Making Games Short

(Extended abstract)

Uriel Feige[*] Joe Kilian[†]

Abstract

We study the complexity of refereed games, in which two computationally unlimited players play against each other, and a polynomial time referee monitors the game and announces the winner. The players may exchange messages with the referee in private, resulting in a game of perfect recall but incomplete information. We show that any EXPTIME statement can be efficiently transformed into a refereed game in which if the statement is true, the first player wins with overwhelming probability, and if the statement is false, the second player wins with overwhelming probability. We also prove matching PSPACE upper and lower bounds on the complexity of statements that have refereed games that take one round of communication.

1 Introduction

In a *refereed game* there are two computationally unlimited players, and one computationally limited trusted referee. The game proceeds by exchanging messages between each player and the referee. Messages are private – a player cannot see the messages exchanged between the other player and the referee. The referee follows a prespecified algorithm known to both players. This algorithm may be randomized, in which case we assume that the coin tosses of the algorithm are private, i.e., hidden from the players. The game ends when the referee declares which of the players has won the game. The purpose of each player is to be declared as the winner.

Viewed as a game between two players, the role of the referee is to ensure that the rules of the game are kept. The need for private coin tosses and private messages may arise in games like Poker, where the trusted referee shuffles the cards and deals to each player his cards in private. To make a move of a player public (as is necessary in games like Chess), the trusted referee can tell each player which messages were received from the other player.

We would like to use refereed games as a decision procedure in scenarios in which two experts differ in opinion regarding the truthfulness of a mathematical statement. The mathematical statement is transformed into a refereed game, with the property that if the statement is true, the first player has a strategy that almost surely wins, whereas if the statement is false, the second player has a strategy that almost surely wins. By letting the two experts play this game against each other, the referee can observe who wins, and accept his opinion on the matter. The underlying rational is that the losing player cannot be called an expert, as either he made a wrong claim about the original mathematical statement, or he cannot play well (and experts are supposed to play well).

We ask which games can be refereed efficiently. The efforts of the referee are measured in terms of the number of messages exchanged with each player, the length of each message, and the computational effort involved in the referee computing the next message to send and in deciding who wins. We want all these to be bounded by a polynomial in the size of the game. To make sense of the word "polynomial", we will consider infinite families of games (e.g., like chess on boards of size n by n).

There is another consideration that is central to efficient refereeing. As we are dealing with games of imperfect information (a player does not see the messages exchanged between the other player and the referee), game theory tells us that the optimal strategy (in the minimax sense) for each player is randomized. We do not expect a player to have a strategy that wins regardless of the strategy used by the other player (such a strategy would be deterministic). Hence a player starting in an infe-

[*]Dept. of Applied Math and Computer Science, The Weizmann Institute. Rehovot 76100, Israel. `feige@wisdom.weizmann.ac.il`. Incumbent of the Joseph and Celia Reskin Career Development Chair. Yigal Alon Fellow.

[†]NEC research Institute, 4 Independence Way, Princeton, New Jersey. `joe@research.nj.nec.com`.

rior position may still win due to lucky coin tosses. To partially remedy this situation, we will be interested in game positions whose minimax value is bounded away from half.

To deal with the complexity of statements that can be decided by refereed games, we consider games that recognize languages.

Definition 1 *A* family of refereed games *is described as follows. For any binary string x there is a game G_x. There is one refereeing algorithm R used for all games in the family. The refereeing algorithm receives x as input, may flip private coins and runs in time bounded by a polynomial in $|x|$.*

Definition 2 *Language L is* refereeable *if there is a family of refereed games such that for any x, if $x \in L$ then Player 1 has a strategy that wins G_x with probability at least 2/3, and if $x \notin L$ then Player 2 has a strategy that wins G_x with probability at least 2/3.*

Definition 2 allows for a certain probability of error in the referee's decision (at most 1/3) even if the truthful player plays optimally. To make this probability arbitrarily small, simply repeat the game several times and take a majority vote. In all of our results, "at least 2/3" can be replaced by "exponentially close to 1."

Hence if a language L is refereeable, then disputes between two players regarding whether or not a certain input x is in the language can be resolved by a referee. The referee can let the players play the game G_x and simply observe who wins. If all communications are indeed private, and if the player claiming the correct status of x with respect to L uses the optimal strategy, then the referee will with high probability correctly decide the status of x, regardless of the strategy employed by the other player. The total effort of the referee (computation and communication) is polynomial in the input length, but the computational effort of the players may be larger, depending on the complexity of L.

Clearly, any language in BPP is refereeable – the random polynomial time referee can announce the status of x with respect to L without having the players play at all. By the well known results on *alternation* [4], any language in PSPACE is also refereeable, by a game of perfect information. Previous to our work, it was unknown whether having private communications (imperfect information) extends the class of refereeable games beyond PSPACE. This question was asked in [9, 10]. We show:

Theorem 1 *Any language in EXPTIME is refereeable.*

This result is best possible, as the work of Koller and Megiddo [13] (and see also [14, 10]) implies that computing the minimax value of a game G_x (that is, the probability that R declares that Player 1 wins assuming that both players use optimal strategies) can be done in time polynomial in size of the game tree. By our requirement that R is polynomial, the game tree associated with G_x is at most exponential in $|x|$.

We also study the round complexity of refereeable games.

Definition 3 *A* round *of a refereeable game is composed of four messages in the following order: a message from the referee to Player 1, a message from the referee to Player 2, a message from Player 1 to the referee, a message from Player 2 to the referee.*

We introduce the following notation. RG denotes a refereeable game. It has two parameters. The first has one of two values: *public*, specifying that all communication is public (each player sees the messages exchanged between the referee and the other player), or *private* (all communication is private, and a player knows of contents of communications of the referee with the other player only if the referee tells the player about them). The other parameter specifies the number of rounds in the refereeable game, where the term *poly* specifies that this number may be polynomial.

Using this notation, the results of [4] imply that RG(public,poly) = PSPACE, and our results imply that RG(private,poly) = EXPTIME. We also study the class RG(private,1). It is easily seen that Σ_2^p is contained in RG(public,1), and that RG(public,k) is contained in the polynomial hierarchy for any fixed k. (This is true even if the referee uses private coin tosses, because approximate counting can be done within the polynomial hierarchy [26].) We show:

Theorem 2 *RG(private,1) = PSPACE.*

That is, any language in PSPACE is refereeable in one round and every language refereeable in one round is in PSPACE.

Our proofs of the lower bounds (i.e., PSPACE $\subseteq$ RG(private,1) and EXPTIME $\subseteq$ RG(private,poly)) are based on algebraic techniques as developed by [18, 25] in the context of interactive proofs. Adapting these algebraic techniques to the design of refereeable games makes use of some extra tricks that have no counterpart in the interactive proofs scenario. The proof of the upper bound (RG(private,1) $\subseteq$ PSPACE) was inspired by the NC algorithm for positive linear programming [17] (though our PSPACE algorithm and its analysis are quite different from the algorithm of Luby and Nisan).

Our results may be viewed as showing that private communication can reduce the round complexity of the refereeing process. With some abuse of notation, we may denote by RG(public, exp) the class of alternating PSPACE, in which there are an exponential number of communication rounds, but the refereeing effort for each round is polynomial. Combining our results with those of [4] on alternation we see that private communication makes games shorter.

Corollary 3 • *RG(private, poly) = RG(public, exp) (= EXPTIME)*

• *RG(private, 1) = RG(public, poly) (= PSPACE)*

An interesting question that remains open is whether RG(private,k) = EXPTIME for some fixed k. Other open questions are related to approximation of P-complete problems. Theorem 1 shows that the value of a refereed game can be decided exactly in EXPTIME, and is EXPTIME-hard to approximate. Scaling everything down by an exponent, we obtain a problem that can be solved exactly in quasipolynomial time, but is "quasipolynomial time"-hard to approximate. By a reduction to linear programming (as that in [14]), one obtains that linear programming cannot be approximated in polylogarithmic space unless P is contained in polylogspace. Similar results were proven earlier using techniques which are more elementary [24, 19]. It remains to be seen whether the "deeper" (i.e., more complicated) nature of our proof can lead to new insights regarding the hardness of approximating P-complete problems in NC. Trying to reverse the connection is also challenging: can results on the P-completeness of approximating linear programming (or some other problem) be used in order to design a refereed game that would prove Theorem 1? Observe that in the related context of PCPs versus approximation, the relation between the inapproximability of max-3SAT and the PCP theorem is bidirectional [1].

2 Related work

The model of refereed games was inspired by models of interactive proofs [11, 2, 3]. It was first explicitly introduced by Feige, Shamir and Tennenholtz [9] (using different terminology), where an analogy was given between refereed games and a court of law. In this analogy, the referee is the judge, and the players are two disputing lawyers, each trying to convince the judge that his client is right. The judge is computationally limited (meaning that he does not have much time to spend on the case), whereas the lawyers are computationally unlimited (meaning that they had a long time to prepare the case). Private communication can be modeled by the judge inviting one of the sides to his office for private questioning. (This may not be customary practice, but our results on round reduction through private communication indicate that this may allow the judge to uncover the truth more quickly). In [9] it was shown that certain weak variants of the model are as strong as PSPACE, whereas variants of the model in which all communication is public but the coin tosses of the referee are private do not extend the power of the model beyond PSPACE. The power of RG(private, poly) was left as the main open problem.

Feige and Shamir [8] studied the version of the model in which the referee is space bounded rather than time bounded. They show that such refereed games can simulate an arbitrary Turing machine. This shows that any recursive language is refereeable when the referee is space bounded but not time bounded, indirectly answering an open question in a preliminary version of [21]. (This result also follows from [7].) They also consider the case of simultaneous time and space bounds on the referee, and show that up to polynomial factors in the time bounds and number of rounds, space bounds on the referee do not further restrict the set of refereeable games. Hence our result that RG(private,poly) = EXPTIME holds even when the referee is limited to constant space.

From the point of view of the players, a refereed game is a game of incomplete information (not seeing the moves of the other player), and of perfect recall (a player remembers all his local past). In this context, we survey some standard results from game theory that are relevant to this paper.

A *pure strategy* for a player is a function from the local view of the player to its next move. A *mixed strategy* is a probability distribution over pure strategies. A game is *zero sum* if the sum of payoffs of both players is always zero. That is, they are playing against each other with no incentive to cooperate. For zero sum games, we shall assume that the first player is trying to maximize the payoff, whereas the second player is trying to minimize it. The rest of the discussion is specialized to zero sum games. The celebrated minimax theorem of game theory [20] says that there is a certain value v such that Player 1 has an optimal mixed strategy that insures an expected payoff of at least v regardless of the strategy played by Player 2, and that Player 2 has an optimal mixed strategy that insures a expected payoff of at most v regardless of the strategy played by Player 1. A game may be represented in *normal form*, as a matrix, where rows of the matrix are pure strategies of one player, columns are pure strategies of the other player, and entries are the payoff of Player 1 in case the respective strategies are played. A game may also be represented in *extensive form*, as a game tree, where vertices are positions and edges are next move relations. In games of imperfect information, some vertices of the game tree may be clustered together as an *information set*, meaning that one of the players cannot distinguish between these vertices (positions) based on his local information.

A *behavior strategy* specifies for each information set in which the player is to move a probability distribution on the outgoing edges (specifying with what probability each possible next move is chosen). Kuhn [15] has shown that for games of perfect recall, any mixed strategy can be transformed into a behavior strategy that achieves the same payoff against any other strategy.

Returning to refereeable games, observe that by computing the minimax value of the game of perfect recall G_x, we could tell whether $x \in L$. Hence the complexity of computing the value of a game of perfect recall is an upper bound on the complexity of languages that are refereeable.

A standard game theory procedure for computing the value of a game is by using linear programming, when the game is represented in normal form. Despite the fact that linear programming has polynomial time algorithms, this does not give an efficient algorithm in our case, because the normal form representation of G_x may be doubly exponential in $|x|$ (even for one round games). Hence we seek an algorithm that works directly on the game tree (whose size is only exponential in $|x|$). Perhaps the most well known algorithm for computing the value of a game tree is the min-max algorithm of Zermelo [27], based on backward induction. For games of perfect information, this gives a PSPACE algorithm. However, this algorithm does not apply to games of perfect recall (where in particular, the optimal strategy is not deterministic).

Koller and Megiddo [13] show that the value of games of perfect recall can be computed in time that is polynomial in the size of the game tree. Their algorithm is based on the Ellipsoid algorithm for linear programming with a separation oracle. Hence RG(private, poly) $\subseteq$ EXPTIME. Koller, Megiddo and von Stengel [14] simplify the linear programming based approach for deciding the value of a game of perfect recall, and show that all constraints can be written explicitly. The transformation from game tree to LP can be done in space proportional to the product of the depth and the log of the branching factor of the tree.

Feigenbaum, Koller and Shor [10] systematically study the relation between types of two person games (perfect information, perfect recall, and imperfect information), and complexity classes (often having definitions in terms of various kinds of interactive proofs). They distinguish between what they call the game theoretic model and the accept reject model. Loosely speaking, the difference between the two models is whether one cares about the exact probability with which player 1 wins, or about the approximate probability. In particular, [10] show that it is EXPTIME-hard to determine the (exact) value of a game of perfect recall in the game theoretic model (this is somewhat analogous to linear programming being P-complete). They leave open the question of whether any language not known to be in PSPACE has a game of perfect recall in the accept-reject model. This last open question is equivalent to the question that we answer regarding the complexity of RG(private, poly).

The EXPTIME-hardness result of [10] in the game theoretic model applies already in the scenario in which all communication is one-way: the referee never sends any message. This EXPTIME hardness result for one-way communication cannot be extended to the accept-reject model (the one we study), unless EXPTIME=PSPACE. In fact, Lipton and Young [16] show that such games have near optimal strategies of small support, hence approximating them can be done within the polynomial hierarchy. An alternative proof that this class of games is limited to PSPACE follows from the fact that they can be represented as exponential size positive linear programs, combined with the NC approximation algorithm of Luby and Nisan [17] for positive linear programs.

There has been a lot of work on models related to refereed games. We already mentioned alternation [4], that characterizes the complexity of games of complete information. Reif [21] deals with the complexity of two player games of incomplete information. The question that he asks is what is the complexity of deciding whether Player 1 has a strategy that wins regardless of the strategy of Player 2. Note that in this case the strategy for Player 1 must be deterministic. For the case of games with polynomially many alternations, the complexity remains PSPACE. But if there is no bound on the number of alternations, then the complexity increases from EXPTIME (which is what happens in the public case), to double exponential time (in the private case). In a preliminary version of his paper, Reif poses the open question of the computational complexity of the payoff of optimal probabilistic strategies for reasonable games. The version of this question in which the number of alternation is not restricted was shown to be undecidable in [8]. The version in which the number of alternations is polynomial is EXPTIME-complete (upper bound follows from [13], and lower bound follows from [10] for exact computation of the payoff, and Theorem 1 for approximating the payoff).

Other work on related models includes [5, 6, 12, 22], and will not be surveyed here because of space limitations.

3 Designing complex games

In the refereed games that we design, the referee is randomized and the player that is truthful employs a deterministic strategy. This situation can be reversed at the cost of an extra round, if each player initially sends the referee a random string in private, and then the referee XORs these strings to get his own private random bits.

We sketch the proof of Theorem 1, that EXPTIME $\subseteq$ RG(private,poly).

Proof:(Sketch) Let L be an EXPTIME language and let M be a one tape Turing machine that accepts L in exponential time. On common input x, Player 1 says $x \in L$ and Player 2 says $x \notin L$. One player always tells the truth (in our protocol, honesty is the best policy), and the other player sometimes cheats.

Consider the tableau of the computation of M on x. W.l.o.g., we assume that it has 2^n rows, 2^n columns, and the first row has polynomially many nonblank characters known to both players which encode the input and initial configuration of M. Each row has an *active region* around where the read/write head of M is located (the tape itself is assumed to also encode at that location the state of the finite control of M). Finally, at the end of its computation, M writes a 1 in the first square if $x \in L$ and a 0 otherwise. Thus, Player 1 claims that the first character of the last row is 1 and Player 2 claims that it is 0. We need to check who is correct.

View each row i of the tableau as a function $f_i : \{0,1\}^n \rightarrow \{0,\ldots,q)$ where q is a large enough to encode the state of M and the characters on the tape, including blank characters, which we assume are encoded by a 0. Let $p > n^2$ be some large prime. All computations are done modulo p. Let $\hat{f}_i : [p]^n \rightarrow [p]$ be the multilinear extension of f_i. That is, $\hat{f}_i(x_1,\ldots,x_n) = \sum_{b_1,\ldots,b_n \in \{0,1\}^n} f(b_1,\ldots,b_n) \prod_{i=1}^n s_i$, where s_i is shorthand notation for x_i when $b_i = 1$, and for $1 - x_i$ when $b_i = 0$.

Observe that $\hat{f}_1$ (corresponding to the first configuration) can be computed in polynomial time by the referee, because on $\{0,1\}^n$ it has only polynomially many nonzero values (by the convention that blanks are encoded as 0). Similarly, if the active region of row i of the tableau is known then $\hat{f}_{i+1} - \hat{f}_i$ can be computed in polynomial time, since this difference only depends on the active region of row i.

The protocol is a binary search on the rows. Clearly, the two players agree on the contents of first row, and disagree on the last. Using (a clever) binary search, the referee finds two rows i and $i+1$ such that the players agree (or at least appear to agree) on row i (a *good* row) and disagree on row $i+1$ (a *bad* row). Then, using the information available on row i and additional questions, the referee catches the cheater with high probability.

To implement the binary search, the referee selects in private two random values $a, b \in [p]^n$ and obtains a parametric line $L_1 = at + b$ in $[p]^n$. The referee sends in private a and b (and hence L_1) to Player 2. The referee selects in private a random value t_r for the parameter t and obtains a random point $p_1 = at_r + b$ on line L_1. The referee sends p_1 to Player 1 in private.

Now for any row j selected by the binary search (the name of the row can be announced in public), Player 1 sends in private a value y_j that is claimed to be $\hat{f}_j(p_1)$, and Player 2 sends in private a polynomial P_j of degree n that is claimed to represent $\hat{f}_j$ on L_1. That is, for every $t \in [p]$, it is intended that $P_j(t) = \hat{f}_j(L_1(t))$. The referee checks whether $P_j(t_r) = y_j$ (implying that the two players agree on $\hat{f}_j(p_1)$). If they disagree, row i is declared *bad*. If they agree, row i is declared *good*.

Consider now a good row j. The referee knows that $\hat{f}_j(p_1) = y = P_j(t_r)$, because at least one player is truthful. Furthermore, the referee may assume that $P_j(t) = \hat{f}_j(L_1(t))$ for all t. This assumption is justified, as otherwise $P_j(t)$ and $\hat{f}_j(L_1(t))$ agree on at most n of the p different values of the parameter t, and as t_r (and p_1) is random and unknown to Player 2, the probability of $\hat{f}_j(p_1) = P_j(t_r)$ holding is at most n/p (but note the "subtlety" at the end of the proof, which leads to a corrected probability of $n/(p - n^2)$), which is negligible for sufficiently large p. We shall use the fact that the referee can now pick some random point $p_2 \neq p_1$ on the line L_1 and compute the value of $\hat{f}_j(p_2)$. We note that row 1 is forced to be good, because the referee can compute $\hat{f}_1$ by himself.

After doing a binary search for n steps, we are left with a good row i and a bad row $i+1$. (Remark: the case that row 2^n appears to be good, and we do not have a bad row, is handled as in case 1 below.) Now the referee asks both players for the location and contents of the active region of row i. There are two possibilities.
1) *Both players agree on the location and contents of the active region of row i.* In this case, the referee has sufficient information to compute by himself the function $\hat{f}_i - \hat{f}_{i+1}$ (this difference depends only on the active region of row i), and hence compute the difference $\hat{f}_i(p_1) - \hat{f}_{i+1}(p_1)$. Since the players agree on $\hat{f}_i(p_1)$ and disagree on $\hat{f}_{i+1}(p_1)$, one of them will be caught cheating.
2) *The players disagree on the location or contents of the active region of row i.* Based on this, the referee now holds a specific point (say p_3) in the good row on which the players disagree. Let L_2 be the line that passes through p_2 (recall that p_2 is on L_1) and p_3. The referee sends some canonical representation of L_2 (that does not disclose p_2) to Player 1. Player 1 must reply with a degree n polynomial that describes $\hat{f}_i$ on this line. If the polynomial does not agree with the value claimed by Player 1 on p_3, then Player 1 is exposed as a cheater and Player 2 wins. Likewise, if the polynomial does not agree with the value that the referee deduced from P_i for $\hat{f}_i(p_2)$, again Player 1 is exposed as a cheater and

Player 2 wins. But if the polynomial agrees with the values on both p_2 and p_3, Player 1 wins. The reason is that this polynomial may be assumed to correctly represent $\hat{f}_i$ on the line L_1 because it gave the correct value on the random point p_2 unknown to Player 1. Hence Player 2 was cheating regarding the value of p_3.

The protocol above requires sending roughly n polynomials. Its underlying assumption is that an incorrect low degree polynomial is incorrect on a random point. A slight subtlety is that a cheating Player 2 may send a polynomial P_i that is correct only on n points, and deduce some information on the location of point p_1 by the fact that a row is declared bad. But this information is limited when the prime p is sufficiently large. The error of the protocol is $O(n^2/p)$. $\square$

We now prove one direction of Theorem 2.

Lemma 4 *PSPACE $\subseteq$ RG(private,1)*

Proof: **(Sketch)** Our proof modifies Shamir's proof [25] that PSPACE $\subseteq$ IP. To illustrate how this modification is done, we will show how a standard protocol for $\#P \subseteq$ IP (as in [18]) can be modified to show $\#P \subseteq$ RG(private,1). The principles for proving PSPACE $\subseteq$ RG(private,1) are similar, and the only added complications are those required for proving that PSPACE $\subseteq$ IP.

Consider an arbitrary 3CNF formula ϕ, and arithmetize it by changing logical ands to products, and a clause to a multilinear polynomial in its three variables, that has value 1 when the clause is satisfied and 0 otherwise. Hence ϕ is transformed into a polynomial $\hat{\phi}$ of degree at most n in each of its variables (assuming there were at most n clauses in ϕ), and under 0/1 assignments to the variables, $\hat{\phi}$ evaluates either to 0 or to 1, depending on whether the assignment satisfies ϕ. Now player P_1 claims that for some value N_0,

$$\sum_{x_1,\ldots,x_n \in \{0,1\}} \hat{\phi}(x_1,\ldots,x_n) = N_0,$$

whereas player P_2 claims otherwise. The referee needs to decide who is correct.

Preliminaries

R chooses at random a sufficiently large prime p that he will send to P_1 and P_2; all computations in the following argument will be performed over the field $F = GF(p)$. We use lowercase letters to denote components of vectors denoted by uppercase letters: if $X_i \in F^j$ then $X_i = (x_1^i,\ldots,x_j^i)$ (vector subscripts turn into component superscripts).

If $Z \in F^i$ and $a \in F$ then $Z|a$ is shorthand for $(z_1,\ldots,z_i,a)$. We write $X \leftarrow S$ to denote the act of choosing X independently and uniformly from S.

Given distinct $t_0,\ldots,t_d \in F$ and $Y_0,\ldots,Y_d \in F^i$, we consider the degree-d parametric curve $C(t) \in F[t]^i$ passing through $(t_0,Y_0),\ldots,(t_d,Y_d)$ as the i unique degree-d polynomials $c_1(t),\ldots,c_i(t)$ such that $c_j(t_i) = y_j^i$. We assume that $C(t)$ is represented in some compact canonical form, independent of how it is derived.

We define $\Phi_0,\ldots,\Phi_n$ by

$$\Phi_i(x_1,\ldots,x_i) = \sum_{x_{i+1},\ldots,x_n \in \{0,1\}} \hat{\phi}(x_1,\ldots,x_n).$$

Thus, P_1 claims that $\Phi_0 = N_0$ (viewing N_0 as an element of $GF(p)$; p exceeds the largest possible value for N_0). Note also that Φ_n is easy to compute and that

$$\Phi_i(Z) = \Phi_{i+1}(Z|0) + \Phi_{i+1}(Z|1).$$

The protocol

The protocol consists of R's private computations and random choices, R's messages to P_1 and P_2, P_1 and P_2's messages to R, and R's decision procedure.

***R*'s computations:** For $0 \leq i \leq n$, R chooses $A_i, B_i \leftarrow F^i$ and $w_i, y_i \leftarrow F$. For $1 \leq i \leq n$, let $C_i(t) \in F[t]^i$ be the unique degree-2 parametric curve going through $(0, A_{i-1}|0)$, $(1, A_{i-1}|1)$ and $(2, B_i)$. Let $D_i(t) \in F[t]^i$ be the unique degree-1 (linear) parametric curve passing through $(w_i, C_i(w_i))$ and (y_i, A_i). We assume that $w_i \neq y_i$) and have R (arbitrarily) declare P_1 the winner if this is not the case.

$\boldsymbol{R} \rightarrow \boldsymbol{P_1}$: $p, A_1,\ldots,A_n, B_1,\ldots,B_n$.

$\boldsymbol{R} \rightarrow \boldsymbol{P_2}$: $p, D_1(t),\ldots,D_n(t)$ (in canonical form).

$\boldsymbol{P_1} \rightarrow \boldsymbol{R}$: For $1 \leq i \leq n$, define $N_i \in F$ and $Q_i(t) \in F[t]$ by $N_i = \Phi_i(A_i)$ and $Q_i(t) = \Phi_i(C_i(t))$. Note that $Q_i(t)$ has degree at most $2i \cdot n$. Note also that P_1 has enough information to compute C_i. P_1 sends $N_1,\ldots,N_n, Q_1(t),\ldots,Q_n(t)$.

$\boldsymbol{P_2} \rightarrow \boldsymbol{R}$: For $1 \leq i \leq n$, define $T_i(t) \in F[t]$ by $T_i(t) = \Phi_i(D_i(t))$. Note that $T_i(t)$ has degree at most $i \cdot n$. P_2 sends $T_1(t),\ldots,T_n(t)$.

Deciding who wins: R chooses $r \leftarrow F$. First, P_1 loses immediately if

- For some i, $Q_i(t)$ has degree greater than $2i \cdot n$ or $Q_i(0) + Q_i(1) \neq N_{i-1}$,
- $N_n \neq \Phi_n(A_n)$, or
- $Q_n(r) \neq \Phi_n(C_n(r))$.

P_2 loses immediately if

- For some i, $T_i(t)$ has degree greater than $i \cdot n$, or
- $T_n(r) \neq \Phi_n(D_n(r))$.

Failing one of these tests reveals an "obvious lie". If no obvious lies are detected, then R finds the smallest i such that $T_i(y_i) = N_i$. If $T_i(w_i) = Q_i(w_i)$ then R declares P_1 the winner; otherwise, R declares P_2 the winner.

Why the protocol works

We sketch why the above protocol allows the correct party to win with high probability. First, by a straightforward application of the definitions it follows that if P_i is correct and plays according to the above rules, then P_i will never make an obvious lie. Second, there are three general principles R can use to determine who is telling the truth.

1. If both players ever explicitly or implicitly agree on the value of $\Phi_i(X)$ for any X then they are both correct.
2. If a player makes a statement about the values of Φ_i on a curve, and turns out to be correct on some random point on that curve, one that it could not predict, then it is safe to assume that the player is telling the truth about all the points on the curve.
3. If a player explicitly or implicitly implies some value for $\Phi_i(X)$ for some X that disagrees with a "safe assumption," then it is safe to assume he is lying.

The first rule follows because the correct player is assumed to tell the truth (this turns out to be a winning strategy), and one of the players is correct. The second rule follows because a player gives a low-degree (compared to p) polynomial to specify the value of Φ_i along a parametric curve: either the polynomial is correct or it agrees with the correct polynomial on a negligible fraction of the points on the curve. The third rule follows because the correct player should always tell the truth, so it never hurts to punish lies.

R checks the players' implicit assertions about Φ_n on various curves by checking random points on these curves (note that R can compute Φ_n by himself). Hence, both are obliged to be correct concerning Φ_n, or they will be caught in an obvious lie. Now if both players were telling the truth always, then it would always hold that $T_i(y_i) = N_i = \Phi_i(A_i)$, by simple algebra. However, they implicitly disagree on $\Phi_0(A_0)$, so (barring obvious lies) there must be a smallest i such that they implicitly agree on $\Phi_i(A_i)$ (i.e. $N_i = T_i(y_i)$) but implicitly disagree on $\Phi_{i-1}(A_{i-1})$. It suffices then to determine which player is giving the correct value for $\Phi_{i-1}(A_{i-1})$.

We first note that

$$\begin{aligned} \Phi_{i-1}(A_{i-1}) &= \Phi_i(A_{i-1}|0) + \Phi_i(A_{i-1}|1) \\ &= \Phi_i(C_i(0) + \Phi_i(C_i(t)) \end{aligned}$$

If $Q_i(t)$ were indeed equal to $\Phi_i(C_i(t))$ then

$$\Phi_{i-1}(A_{i-1}) = Q_i(0) + Q_i(1) = N_{i-1},$$

or P_1 would have been caught in an obvious lie. Hence, it remains to judge whether $Q_i(t)$ is correctly constructed. We next note that by a straightforward probability argument it follows that,

- P_2 has no information concerning the value of y_i, and
- P_1 has no information concerning the value of w_i.

Then, since $N_i = T_i(y_i)$ (i.e. they agree) then R can assume that $T_i = \Phi_i(D_i(t))$ everywhere and in particular, $T_i(w_i) = \Phi_i(D_i(w_i))$. But since $D_i(w_i) = C_i(w_i)$, with high probability $T_i(w_i) = \Phi_i(C_i(w_i))$. If $T_i(w_i) = Q_i(w_i)$ then $Q_i(t)$ is (with high probability) correct at a random unknown point and hence is indeed correct, implying that P_1 is correct. Otherwise, P_1 may safely be assumed to have lied about the value of $Q_i(t)$, and therefore should lose.

To extend the above protocol from $\#P$ to PSPACE, perform it on Shamir's interactive proof for the value of an arithmetized simple QBF [25]. In this case we can take $N_0 = 0$, and let Player 1 be the one claiming that the original QBF is false. Details are omitted from this preliminary version. $\square$

4 Upper bounds on the complexity of refereed games

We prove here the other direction of Theorem 2.

Lemma 5 *RG(private, 1) $\subseteq$ PSPACE.*

The proof of Lemma 5 proceeds in two parts. First, we show a method for successively improving the players' strategies. We show that using polynomially many iterations of this method will result in one of the players having a near-optimal strategy. Next, we show how to implement this method in PSPACE.

4.1 A naive method for evolving good strategies.

Instead of efficiently constructing a strategy S_1 that behaves near optimally against all possible opposing strategies S_2, a much easier goal is to construct a strategy S_1 that works nearly optimally against the members of a small set $\{S_{2,1}, \ldots, S_{2,k}\}$ of opposing strategies. This suggests the following iterative procedure for evolving good strategies.

We denote by $S_{i,1}, S_{i,2}, \ldots$ the sequence of strategies generated for Player i. We define $\mathsf{win}(S_1, S_2)$ to be the probability that Player 1 wins when using strategy S_1 against Player 2 using strategy S_2, $\mathsf{win}(S_1, \cdot)$ to be $\min_{S_2} \mathsf{win}(S_1, S_2)$ and $\mathsf{win}(\cdot, \cdot)$ to be $\max_{S_1} \mathsf{win}(S_1, \cdot)$. We assume that either Player 1 or Player 2 has a strategy that causes it to win with probability at least .99 against any opposing strategy. That is, either $\mathsf{win}(\cdot, \cdot) \leq .01$ or $\mathsf{win}(\cdot, \cdot) \geq .99$. Our goal is to determine which case holds.

We present a naive iterative scheme that makes such a determination, but requires too many iterations to be useful, then modify it to reduce the number of iterations required. First choose strategy $S_{1,1}$ arbitrarily. Then generate a strategy $S_{2,1}$ such that $\mathsf{win}(S_{1,1}, S_{2,1}) < .1$, assuming such a strategy exists. For $i = 2, 3, \ldots$, generate strategy $S_{1,i}$ so that for all $j < i$, $\mathsf{win}(S_{1,i}, S_{2,j}) > .9$ and generate $S_{2,i}$ so that $\mathsf{win}(S_{1,i}, S_{2,i}) < .1$. We call strategies meeting these respective conditions *good*. If no good $S_{1,i}$ exists that meets the above requirements, then for all S_1, $\mathsf{win}(S_1, \cdot) < .9$, so $\mathsf{win}(\cdot,\cdot) < .9$ and therefore, by the assumption about the game, $\mathsf{win}(\cdot,\cdot) \leq .01$. Similarly, if no good $S_{2,i}$ exists, then $\mathsf{win}(\cdot,\cdot) \geq .99$.

The above approach has two main difficulties. First, the number of iterations required before one side does not have a good strategy can trivially be exponential, requiring us to keep track of exponentially many strategies. Second, it is not clear how to efficiently (in PSPACE) find a strategy that beats all the currently selected opposing strategies, or for that matter how to even represent a single such strategy with less than exponential space.

4.2 Reducing the number of iterations.

To reduce the number of iterations required by the naive procedure, we impose a further constraint on $S_{1,i}$. Let $Pr_{R,S}(q, a)$ denote the probability that R asks q to P_1 and that P_1, using strategy S, answers with a. We define $H(S)$ by

$$H(S) = \sum_{(q,a)} -Pr_{R,S}((q,a)) \lg Pr_{R,S}((q,a)).$$

That is, $H(S)$ is the entropy of the distribution on (q, a) induced by R and P_1 using strategy S.

We modify our naive iterative procedure by requiring that $S_{1,i}$ have essentially maximal entropy over all strategies such that $\mathsf{win}(S_{1,i}, S_{2,j}) \geq .9$ for $j < i$. That is, we require that for all strategies S' such that $\mathsf{win}(S', S_{2,j}) \geq .9$ for $j < i$, $H(S') - H(S_{1,i})$ is either negative or $o(1)$. Indeed, we will later show how to find $S_{1,i}$ such that $H(S') - H(S_{1,i})$ is negative or exponentially small. Informally, we choose the most random strategy over all the good strategies. For our analysis it doesn't seem to help us to place any similar constraint on $S_{2,i}$.

Clearly, $H(S_{1,i+1})$ cannot be significantly bigger than $H(S_{1,i})$, or $S_{1,i+1}$ would have been chosen instead of $S_{1,i}$ in stage i. In fact, $H(S_{1,i+1})$ must be significantly smaller than $H(S_{1,i})$ as implied by the following key lemma.

Lemma 6 *For any good $S_{1,i+1}$, $H(S_{1,i}) - H(S_{1,i+1}) > .01$.*

Proof: We give a high-level overview of the proof. The full proof is given in the appendix.

Let D_i denote the distribution on (q, a) induced by R and P_1 using strategy $S_{1,i}$. We want to show that $H(D_{i+1})$ is smaller than $H(D_i)$. We first note that $S_{1,i}$ performs poorly against $S_{2,i}$ but $S_{1,i+1}$ performs quite well against $S_{2,i}$. This implies that D_i is quite different from D_{i+1}. Now, consider the hybrid strategy S that answers according to $S_{1,i}$ with probability $\frac{1}{2}$ and according to $S_{1,i+1}$ otherwise. The distribution induced by S and R is just $D = (D_i + D_{i+1})/2$. Now, if D_{i+1} has nearly the same entropy as D_i, and they are fairly disjoint, then D will have significantly more entropy that D_i. But this yields a contradiction as follows. S meets the same condition as are met by $S_{1,i}$, and therefore $S_{1,i}$ would not have been chosen because it does not have near-optimal entropy. □

Lemma 6 implies that the modified process will terminate after polynomially many steps. Assuming some polynomial bound on the lengths of q and a there is a polynomial upper bound u on $H(S_{1,1})$. This bound and Lemma 6 implies that $H(S_{1,i}) \leq u - (i-1)/100$, and hence that the process must terminate before i reaches $1 + 100u$, since $H(S_{1,i}) \geq 0$.

4.3 Finding and representing good strategies.

It remains to show how to find and represent good $S_{1,i}$ and $S_{2,i}$ within PSPACE. Furthermore, $S_{1,i}$ must have near-maximal entropy. To do this, we guess a good strategy and test and represent it using a simple generalization of Savitch's theorem.

For strategy S, let $S(q, a)$ denote the probability that S outputs a on question q. Hence a strategy can be represented as a two dimensional table. We scan this table row by row to obtain a representation as an exponentially long sequence of numbers $x_1, \ldots, x_N$, one for each pair (q, a), giving the probability that the player should answer a on question q. We denote by $q(j)$ and $a(j)$ the questions and answers referred to by x_j. Unfortunately, we don't have enough space to store such a sequence. However, for our purposes it is sufficient to perform the following tasks.

1. Given oracles for $S_{2,1}, \ldots, S_{2,i-1}$, and a lower bound ℓ, determine in PSPACE if there *exists* a sequence $x_1, \ldots, x_N$ representing a good strategy $S_{1,i}$ such that $H(S_{1,i}) \geq \ell$. (And the analogous problem for $S_{2,i}$, without any entropy constraint).

2. If such a sequence exists, compute x_j for any j in PSPACE, such that the sequence so defined indeed has the above property.

Note that performing the above two tasks allows one to find an $S_{1,i}$ with near-maximal entropy by using binary search on ℓ. Furthermore, it suffices to compute and perform the first task for a sufficiently close approximation to $H(S_{1,i})$ (we use only polynomially many bits of precision in performing this computation).

4.3.1 Checking a witness in PSPACE.

Using the above ordering of $x_1, \ldots, x_N$, one can check all of the conditions this sequence must satisfy by "scanning" the sequence one element at a time. That is, one can read in x_1, perform some computations, then read in x_2 and perform more computations, until one finally reads in x_N and decides whether $x_1, \ldots, x_N$ satisfies the required conditions. Once read, an element may not be read again.

We describe in greater detail the conditions a purported strategy must meet, and how they can be computed. First, the probability assigned to (q, a) must be between 0 and 1 and for each q, the sum of the probabilities assigned to (q, a) (summing over a) must equal 1. As for each q, the variables x_j for which $q = q(j)$ are adjacent, these condition are simple to check in the scanning model.

Let $\rho_i(q)$ denote the probability that R gives question q to P_i and $\rho(q_1, q_2)$ denote the probability that R gives question q_1 to P_1 and q_2 to P_2. These function are trivial to compute in PSPACE. $H(S_{1,i})$ can be computed by

$$H(S_{1,i}) = \sum_j -(\rho_1(q(j))x_j)\lg(\rho_1(q(j))x_j).$$

More precisely, an exponentially close approximation to $H(S_{1,i})$ can so be computed, since we use only polynomially many bits of precision.

Let $\chi(q_1, a_1, q_2, a_2)$ be 1 if R would award the game to Player 1 on seeing (q_1, a_1, q_2, a_2) and 0 otherwise. Assume that an oracle for strategy $S_{2,j}$ is available. Then $\mathsf{win}(S_{1,i}, S_{2,j})$ is $\sum_{q_1,a_1,q_2,a_2} \rho(q_1, q_1)S_{1,i}(q_1, a_1)S_{2,j}(q_2, a_2)\chi(q_1, a_1, q_2, a_2)$ which can be computed using a single scan of $x_1, \ldots, x_N$ by the equivalent formula $\sum_k \sum_{q_2,a_2} \rho(q(k), q_2)x_k S_{2,j}(q_2, a_2)\chi(q(k), a(k), q_2, a_2)$.

Our intention is to perform computations with precision of polynomially many bits, as space is limited. Hence we shall only obtain approximations of $\mathsf{win}(S_{1,i}, S_{2,j})$ up to exponentially small errors, but this suffices for our goal of approximating the value of a $RG(private, 1)$ game. Note also that there is sufficient space to store the intermediate calculations for computing $\mathsf{win}(S_{1,i}, S_{2,j})$ for all j using a single scan of $x_1, \ldots, x_N$. Here we need the fact that i is polynomially bounded.

4.3.2 Using Savitch's Theorem.

Since we can check a witness $x_1, \ldots, x_N$ by making a single scan, as described above, we can view $x_1, \ldots, x_N$ as nondeterministic choices made by a nondeterministic polynomial-space bounded Turing Machine M. Here it is crucial that each x_i is scanned exactly once, since such choices can't be recalled. Thus, by Savitch's theorem, it follows that the existence of a good $x_1, \ldots, x_N$ can be decided in deterministic polynomial time. Furthermore, one can in deterministic PSPACE output the lexicographically first such witness. To do this, we run M up until the point where it makes the first choice (x_1). Let σ be M's configuration at this point. Again using Savitch's construction we can determine for any bound ℓ whether M, in configuration σ, can choose an x_1 such that $x_1 \geq \ell$ (using the lexicographic ordering) and then go on to accept. By binary search, the lexicographically first "safe" x_1 can be chosen. Then M can be run until it needs to choose x_2, etc. Note that since we are making M's choices for it, it always has a unique configuration σ representable in PSPACE. Finally, if we can enumerate the lexicographically first witness in deterministic PSPACE then we can compute x_i on input i just by going through the enumeration.

It is crucial that in generating a strategy we treat the opposing strategies as oracles. This is because in Savitch's construction the deterministic Turing machine requires the square of the space used by the nondeterministic Turing machine. If in generating the strategy $S_{1,i}$ we squared the space cost incurred in generating $S_{2,i-1}$, then we could only proceed for a constant number of iterations. However, by treating $S_{2,i-1}$ as an oracle, we have that the space required to compute $S_{1,i}$ is bounded by the space needed to compute $S_{2,i-1}$ *plus* some polynomial in i and the input size. Thus, as long as the number of iterations is polynomially bounded we can implement the iterative process using polynomial space.

Acknowledgements

We thank Joan Feigenbaum, Daphne Koller and Peter Shor for helpful discussions.

References

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

[2] L. Babai, S. Moran. "Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes." *J. Computer and Sys. Sci.* 36 (1988), 254–276.

[3] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. "Multi-prover interactive proofs:

How to remove intractability assumptions." In *Proc. of the 20th ACM Symp. on the Theory of Computing*, pages 113–131, 1988.

[4] A. Chandra, D. Kozen, L. Stockmeyer. "Alternation". *Journal of the ACM, 28:114–133, 1981.*

[5] A. Condon, J. Feigenbaum, C. Lund, P. Shor. "Probabilistically checkable debate systems and nonapproximability of PSPACE-hard functions". *Chicago Journal of Theoretical Computer Science, 19 October, 1995.*

[6] A. Condon, J. Feigenbaum, C. Lund, P. Shor. "Random debaters and the hardness of approximating stochastic functions". *Proc. of Ninth Annual Structure in Complexity Theory Conference, 280–293, 1994.*

[7] A. Condon and R. Lipton. "On the complexity of space bounded interactive proofs." In *Proc. 30th IEEE Symp. on Foundations of Computer Science*, pages 462–467, 1989.

[8] U. Feige, A. Shamir. "Multi-Oracle Interactive Protocols with Constant Space Verifiers". *JCSS, Vol. 44, pp. 259-271, April 1992.*

[9] U. Feige, A. Shamir, M. Tennenholtz. "The noisy oracle problem". *Advances in Cryptology – Crypto 88 Proceedings, Lecture Notes in Computer Science, 403, Springer-Verlag, Berlin, 1990, 284–296.*

[10] J. Feigenbaum, D. Koller, P. Shor. "A game-theoretic classification of interactive complexity classes". *Proc. of Tenth Annual Structure in Complexity Theory Conference, 227–237, 1995.*

[11] S. Goldwasser, S. Micali, C. Rackoff. "The knowledge complexity of interactive proof-systems." *SIAM J. Comp.* **18** (1989), 186–208.

[12] M. Kiwi, C. Lund, A. Russell, D. Spielman, R. Sundaram. "Alternation in Interaction". *Proc. of Ninth Annual Structure in Complexity Theory Conference, 294–303, 1994.*

[13] D. Koller, N. Megiddo. "The complexity of two-person zero-sum games in extensive form". *Games and Economic Behavior 4, 528–552, 1992.*

[14] D. Koller, N. Megiddo, B. von Stengel. "Fast algorithms for finding randomized strategies in game trees". In *Proc. 26th ACM Symposium on the Theory of Computing, 750–759, 1994.*

[15] H. Kuhn. "Extensive games and the problem of information". In *Contributions to the Theory of Games II* (H. Kuhn and A. Tucker, Eds.), 193–216, Princeton University Press, Princeton, NJ, 1953.

[16] R. Lipton, N. Young. "Simple strategies for large zero-sum games with applications to complexity theory". In *Proc. 26th ACM Symposium on the Theory of Computing, 734–740, 1994.*

[17] M. Luby, N. Nisan. "A parallel approximation algorithm for positive linear programming". In *Proc. 25th ACM Symposium on the Theory of Computing, 448–457, 1993.*

[18] C. Lund, L. Fortnow, H. Karloff, N. Nisan. "Algebraic Methods for Interactive Proof Systems." *J. ACM*, **39** (1992), 859–868.

[19] N. Megiddo. "A note on approximate linear programming". *Information Processing Letters 42 (1992) 53.*

[20] J. von Neumann, O. Morgenstern. "The theory of games and economic behavior". *Princeton University Press, Princeton, 1947.*

[21] J. Reif. "The complexity of two-player games of incomplete information". *Journal of Computer and System Sciences 29, 274–301, 1984.* Preliminary version, titled "Universal games of incomplete information", appeared in *Proc. of 11th ACM Symposium on the Theory of Computing, 288–307, 1979.*

[22] A. Russell, R. Sundaram. "Symmetric alternation captures BPP". To appear in *Computational Complexity.*

[23] W. Savitch. "Relationship between nondeterministic and deterministic tape classes". *JCSS 4, 177–192, 1970.*

[24] M. Serna. "Approximating linear programming is log-space complete for P". *Information Processing Letters 37 (1991) 233–236.*

[25] A. Shamir. "IP=PSPACE". *Journal of the ACM, 39:869–877, 1992.*

[26] L. Stockmeyer. "On approximation algorithms for #P". *SIAM J. Comput. 14 (1985), 849–861.*

[27] E. Zermelo. "Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels". In *E.W. Hobson and A.E.H. Love, editors, Proc. 5th International Congress of Mathematicians II, 501–504*, Cambridge University Press, Cambridge 1913.

A Proof of Lemma 6.

We treat our distributions D_i as vectors where each entry corresponds to the probability of some pair (q, a).

The L_1 *norm* of a vector is $|V| = \sum_i |v_i|$. $H(V) = \sum_i -v_i \lg v_i$.

The following proposition is a quantitative version of a well known inequality concerning entropies.

Proposition 7 *Let U and V be two arbitrary nonnegative n-vectors, each of norm 1. Let $\Delta = |U - V|/2$. Then:*

$$H(\frac{U+V}{2}) \geq \frac{H(U)+H(V)}{2} + f(\Delta)$$

where $f : [0,1] \longrightarrow [0,1]$ satisfies $f(0) = 0$, $f(1) = 1$, and $f(x) \geq x^2/12$.

Proof: If $\Delta = 0$ then $U = V = (U+V)/2$ and we get exact equality in the proposition. If $\Delta = 1$ then U and V are nonzero on a disjoint set of coordinates. Hence $H((U+V)/2) = H(U/2) + H(V/2)$. But $H(U/2) = -\sum(u_i/2)\log u_i/2 = -(1/2)(-H(U) - \sum u_i) = (H(U)+1)/2$, and the proposition follows with equality.

For general values of Δ we use the the following inequalities. For $x > 0$, $\ln(1+x) < x$, and $\ln(1-x) < -x - x^2/2$.

Consider an arbitrary coordinate i and let $u = u_i$ and $v = v_i$. Assume w.l.o.g. that $v \geq u$ and let $\delta = v - u \geq 0$. We bound $d = h((u+v)/2) - (h(u)+h(v))/2$. By simple manipulations, $d = (\log e)d'/2$, where $d' = -(u+v)\ln(u+v)/2 + u\ln u + v\ln v$. Further manipulations give $d' = -u(\ln(u+v)/2 - \ln u) - v(\ln(u+v)/2 - \ln v) = -u\ln(1+\delta/2u) - v\ln(1-\delta/2v)$. Using the bounds on $\ln(1+x)$ we obtain $d' \geq -u\delta/2u + v\delta/2v + v\delta^2/8v^2 = \delta^2/8v$. Hence $d \geq \delta^2 \log e/16v \geq \delta^2/12v$.

Call a coordinate i *good* if either $u_i/v_i < 1 - \Delta$, or $v_i/u_i < 1 - \Delta$. For a good coordinate $\delta/\max(v,u) \geq \Delta$, and $d \geq \delta\Delta/12$.

Consider now $D = H((U+V)/2) - (H(U/2) + H(V/2))/2$, and compute this sum coordinatewise. The contribution of each coordinate is nonnegative, whereas the contribution of good coordinate i is $\delta_i\Delta/12$. Hence $D \geq (\Delta/12)\sum_{i \text{ good}} \delta_i$, where the summation is taken over the good coordinates.

For a coordinate i, let $m_i = \min(u_i, v_i)$. Observe that $\sum m_i \leq 1 - \Delta$. It follows that summing over all coordinates that are not good, $\sum \delta_i \leq \Delta/(1-\Delta)\sum m_i \leq \Delta$. Since for all coordinates, $\sum \delta_i = 2\Delta$, it follows that for the good coordinates $\sum_{i \text{ good}} \delta_i \geq \Delta$. Hence $D \geq \Delta^2/12$. □

To apply Proposition 7 to relate the entropies of D_i, D_{i+1} and D, we first bound $\Delta = |(D_i - D_{i+1})/2|$. Let $D_j(q,a)$ denote the probability that D_j assigns to (q,a) and let $w(q,a)$ denote the probability, conditioned on R asking q of P_1, that P_1 will win if he answers a and P_2 plays according to $S_{2,i}$. Then

$$\begin{aligned} \mathsf{win}(S_{1,i}, S_{2,i}) &= \sum_{(q,a)} w(q,a) D_i(q,a) \text{ and} \\ \mathsf{win}(S_{1,i+1}, S_{2,i}) &= \sum_{(q,a)} w(q,a) D_{i+1}(q,a) \end{aligned}$$

Since $\mathsf{win}(S_{1,i}, S_{2,i}) \leq .1$ and $\mathsf{win}(S_{1,i+1}, S_{2,i}) \geq .9$, we have

$$|\mathsf{win}(S_{1,i}, S_{2,i}) - \mathsf{win}(S_{1,i+1}, S_{2,i})| \geq .8.$$

Thus,

$$|\sum_{(q,a)} w(q,a) D_i(q,a) - \sum_{(q,a)} w(q,a) D_{i+1}(q,a)| \geq .8.$$

which implies that

$$\sum_{(q,a)} w(q,a)|D_i(q,a) - D_{i+1}(q,a)| \geq .8.$$

Finally, since $w(q,a) \leq 1$ then

$$\sum_{(q,a)} |D_i(q,a) - D_{i+1}(q,a)| \geq .8.$$

Thus, $\Delta \geq .4$.

At this point, Proposition 7 implies that

$$H(D) \geq \frac{H(D_i) + H(D_{i+1})}{2} + .4^2/12.$$

Thus, if $H(D_{i+1}) \geq H(D_i) - .01$ then $H(D) > H(D_i) + .008$. Now, for any S_2,

$$\mathsf{win}(S, S_2) = \frac{\mathsf{win}(S_{1,i}, S_2) + \mathsf{win}(S_{1,i+1}, S_2)}{2},$$

so in particular,

$$\mathsf{win}(S, S_{2,j}) \geq .9$$

for $j < i$, since $\mathsf{win}(S_{1,i}, S_{2,j}), \mathsf{win}(S_{1,i+1}, S_{2,j}) \geq .9$. Thus, S meets the constraints required of $S_{1,i}$ yet has significantly higher entropy, which is a contradiction.