. Khachiyan. A polynomial algorithm for linear programming, Soviet Math Dokl. 20, 1979, 191-194.

J. Kilian. Zero knowledge with log-space verifiers, Proceedings of the 29th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, October 1988, 25-35.

F. T. Leighton and R. L. Rivest. Estimating a probability using finite memory, *IEEE Trans. Information Theory* IT-32, 1986, 733-742.

R.J. Lipton. New directions in testing, *Distributed Computing and Cryptography, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, 2 1991, 191-202.

C. Lund, L. Fortnow, H. Karloff and N. Nisan. Algebraic methods for interactive proof systems, Proceedings of the 30th Annual Symposium on the Foundations of Computer Science, IEEE Computer Society Press, October 1990, 2-10. See also *J. ACM* 39(4), October 1992, 859-868.

C. H. Papadimitriou. Games against nature, *J. Comput. System Sci.*, 31, 1985, 288-301.

A. Paz. *Introduction to Probabilistic Automata,* Academic Press, 1971.

G. L. Peterson and J. H. Reif. Multiple-person alternation, Proceedings of 20th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1979, 348-363.

J. H. Reif. The complexity of two-player games of incomplete information, *J. Comput. System Sci.* 29, 1984, 274-301.

W. L. Ruzzo. On uniform circuit complexity, *J. Comput. System Sci.* 22, 1981, 365-383.

A. Shamir. IP=PSPACE, Proceedings of the 30th Annual Symposium on the Foundations of Computer Science, IEEE Computer Society Press, October 1990, 11-15. See also *J. ACM* 39(4), October 1992, 869-877.

I.H. Sudborough. On the tape complexity of deterministic context-free languages, *J. ACM*, 25(3), 1978, 405-114.

S. Toda. PP is as hard as the polynomial time hierarchy, *SIAM J. Comput.*, 20(5), 1991, 865-877.

L. Valiant. The complexity of computing the permanent, *Theoretical Computer Science*, 8, 1979, 189–201.

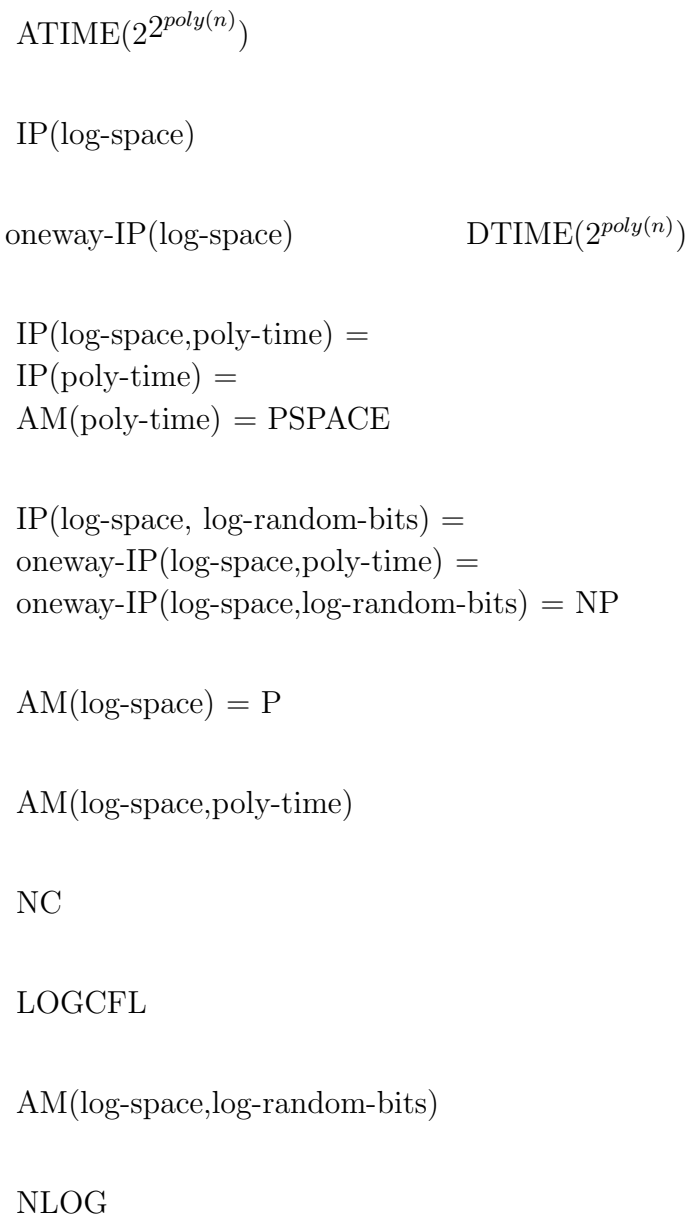

Figure 1: Results on log space bounded interactive proof systems. Each class is contained in the one above it. No containments are known to be proper.

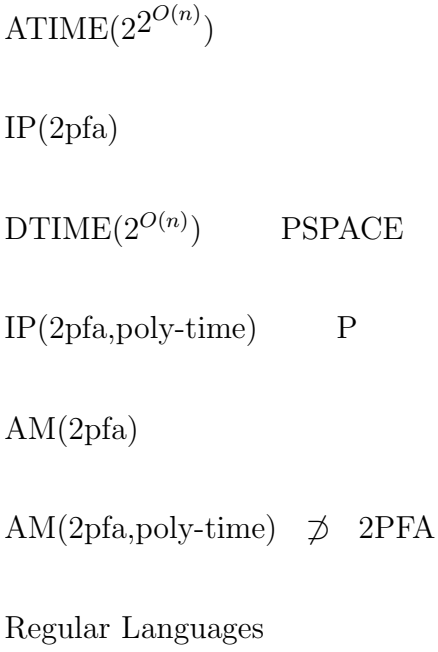

Figure 2: Results on constant space bounded interactive proof systems. Solid arrows denote proper containments, and dotted arrows denote containments which are not known to be proper.