# Finite State Verifiers I: The Power of Interaction

Cynthia Dwork and Larry Stockmeyer

IBM Almaden Research Center
San Jose, California

Abstract. An investigation of interactive proof systems (IPS's) where the verifier is a 2-way probabilistic finite state automaton (2pfa) is initiated. In this model it is shown:

1. IPS's where the verifier uses private randomization are strictly more powerful than IPS's where the random choices of the verifier are made public to the prover.
2. IPS's where the verifier uses public randomization are strictly more powerful than 2pfa's alone, i.e., without a prover.
3. Every language which can be accepted by some deterministic Turing machine in exponential time can be accepted by some IPS.

Additional results concern two other classes of verifiers: 2pfa's which halt in polynomial expected time, and 2-way probabilistic pushdown automata which halt in polynomial time. In particular, IPS's with verifiers in the latter class are as powerful as IPS's where verifiers are polynomial-time probabilistic Turing machines. In a companion paper [7], zero knowledge IPS's with 2pfa verifiers are investigated.

This paper appeared in the *Journal of the ACM*, Vol. 39, No. 4, 1992, pp. 800–828.

## 1. Introduction

Issues in complexity theory and cryptography motivated Babai [1] and Goldwasser, Micali, and Rackoff [11] to introduce the concept of an interactive proof system. Speaking informally, an *Interactive Proof System* (IPS) for membership in a language $L$ is a two-party protocol whereby a "prover" convinces a "verifier" that elements $x \in L$ are actually in $L$. The concept is interesting only if the verifier is not itself sufficiently powerful to decide membership in $L$. Both the prover and the verifier may make random moves (flip coins), so in particular the prover is unable to predict the verifier's questions. The prover's ability to correctly answer all the verifier's questions plays an important role in "convincing" the verifier. In general, however, there is a small probability that the verifier will be fooled, i.e. falsely convinced that a string $x$ not in $L$ is actually in $L$. The hope is to obtain efficiency at the cost of perfect soundness.

To date, most research in interactive proof systems has dealt with the case that the verifier is a probabilistic Turing machine (ptm) which runs in polynomial time. Several flavors of interactive proof systems have been defined; in particular, (1) the verifier may employ either *public coins* or *private coins*, defined, roughly speaking, according to whether the prover can see the outcomes of the random choices made by the verifier; (2) the IPS may or may not be *zero knowledge*. The proof system is zero knowledge if *no* verifier can extract any information from the interactive proof that $x \in L$ except for the fact that $x \in L$, i.e., nothing extra is learned.

For polynomial-time ptm verifiers, proving any *difference* in power between two different flavors of IPS's will be difficult since it would prove that P, the class of languages accepted by deterministic Turing machines in polynomial time, is not equal to PSPACE, the class of languages accepted by Turing machines in polynomial space. This is true because (i) the languages accepted by the most powerful flavor of IPS (namely, private coins and no zero knowledge requirement) are contained in PSPACE, as shown by Condon and Ladner [5] and Feldman (personal communication in [12]) and (ii) any language in P is accepted by the weakest flavor of IPS (namely, an IPS where the verifier does not communicate with a prover). In particular, if there is any language $L \notin$ P such that a prover can help a verifier accept $L$, then P $\neq$ PSPACE. Similarly, some results proving the existence of zero knowledge IPS's are based on the (unproved) assumption that certain types of encryption schemes exist (see, for example, [10]).

The motivating goals of the new work are (1) to understand the structure of the set of languages with interactive proof systems (public coins vs. private coins, zero knowledge vs. arbitrary interactive proof systems); (2) to determine whether interactive proof systems ever really help the verifier determine something it cannot determine on its own; and (3) to obtain results about IPS's without unproved assumptions.

Because the ability to prove lower bounds is crucial to obtaining such results, we restrict the class of verifiers, namely, to 2-way probabilistic finite state automata (2pfa's). We summarize our main results for 2pfa verifiers in the case that zero knowledge is not an issue. The companion paper [7] considers the case of zero knowledge. As in previous work on IPS's, the prover is assumed to be arbitrarily powerful. By way of comparison, we also give the current state of knowledge for polynomial-time probabilistic Turing machine verifiers.

1. IPS's with private coins are more powerful, i.e., can recognize strictly more languages, than IPS's with public coins. This result remains true if the 2pfa verifier is required to halt in polynomial expected time.

   Polynomial time ptm: Private coins and public coins are equally powerful, at least when zero knowledge is not a concern [12].

2. IPS's with public coins are more powerful than 2pfa verifiers alone, i.e., without a prover.

   Polynomial time ptm: Relative power not known.

3. There is an IPS with private coins for any language which can be recognized in exponential time.

   Polynomial time ptm: Shamir [24], building on work of Lund, Fortnow, Karloff and Nisan [20], has shown that any language in PSPACE has an IPS.

In the course of proving results 1 and 2, we give techniques for showing that certain languages cannot be recognized by any 2pfa or by any IPS with a 2pfa verifier using public coins.

As a final piece of motivation, there is growing evidence that the concept of an interactive proof system is fundamental. This concept deserves to be studied in a variety of contexts, in particular, with verifiers other than polynomial-time ptm's. Related work in this direction has been done by Condon and Ladner [5] and Condon [4] who study IPS's with space bounded verifiers.

We now outline the remainder of the paper. Section 2 contains the precise definition of an IPS with a 2pfa verifier and related definitions. Section 3 contains results about IPS's with 2pfa verifiers, including results 1, 2, and 3 above. In Section 4, we show that IPS's where verifiers are polynomial-time 2-way probabilistic pushdown automata recognize exactly the same class of languages as IPS's with polynomial-time ptm verifiers. Open questions are discussed in Section 5.

## 2. Definitions

### 2.1. Informal Discussion

Our definition of an interactive proof parallels the one used in previous papers on interactive proof systems where the verifier is a polynomial-time bounded probabilistic Turing machine (ptm), for example, Goldwasser, Micali, and Rackoff [11], and the one given by Condon and Ladner [4, 5] in a more general setting where the verifier can be either a time bounded or a space bounded Turing machine. The main difference in our case is that the verifier is a 2-way probabilistic finite state automaton (2pfa). A 2pfa consists of a probabilistic finite state control and a 2-way head which reads the input string. In addition, the verifier can communicate with a prover. In our case, the communication is done via a single communication cell which can hold a single symbol from some finite communication alphabet. Whenever the verifier wants a piece of information from the prover, it writes a symbol in the cell, the prover writes its response in the cell, and the verifier continues its computation based on the prover's response. The prover writes a symbol only in response to a symbol written by the verifier. This communication mechanism differs from previous definitions where communication was done by writing strings on a tape. Our definition is consistent, however, with the verifier being a finite state machine; just as we can view a finite state machine as a Turing machine with worktapes of constant size, we can view a single communication cell as a communication tape of constant size. Also note that the previous definitions could just as well have used a communication cell instead of a tape, since the symbols on the tape can be sent through the cell one symbol at a time, assuming that we are not interesting in measuring the number of rounds of interaction, which is the case in this paper.

An important aspect of the definition of an interactive proof (both our definition and previous definitions) is that the prover's view of the verifier's computation is restricted to the view seen through the communication cell. Since the verifier can carry out parts of its computation between communications with the prover, the prover does not in general have complete information about the configuration of the verifier, i.e., its state and the position of its input head. Following [12], we call this aspect *private coins*; in the framework of [5] it is called *partial information.* We shall also consider interactive proofs where the prover has complete information of the configuration of the verifier. As in [12], we say that the verifier uses *public coins* in this case; in [5], this case is called *complete information.* In the terminology of Babai [1], an IPS using public coins is an *Arthur-Merlin proof system.* In the following subsections, these definitions are made more precise.

### 2.2. 2pfa Verifiers

If $S$ is a finite set, a *coin-tossing distribution on $S$* is a mapping $\psi$ from $S$ to $\{0, 1/2, 1\}$ such that $\sum_{a \in S} \psi(a) = 1$. The mapping means "choose $a$ with probability $\psi(a)$." A particular *interactive proof system* (*IPS*) consists of a *verifier $V$* and a *prover $P$*. The verifier and the prover share a common finite *input alphabet* $\Sigma$ and a finite *communication alphabet* $\Gamma$. The verifier $V$ is specified by a finite set of *verifier states $Q$* and a *verifier transition function $\delta$*. The states $Q$ are partitioned into three subsets, $R$, $C$, and $H$, the *reading states*, *communication states*, and *halting states*, respectively. There is a designated reading state $q_0$, the *initial state.* For our purposes, there are two halting states, the *accepting state $q_a$* and the *rejecting state $q_r$*. The symbol $\not{c}$ is used to mark the left and right end of the input, so we assume $\not{c} \notin \Sigma$. For each reading state $q \in R$ and each $\sigma \in \Sigma \cup \{\not{c}\}$, $\delta(q, \sigma)$ is a coin-tossing distribution on $Q \times \{-1, 0+1\}$. The meaning of $\delta$ in this case is that if $V$ is in the reading state $q$ and scanning $\sigma$ on the input tape, then $V$'s next action, i.e., new state and head movement, is chosen according to the distribution $\delta(q, \sigma)$. We assume that $\delta$ is such that the head never moves left (right) when scanning the leftmost (rightmost) endmarker. With each communication state $q \in C$ there is an associated communication symbol $\gamma_q \in \Gamma$. For each $q \in C$ and each $\gamma \in \Gamma$, $\delta(q, \gamma)$ is a coin-tossing distribution on $Q$. When $V$ is in communication state $q$, we imagine that $V$

writes $\gamma_q$ in the communication cell. Letting $\gamma$ be the symbol which the prover then writes in the cell, the next state of $V$ is chosen according to the distribution $\delta(q, \gamma)$. Note that the input head of $V$ does not move in this case. We assume that $V$ halts when it enters a state in $H$, so we do not define $\delta$ on states in $H$. (The separation of input reading and communication into different sets of states is done for convenience in certain proofs, but this is not a real restriction since reading and communication can always be separated by increasing the number of states.)

The prover $P$ is specified by a *prover transition function* $\rho$. For each $x \in \Sigma^*$ and each $y \in \Gamma^*$, $\rho(x, y)$ is a coin-tossing distribution on $\Gamma$. Whenever the input is $x$ and the verifier writes a new symbol in the communication cell, the response of the prover is chosen according to the distribution $\rho(x, y)$, where $y$ is the string of symbols which have been written in the communication cell by both the verifier and the prover, from the beginning of the computation up to and including the new symbol just written by the verifier. In other words, the prover's response can depend on the entire communication history.

The computation of $(P, V)$ on a particular input $x \in \Sigma^*$ starts with the string $\cent x \cent$ written on $V$'s input tape with $V$ in state $q_0$ scanning the leftmost $\cent$. The computation is then governed (probabilistically) by the transition functions $\delta$ and $\rho$ until the verifier enters a halting state. (The computation could be infinite if a halting state is never entered.) For a particular input $x$ and halting state $q$, we define *the probability that* $(P, V)(x)$ *halts in state* $q$ to be the probability, taken over all random choices of the verifier and the prover, that $V$ halts in state $q$ in the computation of $(P, V)$ on input $x$. The term "accepts" (resp., "rejects") abbreviates "halts in state $q_a$ (resp., $q_r$)."

The prover-verifier pair $(P, V)$ is an *interactive proof system for the language* $L \subseteq \Sigma^*$ *with error probability* $\varepsilon$ if

1. for all $x \in L$, $(P, V)(x)$ accepts with probability at least $1 - \varepsilon$, and
2. for all $x \notin L$ and all provers $P^*$, $(P^*, V)(x)$ rejects with probability at least $1 - \varepsilon$.

Let IP(2pfa) be the class of languages $L$ such that $L$ has an interactive proof system with error probability $\varepsilon$ for some $\varepsilon < 1/2$. (We put "2pfa" in the notation as a reminder that the verifier is a 2pfa.)

Let 2PFA denote the class of languages recognized by 2pfa's with error probability $\varepsilon < 1/2$. To avoid making a new set of definitions for 2pfa's, we can view a 2pfa as an IPS where the verifier has no communication states (so the prover can be empty). For example, 2PFA is exactly IP(2pfa) restricted to proof systems $(P, V)$ where $V$ has no communication states.

The above definitions concern IPS's with private coins. The IPS $(P, V)$ uses *public coins* if the communication alphabet is $Q \times \{-1, 0 + 1\}$ and the verifier's transition function is such that whenever the verifier takes a step it sends its new state and head movement to the prover. This information is sufficient for the prover to keep track of the configuration of the verifier. Let AM(2pfa) be defined like IP(2pfa) but restricted to IPS's using public coins (AM abbreviates "Arthur-Merlin").

We shall also use a more general form of recognition called separation. Let $M$ be an IPS or a 2pfa, and let $A, B \subseteq \Sigma^*$ with $A \cap B = \emptyset$. Then $M$ *separates* $A$ *and* $B$ if there is some constant $\varepsilon < 1/2$ (the error probability) such that, for all $x \in A$, $M(x)$ accepts with probability at least $1 - \varepsilon$, and for all $x \in B$, $M(x)$ rejects with probability at least $1 - \varepsilon$. We do not care about the behavior of $M$ on inputs not in $A$ or $B$.

### 2.3. Halting and Time Complexity

The 2pfa verifier $V$ *halts with probability 1* if, for all $x$ and all provers $P^*$, the computation $(P^*, V)(x)$ halts with probability 1. In some results we will want to talk about the *time complexity* of the verifier, defined to be the number of steps taken by the verifier before halting. There are two cases: the *expected time*, averaged over all random choices of the verifier and the prover; and the *worst-case time*, maximized over all such choices. The verifier *runs in expected (resp., worst-case) time* $T(n)$ if, for any input $x$ of length $n$ and any prover $P^*$, the verifier halts in expected (resp., worst-case) time at most $T(n)$ in the computation $(P^*, V)(x)$. By attaching the prefix "ptime" to a class of machines, we restrict the class to those machines that run in polynomial expected time. For example, IP(ptime-ptm) is the class studied in many previous papers, where the verifier is a polynomial-time ptm.

### 2.4. Error Bounds

For polynomial-time ptm verifiers, the error probability of any interactive proof system can be decreased to $2^{-p(n)}$, for an arbitrary polynomial $p(n)$, by running the interactive proof system a polynomial number of times and taking the majority answer. For 2pfa verifiers, we do not know how to achieve such small error probabilities in general, since a finite state machine cannot compute the majority of a non-constant number of answers. However, if the IPS halts with probability 1, this technique can be used to decrease the error probability to an arbitrarily small positive constant by running the IPS a sufficiently large constant number of times. Specifically, if $L$ has an IPS which halts with probability 1 then, for any $\varepsilon > 0$, there is an IPS for $L$ with error probability $\varepsilon$.

### 2.5. Additional Definitions and Remarks

In general, we can always assume that the prover is deterministic, meaning formally that the definition of the prover involves only distributions which map to $\{0, 1\}$. Equivalently, we can imagine that $\rho$ is a function from $\Sigma^* \times \Gamma^*$ to $\Gamma$. This is true because a probabilistic prover can be simulated by a deterministic prover where the verifier provides the prover's coin tosses. (We include probabilistic provers in the general definition since it is not clear that probabilistic provers can be excluded, without loss of generality, when studying zero knowledge IPS's, as in [7] for example.)

In defining verifiers, we could have used distributions which map to the rational numbers in $[0, 1]$. It is easy to see that coin tosses can simulate rational transition probabilities while increasing the expected running time of the verifier by at most a constant factor.

It should be pointed out that the particular examples of interactive proof systems in this paper do not use the full power of the definition of an IPS above. This happens in two ways. First, the verifier always halts with probability 1. Second, although it is convenient to define the prover to be "all powerful" (for example, the function $\rho$ need not be recursive), it is easy to place upper bounds on the time and space used by the prover in particular cases. For most of our specific interactive proofs, the prover can be a simple device such as a deterministic 2-way finite state machine or a logspace Turing machine. In many cases the only role of the prover is to send a certain string $w$ to the verifier, perhaps repeatedly (in general, $w$ depends on the particular input $x$). Since this type of prover arises several times in the sequel, we say more precisely what it means for the *prover to send a string* $w$ *to the verifier.* Let $\Delta$ be the finite alphabet such that $w \in \Delta^*$ and assume $\not c \notin \Delta$ so that $\not c$ can be used as an endmarker. The communication alphabet includes the symbols in $\Delta \cup \{1, \not c\}$. Starting with $i = 1$, whenever the verifier sends a "1" to the prover, the prover responds with the $i$th symbol of $w\not c$ and sets $i \leftarrow i + 1$.

On the other hand, our impossibility results, proved in Sections 3.2 and 3.3, apply to wider types of IPS's. For example, they remain valid if coin-tossing distributions are generalized to distributions which map into the entire real interval $[0, 1]$. The impossibility results also apply under a weaker definition of an IPS where the rejecting condition, 2 above, is replaced by:

2$'$. for all $x \notin L$ and all provers $P^*$, $(P^*, V)(x)$ accepts with probability at most $\varepsilon$.

We use the notation $\mathrm{IP}_w$ for this weaker definition. If $\mathcal{V}$ is any class of verifiers which halt with probability 1, then obviously $\mathrm{IP}(\mathcal{V}) = \mathrm{IP}_w(\mathcal{V})$. For other classes, there can be a difference. Specifically, Lipton [19] shows that $\mathrm{IP}_w$(2pfa) contains precisely the recursively enumerable languages, whereas Condon and Lipton [6] show that every language in IP(2pfa) is recursive. So as not to have too many different notations in the paper, we state all of our results in terms of IP.

When describing a particular 2pfa which flips a fair coin, we let 0 and 1 denote the two possible outcomes, where each outcome has probability 1/2.

A *sweeping 2pfa* is a 2pfa restricted so that the input head can switch its direction of movement only when scanning an endmarker $\not c$.

We let $|S|$ denote the cardinality of the set $S$, and we let $|w|$ denote the length of the word $w$. Logarithms in this paper are to the base 2.

## 3. IP(2pfa)

This section contains several results about the power of interactive proof systems with a 2pfa verifier. It is immediate from definitions that

$$\mathrm{2PFA} \subseteq \mathrm{AM(2pfa)} \subseteq \mathrm{IP(2pfa)}.$$

We show that both of these containments are proper. Moreover, proper containment in both cases is witnessed by a fairly simple language, in particular, a context-free language. The

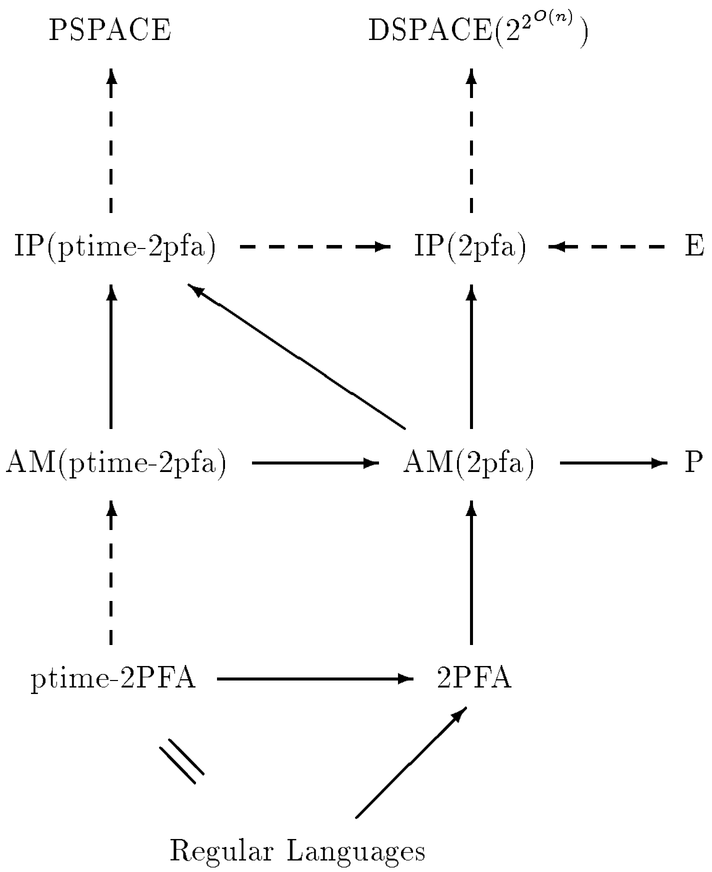


Figure 1: Inclusions among classes. Solid arrows denote proper inclusions. Dashed arrows denote inclusions which are not known to be proper.

second containment remains proper if the 2pfa verifier must halt in polynomial expected time.

Our present knowledge is summarized in Figure 1. In this figure, a solid arrow from class $\mathcal{C}$ to class $\mathcal{D}$ means that $\mathcal{C}$ is properly contained in $\mathcal{D}$. A dashed arrow from $\mathcal{C}$ to $\mathcal{D}$ means that $\mathcal{C} \subseteq \mathcal{D}$ although the containment is not known to be proper. P, PSPACE, and E denote the class of languages recognized by deterministic Turing machines in polynomial time, polynomial space, and time $2^{O(n)}$, respectively. Some of the results shown in Figure 1 are from other papers. In a separate paper [8] we show that ptime-2pfa's recognize only regular languages; this result was also obtained by Kaneps and Freivalds [15]. Results of Condon and Ladner [5] imply that AM(2pfa) $\subseteq$ P and IP(ptime-2pfa) $\subseteq$ PSPACE. Freivalds [9] shows that there is a nonregular language in 2PFA. Condon and Lipton [6] show that IP(2pfa) is contained in the class DSPACE($2^{2^{O(n)}}$) of languages recognized by Turing machines within double-exponential space.

We begin in Section 3.1 with a simple example of an interactive proof system for the palindrome language. In Section 3.2 we prove technical results which can be used to show that certain languages are not in 2PFA, AM(2pfa), or their ptime versions. In Section 3.3 these results are applied to prove various proper containments. In Section 3.4 we investigate the general power of IP(2pfa) and show that this class contains any language which can be recognized in deterministic exponential time. In this section we also give results on the power of IPS's with 2pfa verifiers restricted to halt in polynomial expected time.

### 3.1. An Example

If $x$ is a string, let $x^R$ be $x$ written backwards. Define the set of palindromes:

$$Pal = \{\, x \in \{0,1\}^* \mid x = x^R \,\}.$$

We describe an IPS $(P, V)$ for $Pal$ with error probability $\varepsilon$ for any constant $\varepsilon > 0$. If $x$ is a palindrome, the interaction involves $k$ iterations, where $k = \lceil \log_2(1/\varepsilon) \rceil$. On each iteration, the prover $P$ sends $x$ to the verifier. At the start of each iteration, the verifier $V$ (privately) tosses a fair coin. Letting $w$ denote the string received from the prover during this iteration, if the outcome of the coin toss is 0 then $V$ checks that $w = x$ and rejects if not. If the outcome is 1 then $V$ checks that $w = x^R$ and rejects if not. If the check succeeds for all $k$ iterations, then $V$ accepts.

If $x$ is a palindrome, then obviously $(P, V)(x)$ accepts with probability 1. Suppose that $x$ is not a palindrome and let $P^*$ be any prover. At any iteration, if $w$ is the string which $P^*$ sends to $V$, then either $w \neq x$ or $w \neq x^R$ (or both), so $V$ will reject during this iteration with probability at least $1/2$. Therefore, the probability that $V$ does not reject during any of the $k$ iterations is at most $2^{-k} \leq \varepsilon$.

We have thus shown:

**Theorem 3.1.** *$Pal \in$ IP(2pfa). Moreover, for any error probability $\varepsilon > 0$, there is an IPS for Pal where the verifier is a sweeping 2pfa which runs in worst-case time $O(n)$.*

Note in the IPS above that a cheating $P^*$ could easily cause $V$ to accept a non-palindrome if the verifier's coins were public. We show in Section 3.3 that there is no IPS for $Pal$ using public coins.

### 3.2. Impossibility Results

The results in this section are based on a simple idea which has been used previously by Rabin [21] and Greenberg and Weiss [13] to analyze probabilistic finite state automata. Let $M$ be a 2pfa with accepting state $q_a$ and rejecting state $q_r$, and let $w$ be a word over the input alphabet of $M$. We define the *word probabilities of $M$ on $w$* as follows. A *starting condition* is a pair $(q, \xi)$ where $q$ is a state of $M$ and $\xi \in \{\text{Left}, \text{Right}\}$; its intuitive meaning is "start $M$ on the $\xi$ end of $w$ in state $q$." A *stopping condition* is either

1. a pair $(q, \xi)$ as above, meaning "the head of $M$ falls off the $\xi$ end of $w$ with $M$ in state $q$,"

2. "Loop", meaning "the computation of $M$ loops forever within $w$,"

3. "Accept", meaning "$M$ halts in state $q_a$ before the head falls off either end of $w$," or

4. "Reject", meaning "$M$ halts in state $q_r$ before the head falls off either end of $w$."

For each starting condition $\sigma$ and each stopping condition $\tau$, let $p(w, \sigma, \tau)$ be the probability that stopping condition $\tau$ occurs given that $M$ is started in starting condition $\sigma$ on $w$. If $W$ is a large set of relatively short words, then a pigeonhole argument shows that there must be two words $w$ and $w'$ in $W$ which $M$ cannot distiguish, i.e., for each $\sigma$ and $\tau$, the probabilities $p(w, \sigma, \tau)$ and $p(w', \sigma, \tau)$ are very close. If $L$ is a language such that for any two different words $w$ and $w'$ in $W$ there is a word $u$ such that $wu \in L$ iff $w'u \notin L$, then $M$ does not recognize $L$. Under stronger conditions on $L$, the argument can be extended to show that $L$ is not in AM(2pfa).

Since we model computations of 2pfa's by Markov chains, we first give some definitions and results about Markov chains. Basic facts about Markov chains can be found, for example, in [23]. We consider Markov chains with finite state space, say $\{1, 2, \ldots, s\}$ for some $s$. A particular Markov chain is completely specified by its matrix $R = \{r_{ij}\}_{i,j=1}^{s}$ of transition probabilities. If the Markov chain is in state $i$, then it next moves to state $j$ with probability $r_{ij}$. The chains we consider have a designated starting state, say, state 1, and some set $T$ of trapping states, so $r_{kk} = 1$ for all $k \in T$. For $k \in T$, let $a(k, R)$ denote the probability that the Markov chain $R$ is trapped in state $k$ when started in state 1.

We start with a lemma which bounds the effect of small changes in the transition probabilites of a Markov chain. This lemma is a slight restatement of Lemma 1 from [13]. Let $\beta \geq 1$. Say that two numbers $r$ and $r'$ are *$\beta$-close* if either (i) $r = r' = 0$ or (ii) $r > 0$, $r' > 0$, and $\beta^{-1} \leq r/r' \leq \beta$. Two Markov chains $R = \{r_{ij}\}_{i,j=1}^{s}$ and $R' = \{r'_{ij}\}_{i,j=1}^{s}$ are *$\beta$-close* if $r_{ij}$ and $r'_{ij}$ are $\beta$-close for all pairs $i, j$.

**Lemma 3.2.** *Let $R$ and $R'$ be two $s$-state Markov chains which are $\beta$-close, and let $k$ be a trapping state of both $R$ and $R'$. Then $a(k, R)$ and $a(k, R')$ are $\beta^z$-close where $z = 2s$.*

*Proof.* It follows from the Markov chain tree theorem (see Leighton and Rivest [17, 18] or Broder [2]) that $a(k, R) = P/Q$ where $P$ and $Q$ are polynomials of degree at most $s$ in the nonzero $r_{ij}$ and $P$ and $Q$ have nonnegative coefficients, and $a(k, R') = P'/Q'$ where $P'$ is identical to $P$ and $Q'$ is identical to $Q$ except that $r_{ij}$ is replaced everywhere by $r'_{ij}$. Now if $m$ is any monomial of $P$ and $m'$ is the corresponding monomial of $P'$, then $m$ and $m'$ are $\beta^s$-close since $P$ and $P'$ are of degree at most $s$. Since the coefficients of $P$ and $P'$ are nonnegative, it follows that $P$ and $P'$ are $\beta^s$-close. Similarly, $Q$ and $Q'$ are $\beta^s$-close. The conclusion follows immediately. $\square$

We now state and prove an impossibility result for 2pfa's.

**Theorem 3.3.** *Let $A, B \subseteq \Sigma^*$ with $A \cap B = \emptyset$. Suppose there is an infinite set $I$ of positive integers and, for each $m \in I$, a set $W_m \subseteq \Sigma^*$ such that*

1. $|w| \leq m$ *for all* $w \in W_m$,

2. *for every integer* $k$ *there is an* $m_k$ *such that* $|W_m| \geq m^k$ *for all* $m \in I$ *with* $m \geq m_k$, *and*

3. *for every* $m \in I$ *and every* $w, w' \in W_m$ *with* $w \neq w'$, *there are words* $u, v \in \Sigma^*$ *such that either* $uwv \in A$ *and* $uw'v \in B$, *or* $uwv \in B$ *and* $uw'v \in A$.

*Then no 2pfa separates* $A$ *and* $B$.

*Proof.* Suppose to the contrary that the 2pfa $M$ separates $A$ and $B$ with error probability $\varepsilon < 1/2$. Let $c$ be the number of states of $M$.

Consider the word probabilities $p(w, \sigma, \tau)$ defined above. For each word $w$ there are a total of $d = 4c^2 + 6c$ word probabilities. Fix some ordering of the pairs $(\sigma, \tau)$ of starting and stopping conditions and let $\mathbf{p}(w)$ be the vector of these $d$ probabilities according to this ordering.

We first show that if $w \in W_m$ and if $p$ is a nonzero element of $\mathbf{p}(w)$, then $p \geq 2^{-cm}$. Form a Markov chain $K(w)$ with states of the form $(q, l)$ where $q$ is a state of $M$ and $0 \leq l \leq |w| + 1$. The chain state $(q, l)$ with $1 \leq l \leq |w|$ corresponds to $M$ being in state $q$ scanning the $l$th symbol of $w$. Transition probabilities from such states are obtained from the transition probabilities of $M$ in the obvious way. For example, if the $l$th symbol of $w$ is 0, and if $M$ in state $q$ reading a 0 can move its head left and enter state $q'$ with probability 1/2, then the transition probability from state $(q, l)$ to state $(q', l - 1)$ is 1/2. Chain states of the form $(q, 0)$ and $(q, |w| + 1)$ are trap states of $K(w)$ and correspond to the head of $M$ falling off the left or right end, respectively, of $w$. Now consider, for example, $p = p(w, \sigma, \tau)$ where $\sigma = (q_i, \text{Left})$ and $\tau = (q_j, \text{Left})$. If $p > 0$, then there must be some path of nonzero probability in $K(w)$ from state $(q_i, 1)$ to $(q_j, 0)$, and since $K(w)$ has at most $cm$ nontrapping states, there is such a path of length at most $cm$. Since 1/2 is the smallest nonzero transition probability of $M$, it follows that $p \geq 2^{-cm}$. The other three cases $p(w, \sigma, \tau)$ where $\tau$ has the form $(q, \xi)$ are similar. If $\sigma = (q_i, \text{Left})$ and $\tau = \text{Loop}$, there must be a path of nonzero probability in $K(w)$ from state $(q_i, 1)$ to some state $(q_j, l)$ such that there is no path of nonzero probability from $(q_j, l)$ to any trap state of the form $(q, 0)$ or $(q, |w| + 1)$. Again, if there is such a path, there is one of length at most $cm$. The remaining cases are similar.

Fix an arbitrary $m \in I$. Divide $W_m$ into equivalence classes by making $w$ and $w'$ equivalent if $\mathbf{p}(w)$ and $\mathbf{p}(w')$ are zero in exactly the same coordinates. Let $E_m$ be a largest equivalence class, so $|E_m| \geq |W_m|/2^d$.

Let $d'$ be the number of nonzero coordinates of $\mathbf{p}(w)$ for $w \in E_m$. Let $\hat{\mathbf{p}}(w)$ be the $d'$-dimensional vector of nonzero coordinates of $\mathbf{p}(w)$. Note that $\hat{\mathbf{p}}(w) \in [2^{-cm}, 1]^{d'}$ for all $w \in E_m$. Let $\log \hat{\mathbf{p}}(w)$ be the componentwise log of $\hat{\mathbf{p}}(w)$, so $\log \hat{\mathbf{p}}(w) \in [-cm, 0]^{d'}$.

By dividing each coordinate interval $[-cm, 0]$ into subintervals of length $\mu$, we divide the space $[-cm, 0]^{d'}$ into at most $(cm/\mu)^d$ *cells*, each of size $\mu \times \mu \times \ldots \times \mu$. We want to choose $\mu$ large enough that the number of cells is smaller than the size of $E_m$, i.e.,

$$\left(\frac{cm}{\mu}\right)^d < \frac{|W_m|}{2^d}. \tag{1}$$

Since $|W_m|$ is assumed to grow faster than any polynomial in $m$ and since $d$ is a constant depending only on $M$, for any $\mu > 0$ there is an $m_\mu$ such that (1) holds for all $m \in I$ with $m \geq m_\mu$.

Assuming (1), there must be two different words $w, w' \in E_m$ such that $\log \hat{\mathbf{p}}(w)$ and $\log \hat{\mathbf{p}}(w')$ belong to the same cell. Therefore, if $p$ and $p'$ are two nonzero probabilities in the same coordinate of $\mathbf{p}(w)$ and $\mathbf{p}(w')$, respectively, then

$$|\log p - \log p'| \leq \mu.$$

It follows that $p$ and $p'$ are $2^\mu$-close. Therefore, $\mathbf{p}(w)$ and $\mathbf{p}(w')$ are componentwise $2^\mu$-close.

For this $w$ and $w'$, let $u$ and $v$ be the words in Assumption 3 in the statement of the theorem. We describe two Markov chains, $R$ and $R'$, which model the computation of $M$ on $uwv$ and $uw'v$, respectively. Both chains have $4c+4$ states. $4c$ of the states have the form $(q, l)$ where $q$ is a state of $M$ and $1 \leq l \leq 4$. The other states are Initial, Accept, Reject, and Loop. The state $(q, l)$ of $R$ corresponds to $M$ being in state $q$ reading the right end of $\cent u$ if $l = 1$, the left end of $w$ if $l = 2$, the right end of $w$ if $l = 3$, or the left end of $v\cent$ if $l = 4$. The state Initial corresponds to $M$ being in its initial state $q_0$ reading the leftmost endmarker $\cent$, the states Accept and Reject correspond to the computation halting in the accepting state or the rejecting state, respectively, and Loop means that $M$ has entered an infinite loop. The transition probabilites of $R$ are obtained from the word probabilities of $M$ on $\cent u$, $w$, and $v\cent$. For example, the transition probability from $(q_i, 3)$ to $(q_j, 1)$ is just $p(w, (q_i, \text{Right}), (q_j, \text{Left}))$, the transition probability from Initial to $(q_j, 2)$ is $p(\cent u, (q_0, \text{Left}), (q_j, \text{Right}))$, and the transition probability from $(q_i, 4)$ to Accept is $p(v\cent, (q_i, \text{Left}), \text{Accept})$. The states Accept, Reject, and Loop are trap states of $R$. The chain $R'$ is defined similarly, but using $w'$ in place of $w$.

Suppose that $uwv \in A$ and $uw'v \in B$, the other case being symmetric. Let $z = 2(4c+4)$. Let $a$ (resp., $a'$) be the probability that $M$ accepts input $uwv$ (resp., $uw'v$). Then $a$ (resp., $a'$) is exactly the probability that the Markov process $R$ (resp., $R'$) is trapped in state Accept when started in state Initial. Now $uwv \in A$ implies $a \geq 1 - \varepsilon$. Since $R$ and $R'$ are $2^\mu$-close, Lemma 3.2 implies that

$$a'/a \geq 2^{-\mu z}$$

which implies

$$a' \geq (1-\varepsilon)2^{-\mu z}.$$

Since $\varepsilon < 1/2$ and $z$ is constant, for sufficiently large $m$ we can choose $\mu$ so that (1) holds and $\mu$ is so small that $a' > 1/2$. But since $uw'v \in B$, this contradicts the assumption that $M$ separates $A$ and $B$. □

The next result, proved in a very similar way, can be used to show that certain languages are not in AM(2pfa). The conditions on the language must be strengthened, however. We will see in the next section that there is a language in AM(2pfa) which satisfies the conditions of Theorem 3.3.

**Theorem 3.4.** *Let $L \subseteq \Sigma^*$. Suppose there is an infinite set $I$ of positive integers and, for each $m \in I$, an integer $N(m)$ and 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. $|w| \le m$ *for all* $w \in W_m$,

2. *for every integer* $k$ *there is an* $m_k$ *such that* $N(m) \ge m^k$ *for all* $m \in I$ *with* $m \ge m_k$, *and*

3. *for all* $1 \le i, j \le N(m)$, $u_j w_i v_j \in L$ *iff* $i = j$.

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

*Proof.* Suppose to the contrary that there is an IPS $(P, V)$ for $L$ with error probability $\varepsilon < 1/2$ using public coins. We can assume without loss of generality that the prover $P$ uses a *Markov transition function* (sometimes called a *Markov strategy*): for each $x \in L$, whenever the verifier $V$ writes a symbol in the communication cell at some step, the response of $P$ depends (deterministically) only on $x$ and the configuration of $V$, i.e., the state and head position of $V$ at this step. A formal justification that there is no harm in restricting attention to Markov strategies is given by Condon [4, Thm. 2.6].

We define word probabilities on words in $W_m$, $U_m$, and $V_m$ as follows. For each $i$ with $1 \le i \le N(m)$, consider the Markov strategy $S_i$ which $P$ uses on input $u_i w_i v_i$. Since $u_i w_i v_i \in L$, this strategy causes $V$ to halt in its accepting state with probability at least $1 - \varepsilon$. Since $S_i$ is a Markov strategy, it is easy to see that the word probabilities of $(P, V)$ on $\cent u_i$, on $w_i$, and on $v_i \$$ are, in fact, well-defined. (If the prover were not using a Markov strategy, they would not be well-defined. The notion of a starting condition would have to be extended to include also the communication history, since in general the prover's responses depend on the communication history. For Markov strategies, however, this history is irrelevant.) Formally, we use a Markov chain $K(w_i)$, very similar to the Markov chain $K(w)$ in the proof of Theorem 3.3, to model the computation of $(P, V)$ on $w_i$. States of this chain are of the form $(q, l)$ where $q$ is a state of $V$ and $0 \le l \le |w_i| + 1$. Transition probabilities from states $(q, l)$ where $q$ is a reading state are obtained from the transition probabilities of $V$ as before. If $q$ is a communication state and $1 \le l \le |w_i|$, the prover's response according to strategy $S_i$ depends only on the configuration $(q, l)$ of $V$, so transition probabilities from $(q, l)$ are determined by $S_i$ and the transition probabilities of $V$. Now, for example, the word probability $p(w_i, (q_k, \mathrm{Right}), (q_j, \mathrm{Right}))$ is the probability that $K(w_i)$ is trapped in state $(q_j, |w_i| + 1)$ when started in state $(q_k, |w_i|)$. Word probabilities for $\cent u_i$ and $v_i \$$ are defined similarly.

Arguing as in the proof of Theorem 3.3, for every $\mu > 0$ there is an $m_\mu$ such that, for all $m \in I$ with $m \ge m_\mu$, there must be $w_i, w_j \in W_m$ with $i \ne j$ such that the vectors of word probabilities $\mathbf{p}(w_i)$ and $\mathbf{p}(w_j)$ are componentwise $2^\mu$-close.

Consider a prover $P^*$ which, on input $u_i w_j v_i$, uses strategy $S_j$ when $V$ is reading a symbol of $w_j$ and uses strategy $S_i$ when $V$ is reading a symbol of $\cent u_i$ or $v_i \$$. Similar to the proof of Theorem 3.3, we can construct a Markov chain $R$ which models the computation of $(P, V)$ on input $u_i w_i v_i$ and a Markov chain $R'$ which models the computation of $(P^*, V)$ on input $u_i w_j v_i$. Since $(P, V)$ accepts $u_i w_i v_i$ with probability at least $1 - \varepsilon$, since $\mathbf{p}(w_i)$ and $\mathbf{p}(w_j)$ are $2^\mu$-close, and since $P^*$ uses the same strategy as $P$ when $V$ is reading $\cent u_i$ or $v_i \$$, it is easy to argue as before that $(P^*, V)$ accepts $u_i w_j v_i$ with probability $> 1/2$. Since $u_i w_j v_i \notin L$, this contradicts the assumption that $(P, V)$ is an IPS for $L$. $\square$

With a time restriction, these impossibility results can easily be modified to apply to a larger class of languages and separation problems. We first give the modification of Theorem 3.3. The proof of this result, for language recognition rather than separation, is given in [8, Lemma 4.3]. We sketch the proof here.

**Theorem 3.5.** *Let $A, B \subseteq \Sigma^*$ with $A \cap B = \emptyset$, and let $T(n)$ be a nondecreasing unbounded function. Suppose there is an infinite set $I$ of positive integers and, for each $m \in I$, a set $W_m \subseteq \Sigma^*$ such that*

1. *for every integer $k$ there is an $m_k$ such that $|W_m| \geq (\log T(m))^k$ for all $m \in I$ with $m \geq m_k$, and*
2. *for every $m \in I$ and every $w, w' \in W_m$ with $w \neq w'$, there are words $u, v \in \Sigma^*$ such that*
   - *(a) $|uwv| \leq m$ and $|uw'v| \leq m$, and*
   - *(b) either $uwv \in A$ and $uw'v \in B$, or $uwv \in B$ and $uw'v \in A$.*

*Then no 2pfa running in expected time $T(n)$ separates $A$ and $B$.*

*Proof.* Suppose that $A$ and $B$ are separated by the 2pfa $M$ which halts in expected time $T(n)$. Define word probabilities for words in $W_m$ as before. Say that a probability $p$ is *small* if $p < T(m)^{-2}$; otherwise $p$ is *large*. When dividing $W_m$ into equivalence classes, make $w$ and $w'$ equivalent if $\mathbf{p}(w)$ and $\mathbf{p}(w')$ are small in exactly the same coordinates. Again let $E_m$ be a largest equivalence class. For $w \in E_m$, let $\hat{\mathbf{p}}(w)$ be $\mathbf{p}(w)$ restricted to the coordinates containing large probabilities, and let $\log \hat{\mathbf{p}}(w)$ be the componentwise log of $\hat{\mathbf{p}}(w)$. Since the logarithm of any large word probability lies in the interval $[-2\log T(m), 0]$, the analogue of (1) is

$$\left(\frac{2\log T(m)}{\mu}\right)^d < \frac{|W_m|}{2^d}.$$

Since $|W_m|$ grows faster than any polynomial in $\log T(m)$, we can make $\mu$ arbitrarily close to zero as before. Find $w$ and $w'$ such that $\hat{\mathbf{p}}(w)$ and $\hat{\mathbf{p}}(w')$ are componentwise $2^\mu$-close.

The rest of the argument is similar to part of the proof of Theorem 3 in [13]. Model the computation of $M$ on inputs $uwv$ and $uw'v$ by Markov chains $R$ and $R'$, respectively, as before. If $p$ and $p'$ are corresponding transition probabilities in $R$ and $R'$, respectively, then either $p$ and $p'$ are $2^\mu$-close or $p$ and $p'$ are both small. Let $\mathcal{E}$ be the event that, when $R$ is started in state Initial, it is trapped in state Accept or Reject before any transition labeled with a small probability is taken; define event $\mathcal{E}'$ similarly for $R'$. Since $M$ halts in expected time at most $T(m)$ on the inputs $uwv$ and $uw'v$, it is obvious that the probabilities of $\mathcal{E}$ and $\mathcal{E}'$ go to 1 as $m$ increases. Therefore, by changing all small probabilities to zero, we do not significantly change the probabilities that $R$ and $R'$ enter the Accept state, provided that $m$ is sufficiently large. After these changes, we can argue as before that the probabilities of $M$ accepting $uwv$ and $uw'v$ are very close. $\square$

The extension of this argument to AM(ptime-2pfa) is done exactly as in the proof of Theorem 3.4, thus proving the following. For simplicity, we state this result only for polynomial time bounds.

**Theorem 3.6.** *Let $L \subseteq \Sigma^*$. Suppose there is an infinite set $I$ of positive integers and, for each $m \in I$, an integer $N(m)$ and sets $W_m = \{w_1, w_2, \dots, w_{N(m)}\}$, $U_m = \{u_1, u_2, \dots, u_{N(m)}\}$ and $V_m = \{v_1, v_2, \dots, v_{N(m)}\}$ of words such that*

1. *$|u_i w_j v_i| \le m$ for all $i$ and $j$ with $1 \le i, j \le N(m)$,*
2. *for every integer $k$ there is an $m_k$ such that $N(m) \ge (\log m)^k$ for all $m \in I$ with $m \ge m_k$, and*
3. *for all $1 \le i, j \le N(m)$, $u_j w_i v_j \in L$ iff $i = j$.*

*Then $L \notin$ AM(ptime-2pfa).*

### 3.3. Private Coins vs. Public Coins vs. 2PFA

It follows easily from Theorem 3.4 that

$$Pal \notin \mathrm{AM(2pfa)}.$$

Specifically, fix any ordering $w_1, w_2, \dots, w_{N(m)}$ of the words in $\{0,1\}^m$, where $N(m) = 2^m$, and take $v_i = w_i^R$ and $u_i = \lambda$ (the empty word) for all $i$. Together with Theorem 3.1, this shows:

**Theorem 3.7.** *IP(2pfa) $\neq$ AM(2pfa). Moreover, this inequality remains true when verifiers are 2pfa's which run in polynomial expected time or polynomial worst-case time.*

To show that AM(2pfa) $\neq$ 2PFA, we use the language *Center*, defined as the set of binary words of odd length which have the symbol 1 in the center:

$$Center = \{\, u1v \mid u, v \in \{0,1\}^* \text{ and } |u| = |v| \,\}.$$

**Theorem 3.8.** *Center $\notin$ 2PFA and Center $\in$ AM(2pfa).*
*Therefore, AM(2pfa) $\neq$ 2PFA.*

*Proof.* *Center* $\notin$ 2PFA. We use Theorem 3.3 with $A =$ *Center* and $B =$ the complement of *Center*. Let $W_m = \{0,1\}^m$. For any $w, w' \in W_m$ with $w \neq w'$, we can write $w = ybz$ and $w' = y'\bar{b}z'$ where $|y| = |y'|$, $|z| = |z'|$, $b \in \{0,1\}$, and $\bar{b} = 1 - b$. For any $u$ and $v$ with $|u| = |z|$ and $|v| = |y|$, we have $uwv \in$ *Center* iff $uw'v \notin$ *Center*. Since $|W_m| = 2^m$, all the conditions of Theorem 3.3 are satisfied. Therefore, no 2pfa separates *Center* from its complement.

*Center* $\in$ AM(2pfa). The method is similar to the one used by Freivalds [9] to show that the language $\{\, 0^m 1^m \mid m \ge 1 \,\}$ is recognized by a 2pfa.

The protocol consists of a number of iterations. On input $x \in$ *Center* with $|x| = 2m+1$ for integer $m$, the prover $P$ sends $1^{m+1}$ to the verifier at each iteration.

The verifier's algorithm depends on a positive integer parameter $k$ to be chosen later. On input $x$, the verifier $V$ first checks that $n = |x|$ is odd and rejects if not. Assuming $n = 2m+1$, the protocol proceeds for a number of iterations. At each iteration, the verifier uses the string $1^j$ received from the prover to position its head on the $j$th symbol of $x$. If this symbol is not 1, or if $j = 0$ or $j > n$, then $V$ rejects. Otherwise, $V$ has its head positioned on a 1 in $x$ with $a = j - 1$ symbols of $x$ to the left of the head and $b = n - j$ symbols of $x$ to the right of the head. $V$ now (publically) flips a fair coin; let $c$ be the outcome. If $c = 0$, then $V$ flips a fair coin $k(2a + 2)$ times, moving its head left to keep count. If $c = 1$, then $V$ flips a fair coin $k(2b + 2)$ times, moving its head right to keep count. In either case, if all $k(2a + 2)$ or $k(2b + 2)$ flips have outcome 0, then $V$ rejects. Otherwise, $V$ then flips a coin $kn$ times. If all $kn$ outcomes are 0 then $V$ accepts. If $V$ neither accepts nor rejects on this iteration, it proceeds to another iteration.

We argue that this protocol is an IPS for *Center*. Consider first the case that $x \in$ *Center* with $P$ as the prover. At each iteration, $a = b = m$. Therefore, restricting attention to $V$'s probability of accepting or rejecting at any single iteration,

$$\Pr[\,V \text{ rejects}\,] = 2^{-k(2m+2)}$$

and

$$\Pr[\,V \text{ accepts}\,] = \Pr[\,V \text{ does not reject}\,] \cdot 2^{-kn} \geq 2^{-1}2^{-kn} = 2^{-k(2m+1)-1},$$

so

$$\Pr[\,V \text{ accepts}\,] \geq 2^{k-1} \cdot \Pr[\,V \text{ rejects}\,].$$

Therefore, the probability that $V$ accepts before it rejects approaches 1 as $k$ increases. Also, $V$ eventually either accepts or rejects with probability 1. Therefore, the probability that $V$ makes an error in this case approaches zero as $k$ increases.

Consider now the case that $x \notin$ *Center*, and let $P^*$ be an arbitrary prover. Consider an arbitrary iteration. Assuming that $n = 2m + 1$ and that $P^*$ sends $1^j$ to the verifier where the $j$th symbol of $x$ is a 1 (for otherwise, $V$ rejects), we have that either $a \leq m - 1$ or $b \leq m - 1$. Therefore, either $2a + 2 \leq n - 1$ or $2b + 2 \leq n - 1$, and whichever case holds, $V$ will choose this case with probability 1/2. Therefore, again restricting attention to a single iteration,

$$\Pr[\,V \text{ rejects}\,] \geq 2^{-1}2^{-k(n-1)} \quad \text{and} \quad \Pr[\,V \text{ accepts}\,] \leq 2^{-kn},$$

so

$$\Pr[\,V \text{ rejects}\,] \geq 2^{k-1} \cdot \Pr[\,V \text{ accepts}\,].$$

Therefore, the probability that $V$ rejects before it accepts approaches 1 as $k$ increases. □

In the IPS used to prove that *Center* $\in$ AM(2pfa), the verifier halts with probability 1. The expected running time is exponential in $n$, however.

Theorem 3.6 can be applied to show that *Center* is not in AM(ptime-2pfa), and a similar argument works for the "unary palindrome" language defined by

$$Upal = \{\, 0^n 1^n \mid n \geq 1 \,\}.$$

This yields the following two noncontainments.

**Theorem 3.9.** *AM(2pfa) $\not\subseteq$ AM(ptime-2pfa),*
*2PFA $\not\subseteq$ AM(ptime-2pfa).*

*Proof.* We first show that *Center* $\notin$ AM(ptime-2pfa). Given $m$, let $u_i = \lambda$, $w_i = 0^i$, and $v_i = 10^i$ for $1 \leq i \leq N(m)$, where $N(m) = \lfloor (m-1)/2 \rfloor$. Since $u_j w_i v_j \in$ *Center* iff $i = j$, it follows from Theorem 3.6 that *Center* $\notin$ AM(ptime-2pfa). The argument for *Upal* is similar; take $v_i = 1^i$. We have shown in Theorem 3.8 that *Center* $\in$ AM(2pfa), and Freivalds [9] shows that *Upal* $\in$ 2PFA. □

Even though the main subject of this paper concerns 2pfa verifiers, it is worth mentioning that the proper inclusions of Theorems 3.7 and 3.8 apply also to interactive proof systems where the verifier is a space-bounded probabilistic Turing machine (ptm) where the space bound $S(n)$ is *sublogarithmic*, i.e., $\lim S(n)/\log n = 0$. Such ptm's have a read-only input tape and a space $S(n)$ bounded read/write worktape; each tape is scanned by a single 2-way head. Let BPSPACE($S$) be the class of languages recognized by ptm's with space bound $S(n)$, and let IP(space-$S$-ptm) (AM(space-$S$-ptm)) be the class of languages having interactive proof systems (using public coins) where the verifier is a space $S(n)$ bounded ptm. Definitions are analogous to those in §2; in particular, the error probability is assumed to be bounded below 1/2 in all cases.

**Theorem 3.10.** *If $S(n)$ is sublogarithmic then*

$$\text{IP(space-}S\text{-ptm)} \neq \text{AM(space-}S\text{-ptm)},$$
$$\text{AM(space-}S\text{-ptm)} \neq \text{BPSPACE}(S).$$

*Proof.* It suffices to show that

$$\textit{Center} \notin \text{BPSPACE}(S) \text{ and } \textit{Pal} \notin \text{AM(space-}S\text{-ptm)}$$

if $S(n)$ is sublogarithmic. We sketch the necessary modifications to the proofs of Theorems 3.3 and 3.4.

We view a space $S(n)$ ptm as a 2pfa with a variable number of states; on inputs of length $n$, the 2pfa has $c(n) = b^{S(n)}$ states for some constant $b$. These states correspond to all the possible combinations of the state of the finite control, the worktape contents, and the position of the head on the worktape. If $S(n)$ is sublogarithmic then, for any constant $a > 0$, $c(n) < n^a$ for almost all $n$. The number of word probabilities is now $d(n) = O(c(n)^2)$, so $d(n)$ also grows more slowly than $n^a$ for any constant $a > 0$. If $p$ is a nonzero word probability in $\mathbf{p}(w)$ where $|w| \leq n$, then $p \geq 2^{-c(n)n}$, so $\log p$ lies in the interval $[-c(n)n, 0]$. In the application of Theorems 3.3 and 3.4 to *Center* and *Pal*, we have $|W_m| = 2^m \geq 2^{n/2}$. Putting all of this together, the analogue of (1) is

$$\left(\frac{c(n)n}{\mu}\right)^{d(n)} < \frac{2^{n/2}}{2^{d(n)}}.$$

Taking $\mu = 2^{-\sqrt{n}}$, this inequality is satisfied for all sufficiently large $n$. The number $z$ used in the application of Lemma 3.2 is now $O(c(n))$, so $2^{-\mu z}$ approaches 1 as $n$ increases. In the case of *Center*, for example, for sufficiently large $n$ we can force the space $S(n)$ bounded ptm to accept a word not in *Center* with probability at least $2^{-\mu z}(1-\varepsilon) \geq 1/2$. □

Regarding space bounds at least $\log n$, Condon and Ladner [5] show that for any space constructible $S(n) \geq \log n$,

$$\text{AM(space-}S\text{-ptm)} \subseteq \text{DTIME}(2^{O(S(n))}) \tag{2}$$

where DTIME($T(n)$) is the class of languages recognized by deterministic Turing machines within time $T(n)$. The next result follows immediately from this, from Theorem 3.13 in the next section, and from the hierarchy theorem for time complexity [14].

**Theorem 3.11.** *For any space constructible $S(n) \geq \log n$,*

$$IP(\textit{space-}S\textit{-ptm}) \neq AM(\textit{space-}S\textit{-ptm}).$$

Theorem 3.11 was proved independently by Condon [4]. Showing proper inclusion of BPSPACE($S$) in AM(space-$S$-ptm) for $S(n) \geq \log n$ will be difficult, given (2). For example, proving this proper inclusion for $S(n) = \log n$ will require showing that there is a language in P which cannot be recognized by a ptm in space $\log n$.

### 3.4. The Power of IP(2pfa)

Let E denote the class of languages recognized by deterministic Turing machines within time $2^{cn}$ for some constant $c$. We show that E is contained in IP(2pfa). One interesting consequence of this result is that adding a prover to a 2pfa verifier causes a much greater increase in the power of the verifier than adding a prover to a polynomial-time ptm verifier. It follows from (2) that 2PFA $\subseteq$ P, and Theorem 3.8 shows that this inclusion is proper. Therefore, without a prover, 2PFA is properly contained in the class of languages accepted by polynomial-time ptm's. With the addition of a prover, the situation is different, assuming that E is not contained in PSPACE. It is known that IP(ptime-ptm) $\subseteq$ PSPACE. Therefore, assuming E $\not\subseteq$ PSPACE, this implies that IP(2pfa) $\not\subseteq$ IP(ptime-ptm). Since obviously NP $\subseteq$ IP(ptime-ptm) and since we do not know that NP is contained in IP(2pfa), it is possible that IP(2pfa) and IP(ptime-ptm) are incomparable. Condon and Lipton [6] give an upper bound on the complexity of IP(2pfa); they show that if $L \in$ IP(2pfa) then $L$ is accepted by a deterministic Turing machine within space $2^{2^{O(n)}}$.

**Theorem 3.12.** *$E \subseteq IP(2pfa)$.*

*Proof.* The proof is similar to the proof of Reif [22] that if a language $L$ is recognized by a deterministic Turing machine in time $2^{cn}$, then $L$ is recognized by a private alternating Turing machine within space $\log n$. The proof also has some similarities with proofs of

Condon and Ladner [5] and Condon [4]. In this proof we assume that the reader is familiar with the definition of an alternating Turing machine (atm) [3]. We use the result that any $L \in \mathrm{E}$ is accepted by some atm within space $n$ [3].

For any $L \in \mathrm{E}$, let $M$ be an atm which accepts $L$ within space $n$. We describe a sweeping 2pfa $V$ and a prover $P$ such that $(P, V)$ is an IPS for $L$ with error probability $\varepsilon < 1/2$.

To simplify the proof, we make some inessential assumptions about $M$. We assume that $M$ has one tape which is scanned by a read/write head. $M$ is given input $x$ by writing $\not{c}x\not{c}$ on the tape with $M$ scanning the leftmost $\not{c}$. The endmarkers $\not{c}$ are never erased by $M$. Let $Q$ be the states and $\Gamma$ be the tape alphabet of $M$. A *configuration* of $M$ on an input of length $n$ is a word of the form $uqv$ where $q \in Q$, $uv \in \not{c}(\Gamma - \{\not{c}\})^*\not{c}$, and $|uv| = n + 2$. The meaning is that $uv$ is written on the tape and $M$ is in state $q$ scanning the leftmost symbol of $v$. For example, the *initial configuration* on input $x$ is $q_0\not{c}x\not{c}$; this configuration is assumed to be existential. $M$ has two halting states, one an accepting state and the other a rejecting state. For configurations $\alpha$ and $\beta$, we write $\alpha \vdash \beta$ if $|\alpha| = |\beta|$ and $\alpha$ can reach $\beta$ in one step of $M$. If $\beta$ is nonhalting, we assume that $\beta$ is universal iff $\alpha$ is existential. We assume that for each nonhalting configuration $\alpha$, there are exactly two configurations $\beta_1$ and $\beta_2$ reachable from $\alpha$ in one step of $M$, and we write $\alpha \vdash_k \beta_k$ for $k = 1, 2$ (since $M$ might move deterministically from $\alpha$, we allow $\beta_1 = \beta_2$). A *computation path* of $M$ on $x$ is a word of the form

$$\$\alpha_0\$\alpha_1\$\alpha_2\$\ldots\$\alpha_m\$$$

where $\alpha_0, \alpha_1, \ldots, \alpha_m$ are configurations such that $\alpha_0$ is the initial configuration on $x$, $\alpha_i \vdash \alpha_{i+1}$ for all $i$, and $\alpha_m$ is halting. We assume that $M$ keeps a counter on a separate track of its tape, increments the counter after each action performed on the first track, and halts when the value of the counter reaches $2^{cn}$ where $c$ is a constant. Therefore, the length of any computation path is finite. Moreover, by implementing the counter with a little care, this fact holds if $\alpha_0$ is any configuration, including configurations which are not actually reachable from any initial configuration.

We say that $\alpha$ *leads to acceptance* if

1. $\alpha$ is an accepting configuration, or
2. $\alpha$ is an existential configuration and there exists an $i \in \{1, 2\}$ such that $\alpha \vdash \beta_i$ and $\beta_i$ leads to acceptance, or
3. $\alpha$ is a universal configuration and $\beta_i$ leads to acceptance for all $i \in \{1, 2\}$ with $\alpha \vdash \beta_i$.

So $M$ accepts $x$ iff the initial configuration on input $x$ leads to acceptance.

The interactive proof which simulates $M$ proceeds by a sequence of iterations. The basic idea is that, at each iteration of the interaction, $P$ and $V$ simulate a computation path of $M$ on input $x$, with the prover $P$ sending the computation path to the verifier $V$. The prover $P$ makes the existential choices, and $V$ simulates the universal choices by random choices. $V$ also checks that $P$ is actually sending a valid computation path, and $V$ immediately rejects if it discovers an error by the prover; this check involves private randomization by the verifier. At a given iteration, there is only a very small chance that $V$ will discover an error by the prover, so the interaction must involve enough iterations to make it very likely that a prover's error will be caught at some iteration. Since $P^*$ can send an infinite string to $V$, the verifier must also be able to detect this situation and reject with probabilty at least $1 - \varepsilon$. (If we use the weaker definition $\mathrm{IP}_w$ of an IPS which allows $(P^*, V)(x)$ to run forever with high probability, then this check is not needed and the proof can be simplified.)

We now describe $P$ and $V$ in more detail. Let $x$ be an input and let $n = |x|$. Assuming $x \in L$, at each iteration $P$ sends a computation path $\$\alpha_0\$\alpha_1\$ \ldots \$\alpha_m\$$ to $V$. $P$ chooses the computation path as follows so that all configurations lead to acceptance. $\alpha_0$ is the initial configuration of $M$ on input $x$. Suppose that the path has been chosen up to some configuration $\alpha_i$ which leads to acceptance. If $\alpha_i$ is existential ($i$ is even), then $P$ chooses $\alpha_{i+1}$ so that $\alpha_i \vdash \alpha_{i+1}$ and $\alpha_{i+1}$ leads to acceptance. If $\alpha_i$ is universal ($i$ is odd), then when $V$ receives the symbol \$ terminating the sending of $\alpha_i$, $V$ chooses a random $u(i) \in \{1, 2\}$ and sends $u(i)$ to $P$ (these are the only public coin flips in $V$'s protocol; the others are all private). The prover $P$ then chooses $\alpha_{i+1}$ so that $\alpha_i \vdash_{u(i)} \alpha_{i+1}$. By this choice of the path, it is clear that the sending terminates with some accepting configuration $\alpha_m$. This completes the description of $P$.

The verifier's algorithm has five parts which are run concurrently. One part is to randomly choose the numbers $u(i)$ as described above, the second part is the *form check*, the third part is the *accept check*, the fourth part is the *validity check*, and the fifth part is the *halt test.*

To describe the form, accept, and validity checks, fix an arbitrary iteration and let $w$ be the string received from the prover ($w$ might be infinite). For $l \geq 1$, let $w_l$ be the $l$th symbol of $w$. $V$ performs the form check by checking that $w$ begins with \$ and that the subword of $w$ between every two consecutive \$'s has the form of a configuration of $M$, i.e., $uqv$ where $q \in Q$ and $uv \in \cent(\Gamma - \{\cent\})^*\cent$. The lengths of the configurations are not checked. If any such subword does not have the proper form of a configuration, then $V$ rejects. $V$ performs the form check within its finite state control; the input head is not used.

$V$ performs the accept check by checking that the last configuration in $w$ is accepting, and rejects if not. (This check is done only in case $w$ is finite.)

We next describe the validity check by describing two procedures *Select* and *Verify* that call each other. By calling *Select*, $V$ will decide with some probability to "verify" at the $l$th symbol of $w$; for short, we say that "$V$ verifies at $l$." $V$ does this by calling the procedure *Verify* just after the prover has sent the $l$th symbol of $w$.

To begin the validity check, $V$ requests the first symbol of $w$ from the prover; this must be a \$. $V$ then calls *Select.*

*Select:* The input head is used to implement a counter $C$ with capacity $n + 4$. Initially, $C = 1$. Flip a coin. If the outcome is 0 or if $C = n + 4$, then call *Verify*. Otherwise, increment $C$ by 1 and request another symbol of $w$ from the prover.

*Verify:* Let $l$ be such that $V$ has received the $l$th symbol of $w$ from the prover, but not the $(l + 1)$th. Let $D = n + 4$. If $l = 1$, then $V$ verifies that the first $n + 5$ symbols of $w$ are

$\$q_0 ¢ x ¢ \$$. If $l > 1$, remember $w_{l-2}, w_{l-1}$, and $w_l$, and verify that $w_{l+D-2}, w_{l+D-1}$, and $w_{l+D}$ follow correctly according to the transition rules of $M$. The input head is used to measure the distance $D = n + 4$. (Note that if $w$ is a valid computation path and if $w_l$ is the $j$th symbol of $\alpha_i$, then $w_{l+D}$ is the $j$th symbol of $\alpha_{i+1}$.) If $w_l$ lies in a universal configuration $\alpha_i$, then the verification also depends on the choice $u(i)$ made by $V$. If $w_l \neq \$$, then $V$ also verifies that exactly one of the symbols between $w_l$ and $w_{l+D}$ is a \$. If $w_l = \$$, then $V$ verifies that $w_{l+D} = \$$ and that $w_j \neq \$$ for all $l < j < l + D$. If any of these verifications fail, then $V$ rejects. If all verifications are correct, then call *Select*. (When *Select* is called, the prover has just sent $w_{l+D}$ to $V$.)

The halt test depends on an integer constant $k$ which affects the error probability. To perform the halt test during a particular iteration, $V$ flips a coin $k(n+m)$ times, where $m$ is the number of \$'s in $w$. If all these flips are 0, then $V$ accepts at the end of this iteration, assuming that $V$ has not rejected due to a error found by one of the checking procedures at this iteration. If these flips are not all 0, $V$ continues to another iteration.

We now argue that $(P, V)$ is an IPS for $L$. If $x \in L$ and $P$ is the prover, then $V$ will never reject due to an error found by the form, accept, or validity check. Since the halt test will eventually succeed, $V$ accepts $x$ with probability 1.

Consider now $x \notin L$ and let $P^*$ be any prover. Fix some iteration and let $w$ be the word which $P^*$ sends to $V$ at this iteration. We consider four cases:

Case 1. $w$ does not have the correct form, i.e., the form check rejects $w$;

Case 2. $w$ has the correct form and is infinite;

Case 3. $w$ has the correct form, $w$ is finite, and $w$ is a computation path $\$\alpha_0\$\alpha_1\$\ldots\$\alpha_m\$$ where $\alpha_0 = q_0 ¢ x ¢$ and $\alpha_i \vdash_{u(i)} \alpha_{i+1}$ for all odd $i$ (recall that $\alpha_i$ is universal for all odd $i$);

Case 4. $w$ has the correct form, $w$ is finite, but Case 3 does not hold.

In Case 1, $V$ rejects with probability 1.

In Case 2, we first make some easily verified claims about the probabilities with which $V$ checks symbols of $w$. To avoid technical complications in making these claims, we assume that $V$ continues to run *Select* and *Verify* on the infinite $w$ even if $V$ rejects. For $l \geq 1$, let $p_l$ be the probability that $V$ verifies at $l$. Also define $p_l = 0$ for $l \leq 0$. Observe from the definition of *Verify* and *Select* that: (i) initially, $V$ verifies at $l$ with probability $2^{-l}$ for $1 \leq l \leq n+3$ and $V$ verifies at $n+4$ with probability $2^{-(n+3)}$; and (ii) if $V$ verifies at $j$, then $V$ next calls *Select* at position $j + D$, so $V$ will verify at $j + D + i - 1$ with probability $2^{-i}$ for $1 \leq i \leq n+3$ and $V$ will verify at $j + D + n + 3 = j + 2D - 1$ with probability $2^{-(n+3)}$. It follows that

$$\begin{aligned} p_l &= 2^{-l} \ \text{ for } \ 1 \leq l \leq n+3, & (3) \\ p_{n+4} &= 2^{-(n+3)}, & (4) \\ p_l &= \sum_{i=1}^{n+3} 2^{-i} p_{l-D-i+1} + 2^{-(n+3)} p_{l-2D+1} \ \text{ for } \ l \geq n+5. & (5) \end{aligned}$$

Using induction on $l$, it follows from these equations that

$$p_l \geq 2^{-n-4} \quad \text{for all} \quad l \geq 1. \tag{6}$$

It also follows from the definition of *Select* and *Verify* that: (i) with probability 1, $V$ verifies at some $l$ with $1 \leq l \leq D$; and (ii) if $V$ verifies at $l$ then, with probability 1, $V$ next verifies at some $l'$ with $l + D \leq l' \leq l + 2D - 1$. We therefore have:

*Claim 1.* Consider any run of $V$ and let $v_1, v_2, \ldots,$ be the positions at which $V$ verifies. Then $v_{i+1} - v_i \leq 2D - 1$ for all $i \geq 1$.

We can now prove our main objective.

*Claim 2.* For any infinite set $I$ of positive integers,

$$\Pr[\,(\exists l \in I)(V \text{ verifies at } l)\,] = 1.$$

*Proof of Claim 2.* Let $I' = \{\, l_1, l_2, \ldots \}$ be an infinite subset of $I$ such that $l_{i+1} \geq l_i + 4D$ for all $i \geq 1$. We say that $V$ *misses* $t$ *times* if $V$ does not verify at any of $l_1, l_2, \ldots, l_t$. We want to bound the probability that $V$ verifies at $l_{t+1}$ given that $V$ misses $t$ times. Let

$$J = \{\, j \mid l_{t+1} - 4D + 2 \leq j \leq l_{t+1} - 2D \,\}.$$

Since $J$ contains $2D - 1$ consecutive integers, it follows from Claim 1 that

$$\Pr[\,(\exists j \in J)(V \text{ verifies at } j) \mid V \text{ misses } t \text{ times}\,] = 1. \tag{7}$$

Equation (5) implies

$$p_l \geq 2^{-(n+3)} p_j \quad \text{if} \quad l - 2D + 1 \leq j \leq l - D.$$

Applying this inequality twice gives

$$p_l \geq 2^{-2(n+3)} p_j \quad \text{if} \quad l - 4D + 2 \leq j \leq l - 2D. \tag{8}$$

It follows from (7) and (8) that

$$\Pr[\,V \text{ verifies at } l_{t+1} \mid V \text{ misses } t \text{ times}\,] \geq 2^{-2(n+3)}.$$

Since $V$ verifies at $l_1$ with probability at least $2^{-2(n+3)}$ (by (6)), it follows that, for any $t \geq 1$,

$$\Pr[\,(\exists i, 1 \leq i \leq t)(V \text{ verifies at } l_i)\,] \geq 1 - (1 - 2^{-2(n+3)})^t.$$

Since this probability goes to 1 as $t \to \infty$, Claim 2 is proved. $\square$

We now use Claim 2 to prove that $V$ rejects with probability 1 if $w$ is infinite. First, we can assume that $w$ contains an infinite number of \$'s, for otherwise the validity check will reject with probability 1. This is true because each call to the procedure *Verify* always looks for a new \$ and rejects if one is not found. Let $S$ be the set of $l$ such that $w_l = \$$ and such that the next occurrence of \$ after $w_l$ is not $w_{l+D}$. If $V$ verifies at any $l \in S$, then $V$

rejects. By Claim 2, we can therefore assume that $S$ is finite. Therefore, from some point on, $w$ has the form

$$w = \dots \$\alpha_j\$\alpha_{j+1}\$\dots$$

where, for all $i \geq j$, $\alpha_i$ is a configuration of length $n+3$. Because of the way that the simulated atm $M$ maintains a counter and halts when the counter reaches a certain value, it follows that there are infinitely many $l$ such that $V$ will reject if it verifies at $l$. So by Claim 2, $V$ rejects $w$ with probability 1.

In Case 4, the inequality (6) implies that $V$ rejects with probability at least $2^{-n-4}$ since there must be an $l$ such that $V$ rejects if it verifies at $l$.

It remains to consider Case 3. Since $\alpha_0$ is the initial configuration on $x$ and since $x \notin L$, it follows that $\alpha_0$ does not lead to acceptance. Therefore, no matter how $P^*$ chooses $\alpha_1$, there is a $u(1) \in \{1,2\}$ such that $\alpha_2$ does not lead to acceptance. $V$ will choose such a $u(i)$ with probability at least 1/2. Similarly, no matter how $P^*$ chooses $\alpha_3$, there is a $u(3) \in \{1,2\}$ such that $\alpha_4$ does not lead to acceptance. Continuing in this way, $\alpha_m$ is rejecting with probability at least $2^{-m}$. Therefore, in Case 3 the accept check will reject with probability at least $2^{-m}$.

Only in Cases 3 and 4 can the halt test cause $V$ to accept, and the probability that this happens at a particular iteration is $2^{-k(m+n)}$. We have just argued that $V$ rejects with probability at least $2^{-n-4}$ or $2^{-m}$ at each iteration in these two Cases. Therefore, the error probability, i.e., the probability that $V$ accepts before it rejects, approaches zero as $k$ increases. $\square$

*Remarks.* In the proof of Theorem 3.12, it is sufficient that the prover be a deterministic exponential space Turing machine. Moreover, during each iteration, the computation of the prover requires at most deterministic exponential time. The IPS $(P, V)$ described in this proof has perfect completeness, meaning that, for all $x \in L$, $(P,V)(x)$ accepts with probability 1. The verifier $V$ is a sweeping 2pfa that halts with probability 1. For any constant $c$, the error probability can be bounded above by $2^{-cn}$ by choosing the parameter $k$ large enough.

A straightforward modification to the proof of Theorem 3.12 gives the following.

**Theorem 3.13.** *Let $S(n)$ be space constructible.*

$$DTIME(2^{n2^{O(S(n))}}) \subseteq IP(\textit{space-}S\textit{-ptm}).$$

*Proof.* In this case, the IPS is simulating an atm $M$ which has space bound $n2^{O(S(n))}$. The verifier uses its input head together with its $S(n)$-bounded worktape as a counter to measure distances of size $n2^{O(S(n))}$. $\square$

For $S(n) \geq \log n$, this result was also proved by Condon [4]. The requirement that $S(n)$ be constructible can be removed from Theorem 3.13 if we allow the verifier to use more than $S(n)$ space when communicating with a cheating prover $P^*$. In this case, the correct prover first sends the correct space bound to the verifier by sending the string $1^{S(n)}$.

We close this section with two results on the power of IPS's with 2pfa verifiers which run in polynomial expected time.

**Theorem 3.14.** *Let $(P, V)$ be an IPS using public coins for the language $L$ where $V$ is a ptm which runs in polynomial expected time and linear space. Then $L \in$ IP(ptime-2pfa).*

*Proof.* The proof is very similar to the proof of Theorem 3.12. We only sketch the differences, using terminology from the proof of Theorem 3.12.

The main new tool, taken from Greenberg and Weiss [13], is a procedure by which a 2pfa can flip a biased "coin" with two possible outcomes, 1 and 0, such that the probability of 1 is $n^{-b}$ where $n$ is the length of the input, and where $b$ can be any fixed positive integer. The expected running time of the coin procedure is polynomial in $n$. We denote a coin with this property as $C_b$.

We first describe $C_1$. The 2pfa begins with the input head scanning the first symbol of the input $x$. It then performs a random walk on the input, moving the head left with probability 1/2 or right with probability 1/2 at each step. The random walk ends when the head either scans the left endmarker (outcome 0) or scans the rightmost symbol of $x$ (outcome 1). It is known that the outcome is 1 with probability $n^{-1}$, and that the random walk terminates in expected time polynomial in $n$; see, for example, [23, §4.2]. Now $C_b$ is obtained by running $C_1$ $b$ times; $C_b$ has outcome 1 if all $b$ runs of $C_1$ have outcome 1.

Let $(P, V)$ be an IPS for $L$ as in the statement of the theorem. If $V$ runs in expected time $p(n)$, we modify $V$ to run in worst-case time $n \cdot p(n)$ by having $V$ reject after it has run for $n \cdot p(n)$ steps. This increases the error probability by at most $n^{-1}$. We can also assume that the error probability of $(P, V)$ is at most $2^{-n}$ on inputs of length $n$ (see, for example, the Amplification Lemma in [12]).

We describe a prover $P'$ and a 2pfa verifier $V'$ such that $(P', V')$ is an IPS for $L$ which runs in expected polynomial time. Let the constants $c$ and $k$ be such that $(P, V)$ runs in worst-case time $cn^k$. As before, the interactive proof consists of a sequence of iterations. Fix an input $x$. Assuming $x \in L$, during each iteration, the prover $P'$ sends a computation path

$$\$\alpha_0\$\alpha_1\$\alpha_2\$\ldots\$\alpha_m\$$$

describing a possible computation of $V$ on input $x$ when $V$ communicates with $P$. We have $m \leq cn^k$. $V'$ simulates the public coin tosses of $V$, and $P'$ simulates the responses of $P$. Since $V$ has space bound $n$, the configurations $\alpha_i$ have length $n + 3$ as in the proof of Theorem 3.12.

The protocol of $V'$ is as follows. It depends on two constants $a$ and $b$ which are chosen with $b \gg 1$ and $a \gg k + b$ to make the error probability small. Let $w$ be the string received from the prover. $V'$ flips the coin $C_1$ each time the prover sends a symbol of $w$, until the outcome of $C_1$ is 1; say that the first 1 outcome occurs just after the $j$th symbol of $w$ has been sent. From that point on, $V'$ verifies at $l$ for all $l \equiv j \bmod (n + 4)$. If any verification fails, then $V'$ rejects. Note that there is a constant $d > 0$ such that, for any $l$,

$$\Pr[\,V' \text{ verifies at } l\,] \geq \left(1 - \frac{1}{n}\right)^{n+3} \frac{1}{n} \geq \frac{d}{n}.$$

After each call to the procedure *Verify*, $V'$ flips $C_a$; if the outcome is 1 then $V'$ rejects. The purpose of flipping $C_a$ is to ensure that $V'$ rejects if $w$ is infinite. $V'$ also rejects if the last configuration $\alpha_m$ is not accepting. After each iteration, $V'$ flips $C_b$; if the outcome is 0, then $V'$ starts another iteration; if the outcome is 1, then $V'$ accepts.

Consider first the case that $x \notin L$ and let $P^*$ be any prover. If $w$ is infinite then $V'$ rejects with probability 1. Assume that $w$ is finite. At each iteration, $V'$ rejects with probability very close to $d/n$. (We can say only "very close" because there is a small probability $2^{-n}$ that the original verifier $V$ accepts in error.) If we choose $b \gg 1$, then with probablity approaching 1 as $n$ increases, $V'$ rejects on some iteration before $C_b$ has outcome 1.

Consider now the case that $x \in L$ and $V'$ interacts with the correct prover $P'$. Since we assume that the error probability of $(P, V)$ is at most $2^{-n}$, at each iteration $V'$ rejects with probability at most $2^{-n}$ (this can happen only if $V'$ finds a nonaccepting $\alpha_m$). Therefore, there is only a very small probability (approaching zero as $n$ increases) that $V'$ finds a nonaccepting $\alpha_m$ (causing $V'$ to reject) before $C_b$ has outcome 1 (causing $V'$ to accept). We expect the interactive proof to have $n^b$ iterations. Moreover, by Markov's inequality, the probability that the number of iterations exceeds $n^{b+1}$ is at most $n^{-1}$. Therefore, during the entire interactive computation of $(P', V')(x)$, with probability at least $(1 - n^{-1})$, $C_a$ is flipped at most $cn^k n^{b+1}$ times. By choosing $a \gg k+b$, there is only a very small probability that $C_a$ will have outcome 1 (causing $V'$ to reject) before $C_b$ has outcome 1 (causing $V'$ to accept).

The above analysis is valid for all long enough inputs $x$. We build a table of answers into $V'$ for all the small exceptions. $\square$

If follows from Theorem 3.14 that IP(ptime-2pfa) contains all context-free languages and an NP-complete language. Theorem 3.14 is also useful in proving the following.

**Theorem 3.15.** *AM(2pfa) $\subseteq$ IP(ptime-2pfa).*

*Proof.* Let $L \in$ AM(2pfa). By Theorem 3.14 it suffices to show that $L$ is accepted by some nondeterministic Turing machine $M$ which accepts in polynomial time and linear space.

Let $(P, V)$ be a public coin IPS for $L$. Let $x$ be a given input and let $n = |x|$. As in the proof of Theorem 3.4, $x \in L$ iff there is a Markov transition function (strategy) $\sigma$ such that $V$ accepts $x$ with probability greater than 1/2 when $V$ interacts with the prover $P_\sigma$ which uses Markov strategy $\sigma$. On input $x$, $M$ first nondeterministically guesses a Markov strategy $\sigma$; that is, for each communication state $q$ of $V$ and each $l$ with $0 \leq l \leq n+1$, $M$ guesses a symbol $\sigma(q, l)$ from the communication alphabet; this symbol is the response of $P_\sigma$ whenever $V$ is in state $q$ reading the $l$th symbol of $\not{c}x\not{c}$. For each $l$, the symbols $\sigma(q, l)$ for all communication states $q$ are written on the $l$th square of $M$'s worktape.

Given the choice of $\sigma$, the computation $(P_\sigma, V)(x)$ can be modeled by a Markov chain $K(x)$ very similar to the one described in the first part of the proof of Theorem 3.3. Let $Q$ be the set of states of $V$. The states of this Markov chain have the form $(q, l)$ where $q \in Q$ and $0 \leq l \leq n+1$. If $V$, when reading the $l$th symbol of $\not{c}x\not{c}$ in state $q$, can enter state

$q'$ and move its head in direction $d \in \{-1, 0, 1\}$ with probability $p$, then in the Markov chain $K(x)$ there is a transition from state $(q, l)$ to state $(q', l + d)$ with probability $p$. If $q$ is a communication state, then the transitions out of $(q, l)$ also depend on $\sigma(q, l)$. Let $p((q, l), (q', l'))$ denote the transition probability from state $(q, l)$ to state $(q', l')$ in $K(x)$. Note that

$$|l - l'| > 1 \quad \Rightarrow \quad p((q, l), (q', l')) = 0. \tag{9}$$

For simplicity, assume that when $V$ enters a halting state $q_a$ or $q_r$ it remains in that state forever without moving the head and assume that $V$ enters the accepting state $q_a$ only when reading the left endmarker ¢. Therefore, the probability that $(P_\sigma, V)(x)$ accepts is the probability that $K(x)$ is trapped in state $(q_a, 0)$ when started in state $(q_0, 0)$.

Let $t(q, l)$ be the probability that $K(x)$ is trapped in state $(q_a, 0)$ when started in state $(q, l)$. Let $Z$ be the set of states $(q, l)$ such that there is no path of positive probability from $(q, l)$ to $(q_a, 0)$ in $K(x)$. Recalling (9), the trapping probabilities are the unique solution to the following system of equations which we call the *trapping equations* (see [23, §4.2]):

$$\begin{aligned} t(q_a, 0) &= 1, \\ t(q, l) &= 0 \text{ for all } (q, l) \in Z, \\ t(q, l) &= \sum_{q' \in Q} \sum_{|l-j| \le 1} p((q, l), (q', j))\, t(q', j) \text{ for } (q, l) \notin Z \cup \{(q_a, 0)\}. \end{aligned}$$

We claim that there is a constant $c$ (depending only on $V$) such that any trapping probability $t(q, l)$ can be written as $y/z$ for some integers $y$ and $z$ with $0 \le y \le z \le c^n$. Therefore, any $t(q, l)$ can be represented within space $O(n)$. To prove the claim, note that the trapping probabilities are the solution to a system of $m$ linear equations in $m$ unknowns where $m = O(n)$. Moreover, each equation involves at most $k$ variables, where $k = 3|Q|$ is constant. Since the coefficients of the system are the transition probabilities of $V$, we can transform the system to one where all coefficients are integers with absolute values bounded above by 2. By Cramer's rule, any trapping probability is given by $\det(A)/\det(B)$ where $A$ and $B$ are $m \times m$ matrices with integer entries of absolute value at most 2, and where $B$ has at most $k$ nonzero entries in each row. Since $\det(B)$ is the sum of at most $k^m$ terms, each of magnitude at most $2^m$, we have $\det(B) \le (2k)^m \le c^n$ for some constant $c$. Since $\det(A)$ is an integer, this completes the proof of the claim.

We now continue to describe the operation of $M$. By performing a breadth-first search from each state of $K(x)$, $M$ finds and marks all states in $Z$. Then $M$ nondeterministically guesses $t(q, i)$ for all $(q, i)$ with $0 \le i \le 2$. Using the claim above, any constant number of trapping probabilities $t(q, l)$ can be stored by $M$ in linear space. Having guessed $t(q, i)$ for $0 \le i \le 2$, $M$ verifies that these guesses satisfy the trapping equations for $l = 0$ and $l = 1$ and also verifies that $t(q_0, 0) > 1/2$. If any of these verifications fail, then $M$ rejects. Then sequentially for $i = 2, 3, \ldots, n$, $M$ does the following: erase $t(q, i - 2)$ for all $q$, guess $t(q, i + 1)$ for all $q$, and verify the trapping equations for $l = i$. Finally, $M$ verifies the trapping equations for $l = n + 1$. It is easy to see that the computation of $M$ can be done in polynomial time and linear space, which completes the proof. □

## 4. Pushdown Verifiers

The literature in formal language theory and automata theory contains studies of many different types of automata. The questions addressed in this paper for 2pfa verifiers can also be asked for other types of verifier automata. Although we do not suggest that this entire body of questions is an interesting research area, we do give a result about an obvious next choice for verifier automata, namely, pushdown automata. We consider probabilistic pushdown automata with a single 2-way input head (2ppda). These machines are defined like 2pfa's but have in addition a single pushdown store of unbounded size.

The result shows that for interactive proof systems with private coins which run in polynomial time, a 2ppda verifier is as good as a probabilistic Turing machine verifier.

**Theorem 4.1.** *There is an IPS $(P, V)$ for $L$ where $V$ is a ptm which runs in polynomial worst-case time iff there is an IPS $(P', V')$ for $L$ where $V'$ is a 2ppda which runs in polynomial worst-case time.*

*Proof.* The "if" direction is trivial. The proof of "only if" is very similar to the proof of Theorem 3.14. As in that proof, the prover $P'$ sends a computation $w$ of $(P, V)(x)$ to $V'$ and $V'$ checks that it is correct. There are two major differences. First, the IPS $(P, V)$ with a ptm verifier uses polynomial space instead of linear space, so $V'$ cannot use its input head to measure distance equal to the length of a configuration (this is needed in the procedure *Verify*). Second, we want $(P', V')$ to run in polynomial worst-case time rather than polynomial expected time, so $V'$ cannot use the "coins" $C_a$ and $C_b$ to determine when it should halt because the worst-case running time of the coin procedures is infinite.

We handle both of these differences by using counters. We show how a 2-way pda can initialize a counter to $2^{a\lfloor \log n \rfloor}$, where $a$ is a fixed positive integer, and then repeatedly decrement the counter by 1 until it reaches zero. These procedures are deterministic. The value of the counter is stored in binary on the pushdown store with the least significant bit on the top of the stack. Let $T_a$ denote such a counter.

We first show how to initialize $T_1$ to $2^{\lfloor \log n \rfloor}$. Let $x$ be an input of length $n$. For each integer $i$ with $0 \leq i \leq \lfloor \log n \rfloor$, we define an *$i$-state* of the pda to be a configuration of the pda where the input head is scanning the $2^i$-th symbol of $x$ and $2^i$ is written in binary on the pushdown store. We describe how a pda can move from an $i$-state to an $(i+1)$-state, assuming $i < \lfloor \log n \rfloor$. The pda moves its input head left, pushing the symbol 2 each time the head moves, until the left endmarker is reached. The pda then pops 2's, moving the input head two symbols to the right for every 2 popped until the 2's are exhausted. The pda then pushes a 0 onto the stack. If the right endmarker is reached before the 2's are exhausted, the pda pops the remaining 2's and halts the initialization procedure. To initialize $T_1$, the pda starts in a 0-state, i.e., it pushes a 1 onto the stack and positions the input head on the first symbol of $x$. It then repeatedly moves from an $i$-state to an $(i+1)$-state until the procedure halts. The initialization of $T_a$, for $a > 1$, is similar.

To decrement a counter by 1, the pda pops 0's off the stack, using the input head to keep track of the number of 0's popped, until a 1 is found on the top of the stack. This 1 is changed to a 0. The pda then pushes a number of 1's equal to the number of 0's popped. Similarly, the pda can test if a counter is zero.

Say that $(P, V)$ runs in worst case time and space $cn^k$. By a result of Goldwasser and Sipser [12], we can assume that $(P, V)$ uses public coins. Let $d$ be the smallest integer such that $2^{d\lfloor \log n \rfloor} \geq cn^k + 4$ for all $n \geq 2$, and let $D = 2^{d\lfloor \log n \rfloor}$. We take $D - 1$ to be the length of configurations of $V$, as well as an upper bound on the number of configurations in any computation of $V$. At the start of the interactive proof, $V'$ initializes $T_b$ where $b$ is chosen so that $b \gg d$. At the start of each iteration, $V'$ initializes $T_d$. This counter $T_d$, which we call the *time counter*, is stored on top of the counter $T_b$ on the pushdown. (In general, multiple counters are separated by marker symbols.) $V'$ then initializes a counter $R$ to a randomly chosen integer $j$ with $0 \leq j < D$. The counter $R$ is stored on top of the time counter. The initialization of $R$ is similar to the initialization of $T_d$, except that the counter contains 0 in a 0-state, and in moving from an $i$-state to an $(i+1)$-state, a random bit is pushed instead of a 0. Now $V'$ alternately asks for a symbol of $w$ from the prover and decrements $R$ until $R$ becomes 0. $V'$ then verifies at $l$ for all $l \equiv j + 1 \bmod D$. This ensures that each position of $w$ is verified with probability $\Omega(n^{-d})$. Before each call to *Verify*, $V'$ initializes a copy of $T_d$ which is then decremented to measure the distance $D$ which is needed by *Verify*. An additional complication arises when $V'$ verifies at $l = 1$, since $V'$ must check that $w$ begins with $\$q_0 \cent x$ and must also decrement $T_d$ after each symbol of $w$ is sent to check that the first configuration in $w$ has the correct length. These two tasks cannot be done in parallel because they use the input head in different ways. Therefore, to verify at $l = 1$, $V'$ flips a coin. If the outcome is 0, $V'$ checks that $w$ begins with $\$q_0 \cent x\#$ where $\#$ denotes the blank tape symbol. If this check fails then $V'$ rejects, and if it succeeds then $V'$ terminates this iteration. If the outcome is 1, then $V'$ initializes a copy of $T_d$ and uses this counter to check that the first $D+1$ symbols of $w$ are a word of the form $\$y\#^k \cent \$$ where $k$ is a positive integer and where the word $y$ contains no occurrence of $\#$ or $\$$. If this check fails then $V'$ rejects, and if it succeeds then $V'$ continues to verify at all positions $l \equiv 1 \bmod D$.

After each call to *Verify*, $V'$ decrements the time counter. If the time counter reaches 0 before the prover ends this iteration, then $V'$ rejects (because the prover is sending a computation which is too long). $V'$ also rejects if any of the verifications fail or if the final configuration in the computation $w$ is not accepting. After each successful iteration, $V'$ pops the time counter and decrements $T_b$. If $T_b$ is 0, then $V'$ accepts; else $V'$ requests another iteration. We choose $b \gg d$ to make the error probability small. $\square$

*Remark.* In the proof of Theorem 4.1, the size of the pushdown store of $V'$ is $O(\log n)$. J. Rompel (personal communication in [16]) has shown independently that IPS's with polynomial-time ptm verifiers are as powerful as IPS's with polynomial-time log-space ptm verifiers. Theorem 4.1 shows that, without loss of power, the verifier can be further restricted so that its log-space bounded worktape is a pushdown store.

## 5. Open Questions

This paper is the first attempt to study 2pfa verifiers, and many questions remain open. We mention three questions which we find intriguing.

1. Is there a nonregular language in AM(ptime-2pfa)?

It is not difficult to contrive a *separation* problem which has an IPS using public coins with a ptime-2pfa verifier but which cannot even be solved by a 2pfa. This is done by padding the language *Center* of §3.3. Specifically, define $U$ to be the set of words of the form $x\#^k$ where $x \in \{0,1\}^*$ and $k = 2^{|x|}$, and let $A$ (resp., $B$) be the set of $x\#^k \in U$ such that $x \in \mathit{Center}$ (resp., $x \notin \mathit{Center}$). As a function of the length of the padded input, the public coin IPS for *Center*, described in the proof of Theorem 3.8, uses polynomial expected time. It is easy to apply Theorem 3.3 to show that no 2pfa can separate $A$ and $B$. Unfortunately, it follows from Theorem 3.6 that $U \notin$ AM(ptime-2pfa); this is a serious obstacle to extending this separation problem to a language recognition problem in AM(ptime-2pfa).

2. Close the gap between the lower bound (deterministic single-exponential time) and the upper bound (deterministic double-exponential space [6]) on the complexity of IP(2pfa).

In studying the power of IPS's with 2pfa verifiers, we were naturally led to questions about the power of 2pfa's alone. We have one tool, Theorem 3.3, for showing that certain languages are not in 2PFA, although there are probably cases for which this result is not strong enough to prove nonmembership in 2PFA. One interesting example is the language $D_1$ containing all strings of balanced parentheses of one type; $D_1$ is generated by the context-free grammar with productions $S \to (\,)$, $S \to SS$, and $S \to (S)$.

3. Is $D_1$ in 2PFA?

We conjecture that the answer is "no", although this cannot be proved by Theorem 3.3. If $W_m$ satisfies conditions (1) and (3) in the statement of Theorem 3.3, then $|W_m| = O(m^2)$, so condition (2) is not satisfied.

We note that if $D_k$ is the language containing strings of balanced parentheses of $k$ types, then Theorem 3.3 easily shows that $D_k \notin$ 2PFA for $k \geq 2$.

## References.


[1] Babai, L. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing.* ACM, New York, 1985, pp. 421–429.

[2] Broder, A. Generating random spanning trees. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science.* IEEE, New York, 1989, pp. 442–447.

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

[4] Condon, A. Computational models of games. Ph.D. Thesis, *Tech. Report 87-04-04*, Computer Science Dept., University of Washington, Seattle, WA, 1987.

[5] Condon, A., and Ladner, R. Probabilistic game automata. *J. Comput. Syst. Sci. 36* (1988), 452–489. Preliminary version in *Proceedings of the Conference on Structure in Complexity Theory, Lecture Notes in Computer Science, Vol. 223*, Springer-Verlag, New York, 1986, pp. 144–162.

[6] Condon, A., and Lipton, R. J. On the complexity of space bounded interactive proofs (extended abstract). In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1989, pp. 462–467.

[7] Dwork, C., and Stockmeyer, L. Finite state verifiers II: zero knowledge. *J. ACM 39* (1992), 829–858.

[8] Dwork, C., and Stockmeyer, L. A time complexity gap for two-way probabilistic finite-state automata. *SIAM J. Comput. 19* (1990), 1011–1023.

[9] Freivalds, R. Probabilistic two-way machines. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, *Lecture Notes in Computer Science, Vol. 118*, Springer-Verlag, New York, 1981, pp. 33–45.

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

[11] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18* (1989), 186–208. Preliminary version in *Proceedings of the 17th ACM Symposium on Theory of Computing*. ACM, New York, 1985, pp. 291–304.

[12] Goldwasser, S., and Sipser, M. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th ACM Symposium on Theory of Computing*. ACM, New York, 1986, pp. 59–68.

[13] Greenberg, A. G., and Weiss, A. A lower bound for probabilistic algorithms for finite state machines. *J. Comput. Syst. Sci. 33* (1986), 88–105.

[14] Hopcroft, J. E., and Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA, 1979.

[15] Kaneps, J., and Freivalds, R. Running time to recognize nonregular languages by 2-way probabilistic automata. In *Proceedings of the 18th International Colloquium on Automata, Languages, and Programming*, Springer-Verlag, New York, to appear.

[16] Kilian, J. Zero-knowledge with log-space verifiers. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1988, pp. 25–35.

[17] Leighton, F. T., and Rivest, R. L. The Markov chain tree theorem. *Report MIT/LCS/TM-249*, Laboratory for Computer Science, MIT, Cambridge, MA, 1983.

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

[19] Lipton, R. J. Recursively enumerable languages have finite state interactive proofs. *Report CS-TR-213-89*, Computer Science Dept., Princeton University, 1989.

[20] Lund, C., Fortnow, L., Karloff, H., and Nisan, N. Algebraic methods for interactive proof systems. *J. ACM 39* (1992), 859–868.

[21] Rabin, M. O. Probabilistic automata. *Information and Control 6* (1963), 230–245.

[22] Reif, J. The complexity of two-player games of incomplete information. *J. Comput. Syst. Sci. 29* (1984), 274–301.

[23] Seneta, E. *Non-Negative Matrices and Markov Chains, 2nd Ed.* Springer-Verlag, New York, 1981.

[24] Shamir, A. IP = PSPACE. *J.ACM 39* (1992), 869–877.