DIMACS Technical Report 95-21
July 1995

A Game-Theoretic Classification of Interactive Complexity Classes

by

Joan Feigenbaum[1]
AT&T Bell Labs, Room 2C-473
600 Mountain Avenue
Murray Hill, NJ 07974
jf@research.att.com

Daphne Koller[2]
UC Berkeley
Computer Science Division
Berkeley, CA 94720
daphne@cs.berkeley.edu

Peter Shor[3]
AT&T Bell Labs, Room 2D-149
600 Mountain Avenue
Murray Hill, NJ 07974
shor@research.att.com

[1]Permanent Member of DIMACS
[2]Supported by a University of California President's Postdoctoral Fellowship and by the Air Force Office of Scientific Research under Contract F49620-91-C-0080.
[3]Permanent Member of DIMACS

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

Game-theoretic characterizations of complexity classes have often proved useful in understanding the power and limitations of these classes. One well-known example tells us that PSPACE can be characterized by two-person, perfect-information games in which the length of a played game is polynomial in the length of the description of the initial position [Chandra *et al.*, Journal of the ACM, 28 (1981), pp. 114–133].

In this paper, we investigate the connection between game theory and interactive computation. We formalize the notion of a *polynomially definable game system* for the language L, which, informally, consists of two arbitrarily powerful players P_1 and P_2 and a polynomial-time referee V with a common input w. Player P_1 claims that $w \in L$, and player P_2 claims that $w \notin L$; the referee's job is to decide which of these two claims is true. In general, we wish to study the following question:

> *What is the effect of varying the system's game-theoretic properties on the class of languages recognizable by polynomially definable game systems?*

There are many possible game-theoretic properties that we could investigate in this context. The focus of this paper is the question of what happens when one or both of the players P_1 and P_2 have *imperfect information* or *imperfect recall.*

We use polynomially definable game systems to derive new characterizations of the complexity classes NEXP and coNEXP. We also derive partial results about other exponential complexity classes and isolate some intriguing open questions about the effects of imperfect information and imperfect recall. These results make use of recent work on complexity-theoretic aspects of games, e.g., [Koller *et al.*, Proc. of 26th ACM Symposium on Theory of Computing, 1994, pp. 750–759] and [Newman, Information Processing Letters, 39 (1991), pp. 67–71.]

These results were presented in preliminary form at the 10th IEEE Structure in Complexity Theory Conference, Minneapolis MN, June 1995.

1 Introduction

Game theory provides a framework in which to model and analyze conflict and cooperation among independent decision makers. Many disciplines have benefitted from this framework, including economics, operations research, military strategy, politics, and biology; see, for example, [AH92] for a thorough overview of the theory and its applications. In computer science, game theory has been applied to, e.g., artificial intelligence [Ro93], distributed computing [BL85], security and privacy [FGY93], and lower bounds [Ya77]. Most importantly for structural complexity theorists, games have been used to characterize complexity classes in illuminating ways. For example, a classical result of Chandra *et al.* [CKS81] tells us that PSPACE is characterized by two-person, perfect-information games in which the length of a played game is polynomial in the length of the description of the initial position.

In this paper, we present a general framework in which to study the connection between game theory and traditional complexity classes. Informally, a *polynomially definable game system* for the language L consists of two arbitrarily powerful players P_1 and P_2 and a polynomial-time referee V with a common input w. Player P_1 claims that $w \in L$, and player P_2 claims that $w \notin L$; the referee's job is to decide which of these two claims is true. In general, V is allowed to use randomness, but there are interesting cases in which this power is not needed. Our overall goal is to answer the following question:

> *What is the effect of varying the system's game-theoretic properties on the class of languages recognizable by polynomially definable game systems?*

There are many possible game-theoretic properties that we could investigate in this context. Here, we focus on what happens when at least one P_k has *imperfect information* (does not see all of the communication between the referee and the other player) or *imperfect recall* (does not remember all the information that he himself once knew).

This framework allows us to

1. *Unify and generalize the game-theoretic aspects of ealier work on the complexity of interactive computation [BFL91, CKS81, FRS94, FST88, LFKN92, Sh91].* For example, game theory is a natural framework in which to state the equivalence of "oracle proof systems" and "multiprover proof systems" given in [FRS94], because oracles and teams of noncommunicating provers are both examples of players with imperfect recall. An oracle has imperfect recall (actually, no recall whatsoever) by definition, because it is fixed before the game between the prover and the verifier begins. A team of provers that cannot communicate during the course of the game can be viewed collectively as one imperfect-recall player: When one team member moves, he does not have the information known only to other team members; thus, when viewed as one player, the team can be said to have "forgotten something that it once knew."

2. *Obtain new game-theoretic characterizations of traditional complexity classes such as NEXP and coNEXP that are analogous to the [CKS81] characterization of PSPACE.*

In a game that models a PSPACE-complete language, both players have perfect information. We show that, by depriving one or more players of information about the progress of the game, we obtain games that model languages in higher complexity classes. For example, NEXP-complete languages are modeled by games in which player P_1 has imperfect recall and player P_2 has perfect recall (but possibly imperfect information). Thus, there is an "information hierarchy" that parallels the complexity hierarchy between
PSPACE and NEXP.

3. *Isolate intriguing open questions about other exponential complexity classes and obtain partial answers to these questions.*

One interesting aspect of items 2 and 3 is that they use both results and proof techniques from recent papers on the complexity of solving games [KMvS94, Ne91].

Imperfect information games have been considered in several previous works in complexity theory, including [C89, CL86, KLRSS94, PR79, Re84]. A crucial difference between these works and ours is that the opposing players in [C89, CL86, KLRSS94, PR79, Re84] use *deterministic (pure) strategies*, whereas ours use *randomized (mixed) strategies*.[1] (These and other game-theoretic terms are formally defined in the next section.) We claim that allowing the players to use mixed strategies is a fundamental part of the game-theoretic paradigm. Language acceptance is typically defined in terms of the *existence* of a strategy for player P_1 that achieves certain goals against *all* strategies of player P_2. Intuitively, this paradigm implies that the existential player P_1 chooses his strategy first, and then P_2 can choose the optimal strategy against that particular fixed strategy of P_1. In particular, if P_1 is forced to choose a pure strategy, P_2 knows all of P_1's moves in advance, which essentially gives P_2 perfect information with respect to P_1's moves. This causes an asymmetry in the game: Player P_2 clearly has an advantage. Hence, the outcome of the game, and the definitions of the associated complexity class, depend on the decision that P_1 chooses his strategy first. But if the players are allowed to use randomized strategies, these problems disappear. Intuitively, knowing the other player's strategy is not as useful if that strategy is randomized, since it does not reveal that other player's moves in advance. The famous min-max theorem shows that the players' optimal randomized strategies are in *equilibrium*: If one player plays the optimal strategy, revealing this to the other player does not help the other player at all. It doesn't matter who chooses his strategy first. Hence, the use of mixed strategies allows us

[1] It is clear in [KLRSS94, PR79, Re84] that the players use pure strategies, because they do not toss coins at all (except for one minor point in [KLRSS94], which is not the central model or result of the paper). The players in [C89, CL86] *do* toss coins, and hence the statement that they use pure strategies requires explanation. At a particular node in a game of [C89, CL86], a player may be required to toss a coin, or he may be required to make a strategic move. But the probabilities associated with the coin tosses are fixed in advance and not chosen strategically by the player. Hence, coin tosses by either player correspond to "chance moves" in standard game-theoretic terminology; in our model, these coins would be tossed by the referee, not by either player. Whenever a player in [C89, CL86] is required to make a strategic move, he does so according to a pure strategy.

to use a framework in which the two players play symmetric roles and to study the effects of varying the information of *both* players.

The issue of mixed strategies in the context of imperfect information games was raised by Reif [Re79], although without the observation that the main point of using mixed strategies is the existence of equilibria. Reif left as an open question the effect of using such strategies on the complexity classes definable by imperfect information games. Our results can be viewed as answering Reif's question for certain types of imperfect information games.

Papadimitriou and Yannakakis [PY94] consider complexity classes defined by games in which the players may use mixed strategies. They are interested in the complexity of computing exact values of games, not in the complexity of approximating these values; hence their results are in some cases related to our game-theoretic model (Section 2) but not to our accept-reject model (Section 3). Their framework differs critically from ours, however, in that they consider games where there is no interaction between the two players. Each player independently chooses a strategy, and these determine the outcome of the game. This is essentially a game in *normal form*. Any game in *extensive form* (one where the players interact dynamically) can be converted into a game in normal form by having the players' strategies encode their moves at all possible points in the game. However, this often requires that an exponential amount of information be transmitted to the referee. Hence, a polynomially definable game system does not typically "fit" in their framework. The one case in which their bound is directly comparable to ours is pointed out in Section 4.

In Sections 2 and 3, we review the basics of game theory and give formal definitions of polynomially definable game systems and the questions we address. Our main results and some open questions are given in Section 4.

2 The game-theoretic model

We first recall the standard definition of a game in *extensive form* [Ku53]. The basic structure of the game is a finite directed tree whose nodes denote game states. A play of the game starts at the root, proceeds according to the players' actions, and ends at a leaf. Extensive-form games can have any finite number of players, but we will concentrate on the two-player case. The internal nodes of the tree are of three types: decision nodes of player P_1, decision nodes of player P_2, and nodes for *chance moves*. At each point in the game, the possible actions determine the outgoing edges at the corresponding node; each edge is therefore labelled with some action. The *payoff* function h determines a payoff $h(a)$ for each leaf a. In general, this payoff can be a vector denoting the payoff to each of the players. We will focus on zero-sum games, i.e., on the case where $h(a)$ is a real number denoting the amount that player P_2 must pay to player P_1. The behavior at the chance nodes is specified by a function β that defines a probability distribution over the possible choices at each chance node.

In order to represent situations in which players may not know everything that occurs in the game (for example, in the game of poker one player's hand is not known to other players), the set of decision nodes is partitioned into *information sets*. Each information set belongs to exactly one player P_k. Intuitively, the player cannot differentiate between different nodes in the same information set. Hence, the labels on the outgoing edges must be the same at all nodes in an information set. Constraints on the information available to the players can be determined by an examination of their information sets. Two properties of particular interest are *perfect information* and *perfect recall.* A player is said to have *perfect information* if at every point in the game he knows the entire history of moves up to that point. This corresponds to information sets that are singletons. A player is said to have *perfect recall* if at every point in the game he remembers everything he once knew and all of his own previous moves. Formally, this is the case if for every pair of nodes a, a' in the same information set of player P_k, the paths to a and a' pass through the same information sets of player P_k and the same decisions are taken at each one.

The players plan their actions according to *strategies.* The basic strategy in an extensive-form game is a *pure* (or deterministic) strategy. A pure strategy π^k of player P_k specifies a choice at each of his information sets. Note that, because the player knows only which information set he is at, a strategy must dictate the same move at all nodes in an information set. Let Π^k denote the set of player P_k's pure strategies. A *mixed strategy* μ^k of player P_k is a probability distribution on the set Π^k of his pure strategies. We can define the expected payoff $H(\mu)$ for each pair of mixed strategies $\mu = (\mu^1, \mu^2)$. A strategy pair μ, together with the distribution β over the chance moves, induces a probability distribution on the leaves of the tree. We denote the probability of reaching a leaf a by $\Pr_\mu(a)$. The expected payoff $H(\mu)$ is then defined to be

$$H(\mu) = \sum_a \Pr_\mu(a) h(a). \tag{1}$$

In the case of zero-sum games, there is a natural definition of optimal strategy: the strategy guaranteeing the best worst-case payoff. More precisely, define:

$$\begin{aligned}\tilde{\mu}^1 &= \mathrm{argmax}_{\mu^1} \min_{\mu^2} H(\mu^1, \mu^2)\\ \tilde{\mu}^2 &= \mathrm{argmin}_{\mu^2} \max_{\mu^1} H(\mu^1, \mu^2)\end{aligned}$$

to be the max-min and min-max strategies respectively. The famous min-max theorem of von Neumann implies that $\tilde{\mu}^1$ is the optimal strategy against $\tilde{\mu}^2$, and vice versa. This implies that each player P_k, by playing $\tilde{\mu}^k$, can guarantee that his payoff will be no worse than $H(\tilde{\mu}^1, \tilde{\mu}^2)$. Of course, his payoff may be better if the other player uses some other strategy. The value $H(\tilde{\mu}^1, \tilde{\mu}^2)$ is called the *value of the game.* Note that $\tilde{\mu}^1$ and $\tilde{\mu}^2$ are in equilibrium: Even if one player knows in advance that the other player is playing $\tilde{\mu}^k$, he cannot choose a better strategy. This important property is not guaranteed if the players are barred from randomizing their strategies. In that case, the value of the game is not defined, and knowing the strategy of the other player can significantly affect the payoff (even if the other player is playing "optimally").

In this context, we would like to answer the following question:

> *What is the complexity of computing the value of a game, and how does it depend on the various properties of the game?*

This complexity clearly depends on the way in which the game is represented. We return to this issue later on.

We use the following notation to formalize this problem. Let x and y be parameters that specify the nature of players P_1 and P_2, respectively. They range over the set S consisting of pi (perfect information), pr (perfect recall, although possibly imperfect information), and ir (imperfect recall). This defines nine functions $\mathrm{Value}(x, y)_{x,y \in S}$; in each, the input to the function is a game in which P_1 is of type x and P_2 is of type y, and the output is the value of the game. For any threshold $\lambda \in [0, 1]$, we obtain two corresponding decision problems, namely $\mathrm{Value}_{\leq\lambda}(x, y)$ and $\mathrm{Value}_{\geq\lambda}(x, y)$; the game G is a yes-instance of $\mathrm{Value}_{\leq\lambda}(x, y)$ (resp. $\mathrm{Value}_{\geq\lambda}(x, y)$) if the value of G is at most (resp. at least) λ. For example, $\mathrm{Value}(pi, ir)$ is the function defined on games in which player P_1 has perfect information and player P_2 has imperfect recall, and, on input G, it computes the value of G. Similarly, the yes-instances of $\mathrm{Value}_{\leq\lambda}(pi, ir)$ are games in which player P_1 has perfect information, player P_2 has imperfect recall, and the value is at most λ.

In Section 4 below, we use the following recent results about complexity theoretic aspects of games.

Theorem 2.1 (cf. [Ne91]): In any game with payoffs in $[0, 1]$, each player has a near-optimal mixed strategy that chooses uniformly at random from a logarithmic-size multiset of pure strategies. Specifically, if the opponent has N pure strategies, there exists a multiset of size $\lceil (\ln N)/\epsilon^2 \rceil$ such that the corresponding mixed strategy guarantees a payoff within less than ϵ of optimal.

Theorem 2.2 (cf. [KMvS94]): For any strategy μ in a two-player, zero-sum game in extensive form, there exists an equivalent strategy μ' such that the support of μ' contains at most l pure strategies, where l is the number of leaves in the game tree.

Theorem 2.3 (cf. [KMvS94]): The optimal strategies of a two-player, zero-sum, perfect-recall game in extensive form are the solutions of a linear program whose size is polynomial in the size of the game tree. Furthermore, this linear program can be constructed in time polynomial in the size of the game tree.

3 The accept-reject model

A *polynomially definable game system* for a language L consists of two arbitrarily powerful players P_1 and P_2 and a polynomial-time referee V. The referee may be probabilistic, but there are some interesting cases in which V does not need randomness. In particular, this will be the case in most of our lower bounds. P_1, P_2, and V have a common input tape. On input w, P_1 claims that w is in L, P_2 claims that w is not in L, and V's job is to decide which of these two claims is true.

Each input w to such a system determines a *polynomially definable game* G as follows. The game is essentially run by the referee V. The moves in the game are relayed by the players to V. Neither player sees V's communication with the other, but V can transmit information about the current status of the game to one or both players. This reflects the fact that the players can have imperfect information to varying degrees. When the interaction is finished, V either accepts or rejects w. Since the referee is polynomial-time, G lasts for $poly(n)$ moves, and each move can be written down in $poly(n)$ bits, where $n = |w|$. The resulting game G clearly defines a two-person, zero-sum game tree in which the length of each path is polynomial. If V is probabilistic, then his coin tosses correspond to chance moves in the game tree.

We want to use the game-theoretic notions of "max-min/min-max strategies" and "value of the game" to define the class of languages accepted by such systems. We must therefore allow the players to use mixed strategies. That is, for each possible input w, each player has a probability distribution over the space of his deterministic strategies. At the beginning of the game, the players examine w and independently choose a pure strategy using their respective probability distributions; those pure strategies are then played throughout the game. Since the game tree has exponential size, a pure strategy also has exponential size. An arbitrary mixed strategy could of course have size doubly exponential in n.

We say that a language L is accepted by the polynomially definable game system (P_1, P_2, V) if the following two conditions are satisfied:

- For all $w \in L$, there exists a mixed strategy μ^1 for P_1 such that, for all strategies μ^2 for P_2, V accepts with probability at least $2/3$.
- For all $w \notin L$, there exists a mixed strategy μ^2 for P_2 such that, for all strategies μ^1 of P_1, V accepts with probability at most $1/3$.

Here, the probability is taken over the pure strategies of both players (if they use mixed strategies) and the coin tosses of V (if any).

We would like to study the following general question:

> *What is the effect of varying the system's game-theoretic properties on the class of languages recognizable by polynomially definable game systems?*

There are many possible game-theoretic properties that we could investigate in this context; in this paper, we focus on the information available to the players P_1 and P_2 throughout the game. Note that a player has perfect information only if the referee reveals to him the outcomes of any coin tosses and any moves taken by the other player. A player has perfect recall if at every point in the game he remembers the entire history of his communication with the referee. Hence, the accept-reject model defines nine language classes, which we denote $\{\mathcal{L}(x,y)\}_{x,y \in S}$, where $S = \{pi, pr, ir\}$, x specifies the nature of P_1, and y specifies the nature of P_2, as in the definition of the Value problem in the previous section. Thus $\mathcal{L}(pi, ir)$ is the class of languages defined by game systems in which P_1 has perfect information and P_2 has imperfect recall.

4 Main Results

Given a polynomially definable game system, each input w defines a game tree whose depth is polynomial in $|w|$. We can view w as a compact representation for a game tree. Games constructed from polynomially definable game systems are a subset of the class of games that can be represented succinctly (i.e., in space polynomial in the depth of the game tree).

We are interested in the general problem of computing the value of succinctly represented games, and we approach this problem from two perspectives:

A What is the complexity of computing the value of a succinctly represented game in which P_1 is of type x and P_2 of type y?

B What is the class $\mathcal{L}(x, y)$?

Problem A (the game-theoretic model) addresses the complexity of computing the value of a game *exactly*. Because the definition of a polynomially definable game system requires a (1/3, 2/3) gap in acceptance probabilities, problem B (the accept-reject model) addresses the complexity of *approximating* the value of succinctly represented games.

In this section, we address problems A and B for each pair of values x, y. In the cases x, y in which we upper bound problem A and lower bound problem B by the same complexity class, we obtain tight characterizations of $\mathcal{L}(x, y)$ and $\text{Value}_{\leq\lambda}(x, y)$. These are the cases in which approximating the value of a game and computing this value exactly have the same complexity. In the other cases, we obtain partial results and state related open questions.

In our bounds on the complexity of $\text{Value}(x, y)$, $\text{Value}_{\leq\lambda}(x, y)$, and $\text{Value}_{\geq\lambda}(x, y)$, the length of the input is the depth of a game tree. If the game is represented as an input string w to a polynomially definable game system, then this depth is just $n = |w|$. However, there are games that can be represented in space polynomial in the depth of the tree that do not come from polynomially definable game systems, and hence the way we have stated these results is more general. In results about the language classes $\mathcal{L}(x, y)$, the input length is by definition $n = |w|$, where w is the input of a polynomially definable game system.

We begin by proving that the classical PSPACE characterization of perfect-information games can be stated in our framework.

Theorem 4.1 *Value*(pi, pi) *can be computed in polynomial space.*

Proof: Recall the well-known theorem that, in a perfect-information game (even with chance moves), there is a deterministic optimal strategy for each player [Ze13]. Such optimal strategies, and the resulting value of the game, can easily be computed in polynomial space, using the standard min-max algorithm: The game tree is traversed depth-first. We know the payoff at each leaf. Now, consider a node u for which all the children have been assigned a payoff. If u corresponds to a move of P_1, and P_1 plays optimally, then the payoff at u is the maximum of the payoffs of its children. Similarly, if u corresponds to a move of P_2, then the payoff at u is the minimum of the payoffs of its children. If u corresponds to a random choice of V, then the payoff is the expectation of the payoffs of the children. The value of the game is the expected payoff at the root. This depth-first search of the tree can clearly be executed in space polynomial in the depth of the tree. ∎

Theorem 4.2 PSPACE $\subseteq \mathcal{L}(pi, pi)$.

Proof: This result is a simple corollary of the classical result of Chandra, Kozen, and Stockmeyer [CKS81], which asserts that PSPACE is equal to alternating polynomial time. Consider any language L in PSPACE, and let M be an alternating polynomial time Turing machine that accepts it. We define a game system as follows: The referee plays the role of M, except that, in any existential (resp. universal) step, he asks P_1 (resp. P_2) which branch to take, and moves accordingly. At all times, the entire history of the computation is revealed to both players. It is clear from the definition of alternating polynomial time that, if $w \in L$, then there exists a *deterministic* strategy for P_1 such that for all strategies of P_2, V always accepts. Similarly, if $w \notin L$, there exists a deterministic P_2 such that, for all P_1, V always rejects. Hence, V can decide the winner with zero probability of error. ∎

Corollary 4.3 *For any threshold $\lambda \in [0,1]$, the language $Value_{\leq\lambda}(pi, pi)$ is PSPACE-complete.*

Corollary 4.4 $\mathcal{L}(pi, pi) =$ PSPACE.

We remark that the class $\forall$BC-TIME($poly(n)$) defined in [C89, CL86] is formally equivalent to $\mathcal{L}(pi, pi)$.

Next, we consider games in which at least one player has imperfect information, but both have perfect recall. This class of games provides our main open problem, stated in Question 4.9 below.

Theorem 4.5 *$Value(pr, pr)$ can be computed in deterministic exponential time.*

Proof: The size of a game tree G is at most exponential in its depth. We can now use Theorem 2.3 to construct and solve a linear program whose size is polynomial in the size of G. ∎

Theorem 4.5 is directly comparable to the upper bound in part (c) of the Classification Theorem of [PY94]. Our bound, which makes essential use of Theorem 2.3, is considerably stronger, because our games are presented in extensive form, whereas those of [PY94] are presented in normal form. It is easy to see that a game in normal form (as in [PY94]) can be viewed as a game in extensive form of the same size. However, as we observed, a game in extensive form generates a normal-form game that is typically exponentially larger.

In this case, we can prove a lower bound only in the game-theoretic model and not in the accept-reject model.

Theorem 4.6 *$Value(pr, pr)$ is EXP-hard.*

Proof: Our proof is based on the proof that linear programming is P-hard [DLR79]. This result can be used to show that solving two-person zero-sum games in normal (matrix) form is also P-hard. The following is a long and very technical exposition showing how this P-hardness result can be scaled up to the exponential case.

Consider a deterministic exponential time Turing machine M and some input w. Our basic construction will be as follows. Following the lines of Jones and Laaser [JL74], we describe the computation of M on w using a set $\mathcal{C}$ of propositional Horn clauses. A *Horn clause* has the form $p_1 \wedge \ldots \wedge p_k \Rightarrow q$ where $p_1, \ldots, p_k$ are all positive literals; q can be either positive or negative, and k can be 0. The set of Horn clauses will have a unique satisfying truth assignment if M accepts w and no satisfying assignment if M rejects w. Player P_1 tries to prove that M accepts by assigning truth values to primitive propositions; player P_2 tries to prove that M rejects by picking clauses that P_1 should satisfy. More precisely, V asks P_1 for a single proposition to be set to true and P_2 for a single clause. The payoff (degree of confidence) is determined by whether the proposition is in the tail of the clause, in the head of the clause, or not in the clause at all. We will show that, if M accepts, there exists a randomized strategy for P_1 that guarantees him an expected payoff of at least 0, no matter which clause is chosen. On the other hand, if M rejects, there is no such strategy.

The translation of M into $\mathcal{C}$ is as follows. Let S be the set of states of M, A be the set of tape symbols, and $\Sigma = S \times A \cup A$. Without loss of generality, we assume that M has a single accepting state s_f and that it always terminates with the head on tape location 1, with a 0 written at that location. Let T denote the maximum possible running time of M on an input of length $n = |w|$; T is exponential in n. Clearly T is also an upper bound on the highest tape location reached in the computation of M on w. For each $\sigma \in \Sigma$, each time point $t \in \{1, \ldots, T\}$, and each possible tape location $l \in \{1, \ldots, T\}$, we have a propositional symbol $p[t, l, \sigma]$. Intuitively, for $a \in A$, $p[t, l, a]$ is true iff tape location l contains a at time t; for $(s, a) \in S \times A$, $p[t, l, (s, a)]$ is true iff at time t the state is s, the head is at tape location l, and that location contains a. Clearly, the description of a single proposition requires only polynomially many bits, so that P_1 can relay his move to V in polynomial time. As we suggested, the computation of M on w uniquely determines a truth assignment for all the propositional variables. Let Q be the set of propositions that are assigned true by that truth assignment.

The clauses describe the computation of M in terms of these propositions. There are three types of clauses. Those describing the initial state consist of a single literal; i.e., $p[1, 1, (w_1, s_0)]$, and $\neg p[1, 1, \sigma]$ for all other σ. Those describing the steps of the computation have the form

$$p[t, l-1, \sigma_1] \wedge p[t, l, \sigma_2] \wedge p[t, l+1, \sigma_3] \Rightarrow p[t+1, l, \sigma]$$

and

$$p[t, l-1, \sigma_1] \wedge p[t, l, \sigma_2] \wedge p[t, l+1, \sigma_3] \Rightarrow \neg p[t+1, l, \sigma']$$

for appropriate choices of $\sigma_1, \sigma_2, \sigma_3$ and $\sigma \neq \sigma'$. Finally, there is a single clause asserting that M accepts: $p[T, 1, (s_f, 0)]$. This is the set $\mathcal{C}$ of clauses; note that the tail of each clause contains between zero and three propositions. Thus, as each clause refers to a constant number of propositions, each one can be described using polynomially many bits. Furthermore, given w and the transition table of M, the referee V can easily decide in polynomial time whether a clause received from P_2 is consistent with the initial configuration or describes a legal transition of M. If the clause received is illegal, P_2 gets a very large negative payoff (i.e., V accepts with a very high degree of confidence). Note that the truth assignment encoded by Q satisfies all the clauses, except perhaps $p[T, 1, (s_f, 0)]$. This last clause is satisfied iff M accepts w.

In order to define the game, it remains to define the payoff assigned by V to each pair: a proposition obtained from P_1 and a clause obtained from P_2. Assume that the proposition is r and that the clause C has the form $p_1 \wedge \ldots \wedge p_k \Rightarrow q$, where $0 \leq k \leq 3$. Intuitively, we would like to reward P_1 for satisfying the head of the clause and penalize him for satisfying its tail. Then, if he plays the right strategy, the expected value of each clause will be as desired. However, we need to normalize for the fact that clauses may have a different number of propositions in the tail.

Therefore, for $\alpha = \frac{k-1}{|Q|}$, we define:

$$H(r, C) = \begin{cases} 1 + \alpha & r = q \\ -1 + \alpha & r = p_i \\ \alpha & otherwise. \end{cases}$$

Note that, if the clause has the form $p_1 \wedge \ldots \wedge p_k \Rightarrow \neg q$, then we express it as $p_1 \wedge \ldots \wedge p_k \wedge q \Rightarrow false$ and essentially ignore the first case in the definition of $H(r, c)$.

Now, assume that $w \in L$ and consider the strategy μ^1 that assigns probability $\frac{1}{|Q|}$ to all propositions in Q and 0 to all the rest. This corresponds, intuitively, to assigning "true" to propositions in Q and "false" elsewhere. Now, consider any clause. It is satisfied by this truth assignment. Therefore, either all propositions in it are assigned "true" or some proposition in the tail is assigned "false." In the first case, the expected payoff for the clause is

$$\frac{1}{|Q|}(1 - k + |Q|\alpha) = 0.$$

In the second case, the payoff is at least (when precisely one proposition in the tail is falsified)

$$\frac{1}{|Q|}(-(k-1) + |Q|\alpha) = 0.$$

Since there exists a strategy μ^1 guaranteeing a value of 0 against every clause, the value of the game in this case is at least 0.

The other case, where $w \notin L$, is a little harder to see. It can be proved by equivalence to a certain linear system used by Dobkin, Lipton, and Reiss [DLR79], which in turn is equivalent to unit resolution in Horn clauses. We will sketch a direct proof. The proof asserts that any variable in Q must get probability at least $\frac{1}{|Q|}$. This clearly suffices. The proof is by induction on t, the time to which the proposition $p[t, l, \sigma]$ refers. In the base case, we have clauses determining the precise truth assignment to each variable $p[0, l, \sigma]$. If x_i is the probability assigned to that proposition, then the payoff to that clause is

$$x_i + \alpha \sum_{j=1}^{N} x_j = x_i + \frac{-1}{|Q|},$$

since $k = 0$. In order to ensure a payoff of at least 0 against this clause, we must have $x_i \geq \frac{1}{|Q|}$. Now, consider some proposition $q = p[t+1, l, \sigma] \in Q$. Because of the form of our construction, there must be a clause $p_1 \wedge \ldots \wedge p_k \Rightarrow q$ where all of the p_i are in Q. Let x_i represent the probability of q and $x_{j_1}, \ldots, x_{j_k}$ represent the probabilities of $p_1, \ldots, p_k$, respectively. Then the payoff of this clause is

$$x_i - (x_{j_1} + \cdots + x_{j_k}) + \sum_{j=1}^{N} x_j \alpha = x_i - (x_{j_1} + \cdots + x_{j_k}) + \frac{k-1}{|Q|}.$$

By the inductive hypothesis, $x_{j_1}, \ldots, x_{j_k} \geq \frac{1}{|Q|}$. Hence, if we want the payoff for this clause the be at least 0, we conclude that

$$x_i - k\frac{1}{|Q|} + \frac{k-1}{|Q|} \geq 0,$$

so that $x_i \geq \frac{1}{|Q|}$ as required. ∎

Theorem 4.6 is analogous to the lower bound in part (c) of the Classification Theorem of [PY94]. In this case, the bounds are equally strong, and the proofs are similar.

Corollary 4.7 *For any threshold $\lambda \in [0,1]$, the language $Value_{\leq\lambda}(pr, pr)$ is EXP-complete.*

Corollary 4.8 $\mathcal{L}(pr, pr) \subseteq$ EXP, $\mathcal{L}(pr, pi) \subseteq$ EXP, *and* $\mathcal{L}(pi, pr) \subseteq$ EXP.

Proof: The first inequality follows from Theorem 4.5. The second and third follow from the fact that perfect information is a special case of perfect recall. ∎

Question 4.9 *Is $\mathcal{L}(pr, pr) =$ EXP? This question can be restated as "Can Theorem 4.6 be proven in the accept-reject model?" or "Is approximating the value of a perfect recall game as hard as computing it exactly?"*

Alternatively, is $\mathcal{L}(pr, pr) =$ PSPACE? We can restate this alternative as "Are the game-theoretic and accept-reject models different for perfect recall games?" or "Is approximating the value of a perfect recall game easier than computing is exactly?"

Question 4.9 has arisen before in the context of cryptographic complexity – see [FST88]. It should be pointed out, of course, that the two alternatives stated explicitly in the question are not the only two; $\mathcal{L}(pr, pr)$ may be equal neither to EXP nor to PSPACE.

The next class of games we consider is the one in which exactly one player has imperfect recall. We obtain tight complexity results for this class.

Theorem 4.10 *For any threshold $\lambda \in [0,1]$, the language $Value_{\geq\lambda}(ir, pr)$ is in NEXP.*

Proof: As before, the game tree G has size exponential in its depth n. Let μ be an optimal strategy of P_1 in G. By Theorem 2.2, we may assume without loss of generality that the support of μ is of size singly exponential in n. We now construct a nondeterministic exponential-time machine M that proceeds as follows. It first guesses a support S for μ. Then, it constructs the following game G', with players P_1' and P_2'. In the first phase of G', P_1' chooses one of the strategies in S. In the second phase, P_1' and P_2' play G' exactly as P_1 and P_2 play G, except that whenever P_1' is to move, he must do so according to the strategy he picked in the beginning. Technically, the game subtrees of G' that correspond to the second phase are simply subtrees of G, in which only P_2' gets to move.

The overall game tree of G' is still of depth polynomial in n and of size singly exponential in n. Furthermore, G' is a perfect recall game: Player P_2' has perfect recall, because P_2 has perfect recall in G; by construction, P_1' has nothing to recall, because his one move is made when he chooses a strategy from S in phase one. By Theorem 2.3, the machine M can solve G' optimally in exponential time. If the value of G' is at least λ, then M accepts; otherwise, it rejects. Clearly, M has an accepting computation iff P_1 has a strategy guaranteeing him a payoff of at least λ iff the value of the game is at least λ. ▮

We could also prove a variant of this theorem utilizing Theorem 2.1 rather than Theorem 2.2. This would result in a weaker theorem that can only approximately determine the value of the game. However, this version would still suffice to prove Corollaries 4.14 and 4.15 below, because the polynomially definable game system model requires a gap in the value of the game.

Theorem 4.11 NEXP $\subseteq \mathcal{L}(ir, pi)$.

Proof: Recall that a language is decidable in nondeterministic exponential time if and only if it has a multiprover interactive proof system [BFL91]. The roles of all provers in such a system can be played collectively by one player with imperfect recall. This is implied by the theorem of Fortnow *et al.* [FRS94], which says that a language has a multiprover proof system if and only if it has a one-prover "oracle" proof system; to see this, note that an oracle is an example of a player with imperfect recall (actually with no recall at all). More directly, we can regard the behavior of a multiprover system on a particular input w as a perfect recall game played by the provers and the verifier; at every leaf of the game tree, the payoffs to all provers are the same. Consider a single player that plays the roles of all of the provers. Whenever it is this player's turn to move, his information set is that of the particular prover whose role he is playing at the time. Because he does *not*, during this turn, have the information available to the other provers, he has imperfect recall.

It is now clear that any nondeterministic exponential-time language L is recognizable by a polynomially definable game system in which P_1 has imperfect recall and P_2 has perfect information (and *a fortiori* has perfect recall). To test whether w is in L, the game system's referee requires P_1 to prove that $w \in L$ exactly as P_1 would do by playing the roles of all provers in a multiprover system for L, and V accepts if and only if the proof system's verifier would accept. This system does not require V to receive any communication from P_2, and thus he can give P_2 perfect information about his interaction with P_1 without affecting the outcome of the game.

Observe that it is possible to prove the same result using a deterministic referee who gets bits from both players and uses their exclusive or as random bits. In this proof, the perfect information of P_2 can be maintained by asking him for the bits at the beginning of the game. ∎

Corollary 4.12 *For any threshold $\lambda \in [0,1]$, the languages $Value_{\geq\lambda}(ir, pi)$ and $Value_{\geq\lambda}(ir, pr)$ are NEXP-complete.*

Corollary 4.13 *For any threshold $\lambda \in [0,1]$, the languages $Value_{\leq\lambda}(pi, ir)$ and $Value_{\leq\lambda}(pr, ir)$ are coNEXP-complete.*

Corollary 4.14 $\mathcal{L}(ir, pi) = \mathcal{L}(ir, pr) = \mathrm{NEXP}$.

Corollary 4.15 $\mathcal{L}(pi, ir) = \mathcal{L}(pr, ir) = \mathrm{coNEXP}$.

The last class of games is that in which both players have imperfect recall. We give nontrivial upper and lower bounds for this class, but they do not quite yield a tight characterization. (See Question 4.19 below.)

Theorem 4.16 *For any threshold $\lambda \in [0,1]$, the language $Value_{\geq\lambda}(ir, ir)$ is in $\Sigma_2^{\mathrm{EXP}} \cap \Pi_2^{\mathrm{EXP}}$.*

Proof: As in the proof of Theorem 4.10, we use Theorem 2.2 to conclude that the optimal strategies for P_1 and P_2 in G have supports, say S_1 and S_2, that are of size singly exponential in n. The Σ_2^{EXP} machine M first guesses a support S_1. For any support S_2, a derived game G' with players P_1' and P_2' is obtained as follows. In phase 1 of G', P_1' and P_2' choose pure strategies from S_1 and S_2, respectively. In phase 2, G' is played exactly as G, subject to the constraint that the players must use the strategies that they chose in phase 1. As in the proof of Theorem 4.10, the game tree of G' is of exponential size. Furthermore, G' is a perfect recall game, because the players each make only one nontrivial move (in phase 1), after which there is nothing that they need to recall. Thus G' can be solved in exponential time, by Theorem 2.3. M accepts iff the value of G' is at least λ. Clearly, the value of G is at least λ iff there exists a support S_1 such that for all supports S_2, the value of the resulting game G' is at least λ. The Π_2^{EXP} machine M' is similar; it verifies that for all supports S_2, there exists a support S_1 such that the value of the resulting game is at least λ. By the min-max theorem, M' is equivalent to M. ∎

Once again, we could prove a weaker variant of this theorem utilizing Theorem 2.1, which would suffice for our result concerning the accept-reject model (Corollary 4.18).

Theorem 4.17 *In the accept-reject model, every language decidable in deterministic exponential time with an NP oracle has a polynomially definable game system in which both players have imperfect recall.*

Proof: Throughout, we will use the fact, explained in the proof of Theorem 4.11, that a multiprover interactive proof can be modelled as a player with imperfect recall. Let $L = L(M^A)$, where M is a deterministic, exponential-time oracle machine and A is an NP oracle. Because M is deterministic, there is, for every word w, a unique, exponential-length string s of correct answers to the oracle queries that M makes on input w. The statement "M accepts (resp. rejects) w if the oracle answers are s" can be verified using an MIP proof system.

The difficulty lies in verifying that the bits of s are correct; because there are exponentially many of them, V cannot ask the P_i's to provide an MIP proof for each bit. Instead, P_1 and P_2 must encode their claimed values (say s_1 and s_2) for s in a way that allows V to use binary search to discover the first bit in which s_1 and s_2 differ. If this first difference is the j^{th} bit, then V asks whichever of P_1 or P_2 claims that the j^{th} oracle query q_j is in A to prove this claim using a standard MIP proof system. V then accepts this P_k's claim about the membership of w in L if and only if he accepts the proof that $q_j \in A$.

In order to permit binary search, we require P_1 and P_2 to encode s and all prefixes of s in a good error-correcting code. Here "good" means that V can be convinced with high probability in time $poly(n)$ that a string of length $2^{poly(n)}$ is close to a codeword and that two unequal strings of length $2^{poly(n)}$ that are both close to codewords are indeed unequal. Multivariate polynomials provide good codes, as we now explain.

For concreteness, assume that s is of length 2^n; the argument goes through *mutatis mutandis* for strings s of length $2^{p(n)}$, where p is an arbitrary polynomial, and strings whose lengths are not powers of two can be padded appropriately. Interpret s as a boolean function f_n on n input bits. Let $\{K_n\}_{n\geq 1}$ be a fixed sequence of finite fields such that $n < |K_n| = poly(n)$; note that a polynomial-time V can do arithmetic in these fields. It is well-known that there is a unique multilinear polynomial $g_n \in K_n[X_1, \ldots, X_n]$ that "extends" f_n over K_n in the sense that g_n agrees with f_n on inputs in $\{0,1\}^n$, where boolean 0's and 1's are identified with the corresponding field elements. (See [Fe93] for an explicit formula for g_n and an overview of the properties of multilinear extensions and their many uses in complexity theory.) The encoding of s is simply the multilinear polynomial g_n, written as a ($|K_n|^n$-sized) table of values.

For any $0 \leq \delta \leq 1$, a function $h : K_n^n \mapsto K_n$ is said to be δ-*close* to g_n if the number of domain elements $x \in K_n^n$ on which $h(x) \neq g_n(x)$ is less than $\delta|K_n|^n$. The results of Babai, Fortnow, and Lund [BFL91] yield the following useful fact.

Multilinearity testing: Let h be a function mapping K_n^n to K_n. There is a probabilistic $poly(n)$-time oracle machine T such that, if h is an n-variable, multilinear polynomial, then T^h always accepts, and, if h is not 1/4-close to an n-variable, multilinear polynomial, then T^h accepts with probability at most $2^{-poly(n)}$. The specific polynomial in the exponent of the error probability can be made as large as desired, at the cost of increasing the (still polynomial) running time of T.

Using this multilinearity test, the referee V in a polynomially definable game system can test with very high accuracy that all of the strings supplied by P_1 and P_2 are 1/4-close to codewords. If any of the strings supplied by P_k fails a multilinearity test, then V accepts P_{3-k}'s claim about whether or not $w \in L$. It remains to show how V can test equality of two functions that are both 1/4-close to multilinear polynomials in $K_n[X_1, \ldots, X_n]$. This can also be done using a well-known fact about polynomials.

Schwartz's Lemma [Sc80]: Two distinct polynomials in $K_n[X_1, \ldots, X_n]$, each of total degree at most n, must differ on at least $(1 - n/|K_n|) \cdot |K_n|^n$ of the $|K_n|^n$ points in their domain.

This lemma clearly implies that, if P_k supplies a function h_k that is 1/4-close to a multilinear polynomial in $K_n[X_1, \ldots, X_n]$, and $h_1 \neq h_2$, then the referee V can, with probability $1 - 2^{-poly(n)}$, find a point x such that $h_1(x) \neq h_2(x)$ by sampling polynomially many points in the domain, independently and uniformly at random. Once again, the polynomial exponent in the error probability can be made as large as desired at the cost of increasing the (still polynomial) number of sample points. ∎

Corollary 4.18 $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathcal{L}(ir, ir) \subseteq \Sigma_2^{\mathrm{EXP}} \cap \Pi_2^{\mathrm{EXP}}$.

Question 4.19 *What is the exact complexity of* $\mathcal{L}(ir, ir)$*? In particular, can the upper bound be improved to* $\mathrm{EXP}^{\mathrm{NP}}$*?*

Finally, we include two general open questions about polynomially definable game systems.

Question 4.20 *Reif [Re84, Re79] defines a richer classification of information structure than we (or traditional game theorists) do and develops an associated complexity hierarchy. Can our model, in which mixed strategies are essential, be combined with his?*

Question 4.21 *Perfect-information games in which the depth of the game tree is polynomial retain the same language-recognition power when we replace one of the all-powerful players by a "random player." This is true in both the bounded-error model [GS89] and the unbounded-error model [Pa85]. Does a similar result about random players hold for games of imperfect information or imperfect recall?*

References

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

[BFL91] L. Babai, L. Fortnow, and C. Lund, "Nondeterministic Exponential Time has Two-Prover Interactive Protocols," *Computational Complexity*, 1 (1991), pp. 3–40.

[BL85] M. Ben-Or and N. Linial, "Collective Coin Flipping, Robust Voting Games, and Minima of Banzhaf Values," *Proceedings of the 26th Symposium on Foundations of Computer Science*, IEEE Computer Society, Los Alamitos, 1985, pp. 408–416.

[CKS81] A. Chandra, D. Kozen, and L. Stockmeyer, "Alternation," *Journal of the ACM*, 28 (1981), pp. 114–133.

[C89] A. Condon, *Computational Models of Games*, ACM Distinguished Dissertation, MIT Press, Cambridge, 1989.

[CL86] A. Condon and R. Ladner, "Probabilistic Game Automata," *Proceedings of the 1st Structure in Complexity Theory Conference*, Lecture Notes in Computer Science, vol. 223, Springer, Berlin, 1986, pp. 144–162.

[DLR79] D. Dobkin, R. Lipton, and S. Reiss, "Linear Programming is LOG-SPACE Hard for P," *Information Processing Letters*, 8 (1979), pp. 96–97.

[FST88] U. Feige, A. Shamir, and M. Tennenholtz, "The Noisy Oracle Model," *Advances in Cryptology – Crypto '88*, Lecture Notes in Computer Science, vol. 403, Springer, Berlin, 1990, pp. 284–296.

[Fe93] J. Feigenbaum, "Locally Random Reductions in Interactive Complexity Theory," in *Advances in Computational Complexity Theory*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 13, AMS, Providence, 1993, pp. 73–98.

[FRS94] L. Fortnow, J. Rompel, and M. Sipser, "On the Power of Multiprover Interactive Protocols," *Theoretical Computer Science*, 134 (1994), pp. 545–557.

[FGY93] M. Franklin, Z. Galil, and M. Yung, "Eavesdropping Games: A Graph-Theoretic Approach to Privacy in Distributed Systems," *Proceedings of the 34th Symposium on Foundations of Computer Science*, IEEE Computer Society, Los Alamitos, 1993, pp. 670–679.

[GS89] S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," *Randomness and Computation*, Advances in Computing Research, vol. 5, JAI Press, Greenwich, 1989, pp. 73–90.

[JL74] N. Jones and W. Laaser, "Complete problems for deterministic polynomial time," *Proceedings of the 6th Symposium on Theory of Computing*, ACM, New York, 1974, pp. 40–46.

[KLRSS94] M. Kiwi, C. Lund, A. Russell, D. Spielman, and R. Sundaram, "Alternation in Interaction," *Proceedings of the 9th Conference on Structure in Complexity Theory*, IEEE Computer Society, Los Alamitos, 1994, pp. 294–303.

[KMvS94] D. Koller, N. Meggido, and B. von Stengel, "Fast Algorithms for Finding Randomized Strategies in Game Trees," *Proceedings of the 26th Symposium on Theory of Computing*, ACM, New York, 1994, pp. 750–759.

[Ku53] H. Kuhn, "Extensive games and the problem of information," *Contributions to the Theory of Games II*, H. Kuhn and A. Tucker, editors, Princeton University Press, Princeton, 1953, pp. 193–216.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM*, 39 (1992), pp. 859–868.

[Ne91] I. Newman, "Private vs. Common Random Bits in Communication Complexity," *Information Processing Letters*, 39 (1991), pp. 67–71.

[Pa85] C. Papadimitriou, "Games Against Nature," *Journal of Computer and System Sciences*, 31 (1985), pp. 288–301.

[PY94] C. Papadimitriou and M. Yannakakis, "On Complexity as Bounded Rationality," *Proceedings of the 26th Symposium on Theory of Computing*, ACM, New York, 1994, pp. 726–733.

[PR79] G. Peterson and J. Reif, "Multiple Person Alternation," *Proceedings of the 20th Symposium on Foundations of Computer Science*, IEEE Computer Society, Los Alamitos, 1979, pp. 348–363.

[Re84] J. Reif, "The Complexity of Two-Player Games of Incomplete Information," *Journal of Computer and System Sciences*, 29 (1984), pp. 274–301.

[Re79] J. Reif, "Universal Games of Incomplete Information," *Proceedings of the 11th Symposium on Theory of Computing*, ACM, New York, 1979, pp. 288–307.

[Ro93] J. Rosenschein, "Consenting Agents: Negotiation Mechanisms for Multi-Agent Systems," *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, San Mateo, 1993, pp. 792–799.

[Sc80] J. Schwartz, "Fast Probabilistic Algorithms for Verification of Polynomial Identities," Journal of the ACM, 27 (1980), 701-717.

[Sh91] A. Shamir, "IP = PSPACE," *Journal of the ACM*, 39 (1992), pp. 869–877.

[Ya77] A. Yao, "Probabilistic Computation: Towards a Unified Measure of Complexity," *Proceedings of the 18th Symposium on Foundations of Computer Science*, IEEE Computer Society, Los Alamitos, 1977, pp. 222–227.

[Ze13] E. Zermelo, "Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels," *Proceedings of the Fifth International Congress of Mathematicians II*, E. W. Hobson and A. E. H. Love, editors, Cambridge University Press, Cambridge, 1913.