Chicago Journal of Theoretical Computer Science

The MIT Press

Volume 1995, Article 4
19 October 1995

ISSN 1073–0486. MIT Press Journals, 55 Hayward St., Cambridge, MA 02142 USA; (617)253-2889; *journals-orders@mit.edu, journals-info@mit.edu*. Published one article at a time in LaTeX source form on the Internet. Pagination varies from copy to copy. For more information and other articles see:

- *http://www-mitpress.mit.edu/jrnls-catalog/chicago.html*
- *http://www.cs.uchicago.edu/publications/cjtcs/*
- *gopher.mit.edu*
- *gopher.cs.uchicago.edu*
- anonymous *ftp* at *mitpress.mit.edu*
- anonymous *ftp* at *cs.uchicago.edu*

The *Chicago Journal of Theoretical Computer Science* is a peer-reviewed scholarly journal in theoretical computer science. The journal is committed to providing a forum for significant results on theoretical aspects of all topics in computer science.

Probabilistically Checkable Debate Systems and Nonapproximability of *PSPACE*-Hard Functions

Anne Condon Joan Feigenbaum Carsten Lund
Peter W. Shor

19 October, 1995

Abstract

Abstract-1 We initiate an investigation of *probabilistically checkable debate systems* (PCDS), a natural generalization of probabilistically checkable proof systems (PCPS). A PCDS for a language L consists of a probabilistic polynomial-time verifier V and a debate between player 1, who claims that the input x is in L, and player 0, who claims that the input x is not in L. We show that there is a PCDS for L in which V flips $O(\log n)$ random coins and reads $O(1)$ bits of the debate if and only if L is in *PSPACE*. This characterization of *PSPACE* is used to show that certain *PSPACE*-hard functions are as hard to approximate closely as they are to compute exactly.

These results first appeared in our Technical Memorandum [CFLS93a]. They were presented in preliminary form at the 25th Annual ACM Symposium on Theory of Computing, San Diego, CA, May 1993, titled "Probabilistically Checkable Debate Systems and Approximation Algorithms for *PSPACE*-Hard Functions" [CFLS93b].

1 Introduction

1-1 Suppose that two candidates, B and C, agree to a debate format. Voter V is too busy to catch more than a very small number of bits of the debate.

How does V decide whether B or C won the debate? In this paper, we show that if B and C choose the right debate format, V's problem is solved. By listening to a few randomly chosen sound bites of the debate, V can with near certainty figure out who won.

1-2 Similarly, suppose that B or C is giving a speech to a set of voters $V_1, \ldots, V_n$, represented by finite automata. He would like to give the speech that results in acceptance (votes) by the greatest number of voters V_i. We show that not only can he not compute this maximum exactly, but he cannot come within an arbitrary constant factor, unless he has access to an oracle (political consultant) with the full power of *PSPACE*.

1-3 Our work builds on recent progress in the theory of *probabilistically checkable proof systems* (PCPS). Results about the language-recognition power of PCPSs have led to lower bounds on the difficulty of approximating *NP*-hard functions. In this paper, we define *probabilistically checkable debate systems* (PCDSs). We prove several results about the language-recognition power of PCDSs and then use them to obtain lower bounds on the difficulty of approximating *PSPACE*-hard functions.

1-4 Let us describe the background for this work in more detail. Loosely speaking, a language L has a PCPS if, for every $x \in L$, there is a string π such that a probabilistic verifier V taking x and π as input can be convinced with high probability that $x \in L$. The class $PCP(r(n), q(n))$ consists of those languages recognizable by PCPSs in which the verifier uses $O(r(n))$ coin flips and looks at $O(q(n))$ bits. It is known that $PCP(\log n, 1) = NP$ (cf. [ALM$^+$92] and [AS92]).

1-5 Results on the power of classes $PCP(r(n), q(n))$ can be used to show that many approximation problems are hard, unless there is some unexpected collapse of complexity classes. The first result along these lines was proven by Condon [Con93]. Additional results concern the MAX-CLIQUE problem [GJ79]—given an undirected graph, find a maximal-size set of nodes with an edge between every pair in the set. In a seminal paper, Feige et al. [FGL$^+$91] showed that MAX-CLIQUE is difficult to approximate. The result of [FGL$^+$91] has been improved several times, and it is now known that there is an ϵ such that approximating MAX-CLIQUE within a factor of n^ϵ is as difficult as solving *NP*-complete problems exactly [ALM$^+$92]. Furthermore, there is a large class of natural optimization problems, those hard for the class *MAX-SNP* defined in [PY91], that do not have polynomial-time approximation schemes unless $P = NP$; that is, for each of these problems, there is an ϵ such that approximating the optimal solution within ratio ϵ is as hard as solving *NP*-complete problems exactly [ALM+92]. This result on *MAX-SNP* shows that many well-known optimization problems are hard to approximate closely, including Traveling Salesman with Triangle Inequality, MAX-SAT, and MAX-CUT.

1-6 A PCDS is a generalization of a PCPS. In a PCDS for L, there are two computationally powerful players, 1 and 0 (called B and C at the beginning of this section) and a probabilistic polynomial-time verifier V. Players 1 and 0 play a game in which they alternately write out strings on a debate tape π. Player 1's goal is to convince V that an input $x \in L$, and player 0's goal is to convince V that $x \notin L$. When the debate is over, V looks at x and π and decides whether $x \in L$ (player 1 wins the debate) or $x \notin L$ (player 0 wins the debate). Suppose V flips $O(r(n))$ random coins, and reads $O(q(n))$ bits of π. If, under the best strategies of players 1 and 0, V's decision is correct with high probability, then we say that L is in $\mathit{PCD}(r(n), q(n))$.

1-7 Specifically, we say that a language L is in $\mathit{PCD}(r(n), q(n))$ if it has a nonadaptive PCDS with one-sided error in which players 1 and 0 write on a debate tape, and then V makes $O(r(n))$ coin flips and queries $O(q(n))$ bits based on these coin flips. By "nonadaptive," we mean that the choice of bits queried by V is based solely on the input and the coin flips. By "one-sided error," we mean that whenever $x \in L$, V must correctly decide that $x \in L$, no matter which sequence of $O(r(n))$ coins are flipped (assuming correct play on the part of player 1). When $x \notin L$, V is allowed to conclude incorrectly that $x \in L$ with probability at most ϵ, for some fixed $\epsilon < 1$.

1-8 Note that we defined PCDSs so that the two players must be deterministic, whereas the verifier can use randomization. Allowing the players to use randomization would not change the class $\mathit{PCD}(r(n), q(n))$; this follows from the standard game-theoretic result that, in perfect information games, players always have deterministic strategies that are optimal [AH92].

1-9 With the above definition in hand, we can state our main results about the language-recognition power of PCDSs.

Theorem 3 (Section 3) $\mathit{PSPACE} = \mathit{PCD}(\log n, 1)$.

This result is the best possible, because one can show that $\mathit{PCD}(\log n, q(n))$ is contained in *PSPACE*, for every function q.

1-10 The following is a technical building block, interesting in its own right, that is used in the proof that $\mathit{PSPACE} = \mathit{PCD}(\log n, 1)$: If $r(n) = \Omega(\log n)$, then $\mathit{PCD}(r(n), q(n))$ contains the same languages if the verifier reads $O(q(n))$ *rounds* of the debate as it does if the verifier reads $O(q(n))$ *bits* of the debate.

1-11 We use our main result about the language-recognition power of PCDSs to prove lower bounds on the difficulty of approximating *PSPACE*-hard functions. Let MAX-Q3SAT be the following natural optimization version of the canonical *PSPACE*-complete language QBF (the set of true quantified Boolean formulae). Suppose

$$\Phi = Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$$

is a quantified Boolean formula, with $Q_i \in \{\exists, \forall\}$, and ϕ in 3CNF. Suppose that the variables of the formula are chosen, in order of quantification, by two players 0 and 1, where player 0 chooses the universally quantified variables and player 1 chooses the existentially quantified variables. If player 1 can guarantee that k clauses of ϕ will be satisfied by the resulting assignment, regardless of what player 0 chooses, we say that k clauses of Φ are *simultaneously satisfiable*. We let MAX-Q3SAT be the function that maps a quantified 3CNF formula Φ to the maximum number of simultaneously satisfiable clauses.

Theorem 4 (Section 4) *There is a constant* $0 < \epsilon < 1$ *such that approximating* MAX-Q3SAT *within ratio* ϵ *is PSPACE-hard.*

Thus MAX-Q3SAT is as hard to approximate closely as it is to compute exactly.

1-12 We use reductions to prove that certain other *PSPACE*-hard functions are *PSPACE*-hard to approximate in a stronger sense. These include maximization versions of the Finite Automata Intersection problem, shown *PSPACE*-complete by Kozen [Koz77], and the Generalized Geography problem, shown *PSPACE*-complete by Schaefer [Sch78]. We show that there is a constant ϵ such that approximating these problems within ratio n^ϵ is *PSPACE*-hard.

1-13 The rest of this paper is organized as follows. We define PCDSs, and all of our other terms, precisely in Section 2. Our results on the language-recognition power of PCDSs are given in Section 3. Those on approximation of *PSPACE*-hard functions are given in Section 4. Section 5 contains open questions and a discussion of subsequent related results.

2 Preliminaries

2-1 We first review the definition of a PCPS (probabilistically checkable proof system). A *verifier* is a probabilistic polynomial-time Turing machine that takes as input a pair x, π, where $\pi \in \{0,1\}^*$, and either accepts or rejects. A language L has a probabilistically checkable proof system, or PCPS, with error probability ϵ if there is a verifier V with the following properties:

- for all x in L, there is a string π such that V accepts with probability 1 on input x, π; and
- for all x not in L, on all strings π, V accepts with probability at most ϵ on input x, π.

2-2 We say that the verifier makes $q(n)$ queries if the number of bits of π read by the verifier is at most $q(n)$, when the input is of size n. $PCP(r(n),q(n))$ is the class of languages that have probabilistically checkable proof systems with error probability $\frac{1}{2}$ in which the verifier uses $O(r(n))$ random bits and makes $O(q(n))$ queries.

2-3 We next extend this to define PCDSs. A probabilistically checkable debate system, or PCDS, consists of a verifier V and a *debate format* D. As before, the verifier is a probablistic polynomial-time Turing machine that takes as input a pair x, π, where $\pi \in \{0,1\}^*$, and outputs 1 or 0. We interpret these outputs to mean "player 1 won the debate" and "player 0 won the debate," respectively.

2-4 A debate format is a pair of functions $f(n), g(n)$. Informally, for a fixed n, a debate between two players, 0 and 1, consistent with format $f(n), g(n)$, contains $g(n)$ rounds. At round $i \geq 1$, player $i \bmod 2$ chooses a string of length $f(n)$.

2-5 For each n, corresponding to the debate format D is a *debate tree*. This is a complete binary tree of depth $f(n)g(n)$ such that, from each node, one edge is labeled 0 and the other is labeled 1. A *debate* is any string of length $f(n)g(n)$. Thus, there is a one-to-one correspondence between debates and the paths in the debate tree. Moreover, a debate is the concatenation of $g(n)$ substrings of length $f(n)$. Each substring is called a *round* of the debate, and each debate of this debate tree has $g(n)$ rounds.

2-6 Again for a fixed n, a *debate subtree* is a subtree of the debate tree of depth $f(n)g(n)$ such that each node at level i (the root is at level 0) has one child if i div $f(n)$ is even, and it has two children if i div $f(n)$ is odd. Informally, a debate subtree corresponds to a list of "responses" of player 1, against *all possible* "arguments" of player 0 in the debate. For this reason, we also refer to a debate subtree as a *strategy* of player 1. A strategy of player 0 can be defined in a similar way (where the definition of debate subtree is modified so that each node at level i has one child if $i \operatorname{div} f(n)$ is odd, and two children if $i \operatorname{div} f(n)$ is even). Thus, a pair of strategies, one for each player, defines a unique debate—namely, the unique path in the intersection of the strategies (represented as trees) of the players.

2-7 A strategy of player 1 could also be defined as a function from nodes of the complete binary tree on levels i where $i \operatorname{div} f(n)$ is even into $\{0, 1\}$, i.e., into responses of player 1. The debate subtree corresponding to such a function is simply the subtree of the complete binary tree that is reachable from the root via paths of the following form: At nodes on level i where $i \operatorname{div} f(n)$ is even, follow the outgoing edge selected by the strategy function; at nodes on level j where $j \operatorname{div} f(n)$ is odd, follow either outgoing edge. However, representing a strategy as a function requires determining responses for many nodes of the game tree that can never be reached with that strategy. Furthermore, the proof of our main result is more naturally expressed in terms of subtrees than functions. For these reasons, we define strategies as subtrees.

2-8 A language L has a PCDS *with error probability* ϵ if there is a pair (D, V) where $D = (f(n), g(n))$, with the following properties:

- For all x in L, there is a debate subtree such that, for all debates π labeling a path of this subtree, V outputs 1 with probability 1 on input x, π. In this case, we say that x is accepted by (D, V).

- For all x not in L, on all debate subtrees, there exists a debate π labeling some path of the subtree such that V outputs 1 with probability at most ϵ on input x, π. In this case, we say that x is rejected by (D, V).

Equivalently, the first condition states that on all x in L, player 1 has a strategy such that, for every strategy of player 0, V outputs 1 with probability 1 on the debate defined by the pair of strategies. The second condition states that on all x not in L, player 0 has a strategy such that, for every strategy of player 1, V outputs 1 with probability at most ϵ on the debate defined by the pair of strategies.

2-9 This definition allows "one-sided error," analogous to the type of errors that are allowed in the complexity class *coRP*. We could also define a class of PCDSs with "zero-sided error," with three possible outputs, 1, 0, and Λ, for "player 1 won," "player 0 won," and "I don't know who won," respectively. In this case, the verifier must never declare the losing player to be a winner, but it may, both in the case that $x \in L$ and in the case that $x \notin L$, say that it doesn't know who won. We will see in Corollary 2 that this definition also gives the class *PSPACE*.

2-10 As in the theory of PCPSs, we say that the verifier makes $q(n)$ queries if the number of bits of π read by the verifier is at most $q(n)$ when the input is of size n. The verifier V in a PCDS is required to be nonadaptive (the bits of π read by V depend solely on the input and the coin flips). If L has a PCDS with error probability $\frac{1}{2}$ in which V flips $O(r(n))$ coins and reads $O(q(n))$ bits of π, we say that $L \in PCD(r(n), q(n))$.

2-11 It will be convenient in later proofs to reason about a generalized class, $GPCD(r(n), q(n))$. This class is defined exactly as $PCD(r(n), q(n))$, except that the verifier of a GPCDS (Generalized Probabilistic Debate System) nonadaptively queries $O(q(n))$ *rounds* of the debate π (rather than $O(q(n))$ *bits* of the debate). Thus, in a GPCDS, there is no restriction on the number of bits queried by the verifier in each round.

2-12 Next, we give some definitions relating to approximability of *PSPACE*-hard functions. Let f be any real-valued function with domain $D \subseteq \{0,1\}^*$. Let A be an algorithm that, on input $x \in \{0,1\}^*$, produces an output $A(x)$. We say that A *approximates* f *within ratio* $\epsilon(n)$, $0 < \epsilon(n) < 1$, if for all $x \in D$, $\epsilon(|x|) \leq A(x)/f(x) \leq 1/\epsilon(|x|)$. If $\epsilon(n) > 1$, then "A approximates f within ratio $\epsilon(n)$" means that $1/\epsilon(|x|) \leq A(x)/f(x) \leq \epsilon(|x|)$. If algorithm A computes the function g, we also say that g approximates f within ratio ϵ.

2-13 The function f has a *polynomial-time approximation scheme*, or PTAS, if for each ϵ, $\epsilon > 0$, there is a polynomial-time algorithm A that approximates f within ratio ϵ [GJ79].

2-14 We say that a function g is *PSPACE-hard* if $PSPACE \subseteq P^g$, i.e., if every language in *PSPACE* is polynomial-time reducible to g. By "approximating f within ratio $\epsilon(n)$ is *PSPACE*-hard," we mean that, if g approximates f within ratio $\epsilon(n)$, then g is *PSPACE*-hard.

2-15 Finally, we review some facts about algebraic techniques for encoding strings. We will use them to prove that $PCD(\log n, q(n)) = GPCD(\log n, q(n))$, which is Theorem 2 below. Let x be an element of $\{0,1\}^n$. The *robust encoding* $E_R(x)$ of x is an element of $\{0,1\}^{2^n}$, indexed by elements v of $\{0,1\}^n$, such that the vth bit of $E_R(x)$ is $\sum_{i=1}^n v_i x_i \bmod 2$. Let l be $\lceil \log n / \log\log n \rceil$ and p be a prime in the interval $[\log^c n, 2\log^c n]$, where c is a constant determined in the proof of Lemma 1. Let $I = \{1, 2, \ldots, \lceil \log n \rceil\} \subset Z_p$. Since $|I^l| \geq n$, we can fix an injective map from $\{0,1\}^n$ to the set of functions that map I^l to $\{0,1\}$. Regard x as one of these functions $I^l \rightarrow \{0,1\}$. There exists an l-variable polynomial X over Z_p, of degree at most $l(|I|-1)$, that agrees with x on all $\alpha \in I^l$. The *low-degree encoding* $E_P(x)$ of x is any such function $X\colon Z_p^l \to Z_p$. Let $(y_1, y_2, \ldots, y_{p^l})$ denote $E_P(x)$.

2-16 The encoding E that is used in the proofs of Theorems 1 and 2 is given by the formula:

$$E(x) \equiv (E_R(y_1), E_R(y_2), \ldots, E_R(y_{p^l}))$$

Note that $|E(x)| = \mathrm{poly}(|x|)$.

2-17 The following expression $\Delta_{E'}$ is defined for every function $E'\colon S \to \Sigma^{n'}$, every set S, and every alphabet Σ. Typically, E' will be an error-correcting code, and $\Delta_{E'}(y)$ measures the fraction of symbols of y that must be changed to transform y into a code word. Let y be an element of $\Sigma^{n'}$. Then

$$\Delta_{E'}(y) \equiv \frac{\min_{x \in S}(\mathrm{Ham}(y, E'(x)))}{n'}$$

where, if $y_i = \sigma_{i1} \cdots \sigma_{in'}$, $1 \le i \le 2$, $\sigma_{ij} \in \Sigma$, then $\mathrm{Ham}(y_1, y_2)$ is the *Hamming distance* between y_1 and y_2, i.e., the number of j for which $\sigma_{1j} \neq \sigma_{2j}$. Also, we define $\Delta(y_1, y_2)$ as the fractional Hamming distance between y_1 and y_2, defined on pairs in which y_1 and y_2 have the same length. That is, $\Delta(y_1, y_2) = \mathrm{Ham}(y_1, y_2)/n$ where $|y_1| = |y_2| = n$.

3 Complexity-Theoretic Results

3-1 Our first theorem on the language-recognition power of PCDSs addresses the question of whether verifiers that read $O(q(n))$ rounds of the debate tape have more power than verifiers that read $O(q(n))$ bits of the debate tape. Surprisingly, for $r(n) = \Omega(\log n)$, the answer is no. This result relies heavily on the following fact about probabilistically checkable proofs.

Theorem 1 (Arora et al. [ALM+92]) *Let k, n_1, $n_2, \ldots,$ and n_k be integers and $\varphi(x_1, x_2, \ldots, x_k)$ be an NP predicate, where $|x_i| = n_i$, for $i = 1, 2, \ldots, k$. Let $n = \sum_{i=1}^k n_i$. Then there exists a verifier V that uses $O(\log n)$ random bits and reads $O(k)$ bits of a proof $\pi = (\pi_1, \pi_2, \ldots, \pi_k, y)$ of length* poly(n) *with the following properties:*

- *If $\varphi(x_1, x_2, \ldots, x_k) = 1$, there is a y such that, with probability 1, V accepts $\pi = (E(x_1), E(x_2), \ldots, E(x_k), y)$.*

- *For every $\pi = (\pi_1, \ldots, \pi_k, y)$, if V accepts with probability greater than $\frac{1}{2}$, then $\varphi(E^{-1}(\pi_1), E^{-1}(\pi_2), \ldots, E^{-1}(\pi_k)) = 1$.*

3-2 In the next theorem, given an *NP* predicate φ of arity $\Theta(q(n))$, we will need to refer to the string y whose existence is guaranteed by the first bullet above. Thus, we say y is a *PCP*$(\log n, q(n))$ *proof* for the predicate $\varphi(x_1, x_2, \ldots, x_{\Theta(q(n))})$, and we refer to $E(x_1), E(x_2), \ldots, E(x_{\Theta(q(n))})$ as the *inputs* to the *PCP*$(\log n, q(n))$ proof y.

Theorem 2 *For every $q(n)$, PCD$(\log n, q(n))$ = GPCD$(\log n, q(n))$.*

Proof of Theorem 2-1 **Proof of Theorem 2** The direction *PCD*$(\log n, q(n)) \subseteq$ *GPCD*$(\log n, q(n))$ is immediate; we consider the other direction. Given a GPCDS (D, V) with players 1 and 0, we construct an equivalent PCDS (D', V') with players $1'$ and $0'$. Suppose that D has N rounds on a given input and assume without loss of generality that N is even. Then D' has $N+1$ rounds. Roughly, the idea is that in rounds 1 through N, the players $0'$ and $1'$ play as in debate D, except they encode their moves using the encoding E defined in Section 2. In round $N+1$, player $1'$ writes additional information, in order to convince V' that V would have accepted on the decoded debate in rounds 1 through N, or that player $0'$ has not properly encoded some of its moves.

Proof of Theorem 2-2 We first describe a strategy of player $1'$ on input $x \in L$ that causes V' to accept with probability 1. Since $x \in L$, there exists a strategy for 1 on x such that V accepts with probability 1. This induces the following strategy for $1'$ in D' in the first N rounds. At the ith round, player $1'$ first "decodes" each of the previous rounds 1 through $i-1$. Then, with respect to this sequence of moves, $1'$ finds the move m that 1 would write in round i according to its winning strategy in D. Player $1'$ plays $E(m)$ in round i. Here, by "decoding" a given move, we mean finding the move M that minimizes the Hamming distance from $E(M)$ to the given move.

Proof of Theorem 2-3 The string written by $1'$ in round $N+1$ is constructed so that V' can check that, for each random string R of V, either V outputs 1 if it is given this random string and the decoded debate of rounds $1, \ldots, N$ or that some move of $0'$ read by V on random string R is a bad encoding. We say that a string y is a bad encoding if $\Delta_E(y) > \epsilon$, where $\epsilon > 0$ is a parameter that is determined in Lemma 1 below, and $\Delta_E(y)$ is as defined at the end of Section 2. Let $V(R)$ denote the execution of verifier V with random string R.

Proof of Theorem 2-4 More precisely, the move of $1'$ in round $N+1$ contains the following strings. First, it contains the encoding of each move of player $0'$. Let x_i be the encoding of the move player $0'$ in the ith round. Note that if, in round i, player $0'$ writes an encoding of a move of debate system D, then x_i is the encoding of the encoding of a move in the debate system D. Second, it contains a $PCP(\log n, 1)$ proof π_{ij} for each bit of each move that $0'$ played. The (i, j)th of these proofs proves that the string encoded by $1'$ has the same value as the jth bit of the ith move played by $0'$. (These proofs enable the verifier V' to check that $1'$ properly encoded $0'$'s moves.) Note that all these proofs have as input only one string encoded by $1'$ and one bit played by player $0'$; therefore $PCP(\log n, 1)$ proofs exist, by Theorem 1. Lastly, for each random seed R, the move of $1'$ in round $N+1$ contains a $PCP(\log n, q(n))$ proof π_R that has one of the following properties:

- when the moves played by $1'$ and $0'$ are decoded and used as the debate tape in D, $V(R)$ outputs 1, or

- at least one of the moves corresponding to moves of player 0 that $V(R)$ reads is a bad encoding.

The inputs to the above statement are $O(q)$ moves by $1'$, corresponding to the $O(q)$ moves of player 1 read by $V(R)$, and also the encoding of the moves of player $0'$ that are in the last move of player $1'$. Observe that all the inputs are encoded by $1'$. Lemma 1 shows that the problem of recognizing a bad encoding is in NP. Thus, by Theorem 1, the $PCP(\log n, q(n))$ proof needed in the second case exists.

Proof of Theorem 2-5 To summarize, in the last round player $1'$ writes a string of the form

$$((x_i)_i, (\pi_{ij})_{ij}, (\pi_R)_R)$$

where R ranges over random seeds of V, $i \in \{2, 4, \ldots, N\}$ and $j \in \{1, 2, \ldots, l_i\}$, where l_i is the number of bits in the ith move of player $0'$. See Figure 1.

Proof of Theorem 2-6 Now we can describe the verifier V'. First V' chooses a random seed R and computes the indices $i_1, i_2, \ldots, i_{q(n)}$ of rounds that V queries using the random seed R. It then probabilistically checks π_R with encoded inputs m_{i_k} for each even i_k, and x_{i_k} for each odd i_k, where m_i is the move in round i. Additionally, for all odd i_k, V' chooses a $j \in \{1, 2, \ldots, l_{i_k}\}$ uniformly at random and probabilistically checks $\pi_{i_k j}$ with input x_{i_k} and the j bit of m_{i_k}.

Proof of Theorem 2-7 Let us show that the debate system (D', V') is a PCDS for L with error probability $1 - \frac{\epsilon}{2}$. (Note that this implies the theorem since the error

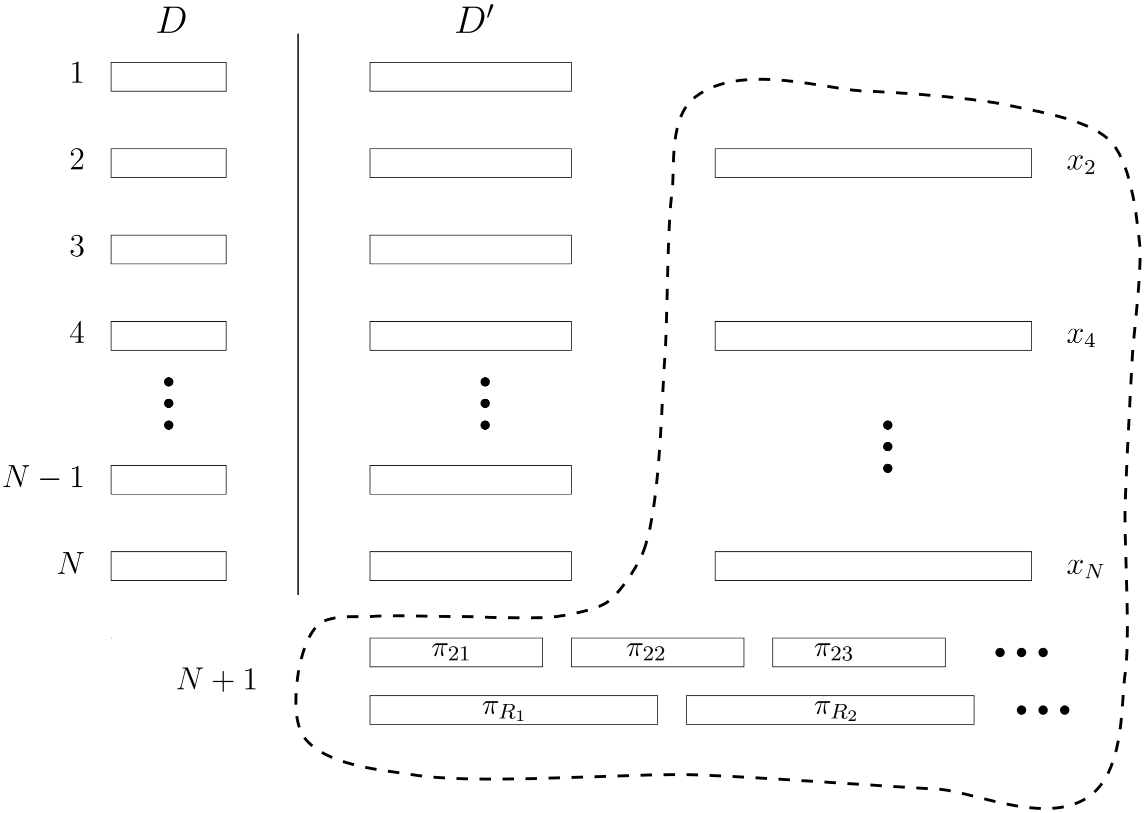

Figure 1: The debate transformation. Everything within the dashed contour is part of move $N+1$ of $1'$.

probability can be made less than $\frac{1}{2}$ by repeating the verification a constant number of times in parallel.)

Proof of Theorem 2-8 In the case that $x \in L$, it follows easily from the construction that V' accepts with probability 1, on the strategy for player $1'$ described above.

Proof of Theorem 2-9 If $x \notin L$, then 0 has a strategy such that V rejects with probability at least $\frac{1}{2}$. Consider the strategy for $0'$ induced by this strategy of player 0 (defined in the same way as the induced strategy for player $1'$ was defined for rounds 1 through N). Note that player 0 is declared the winner on input x for at least half of the random seeds R. Fix some such R. Then, one of two events must be true. The first is that the proof π_R causes the verifier V' to reject with probability at least $\frac{1}{2}$ (by Theorem 1). The second is that, for some i such that round i is read by $V(R)$, the string x_i written by player 1 in round $N+1$ is not the encoding of the string m actually played by $0'$ in round i, but of another string, say x, where $\Delta(x,m) > \epsilon$ (where Δ is the fractional Hamming distance function defined at the end of Section 2).

Proof of Theorem 2-10 Thus, with probability at least ϵ, V' chooses a j such that $x_j \neq m_j$, and the proof π_{ij} causes V' to accept with probability at most $\frac{1}{2}$. Thus V' rejects in the second event with probability at least $\frac{\epsilon}{2}$.

Proof of Theorem 2 □

3-3 We now prove a technical lemma that was used in the previous argument.

Lemma 1 *For some $\epsilon > 0$, there exists a polynomial-time predicate F with the following property. For every y, there exists a z such that $F(y,z) = 1$ if and only if $\Delta_E(y) \geq \epsilon$.*

Proof of Lemma 1-1 **Proof of Lemma 1** The code E is efficiently decodable in the following sense: There exists a polynomial-time computable function G such that, for every z, x:

$$\Delta(z, E(x)) \leq \frac{1}{12} \Rightarrow G(z) = x$$

Let $F(z)$ be the polynomial-time computable predicate $\Delta(z, E(G(z))) > \frac{1}{12}$.

Proof of Lemma 1-2 The decoding function G is constructed from the decoding functions for the two codes that E is composed of. Let $z = (z_1, z_2, \ldots, z_{p^l})$. First note that for each $i \in \{1, 2, \ldots, p^l\}$, we can find, by exhaustive search, y_i that minimizes $\Delta(z_i, E_R(y_i))$, with ties broken arbitraily. This defines a function $g\colon Z_p^l \rightarrow Z_p$.

Proof of Lemma 1-3 It is easy to see that the multivariate polynomial self-corrector due to Gemmell and Sudan [GS92] can be used to construct a decoding function H such that, for every g, x:

$$\Delta(g, E_P(x)) \leq \frac{1}{3} \Rightarrow H(g) = x \tag{1}$$

The self-corrector in [GS92] is randomized, but in our context it uses $O(\log n)$ random bits and can therefore be made deterministic at the expense of a polynomial factor in running time.

Proof of Lemma 1-4 Assume that $\Delta_E(z) \leq \frac{1}{12}$. Thus there exists x such that $\Delta(z, E(x)) < \frac{1}{12}$. Let f be the multivariate polynomial $E_P(x)$. Note that, for every $a \neq a'$, $\Delta(E_R(a), E_R(a')) = \frac{1}{2}$. This implies that for each i such that $g(\alpha_i) \neq f(\alpha_i)$, $\Delta(z_i, E_R(f(\alpha_i))) \geq \frac{1}{4}$. Hence $\Delta_E(z) \geq \frac{1}{4}\Delta(f, g)$. Thus $\Delta(f, g) \leq \frac{1}{3}$, and Equation 1 implies that $H(g) = x$.

Proof of Lemma 1 □

3-4 If $r(n) = o(\log n)$, then it is not necessarily true that $GPCD(r(n), q(n)) \subseteq PCD(r(n), q(n))$. For example, it is clear that $GPCD(0, 1)$ is the entire polynomial-time hierarchy, whereas $PCD(0, 1)$ is just P.

3-5 We now turn to the proof of our main theorem: Every language in *PSPACE* is recognized by a debate system in which the verifier uses $O(\log n)$ random bits and reads $O(1)$ rounds (equivalently, by Theorem 2, $O(1)$ bits) of the debate. The following notation is used in the proof. Let $\Phi = \exists x_1 \forall x_2 \ldots \exists x_n \phi(x_1, \ldots, x_n)$ be an instance of the problem (QBF); without loss of generality, we assume that quantifiers alternate strictly. This instance Φ of QBF can be thought of as a game between two players, an "existential" player (player 1) who sets the odd-numbered variables, and a "universal" player (player 0) who sets the even-numbered variables. This view of the QBF problem as a game motivates the following definitions. The assignment tree A for Φ is the complete binary tree of depth n, where one edge from every internal node is labeled "true" and the other "false." Each path P in the tree corresponds to an assignment of the variables; we say P satisfies ϕ if this assignment satisfies ϕ. Call edges at odd-numbered levels (that is, corresponding to existentially quantified variables) 1-edges and edges at even-numbered levels 0-edges. An $\exists$-strategy subtree A_1 is a subtree of A that has two 0-edges from each node at each even level and one 1-edge from each node at each odd level. Similarly, a $\forall$-strategy subtree A_0 is a subtree of A that has two 1-edges from each node at each odd level and one 0-edge from each node at each even level. An $\exists$-strategy subtree A_1 is *optimal* if it maximizes (over all $\exists$-strategy subtrees) the number of paths that satisfy ϕ. Similarly, a $\forall$-strategy subtree A_0 is optimal if it maximizes (over all $\forall$-strategy subtrees) the number of paths that do not satisfy ϕ. Note that if $\Phi \in$ QBF, all paths of an optimal $\exists$-strategy subtree satisfy ϕ, whereas if $\Phi \notin$ QBF, then no path of an optimal $\forall$-strategy subtree satisfies ϕ.

3-6

Theorem 3 *PSPACE* $=$ *GPCD*$(\log n, 1)$.

Proof of Theorem 3-1 **Proof of Theorem 3** The direction *GPCD*$(\log n, 1) \subseteq$ *PSPACE* is straightforward. To prove the other direction, we show that QBF $\in$ *GPCD*$(\log n, 1)$. Let $\Phi = \exists x_1 \forall x_2 \ldots \exists x_n \phi(x_1, \ldots, x_n)$ be an instance of QBF in which quantifiers alternate strictly. Let A_0, A_1 be optimal $\forall$- and $\exists$-strategy subtrees of Φ, respectively. Note that $\Phi \in$ QBF if and only if the unique path P of length n that is in both A_0 and A_1 satisfies ϕ.

Proof of Theorem 3-2 We give a debate format and a protocol of players 0 and 1 that enable the players to record parts of the strategy subtrees A_0 and A_1 in such a way that a verifier can efficiently check whether or not path P satisfies ϕ.

Proof of Theorem 3-3 In the debate, the players alternately play rounds, starting with player 1. Roughly, in one round, player i does two things: presents a challenge to player $1-i$ and responds to previous challenges written by player $1-i$. A *challenge* by player $1-i$ to player i is simply a path of the strategy subtree A_i that ends in a $(1-i)$-edge. The *response* of player i to this challenge is the edge that extends this path in A_i (if the path is not already of length n).

Proof of Theorem 3-4 Note that a challenge *to* player 1 (respectively 0) has to be a path of A_1 (respectively A_0). How can player 0 write down such a path *without knowing* A_1? Essentially, in round t, player 0 must present paths that are consistent with what player 1 wrote in rounds 1 through $t-1$. We will elaborate on this point below.

Proof of Theorem 3-5 Play proceeds as follows: In round t, one of the players writes a tree T_t. If player i is honest, then in round $t \equiv i \bmod 2$, player i writes the (unique) smallest subtree T_t of A_i such that T_t contains all challenges by player $1-i$ at rounds $j < t$. The debate format is thus $(f(n), g(n))$, where $g(n)$ is chosen to make the error probability small enough and $f(n)$ is chosen to allow encoding of binary trees of the appropriate form; we will explain in detail how to choose f and g when we prove correctness of the protocol.

Proof of Theorem 3-6 Before specifying the algorithm of the verifier V, we give some examples of debates in which player 1 is honest. The values assigned to the variables depend on the input formula and are unimportant for the purposes of this discussion.

Example 1 Suppose that player 0 is honest as well (see Figure 2). In round t, $1 \leq t \leq n$, player $i \equiv t \bmod 2$ assigns a value to x_t by writing down a path of length t that extends the path written in round $t-1$. The path written in round n is P, the intersection of A_1 and A_0. In rounds $n+1$ through g, the path P is repeated in each round. Verifier V will declare player 1 the winner (i.e., accept the input) if and only if P satisfies ϕ.

Example 2 Suppose that $\Phi \in$ QBF and that player 0 cheats in an effort to convince V to reject (see Figure 3). One way player 0 may do this is to play a move that is not a subtree of A_1—that is, to lie about player 1's previous moves.

Note that, in round 3, the honest player 1 need only extend the FF path of the tree played by 0 in round 2. Because the TT path that appears in round 2 is not in A_1, i.e., because it is not consistent with player 1's move in round 1, it does not satisfy the definition of a challenge.

Example 3 Once again, suppose that Φ is true and that player 0 cheats. This time, player 0 does so by lying about player 0's own moves, rather than those of player 1 (see Figure 4).

In this example, both the FF path of round 4 and the FT path of round 2 require a response by player 1 in round 5, because both are legitimate challenges, i.e., neither is inconsistent with player 1's previous moves.

Proof of Theorem 3-7 A move T_t by an honest player i must have the following properties: (i) at most one edge from every node is an i-edge (because T_t is a subtree of A_i); and (ii) the edge to every leaf of depth $< n+1$ is an i-edge (because player i responds to each recorded challenge by player $1-i$). A tree T_t satisfying these two properties is *valid*. (In Example 3 above, player 0 should choose which of the paths FTT or FFF of round 5 to extend in round 6, because extending both of them would result in an invalid move T_6.) Also, we define a *valid challenge* by player $1-i$ to player i at round j to be the (unique) longest challenge that lies in T_j, if T_j is valid.

Proof of Theorem 3-8 Note that if player i is honest, then T_{t-2} is a subtree of T_t. This is because both T_{t-2} and T_t respond to all valid challenges from rounds $j < t-2$. Also,

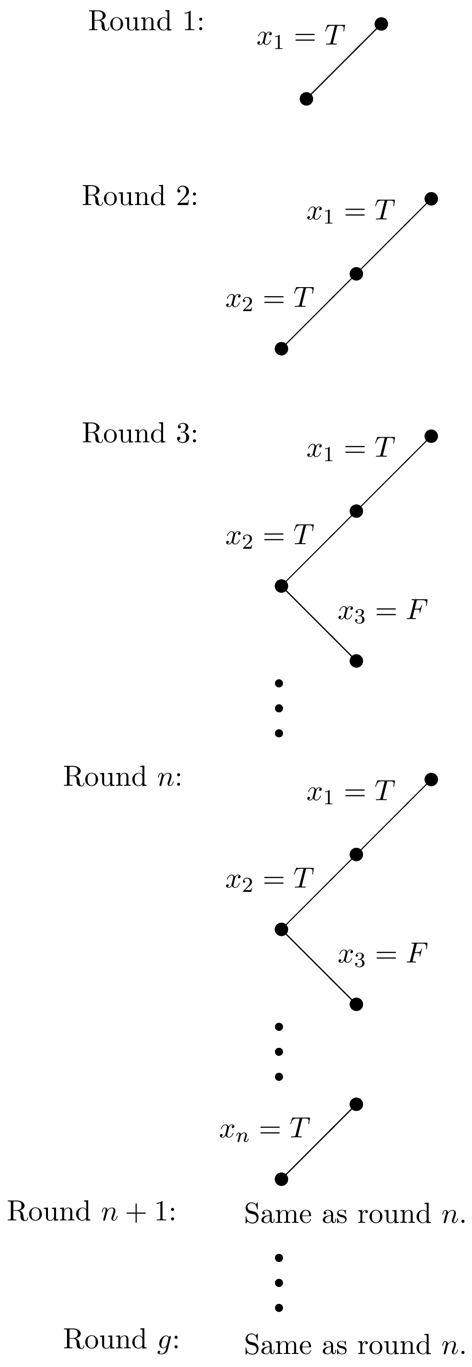

Figure 2: Both player 1 and player 0 are honest.

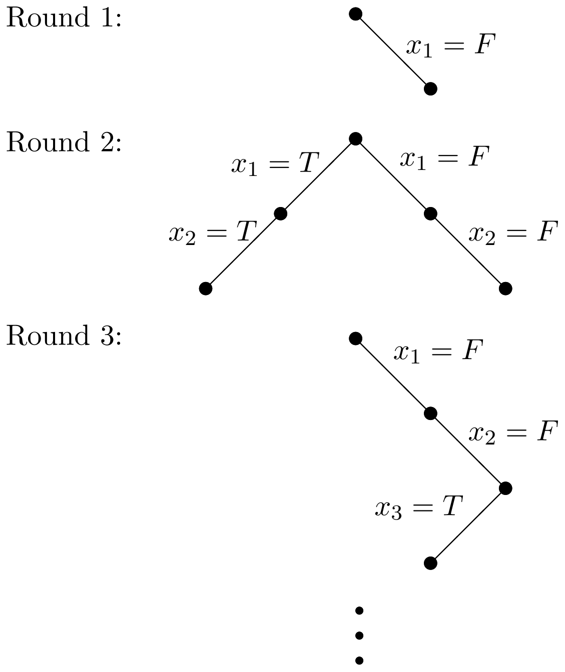

Figure 3: Player 1 is honest and player 0 is lying about player 1's moves.

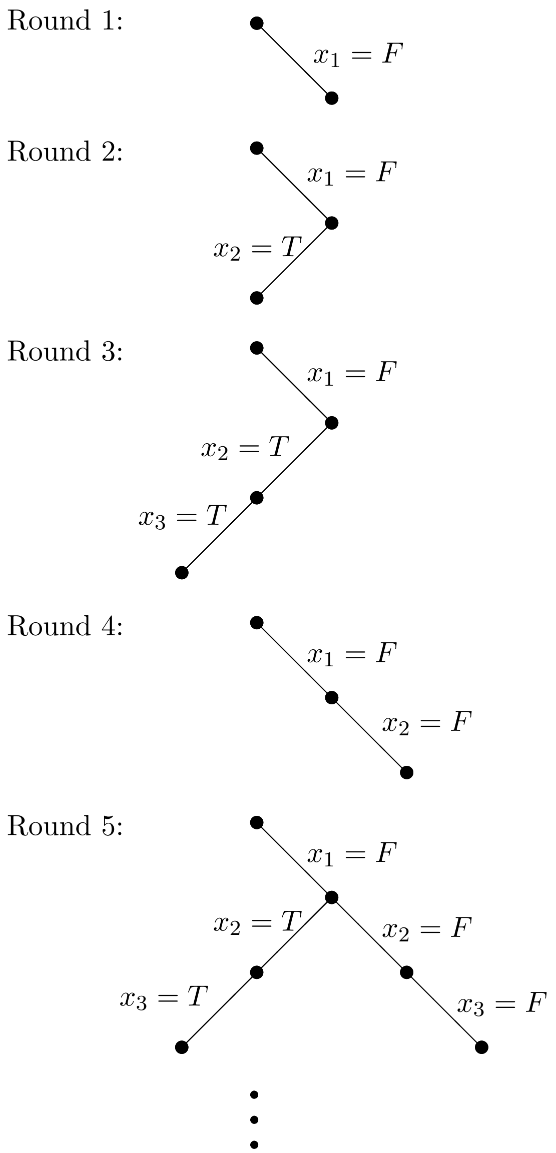

Figure 4: Player 1 is honest and player 0 is lying about player 0's own moves.

T_t has at most one more leaf than T_{t-2}, namely, the leaf of the path that contains the valid challenge at round $t-1$, if it lies on a different path from previous valid challenges.

Proof of Theorem 3-9 We take the number of rounds to be $g = g(n) > 4n$; this will ensure that the error probability is no more than $2(n-1)/(g-3)$, as explained below. We let $f(n)$, the length of a round, be such that every binary tree of depth at most n with at most $g(n)$ leaves can be described using $f(n)$ bits, via some simple encoding of trees to strings. This is sufficient, since the number of leaves of $T_{g(n)}$ is at most $g(n)$.

Proof of Theorem 3-10 Below, we state V's algorithm formally and prove it correct, but first we explain intuitively how V can catch a cheating player by examining only a constant number of rounds of the debate. Suppose that the input formula is true and hence that player 1 follows the protocol honestly. If they are valid, the trees T_g (player 1's last move) and T_{g-1} (player 0's last move) intersect in a unique path, say $P(g)$. If it is of length n, then $P(g)$ satisfies ϕ, and V will certainly declare player 1 the winner. Thus player 0 must try to "stall" and prevent $P(g)$ from growing to length n. Consider a move T_t, where $t < g-1$ is even. The move T_t contains a (unique) longest prefix of $P(g)$, say $P(t)$. Informally, we say that player 0 "cooperates" if $|P(t)| > |P(t-1)|$. Player 0 can cooperate in fewer than n rounds, or else $P(g)$ is of length n. Essentially, our formal proof shows that, if player 0 indeed cooperates in fewer than n rounds, then many intermediate moves T_t are either invalid or not subtrees of the final move T_{g-1}. Either way, V can detect with constant probability that player 0 is cheating by examining only a constant number of randomly chosen moves.

Proof of Theorem 3-11 We now give a formal statement of V's algorithm and a proof of its correctness. Verifier V first examines the last subtrees, T_{g-1} and T_g, written by players 0 and 1 respectively. If T_{g-1} is not valid, V accepts; if T_g is not valid, V rejects. Otherwise, let $P(g)$ be the (unique) path in both T_{g-1} and T_g. If the length of $P(g)$ is n and $P(g)$ satisfies ϕ, V accepts. If the length of $P(g)$ is n and $P(g)$ does not satisfy ϕ, then V rejects.

Proof of Theorem 3-12 Otherwise, the length of $P(g)$ is less than n. In this case, V chooses a random round t, $1 < t < g$, in which player 1 plays, and examines rounds t and $t-1$. If T_t is not valid or is not a subtree of T_g, V rejects. Similarly, if T_{t-1} is not valid or is not a subtree of T_{g-1}, V accepts. Otherwise, let P' be the longest path in both T_{t-1} and T_t. If the last edge of P' is not a 0-edge, then V rejects. Otherwise, V accepts. This completes the statement of V's algorithm.

Proof of Theorem 3-13 Now, suppose that $\Phi \in$ QBF and that player 1 is honest. We show that V accepts with probability 1. This is true if the length of $P(g)$ is n, since in this case $P(g)$ is a path in A_1 and hence satisfies ϕ. If the length of $P(g)$ is less than n, then for all rounds t in which player 1 plays, T_t is valid and a subtree of T_g. Suppose that T_{t-1} is also valid and a subtree of T_{g-1}. It remains to show that the longest path P' in both T_{t-1} and T_t must end in a 0-edge.

Proof of Theorem 3-14 First, note that the length of P' must be $< n$; otherwise P' is contained in both T_{g-1} and T_g and therefore $P' = P(g)$, which contradicts our assumption that the length of $P(g)$ is $< n$. Also, since T_{t-1} is valid, all paths of T_{t-1} of length $< n$ that end in a 1-edge are followed by a 0-edge. Furthermore, if P' ends in a 1-edge, the 0-edge following P' in T_{t-1} must also be in T_t, since the path formed by P' and this 0-edge is a prefix of the valid challenge of player 0 to player 1 at round $t-1$, and player 1 is honest. This contradicts the fact that P' is the longest path in both T_{t-1} and T_t. Hence P' must end in a 0-edge.

Proof of Theorem 3-15 We next show that if $\Phi \notin$ QBF and player 0 is honest, then V accepts with probability at most $2(n-1)/(g-3)$. If the length of $P(g)$ is n, V rejects, since in this case $P(g)$ is a path in A_0, and therefore does not satisfy ϕ. Hence suppose that the length of $P(g)$ is less than n; thus player 1 cannot be honest. We claim that in this case, there can only be $n-1$ values of t such that V accepts when rounds t and $t-1$ are examined. Since V chooses t randomly and uniformly from $(g-3)/2$ choices, the error probability is at most $2(n-1)/(g-3)$. The number of choices for t is $(g-3)/2$, because V never chooses player 1's first move, since it is not preceded by a move of player 0.

Proof of Theorem 3-16 If T_t is valid, let $p(t)$ be the length of $P(t)$, where $P(t)$ is the longest prefix of $P(g)$ in T_t. For every t such that player 1 plays in round t, we show that, if V accepts on examining rounds t and $t-1$, then $p(t) > p(j)$ for all $j < t$ such that T_j is valid. This implies the claim, since then V accepts only the first round for which $p(i) = 1, p(i) = 2, \ldots, p(i) = n-1$, and there are at most $n-1$ such rounds.

Proof of Theorem 3-17 Suppose, then, that V accepts on examining rounds t and $t-1$. Suppose that $j < t$ is a round of player 1, where T_j is valid. We need to show $p(j) < p(t)$. We first show that $P(j)$ is a prefix of $P(t-1)$, which implies that $p(j) \leq p(t-1)$. This is because $P(j)$, with the last edge removed if it is a 0-edge, is the prefix of a valid challenge of player 1 to player 0 at round j, and since player 0 is honest, player 0 responds to this challenge at round

$t-1$. To complete the proof, we show that $p(t-1) < p(t)$. Note that it must be the case that P', the longest path in both T_{t-1} and T_t, ends in a 0-edge, since V accepts. Also, this path must be a prefix of $P(t-1)$ and $P(t)$. In fact, this path must equal $P(t-1)$. (Otherwise, the 1-edge following P' in $P(t-1)$ must be in T_g. Since T_t is a subtree of T_g, this 1-edge must be the 1-edge of T_t following P', contradicting the fact that P' is the longest path in both T_{t-1} and T_t.) Finally, the 1-edge following P' in T_t must be in T_g (since V checks for this) and also in T_{g-1} (since player 0 is honest). Thus $P(t)$ contains $P(t-1)$ as a prefix, and it also contains an additional 1-edge. This implies that $p(t) > p(t-1) \geq p(j)$, as required.

Proof of Theorem 3 □

Combining Theorem 3 with Theorem 2, we obtain the following result.

Corollary 1 *PSPACE* $=$ *PCD*$(\log n, 1)$.

3-7 We can also obtain a characterization of debate systems that allow "zero-sided" error. Let a ZPCDS be a PCDS for which the verifier returns one of the three possibilities "player 1 wins," "player 0 wins," and "I don't know who wins," in which the verifier is always right in the first two cases, and the probability of the third case is at most $\epsilon < \frac{1}{2}$.

Corollary 2 *PSPACE* $=$ *ZPCD*$(\log n, 1)$.

Proof of Corollary 2 This follows from the fact that *PSPACE* is closed under complement. Given an $L \in$ *PSPACE*, our main result shows that L and the complement of L have one-sided PCDSs (D, V) and $(\overline{D}, \overline{V})$, respectively, with error probability $\frac{1}{2}$. Now consider the debate in which both D and $\overline{D}$ are performed, and the verifier declares 0 the winner if V rejects, declares 1 the winner if $\overline{V}$ rejects, and otherwise says that it does not know the winner. It is easy to see that this verifier will have zero-sided error and will declare a winner with probability at least $\frac{1}{2}$.

Proof of Corollary 2 □

4 Functions that are *PSPACE*-Hard to Approximate

4-1 In this section, we give many examples of *PSPACE*-hard functions that are hard to approximate. We consider maximization versions of the problem of deciding whether a quantified Boolean formula is true and show that one version can be approximated within a ratio of $\frac{1}{2}$, yet there is some constant $\epsilon < 1$ such that approximating the function within a ratio of ϵ is *PSPACE*-hard. We prove even stronger results for the maximization versions of several other *PSPACE*-complete problems. For example, there is a constant $\epsilon > 0$ such that approximating Finite Automata Intersection (cf. Kozen [Koz77]) and Generalized Geography (cf. Schaefer [Sch78]) within ratio n^ϵ is *PSPACE*-hard.

4-2 We first consider variants of a well-known *PSPACE*-complete problem, that of deciding whether a quantified Boolean formula is satisfiable. In what follows, we consider quantified (Boolean) formulas in CNF (conjunctive normal form); that is, quantified formulas of the form:

$$\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$$

where each $Q_i \in \{\exists, \forall\}$, each x_i is a Boolean variable, and ϕ is in conjunctive normal form. If each clause of ϕ has exactly three literals, we say the quantified formula is in 3CNF. Let QSAT and Q3SAT be the sets of satisfiable quantified formulas in CNF and 3CNF, respectively.

4-3 Suppose that the variables of the formula are chosen, in order of quantification, by two players 0 and 1, where player 0 chooses the universally quantified variables and player 1 chooses the existentially quantified variables. If player 1 can guarantee that k clauses of ϕ will be satisfied by the resulting assignment, regardless of what player 0 chooses, we say that k clauses of Φ are *simultaneously satisfiable*. We let MAX-QSAT (respectively MAX-Q3SAT) be the function whose domain is the set of quantified formulas that maps a quantified formula (respectively quantified 3CNF formula) Φ to the maximum number of simultaneously satisfiable clauses. The results in [ALM$^+$92] and [AS92] show that MAX-QSAT is *NP*-hard to approximate within certain ratios.

Theorem 4 *There is a constant* $0 < \epsilon < 1$ *such that approximating* MAX-Q3SAT *within ratio* ϵ *is PSPACE-hard.*

Proof of Theorem 4-1 **Proof of Theorem 4** Let L be a language in *PSPACE*. We showed in Section 3 that L is in *PCD*$(\log n, 1)$. We reduce the problem of deciding whether a string x is in L to the problem of approximating the number of simultaneously satisfiable assignments of a quantified 3CNF formula.

Proof of Theorem 4-2 Let (D, V) be a PCDS for L, where V is polynomial-time bounded and uses $r(n) = O(\log n)$ random bits and $O(1)$ queries. Let $D = (f(n), g(n))$. Without loss of generality, we can assume that $f(n)$ and $g(n)$ are polynomials.

Proof of Theorem 4-3 Given an instance x of L, say of length n, we construct a quantified formula from (D, V) as follows. There are $f(n)g(n)$ ordered variables, one for each bit of a debate corresponding to the debate format. The first $f(n)$ variables, which correspond to the first round of a debate, are existentially quantified, the next $f(n)$ variables, which correspond to the second round, are universally quantified, and so on.

Proof of Theorem 4-4 For each sequence of random bits R of length $r(n)$, there is a subformula with $s = O(1)$ clauses, with variables corresponding to the bits of a debate that are queried on random sequence R. The subformula is satisfied by a truth assignment to the variables if and only if the verifier outputs 1, when the query bits are as in the truth assignment.

Proof of Theorem 4-5 If x is accepted by (D, V), then there exists a debate subtree such that, on each debate (or path of the tree), V outputs 1 on all of the random strings. Thus, player 1 can choose the values of the existential variables so that all clauses of the subformulas are simultaneously satisfiable.

Proof of Theorem 4-6 If the input x is not accepted by (D, V), then in every debate subtree, there is a debate on which V outputs 1 on at most $\frac{1}{2}$ of the random strings. Thus, no matter what truth assignment player 1 chooses for the existential variables, there is a choice for the universal variables such that at most $\frac{1}{2}$ of the subformulas are satisfied. Hence, at most $\frac{1}{2}$ of the subformulas are simultaneously satisfiable. Since each subformula contains $O(1)$ clauses, it follows that at most a constant fraction < 1 of the clauses are simultaneously satisfiable.

Proof of Theorem 4 □

4-4 Let $(\log n)$-MAX-Q-FORMULA be the function whose domain is the set of quantified formulas in which the "clauses" are general formulas with at most $\log n$ variables instead of being in CNF. The above result can be extended to prove the following.

Theorem 5 *There is a constant* $0 < \epsilon < 1$ *such that approximating* $(\log n)$-MAX-Q-FORMULA *within ratio* n^ϵ *is PSPACE-hard.*

Proof of Theorem 5 Use standard pseudorandom sampling techniques [CW89, IZ89].

□

4-5 We next consider a variant of Q3SAT called *Balanced* Q3SAT (QB3SAT). We say a quantified formula is balanced if every clause of the formula contains some existentially quantified variable. Balanced Q3SAT consists of those true quantified formulas in 3CNF form that are also balanced. This language is easily seen to be *PSPACE*-complete, by the following reduction from Q3SAT. An instance

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$$

of MAX-QSAT, where ϕ has m clauses, is mapped to the instance

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \exists w_1 \ldots \exists w_m \phi'(x_1, x_2, \ldots, x_n, w_1, \ldots, w_m)$$

where ϕ' is obtained from ϕ by replacing each clause $C_j = (l_1 \vee l_2 \vee l_3)$ of ϕ by the two clauses $(w_j \vee l_1 \vee l_2)$ and $(\bar{w}_j \vee l_3 \vee l_3)$, $1 \leq j \leq m$.

4-6 We define the corresponding function MAX-QB3SAT to be the function MAX-Q3SAT, restricted to the domain of balanced quantified formulas. This provides an example of a function that *can* be approximated to within some constant ratio but *cannot* be approximated to within an arbitrary constant ratio, unless *PSPACE* = *P*.

Lemma 2 *There is a polynomial-time algorithm that approximates* MAX-QB3SAT *within ratio* $\frac{1}{2}$.

Proof of Lemma 2-1 **Proof of Lemma 2** Let $\Phi = Q_1 x_1 \ldots Q_n x_n \phi(x_1, \ldots, x_n)$ be a balanced quantified formula in 3CNF. Let ϕ' be the formula obtained from ϕ by eliminating all universally quantified variables. Since ϕ is balanced, note that the number of clauses of ϕ' is equal to the number of clauses of ϕ (however, clauses may now have only one literal).

Proof of Lemma 2-2 Johnson [Joh74] showed that a truth assignment to the variables of ϕ' that satisfies at least $\frac{1}{2}$ of the clauses can be found in polynomial time. Player 1 can use this assignment to ensure that at least $\frac{1}{2}$ of the clauses of ϕ are satisfied, no matter what the values of the universally quantified variables are.

Proof of Lemma 2 □

Lemma 3 *There is a constant* $0 < \epsilon < 1$ *such that approximating* MAX-QB3SAT *within ratio* ϵ *is PSPACE-hard.*

Proof of Lemma 3 In Theorem 4, we reduced the problem of deciding whether an input x is accepted by a PCDS (D, V) to the problem of approximating a quantified formula Φ. The formula Φ can be converted into a balanced subformula as described in the discussion preceding Lemma 2. It is straightforward to show that the resulting balanced quantified formula Φ' also has the property that, if x is accepted by (D, V), then all clauses of Φ' are simultaneously satisfiable, but if x is not accepted, at most a constant fraction < 1 are simultaneously satisfiable.

Proof of Lemma 3 □

4-7 We next consider a problem from automata theory. Let FA-INT be the set of sequences $A_1, A_2, \ldots, A_m$ of deterministic finite-state automata having the same input alphabet Σ such that there exists a string w that is accepted by all the automata. Kozen [Koz77] showed the problem to be *PSPACE*-complete.

4-8 The function MAX-FA-INT has as its domain the set of all sequences $A_1, A_2, \ldots, A_m$ of deterministic finite-state automata having the same input alphabet Σ, and maps the sequence to the largest number k such that there exists a string w that is accepted by k of the automata. We prove a non-approximability result for MAX-FA-INT.

Theorem 6 *There is a constant* $0 < \epsilon < 1$ *such that approximating* MAX-FA-INT *within ratio* n^ϵ *is PSPACE-hard.*

Proof of Theorem 6-1 **Proof of Theorem 6** We describe a reduction from a new variant of MAX-Q3SAT. The function MAX-FIX-QSAT differs from MAX-Q3SAT in two ways. First, the domain is the set of quantified formulas, where the "clauses" are now the conjunction of $O(\log n)$ "subclauses," each the disjunction of three literals. Second, given an instance of this domain, the function outputs the maximum size k of a set of clauses that player 1 can guarantee will be satisfied, regardless of what assignment player 0 chooses for the universal variables. Thus for MAX-FIX-QSAT, the set of k satisfied clauses must be fixed in advance, that is, the set must be the same for all assignments of player 0. However, for MAX-Q3SAT, the set of k simultaneously satisfied clauses may depend on the assignments of player 0. The proof of Theorem 4 can be extended to show that there is some constant $\epsilon > 0$ such that approximating MAX-FIX-QSAT to within ratio n^ϵ is *PSPACE*-hard.

Proof of Theorem 6-2 We now describe our reduction from MAX-FIX-QSAT to MAX-FA-INT such that an instance of MAX-FIX-QSAT has a set of k clauses that player 1 can guarantee will be satisfied, if and only if the instance of MAX-FA-INT has k automata that accept the same string. Let

$$\Phi = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$$

be an instance of MAX-FIX-QSAT, where ϕ has m clauses. Moreover, assume without loss of generality that each variable appears in some clause.

Proof of Theorem 6-3 We first describe a set Valid of strings w; roughly, each string in this set describes possible choices of player 1 for the existentially quantified variables, against all possible choices of player 0 for the universally quantified variables. We will later construct m sets of automata, one set per clause, such that all automata in each of k sets accept some string w, which happens to lie in the set Valid, if and only if the quantified formula Φ has k simultaneously satisfiable clauses.

Proof of Theorem 6-4 Each string w in the set Valid is of the form $\$w_1\$w_2\$\ldots\$w_N\$$, where each $w_i = w_{i1}w_{i2}\ldots w_{in}$ is a binary string of length n, corresponding to a truth assignment to the variables of ϕ, and $N = 2^u$, where u is the number of universally quantified variables in Φ. Moreover, w must have properties (1), (2), and (3) below.

Proof of Theorem 6-5 Roughly, these properties are necessary and sufficient to ensure that w is in Valid if and only if the strings w_i correspond to paths of an $\exists$-assignment subtree that describes the assignments of player 1 against all possible assignments of player 0 (as explained in the paragraph preceding Theorem 3). Note that such a tree has N leaves. Moreover, these paths are enumerated in order, from left to right of the assignment subtree. We now list the three properties.

1. Suppose that x_j is universally quantified. Then, $w_{1j} = 0$ and $w_{Nj} = 1$. Moreover, if w_i is such that $w_{ij} = 1$ for all j such that x_j is universally quantified, then $i = N$.

2. Suppose that x_j is universally quantified and $i > 1$. Then, $w_{ij} = \bar{w}_{i-1,j}$ if for all $j' > j$ such that $x_{j'}$ is universally quantified, $w_{i-1,j'} = 1$, and $w_{ij} = w_{i-1,j}$ otherwise.

3. Suppose that x_j is existentially quantified and $i > 1$. Then, $w_{ij} = w_{i-1,j}$, unless for all $j' > j$ such that $x_{j'}$ is universally quantified, $w_{i-1,j'} = 1$. In the latter case, there is no restriction on w_{ij}.

We say w is not valid at index j, if property (2) or (3) fails for j.

Proof of Theorem 6-6 We consider two types of automata: "syntax checking" automata and "clause checking" automata. We will later use these automata to construct the automata that are used in the MAX-FA-INT instance. For $1 \leq j \leq n$, automaton S_j checks that the jth bit of each w_i has the property (2) or (3) above, depending on whether j is universally or existentially quantified. Automaton S_{n+1} checks that the $s are separated by strings of length exactly n, and that property (1) above holds. Clearly, a string is accepted by all $n+1$ of the automata if and only if it is in the set Valid. Moreover, each of these automata can be constructed to have poly(n) states.

Proof of Theorem 6-7 There are m clause-checking automata $C_1, \ldots, C_m$, each with poly(n) states, such that a string $w \in$ Valid is accepted by C_j if and only if on all paths of the corresponding assignment subtree, the jth clause is satisfied.

Proof of Theorem 6-8 We now construct m automata. The ith automaton A_i does the following checks on its input string.

1. Perform the check done by automaton S_{n+1}.
2. If x_j or $\bar{x}_j$ is a literal of C_i, then perform the check of S_j. (In this case, say that A_i *examines* bit x_j.)
3. Perform the check done by automaton C_i.

Check (2) can be done by an automaton with poly(n) states. Roughly, for each i, the automaton stores the the bits $w_{i-1,j}$ and w_{ij}, where bit x_j is examined by C_i. Also, the position of the rightmost universal index with value 0 in w_i is stored. This information is sufficient to perform the check of S_j on the string w_i. Since A_i performs the checks of three automata of size poly(n), A_i is also of size poly(n).

Proof of Theorem 6-9 If player 1 can guarantee that a fixed set of k clauses of ϕ are satisfied for all assignments of player 0, then there is a corresponding string w that is accepted by k automata.

Proof of Theorem 6-10 Conversely, suppose that $k > 0$ automata all accept some string w. Note that w may not be a member of Valid. However, w must pass check (1), because $k > 0$. Hence, suppose that w is not valid at I indices. We prove by induction on I that there exists a string w' in Valid such k automata accept w'. From this it follows that there is a set of k clauses of Φ that player 1 can guarantee will be satisfied.

Proof of Theorem 6-11 The base case, when $I = 0$, is immediate, for in that case $w = w'$. Suppose $I > 0$, and let j be the smallest invalid index. In what follows, if x_j is existentially (universally) quantified, we say that j is an existential (universal) index.

Proof of Theorem 6-12 If j is an existential index, we define w'_i as follows for $1 \leq i \leq N$: Let $w'_{ij} = 0$ and let $w'_{is} = w_{is}$ for $s \neq j$. The resulting string has $I - 1$ invalid indices. Furthermore, it is still accepted by k automata. (This is because none of the k accepting automata can possibly examine bit j.) We can now apply induction to complete the proof.

Proof of Theorem 6-13 Next, suppose that j is a universal index. The procedure to construct w' from w is more complicated in this case. We do it in stages. For each set of bits $b_1 \ldots b_{j-1}$, we consider separately the (unique) longest contiguous substring $w_r\$ \ldots \$w_{r'}\$$ of w (if any) such that $b_1 \ldots b_{j-1}$ is a prefix of each w_i, $r \leq i \leq r'$. Call this substring $w[b_1 \ldots b_{j-1}]$. In the next paragraph, we describe a new substring $w'[b_1 \ldots b_{j-1}]$ that is obtained from $w[b_1 \ldots b_{j-1}]$. We then let w' be the string obtained by concatenating the substrings $w'[b_1 \ldots b_{j-1}]$ in the appropriate order. Our construction ensures that w' has $I-1$ invalid indices and is accepted by all k automata that accept w.

Proof of Theorem 6-14 Therefore, fix $b_1 \ldots b_{j-1}$, and consider only the substring $w_r\$ \ldots \$w_{r'}\$$. Note that, for all universal indices $j' > j$, $w_{rj'} = 0$ if j' is valid, and $w_{r'j'} = 1$. (If this were not the case, it would contradict the facts that $1 \ldots j-1$ are valid indices and that $w_r\$ \ldots \$w_{r'}\$$ is the longest substring of w such that $b_1 \ldots b_{j-1}$ is a prefix of each w_i.) Let l be the smallest number such that, for all universal indices $j' > j$, $w_{lj'} = 1$.

Proof of Theorem 6-15 Let $w'_r\$ \ldots \$w'_l\$$ be such that, for $1 \leq i \leq l$, $w'_{ij} = 0$, and, for $s \neq j$, $w'_{is} = w_{is}$. Similarly, let $w''_r\$ \ldots \$w''_l\$$ be such that, for $1 \leq i \leq l$, $w''_{ij} = 1$ and, for $s \neq j$, $w''_{is} = w_{is}$.

Proof of Theorem 6-16 Then, the new string $w'[b_1 \ldots b_{j-1}]$ is $w'_r\$ \ldots \$w'_l\$w''_r\$ \ldots \$w''_l\$$.

Proof of Theorem 6-17 This completes the description of w'. The construction guarantees that (i) w' has $I-1$ invalid indices (namely, those invalid indices $j' > j$ of w) and (ii) w' is accepted by the k automata that accept w. Again, induction can be applied to complete the proof.

Proof of Theorem 6 □

4-9 Generalized Geography is an abstraction of a popular car game in which two players alternately list the names of countries, each beginning with the last letter of the previous country, until one player cannot list a new country. A corresponding game can be played on a directed graph G that has a distinguished start node s. A marker is initially placed on s, and two players, 0 and 1, alternately move it along an edge, with the constraint that player 1 starts and each edge can be used only once. The first player unable to move loses. Schaefer [Sch78] defines GGEOG to be the set of pairs (G, s) such that player 1 has a winning strategy.

4-10 Informally, a natural optimization version of GGEOG is to compute how long player 1 can keep the game going, even if player 1 does not eventually win the game. We say (G, s) can be played for k rounds if player 1 has a strategy that causes the marker to move along k edges of the graph before the game ends. We define MAX-GGEOG to be the function whose domain is the set of pairs (G, s), where G is a directed graph with node s, that maps a pair (G, s) to the maximum number k of rounds that (G, s) can be played.

Theorem 7 *There is a constant* $0 < \epsilon < 1$ *such that it is PSPACE-hard to approximate* MAX-GGEOG *within ratio* n^ϵ*.*

Proof of Theorem 7-1 **Proof of Theorem 7** We first modify Schaefer's reduction [Sch78] to obtain a reduction from MAX-Q3SAT to MAX-GGEOG. We later describe a simple modification of our construction, to obtain a reduction from $(\log n)$-MAX-Q-FORMULA to MAX-GGEOG.

Proof of Theorem 7-2 Let $\Phi = Q_1 x_1 Q_2 x_2 \dots Q_n x_n \phi(x_1, \dots, x_n)$ be an instance of MAX-Q3SAT, where ϕ contains m clauses. In what follows, we assume that n is even and that $Q_n = \forall$; the reduction can easily be modified to handle the other cases.

Proof of Theorem 7-3 We construct from Φ an instance (G, s) of MAX-GGEOG such that, if a maximum of k clauses of Φ are simultaneously satisfiable, then (G, s) can be played for a maximum of $4n + kn^2 + O(1)$ steps. From this property, it follows that, given an approximate value for the length of the generalized geography game, an approximate value for the number of satisfiable clauses can be deduced. The graph G is composed of a "variable-setting" component, a "clause-testing" component and a "line." We describe each of these components in turn and also describe how they are interconnected.

Proof of Theorem 7-4 We first describe the variable-setting component. The node set is:

$$\begin{aligned} V_1 &= \{\, x_i, \bar{x}_i, u_i, v_i \mid Q_i = \exists, 1 \le i \le n \,\} \\ &\cup \{\, x_i, \bar{x}_i, u_i, v_i, w_i, \bar{w}_i, z_i \mid Q_i = \forall, 1 \le i \le n \,\} \\ &\cup \{u_{n+1}\} \end{aligned}$$

Proof of Theorem 7-5 Node u_1 is the start node s. The nodes $x_i, \bar{x}_i, 1 \le i \le n$ are referred to below as “literal” nodes. The edge set is:

$$\begin{array}{rcl} E_1 & = & \{\, (u_i, x_i), (u_i, \bar{x}_i), (x_i, v_i), (\bar{x}_i, v_i), (v_i, u_{i+1}) \mid Q_i = \exists, 1 \le i \le n \,\} \\ & \cup & \left\{ \begin{array}{l} (u_i, w_i), (u_i, \bar{w}_i), (w_i, x_i), (\bar{w}_i, \bar{x}_i) \\ (x_i, v_i), (\bar{x}_i, v_i), (v_i, z_i), (z_i, u_{i+1}) \end{array} \middle| Q_i = \forall, 1 \le i \le n \right\} \end{array}$$

Proof of Theorem 7-6

Thus, the variable-setting component consists of n diamond-shaped gadgets that are strung together. The construction ensures that for each i, the choice of whether to follow the path through x_i or $\bar{x}_i$ is made by player 0 if x_i is a universally quantified variable, and by player 1 if x_i is an existentially quantified variable. Informally, this choice determines a truth assignment to the variable x_i.

Proof of Theorem 7-7

We next describe the clause-testing component. The node set is:

$$V_2 = \{\, y_k, y'_k, y''_k \mid 1 \le k \le m \,\} \cup \{\, y_{kj} \mid 1 \le k \le m, 1 \le j \le n^2 - 3 \,\}$$

The edge set is

$$\begin{array}{rcl} E_2 & = & \{\, (u_{n+1}, y_k) \mid 1 \le k \le m \,\} \\ & \cup & \{\, (y_k, y'_k), (y_k, y''_k), (y'_k, y_{k1}) \mid 1 \le k \le m \,\} \\ & \cup & \{\, (y_{kj}, y_{kj+1}) \mid 1 \le k \le m, 1 \le j \le n^2 - 4 \,\} \\ & \cup & \{\, (y_{kn^2-3}, u_{n+1}) \mid 1 \le k \le m \,\} \end{array}$$

Proof of Theorem 7-8

Note that, because we are assuming that the last quantifier is ∀, player 1 chooses, from u_{n+1}, an edge to some y_k. Informally, node y_k corresponds to a clause C_k, which player 1 claims is true. At that point, player 0 can either move to y''_k, in which case player 0 is challenging player 1 that clause C_k is false, or y'_k, in which case player 0 is not challenging. If player 0 does not challenge, a path of length n^2 is followed, back to u_{n+1}, where this is repeated.

Proof of Theorem 7-9

Other interconnections between the clause-testing and variable-setting components are as follows:

$$\begin{array}{rcl} E_{12} & = & \{\, (y''_k, x_i) \mid x_i \text{ occurs unnegated in clause } k \,\} \\ & \cup & \{\, (y''_k, \bar{x}_i) \mid x_i \text{ occurs negated in clause } k \,\} \end{array}$$

Proof of Theorem 7-10

Thus, if player 0 challenges clause C_k, player 1 chooses a literal of the clause. If the literal is false, player 0 can follow an edge of the diamond and force player 1 to lose in one more move.

Proof of Theorem 7-11 We finally describe the line. The node set is $V_3 = \{\, l_i \mid 1 \leq i \leq n^4 \,\}$. The edge set is $E_3 = \{\, (l_j, l_{j+1}) \mid 1 \leq j \leq n^4 - 1 \,\}$. Thus, this is simply a line with $n^4 - 1$ edges.

Proof of Theorem 7-12 The following edges connect each literal node of the variable-testing component to the line:

$$E_{13} = \{\, (x_i, l_1), (\bar{x}_i, l_1) \mid 1 \leq i \leq n \,\}$$

The purpose of this line is to ensure that player 0 never challenges player 1 on a clause that is actually true. Otherwise, player 1 can force the game to end up on the line and thus take n^4 steps, and also player 0 loses.

Proof of Theorem 7-13 Thus, the whole reduction gives $G = (V, E)$ where $V = V_1 \cup V_2 \cup V_3$ and $E = E_1 \cup E_2 \cup E_{12} \cup E_3 \cup E_{13}$.

Proof of Theorem 7-14 This completes the reduction from MAX-Q3SAT and implies that there is a constant ϵ such that approximating MAX-GGEOG within ratio ϵ is *PSPACE*-hard. By reducing from $(\log n)$-MAX-Q-FORMULA instead of MAX-Q3SAT, we improve the result to ratio n^ϵ. In this new reduction, y''_k is connected to a subgraph that simulates the kth formula ϕ_k in the following way. There is a node for each of the operators of ϕ_k and possibly some auxiliary nodes. If $\phi_k = \phi'_k \vee \phi''_k$, we make sure (possibly by adding an auxiliary node) that player 1 makes the move from the node corresponding to the $\vee$ operator. Thus, player 1 chooses the subformula ϕ'''_k such that $\phi'''_k = 1$. If $\phi_k = \phi'_k \wedge \phi''_k$ we make sure (possibly by adding an auxiliary node) that player 0 makes the move from the node corresponding to the $\wedge$ operator. If ϕ_k is a literal, we connect it to the diamonds, as before. The paths from the nodes y'_k to u_{n+1} are longer than before and depend on ϵ.

Proof of Theorem 7 □

5 Subsequent Related Work

5-1 This section contains a brief discussion of related work that has been done since our results first appeared [CFLS93a, CFLS93b].

5-2 A polynomial-round Arthur-Merlin game with a polynomial-time verifier [BM88] can be thought of as a PCDS in which $r(n)$ and $q(n)$ are both arbitrary polynomials and one of the debaters simply makes random moves. Let $AM(\text{poly}(n))$ denote the class of languages accepted by such Arthur-Merlin games. In this context, the fact that $AM(\text{poly}(n)) = PSPACE$ (cf. [LFKN92] and [Sha92]) means that, if $r(n)$ and $q(n)$ are both arbitrary polynomials, then the universal debater in a PCDS can be replaced by a random debater without loss of generality. In [CFLS94] we show that, even if $r(n) = \log n$ and $q(n) = 1$, one can replace the universal debater by a random debater and still retain the power to recognize every language in *PSPACE*. This fact has implications for the hardness of approximating *stochastic PSPACE*-hard functions, of the type studied by Papadimitriou [Pap85].

5-3 We have described *PSPACE*-hard functions that do not have PTASs, unless some unexpected collapse occurs. It is not hard to define a *PSPACE*-hard function that *does* have a PTAS, but the straightforward examples are artificial. We were thus led to ask in [CFLS93a] and [CFLS93b] whether there is a natural *PSPACE*-hard function that has a PTAS. A positive answer to this question is provided in [MHSR94].

5-4 Bodlaender (private communication) has extended our results by showing that MAX-Q3SAT can be approximated within some $\epsilon > 0$, and by providing a simpler proof that MAX-GGEOG is *PSPACE*-hard to approximate; his proof that approximating MAX-GGEOG is hard does not involve PCDSs. Hunt et al. [HMS94] showed, also using direct-reduction arguments, that it is *PSPACE*-hard to approximate several other constrained optimization problems within certain factors. These problems include MAX-Q-FORMULA, a generalization of ($\log n$)-MAX-Q-FORMULA, where the "clauses" are general formulas (with no restrictions on the number of variables per "clause").

5-5 It is not known whether characterizations of *EXP* and *NEXP* can be found that are similar to the *PCP* and *PCD* characterizations of *NP* and *PSPACE*, respectively, and that lead to interesting nonapproximability results for problems that are complete for *EXP* or *NEXP*.

6 Acknowledgments

6-1 We thank Lance Fortnow and Mario Szegedy for helpful discussions in the formative stages of this work. We thank Jin-yi Cai, Lenore Cowen, Uriel Feige, David Johnson, Madhu Sudan, and Mihalis Yannakakis for their comments on earlier versions. Finally, we thank Nick Reingold for helping us typeset Section 3.

Acknowledgement of support: Anne Condon's work was supported in part by NSF grants CCR-9100886 and CCR-9257241.

References

[AH92] R. Aumann and S. Hart, editors. *Handbook of Game Theory*, volume 1. North Holland, Amsterdam, 1992.

[ALM$^+$92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Symposium on Foundations of Computer Science*, pages 14–23, Los Alamitos, CA, October 1992. IEEE Computer Society Press.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs. In *Proceedings of the 33rd Symposium on Foundations of Computer Science*, pages 2–13, Los Alamitos, CA, October 1992. IEEE Computer Society Press.

[BM88] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, April 1988.

[CFLS93a] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions. Technical Report 93-10, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) , Berkeley, CA, March 1993.

[CFLS93b] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions (extended abstract). In *Proceedings of the 25th ACM Symposium on the Theory of Computing*, pages 305–314, New York, May 1993. Association for Computing Machinery.

[CFLS94] A. Condon, J. Feigenbaum, C. Lund, and P. Shor. Random debaters and the hardness of approximating stochastic functions. In *Proceedings of the 9th Conference on Structure in Complexity Theory*, pages 280–293, Los Alamitos, CA, June 1994. IEEE Computer Society Press. Final version to appear in *SIAM Journal on Computing.*

[Con93] A. Condon. The complexity of the max word problem and the power of one-way interactive proof systems. *Computational Complexity*, 3(3):292–305, 1993.

[CW89] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th Symposium on Foundations of Computer Science*, pages 14–19, Los Alamitos, CA, October 1989. IEEE Computer Society Press.

[FGL$^+$91] U. Feige, S. Goldwasser, L. Lovász, M. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Symposium on Foundations of Computer Science*, pages 2–12, Los Alamitos, CA, October 1991. IEEE Computer Society Press.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Fransisco, 1979.

[GS92] P. Gemmell and M. Sudan. Highly resilient correctors for polynomials. *Information Processing Letters*, 43(4):169–174, June 1992.

[HMS94] H. Hunt III, M. Marathe, and R. Stearns. Generalized CNF satisfiability problems and non-efficient approximability. In *Proceedings of the 9th Conference on Structure in Complexity Theory*, pages 356–366, Los Alamitos, CA, June 1994. IEEE Computer Society Press.

[IZ89] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *Proceedings of the 30th Symposium on Foundations of Computer Science*, pages 248–253, Los Alamitos, CA, October 1989. IEEE Computer Society Press.

[Joh74] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, December 1974.

[Koz77] D. Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Symposium on Foundations of Computer Science*, pages 254–266, Los Alamitos, CA, October 1977. IEEE Computer Society Press.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the Association for Computing Machinery*, 39(4):859–868, October 1992.

[MHSR94] M. Marathe, H. Hunt III, R. Stearns, and V. Radhakrishnan. Hierarchical specifications and polynomial-time approximation schemes for PSPACE-complete problems. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pages 468–477, New York, May 1994. Association for Computing Machinery.

[Pap85] C. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31(2):288–301, October 1985.

[PY91] C. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, December 1991.

[Sch78] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, April 1978.

[Sha92] A. Shamir. IP=PSPACE. *Journal of the Association for Computing Machinery*, 39(4):869–877, October 1992.