The Bulletin of Symbolic Logic
Volume 2, Number 3, Sept. 1996

LINEAR LOGIC PROOF GAMES AND OPTIMIZATION

PATRICK D. LINCOLN, JOHN C. MITCHELL AND ANDRE SCEDROV

§1. Introduction. Perhaps the most surprising recent development in complexity theory is the discovery that the class NP can be characterized using a form of randomized proof checker that only examines a constant number of bits of the "proof" that a string is in a language [6, 5, 31, 3, 4]. More specifically, writing $|x|$ for the length of a string x, a language L in the class NP of languages recognizable in Nondeterministic Polynomial time is traditionally given by a polynomial p and a polynomial-time predicate P such that a string x is in L iff there is some string y satisfying $P(x, y)$, where $|y| \leq p(|x|)$. Intuitively, we can think of a string y as a possible proof that $x \in L$, with the predicate P some kind of proof checker that distinguishes good proofs from bad ones. A string x is therefore in a language $L \in$ NP if there is a short proof that $x \in L$, and not in L otherwise. The surprising discovery is that the deterministic proof checker that reads the entire input and proof can be replaced by a probabilistic verifier that on input x and possible proof y, where y is presented in a certain way, flips at most $O(\log |x|)$ coins and reads at most a constant number of bits of x and y. Obviously, a probabilistic verifier that does not read the whole proof will sometimes make mistakes. However, the surprising and essentially non-intuitive mathematical fact is that for each language L in NP, it is possible to find a polynomial q and verifier V flipping a logarithmic number of coins and reading a constant number of bits such that, for any $x \in L$, there exists y with $|y| \leq q(|x|)$ and with $V(x, y)$ accepting with probability 1 and, for $x \notin L$, there is no y with $|y| \leq q(|x|)$ and with $V(x, y)$ accepting with probability $\geq 1/4$. This idea can also be extended to PSPACE [10, 9], using a game that alternates between two players instead of a proof presented by a "single player."

Since the construction of reliable probabilistic verifiers is not very intuitive, it might help to consider some analogies. One is a parallel between probabilistic verifiers and a judge on a thesis review panel. In the traditional

Received January 31, 1996, and in revised form, May 25, 1996.

During the preparation of this paper, Patrick Lincoln was supported by NSF Grant CCR-9224858 and ONR Grant N00014-95-C-0168; John Mitchell was partially supported by NSF; and Andre Scedrov was partially supported by NSF Grant CCR-94-00907, by ONR Grant N00014-92-J-1916, and by a Centennial Research Fellowship from the American Mathematical Society.

1079-8986/96/0203-0003/$2.70

presentation of NP, a thesis is reviewed by a very careful judge who listens to every word and tries to find a flaw, rejecting if one flaw is found. In the probabilistic version, a thesis is reviewed by a busy and distracted, but intelligent professor who has several other things on her mind. Instead of listening to every word, her thoughts randomly drift away from the presentation and back. However, when she listens to the presentation, she analyzes the argument carefully. The theorem says, intuitively, that it is possible for such a person to distinguish reliably good theses from bad ones. An important aspect of this claim is that in replacing a thorough, attentive judge with a probabilistic one, we can ask the student to present the thesis according to any guidelines we choose. In particular, we can ask that parts of the thesis be repeated often, in various ways (perhaps like one of those interminable Faulkner passages). To draw another analogy, we could ask for a presentation that resembles a play of the game sometimes called "My father went to New York," which goes,

1. my father went to New York and he took a suitcase,
2. my father went to New York and he took a suitcase and a tie,
3. my father went to New York and he took a suitcase and a tie and a pair of socks,
4. my father went to New York and he took a suitcase and a tie and a pair of socks and one of those golf shirts with an alligator,

and so on.

The probabilistic verifier characterization of NP may be used to show that a number of optimization problems are hard to approximate in the sense that computing an approximation would also solve every known recognition problem in NP. For instance, the size of the largest clique in a graph is NP-hard to approximate up to any constant factor. See [4, 27] for an overview. Let us also mention recent work [18] that obtains several NP lower bounds for approximation problems by traditional complexity-theoretic techniques.

The probabilistic verifier characterization of PSPACE is given in [10, 9]. A traditional characterization of PSPACE is that a language L is given by a polynomial p and a polynomial-time predicate P on strings, with a string x in L iff $\exists y_1 \forall y_2 \exists y_3 \dots \forall y_{p(|x|)} P(xy_1y_2y_3 \dots y_{p(|x|)})$, where each y_i may be taken to be a single bit $\{0, 1\}$ or, alternatively, a string of length polynomial in $|x|$. In [10], it is shown that one can replace the deterministic polynomial-time predicate P by a probabilistic one that flips a logarithmic number of coins and reads a constant number of bits of the concatenation of strings $xy_1y_2y_3 \dots y_{p(|x|)}$, when y_i's are presented in a certain way. This result is strengthened in [9] to a characterization that also allows the universally-quantified choices ($y_2, y_4, \dots$ above) to be chosen randomly. This is most easily understood using games. In the traditional characterization, PSPACE is the class of languages characterized by games that alternate between players $\exists$ (proponent) and $\forall$ (opponent), with the outcome determined by a polynomial-time verifier (or referee) that looks at all the moves and decides the winner. In the first probabilistic characterization, the play proceeds in the same way, but verifier (referee) flips coins to decide how to review the game, looks at a constant number of moves (not depending on the length of the game) and chooses a winner. In the second probabilistic characterization, one keeps this form of probabilistic verifier, but replaces the $\forall$ player (which may need to make clever moves in order to win) by a simple opponent ¶ who moves purely randomly. Formulas that begin with a string of $\exists$, ¶ quantifiers followed by a list of disjunctive clauses may be called *stochastic* formulas. Among the other possible variations one must mention "interactive proof systems" or IP [7, 16, 25, 30, 27, 26], which provided an important technical source for the more recent work mentioned above. A relevant result of [9], quoted as Theorem 5.3 in this paper, is that it is PSPACE-hard to approximate the expected number of clauses in a stochastic formula that player $\exists$ can satisfy when playing optimally against a random opponent ¶.

After several expository sections that discuss this material in more detail, in Section 6 we develop several variations of the linear logic proof game discussed in [22]. These games are played on linear logic formulas, and their moves are actually instances of inference rules of *linear logic* [14, 15, 21]. Linear logic is a natural choice in this regard because of its intrinsic ability to reflect computational resources [15, 28, 29].

In our context, there are two players, called proponent and opponent, and a separate verifier. Proponent's goal is to play a sequence of moves that constitute a formal proof of an input formula, consisting of axioms and matching inference rules. Opponent tries to force the direction of proponent's evidence in a way that makes it impossible for proponent to obtain a formal proof. Each version of the game has a numeric score that reflects the number of certain preferred axioms used in a complete or partial formal proof.

Our main results can be explained as representations of complexity-theoretic games in the linear logic proof game, with the new complexity results obtained as corollaries of the complexity properties of games from the literature, such as those mentioned above. More specifically, the game representations we consider in Section 7 are defined in a move-by-move fashion; that is, they preserve proponent's moves, opponent's moves, proponent's strategies, as well as proponent's optimal strategies (that is, optimal with respect to the score).

In this way, one transfers to the linear logic proof game the complexity lower bounds for the approximation of the expected score when proponent plays optimally. In the case of the PSPACE-complete multiplicative-additive fragment of propositional linear logic [21], it is shown in Section 7.1 that it is as hard to compute an approximation of the optimal score as it is to determine proponent's optimal strategy. One way to explain this intuitively and informally is that provability in linear logic is not only globally hard, but also locally hard. In other words, the linear logic proof game shares with many other intricate games such as chess the feature that choosing the best next move seems just as hard as developing a complete winning strategy.

Our results have analogs for the NP-complete extended multiplicative fragment MLL$\top$ where opponent is absent. This is discussed in [24].

§2. Stochastic quantified boolean formulas and games. We consider boolean matrices in conjunctive normal form. Prenex boolean formulas are defined as usual, but we allow the "random" quantifier ¶ in addition to $\exists$ and $\forall$. A formula is a *k-CNF* if every clause in its matrix has exactly k literals. For the purposes of establishing a succinct terminology, we say a formula is *classical* if it is closed and all of the quantifiers are $\forall$ or $\exists$, *existential* if it is closed and all of the quantifiers are $\exists$, and *stochastic* if it is closed and all of the quantifiers are ¶ or $\exists$. It is possible to consider formulas that contain other combinations of quantifiers (such as $\forall$ and ¶), but we will not need these other classes of formulas.

A classical formula is either valid or invalid, according to the usual interpretation. One way of understanding the value of a classical formula that will be useful in comparing classical and stochastic formulas is through a very simple game with two players called "$\forall$" and "$\exists$". For a formula $Q_0x_0Q_1x_1\dots Q_nx_nM$, the play follows the quantifier order $Q_0, Q_1, \dots, Q_n$ from left to right, with each player selecting a truth value (*true* or *false,* or, equivalently, 1 or 0) for each variable identified with that player. Informally, the goal of player "$\exists$" is to choose values for the existentially-quantified variables so that the matrix is true. Player "$\forall$" tries to do the opposite, choosing values for the universally-quantified variables that will make the formula false.

It is easy to see that, for any classical formula, if both players continue until the quantifier prefix is exhausted, player "$\exists$" has the ability to win against any possible "$\forall$" opponent precisely if the formula is valid. When player "$\exists$" has a way of winning, regardless of how player "$\forall$" plays, we say player "$\exists$" has a winning strategy, and similarly for player "$\forall$." More formally, a *strategy* is a function from positions (which may be represented by the sequence of moves made so far in the game) to moves. A strategy is a *winning strategy* for a given player if this player is guaranteed a win by following the strategy. Using this standard terminology from game theory, we say that a classical formula is valid iff player "$\exists$" has a winning strategy. We can think of the player "$\exists$" as "proponent" and the player "$\forall$" as "opponent."

For stochastic formulas, we associate a probability PROB-STOC(ϕ) with each formula ϕ. One way of explaining this probability is using a variant of the classical formula game described above, this time between players called "¶" (opponent) and "∃" (proponent). The game is played in essentially the same way as the classical game, except for the way that player "¶"chooses the values of variables. Specifically, the play follows the quantifier order from left to right, with each player selecting a truth value for each variable with the appropriate quantifier. The value of a formula is computed by associating a specific strategy with each player. The simpler of the two is player "¶", who chooses truth values at random. That is, the player "¶" assigns independently to each variable 1 or 0, each with probability $1/2$. Player "∃" chooses truth values so as to maximize the probability that the formula is satisfied. In other words

$$\text{PROB-STOC}(\phi) \overset{\text{def}}{=} \text{Probability that } \phi \text{ is satisfied if "}\exists\text{" plays optimally.}$$

This informal description may be made more precise by defining PROB-STOC(ϕ) by recursion on the length of the quantifier sequence. Specifically, suppose $Q_{i+1}x_{i+1} \dots Q_n x_n M$ is a formula with ¶ and ∃ quantifiers, and free boolean variables $x_0, \dots, x_i$. The value $\mathcal{V}(b_0, \dots, b_i \mid Q_{i+1}x_{i+1} \dots Q_n x_n M)$ of formula $Q_{i+1}x_{i+1} \dots Q_n x_n M$ given truth values $b_0, \dots, b_i$ is defined recursively as follows:

$$\begin{aligned}
\mathcal{V}(b_0, \dots, b_n \mid M) \quad &= \quad \begin{cases} 1 & \text{if } M \text{ is satisfied by } b_0, \dots, b_n \\ 0 & \text{if } M \text{ is not satisfied by } b_0, \dots, b_n \end{cases} \\
\mathcal{V}(b_0, \dots, b_i \mid & \\
\P x_{i+1} \dots Q_n x_n M) \quad &= \quad 1/2(\mathcal{V}(b_0, \dots, b_i, 0 \mid Q_{i+2}x_{i+2} \dots Q_n x_n M) \\
&\qquad +\mathcal{V}(b_0, \dots, b_i, 1 \mid Q_{i+2}x_{i+2} \dots Q_n x_n M)) \\
\mathcal{V}(b_0, \dots, b_i \mid & \\
\exists x_{i+1} \dots Q_n x_n M) \quad &= \quad \max\{\mathcal{V}(b_0, \dots, b_i, 0 \mid Q_{i+2}x_{i+2} \dots Q_n x_n M), \\
&\qquad \mathcal{V}(b_0, \dots, b_i, 1 \mid Q_{i+2}x_{i+2} \dots Q_n x_n M)\}.
\end{aligned}$$

Another view of the value of a stochastic formula game is that this is the probability that player "∃" will win a game played on the classical formula obtained by replacing each ¶ by ∀, when ∀ follows a random strategy and ∃ plays optimally.

§3. Recognition and optimization problems. There are three important recognition problems associated with the classical, existential and stochastic boolean formulas defined above. In this extended abstract we mention only the latter case. This is given by the "language"

$$\text{STOC} \overset{\text{def}}{=} \{\, \phi \mid \phi \text{ is a stochastic boolean formula with PROB-STOC}(\phi) > 1/2 \,\}.$$

The language STOC is often called SSAT in the literature. (Both acronyms are meant to abbreviate "stochastic satisfiability.") A standard result of "exact-solution" complexity theory [27] establishes that STOC is polynomial-space (PSPACE) complete.

This language-recognition problem has an associated number function that one may attempt to approximate. The main idea is to generalize the interpretation of a boolean matrix so that one counts the number of satisfied clauses and then ask for the maximum possible count. More specifically, if one interprets a matrix as a function from $\{0,1\}^n$ to $\mathcal{N}$, the set of natural numbers, by counting the number of satisfied clauses, then one can see that some vectors in $\{0,1\}^n$ come closer to satisfying the full matrix than others.

A useful intuition is to think of a matrix $M = C_1 \wedge \cdots \wedge C_j$ with j clauses as stating a multiset of j conditions to be satisfied simultaneously. While it would be best to satisfy all conditions, this may not be possible. In this case, one would like to know how close one can come to this goal. In the associated game for stochastic formulas, player "$\exists$" (*i.e.*, proponent) tries to satisfy as many clauses as possible, but this time against an opponent that plays randomly. The number function MAX-STOC gives the expected number of clauses that "$\exists$" will be able to satisfy (when playing optimally), *i.e.*,

$$\textsc{max-stoc}(\phi) \stackrel{\text{def}}{=} \sum_k k \; \mathit{Prob} \begin{pmatrix} \text{exactly } k \text{ clauses of } M \text{ are satis-} \\ \text{fied, when "}\exists\text{" plays optimally} \end{pmatrix}.$$

This has a precise recursive definition as above, except that the base case is replaced by:

$$\mathcal{V}(b_0, \ldots, b_n \mid M) = \text{the number of clauses of } M \text{ satisfied by } b_0, \ldots, b_n.$$

Intuitively, MAX-STOC is the expected score for "$\exists$" when "$\exists$" plays optimally against a random adversary. Note that this use of boolean formulas does *not* respect logical equivalence: It is easy to find logically equivalent purely existential formulas ϕ and ϕ' with $\textsc{max-stoc}(\phi) \neq \textsc{max-stoc}(\phi')$.

The games just described and their associated optimal score functions readily extend to a more general setting where the conditions to be satisfied simultaneously are boolean matrices rather than disjunctive clauses. That is, one counts the number of simultaneously satisfiable boolean matrices from a given multiset $\{M_1, \ldots, M_j\}$. The game is played on *generalized prenex formulas* $Qx_0Qx_1 \ldots Qx_n\{M_1, \ldots, M_j\}$. The terminology regarding formulas is also used for generalized formulas. Also, for instance, the base case of the two previous recursive definitions is replaced by

$$\begin{aligned} &\mathcal{V}(b_0, \ldots, b_n \mid \{M_1, \ldots, M_j\}) = \\ &\qquad \text{the number of matrices in } \{M_1, \ldots, M_j\} \text{ satisfied by } b_0, \ldots, b_n. \end{aligned}$$

The game played on formulas is a special case when the boolean matrices $M_1, \ldots, M_j$ are disjunctive clauses.

§**4. Approximate solutions to optimization problems.** Intuitively, we approximate a function f by computing a function in some "neighborhood" of f. Let $\mathcal{D} \subseteq \{0,1\}^*$, and let $\mathcal{R}$ be the set of real numbers. For a function $f\colon \mathcal{D} \to \mathcal{R}$, the neighborhood of f consists of all functions $g\colon \mathcal{D} \to \mathcal{R}$ such that, for every string $x \in \mathcal{D}$, the difference between $f(x)$ and $g(x)$ is relatively small. One measure of error that appears in the literature [12, 6, 5] is

DEFINITION 4.1. Let $\mathcal{D} \subseteq \{0,1\}^*$, and let f be a real-valued function on $\mathcal{D}$. Let $0 < \varepsilon < 1$, where ε may depend on $|x|$, the length of a string $x \in \mathcal{D}$. The ε-neighborhood of f is the set

$$\varepsilon\text{-}nbhd(f) = \left\{ g\colon \mathcal{D} \to \mathcal{R} \mid \forall x \in \mathcal{D}, \quad \varepsilon(|x|) \leq \frac{g(x)}{f(x)} \leq \frac{1}{\varepsilon(|x|)} \right\}.$$

Intuitively, this is approximation up to factor $1/\varepsilon$. Also this is basically approximation with relative error at most $1 - \varepsilon$.

§**5. Lower bounds for approximation problems.** The approximation problems for PROB-STOC and MAX-STOC are *hard,* in the technical sense that computing a close approximation would also solve every known recognition problem in an established complexity class, in this case PSPACE. However, the precise form of lower bound that can be proved using current techniques differs from one problem to the next. Let us give a short intuitive explanation of one approach.

Standard complexity classes are formulated using languages, or recognition problems. Therefore, one natural and appealing form of lower bound for approximation is to show that if one can compute a function efficiently in an ε-neighborhood of f, then one can efficiently decide any language in NP, for example. Since there is a substantial body of literature classifying problems in NP (such as [13]), and historical attempts to develop efficient algorithms for the NP-complete problems have not been successful, this would give strong evidence that no polynomial-time algorithm exists for computing a function within an ε-neighborhood of f. A similar argument for PSPACE would suggest that no such approximation could be computed in less than polynomial space. Little was known about such lower bounds for approximations until recently. However, recently there has been substantial progress in this area. We refer the reader to the monographs [31, 3] and to the survey paper [4] for NP lower bounds.

In general, a reduction from a language recognition problem to an approximation must translate a yes/no question about a language to a yes/no question about functions.

DEFINITION 5.1. Let $\mathcal{D} \subseteq \{0,1\}^*$, and let g be a rational-valued function on $\mathcal{D}$. The function g is PSPACE-hard if, for any language $L \in$ PSPACE and its associated characteristic function χ_L, there exist polynomial-time computable functions h_1, h_2 such that $\chi_L = h_2 \circ g \circ h_1$.

DEFINITION 5.2. Let $\mathcal{D} \subseteq \{0,1\}^*$, and let f be a real-valued function on $\mathcal{D}$. Let ε be a real-valued function on the natural numbers such that $0 < \varepsilon(n) < 1$, for all n. The function f is PSPACE-hard to approximate within multiplicative ratio ε if every rational-valued function $g \in \varepsilon\text{-}nbhd(f)$ is PSPACE-hard.

One way to establish the required reductions[1] for language L and function f is to translate any string x to a function argument $s(x)$ such that

$$\begin{array}{lll} x \in L & \text{implies} & f(s(x)) \geq d_2(x), \\ x \notin L & \text{implies} & f(s(x)) \leq d_1(x), \end{array}$$

where $d_1(x) < d_2(x)$ are sufficiently far apart so that, for any $g \in \varepsilon\text{-}nbhd(f)$, one can decide membership in L by computing $g(s(x))$.

The following theorem is proved in [9].

THEOREM 5.3 ([9]). *There exists a positive real constant $\varepsilon < 1$ such that the function* MAX-STOC *on 3-CNF stochastic formulas is* PSPACE*-hard to approximate within multiplicative ratio ε.*

That is, computing any function h in the ε-neighborhood of MAX-STOC would allow us to decide membership in any language L in PSPACE, using time and space at most a polynomial greater than the time and space needed to compute h. A similar theorem regarding the clause-counting game on classical formulas is obtained in [10]. In [9] it is also observed that

THEOREM 5.4 ([9]). *There exists a positive constant c such that* PROB-STOC *on stochastic formulas is* PSPACE*-hard to approximate within multiplicative ratio 2^{-n^c}, where n is the length of a stochastic formula.*

Using the latter theorem and the techniques of [18], it can be shown that

[1] PSPACE-hardness of approximation was defined in [9] and [10] in terms of Turing-reducibility, but their results as well as ours satisfy the more restrictive definition in terms of many-one reducibility used here. It is possible that some approximation problems exist that are PSPACE-hard with respect to Turing reductions, but not many-one reducibility. For such problems our definition would be too restrictive, but for all the problems we have studied our definition is more convenient.

THEOREM 5.5. *There exists a positive real number c such that the function* MAX-STOC *on generalized stochastic formulas is* PSPACE-*hard to approximate within multiplicative ratio n^{-c}, where n is the length of the generalized stochastic formula.*

§6. Linear logic proof games. Linear logic, introduced in [14], is a refinement of classical logic. While the central notions of *truth* (emphasized in classical logic) and *proof construction* (emphasized in intuitionistic logic) remain important in linear logic, it might be said that the emphasis in linear logic is on *state*. Linear logic is sometimes described as being *resource sensitive* because it provides an intrinsic and natural accounting of process states, events, and resources [15, 28, 29].

A connection between linear logic and randomized interaction was investigated in [22], which introduced a game semantics of a fragment of linear logic by means of probabilistic games. Other notions of game semantics for linear logic were considered in [8, 1, 2, 17, 20, 19, 11]. Another direction is emphasized here: linear logic proof game simulations of probabilistic games from complexity theory. Such simulations are then used to show that certain problems in linear logic that are known to be hard to decide are also hard to approximate.

We shall consider so-called multiplicative-additive fragment of propositional linear logic, MALL. Its proof rules are presented below. This fragment is PSPACE-complete [21]. Although this fragment does not contain linear logic modalities (or: exponentials), some of the other aspects of linear logic may be seen already in MALL. For instance, two standard "structural" rules contraction and weakening are removed, which produces a *linear* system in which each assumption must be used *exactly once* in a proof. In this system, formulas indicate finite resources that cannot necessarily be discarded or duplicated without effort. Another feature of linear logic is the distinction between two natural forms of conjunction and disjunction, one form called "multiplicative," and the other "additive." For instance, considering the hypotheses as resources, a proof of a multiplicative conjunction $\otimes$ as a conclusion must partition the resources used to establish each conjunct, whereas the additive conjunction $\&$ requires the sharing of all of the resources. The distinction between the multiplicative and the additive connectives is reflected in the MALL inference rules, given below.

Let p, q be propositional atoms, let A, B be MALL formulas, let $\Gamma, \Delta, \Theta, \Xi$ be finite multisets of MALL formulas, and let Σ be a finite multiset of literals or constants $1, 0$. We write $\Delta \uplus \Theta$ for the (disjoint) multiset union of Δ and Θ. As usual, we write Γ, A for the multiset obtained by adding an instance of A to Γ. An expression of the form $\vdash \Gamma$ is called a *sequent*. An expression of the form $\vdash \Sigma$ is called a *primitive sequent*.

The English names for MALL inference rules are: *identity*, *cut*, *par*, *tensor*, *bottom*, *one*, *plus*, *with*, and *top*. $\otimes$ and $\parr$ are *multiplicative* connectives; 1 and $\perp$ are *multiplicative* propositional constants. $\oplus$ and & are *additive* connectives; 0 and $\top$ are *additive* propositional constants. There is no rule for 0. *Linear negation* $^\perp$, mentioned in the identity and cut rules, is defined by recursion on the structure of formulas: $(p^\perp)^\perp$ is p, $(A \otimes B)^\perp$ is $A^\perp \parr B^\perp$, $(A \parr B)^\perp$ is $A^\perp \otimes B^\perp$, $1^\perp$ is $\perp$, $\perp^\perp$ is 1, $(A \,\&\, B)^\perp$ is $A^\perp \oplus B^\perp$, $(A \oplus B)^\perp$ is $A^\perp \,\&\, B^\perp$, $\top^\perp$ is 0, and $0^\perp$ is $\top$.

MALL proof rules are

$$
\begin{array}{cccc}
\mathbf{I} & \dfrac{}{\vdash p, p^\perp} & \dfrac{\vdash A, \Gamma \qquad \vdash A^\perp, \Delta}{\vdash \Gamma, \Delta} & \mathbf{Cut} \\[2ex]
\boldsymbol{\parr} & \dfrac{\vdash A, B, \Gamma}{\vdash (A \parr B), \Gamma} & \dfrac{\vdash A, \Gamma \qquad \vdash B, \Delta}{\vdash (A \otimes B), \Gamma, \Delta} & \otimes \\[2ex]
\perp & \dfrac{\vdash \Gamma}{\vdash \perp, \Gamma} & \dfrac{}{\vdash 1} & 1 \\[2ex]
\oplus\mathbf{1} & \dfrac{\vdash A, \Gamma}{\vdash (A \oplus B), \Gamma} & \dfrac{\vdash A, \Gamma \qquad \vdash B, \Gamma}{\vdash (A \,\&\, B), \Gamma} & \mathbf{\&} \\[2ex]
\oplus\mathbf{2} & \dfrac{\vdash B, \Gamma}{\vdash (A \oplus B), \Gamma} & \dfrac{}{\vdash \top, \Gamma} & \top
\end{array}
$$

MALL enjoys the *cut-elimination property* and the *subformula property* [14, 21]. In particular, if a MALL formula is provable, then it is provable without the use of the cut rule, and the required proof rules involve only subformulas of the given formula.

Let us describe several variations of the proof game discussed in [22], all involving the same moves. There are two players, called proponent and opponent, and a separate verifier. Proponent's goal is to play a number of moves demonstrating or giving evidence for a sequent. In order to do this, proponent plays proof rule instances. Opponent tries to force the direction of proponent's evidence in a way that makes it impossible for proponent to win. Opponent plays special markers that may block one side of proponent's & moves. If proponent plays a $\otimes$ move, then opponent does not block either of the premises.

Verifier scores completed plays of the game. Various forms of the game differ in the way they are scored. The main objective of proponent is to never allow opponent to succeed. However, in some forms of the game proponent will be more ambitious, that is, in addition to the main requirement, proponent will try to achieve the best score possible.

Let us first consider a simple version of the game against a randomized opponent, which can be described as an avg/max game played on MALL sequents. Proponent chooses the inference rule to be applied. In the case $\otimes$, proponent chooses a partition and requires both associated expressions to be evaluated. In the case $\oplus$, proponent chooses which of the two expressions will be evaluated. In the case &, opponent chooses by a fair coin toss which of the two expressions will be evaluated. In the case of a primitive sequent, verifier simply computes the value. Each sequent containing the constant $\top$, each identity axiom, and each primitive sequent containing only the constant 1 is scored 1 by verifier. All other primitive sequents are scored 0. Each completed play of the game is scored as the *minimum* of the scores of terminal sequents obtained in the play. Note that the number of moves is finite; indeed, it is polynomial in the size of a given MALL sequent. Proponent wins when each encountered primitive sequent is an identity axiom or the constant 1.

Let us define the function μ, which represents the expected score when proponent plays optimally.

$$\mu(\Gamma) = \max\{\mu(\Gamma'; A) \mid \Gamma = \Gamma', A\},$$
$$\mu(\Gamma; A \otimes B) = \max\{\min\{\mu(\Delta, A), \mu(\Theta, B)\} \mid \Delta \uplus \Theta = \Gamma\},$$
$$\mu(\Gamma; A \parr B) = \mu(\Gamma, A, B),$$
$$\mu(\Gamma; A \oplus B) = \max\{\mu(\Gamma, A), \mu(\Gamma, B)\},$$
$$\mu(\Gamma; A \& B) = \frac{1}{2}[\mu(\Gamma, A) + \mu(\Gamma, B)],$$
$$\mu(\Gamma; \bot) = \mu(\Gamma),$$
$$\mu(\Gamma; \top) = 1,$$
$$\mu(\Sigma) = \begin{cases} 1 & \text{if } \Sigma \text{ is 1 or an axiom,} \\ 0 & \text{otherwise.} \end{cases}$$

Let us emphasize that, for any MALL sequent $\vdash \Xi$, the value $\mu(\Xi)$ is the maximum possible value satisfying these recursive conditions. Specifically, if any encountered sequent contains composite formulas, then several clauses regarding $\mu(\Gamma; A)$ might be applicable.

It may be seen that a MALL sequent $\vdash \Xi$ is provable iff $\mu(\Xi) = 1$. Furthermore, if $\vdash \Xi$ is unprovable and does not contain &, then $\mu(\Xi) = 0$. However, $\mu(\Xi)$ may be arbitrarily close to 1 if $\vdash \Xi$ is unprovable and contains &.

The more involved version of the linear logic proof game against a randomized opponent may also be presented as an avg/max game. The players' moves and the winning condition are the same as in the simple game just described. However, in this version of the game, proponent also attempts to use as many certain preferred axioms as possible. Preferred axioms are, say, instances of a distinguished axiom $\vdash d, d^{\perp}$, where the propositional atom d is fixed in advance. In this version, proponent gets one point for each instance of the distinguished axiom $\vdash d, d^{\perp}$ encountered in a play, but no points are awarded if a primitive sequent $\vdash \Sigma$ is any other identity axiom $\vdash p, p^{\perp}$, or contains $\top$, or consists of a single constant $\vdash 1$. If Σ is any other multiset of literals or constants, that is, a possibly empty multiset of literals or constants $1, 0$ other than an identity axiom or the single constant 1, then proponent receives a penalty of -2^n points, where n is length of the original sequent. Each completed play of the game is now scored as the *sum* of the scores of primitive sequents obtained at the end of the play. Note that proponent wins iff this sum is ≥ 0.

Let us define the function ρ, which represents the expected score for proponent when proponent plays optimally.

$$\rho(\Gamma) = \rho_n(\Gamma), \text{ where } n = |\Gamma|,$$

$$\rho_n(\Gamma) = \max\{\rho_n(\Gamma'; A) \mid \Gamma = \Gamma', A\},$$

$$\rho_n(\Gamma; A \otimes B) = \max\{\rho_n(\Delta, A) + \rho_n(\Theta, B) \mid \Delta \uplus \Theta = \Gamma\},$$

$$\rho_n(\Gamma; A \parr B) = \rho_n(\Gamma, A, B),$$

$$\rho_n(\Gamma; A \oplus B) = \max\{\rho_n(\Gamma, A), \rho_n(\Gamma, B)\},$$

$$\rho_n(\Gamma; A \& B) = \frac{1}{2}[\rho_n(\Gamma, A) + \rho_n(\Gamma, B)],$$

$$\rho_n(\Gamma; \perp) = \rho_n(\Gamma),$$

$$\rho_n(\Gamma; \top) = 0,$$

$$\rho_n(\Sigma) = \begin{cases} 1 & \text{if } \Sigma \text{ is the distinguished axiom } d, d^{\perp}, \\ 0 & \text{if } \Sigma \text{ is another axiom or } 1, \\ -2^n & \text{otherwise.} \end{cases}$$

Observe again that, for any MALL sequent $\vdash \Xi$, the value $\rho(\Xi)$ is the maximum possible value satisfying these recursive conditions, now with n fixed as the number of symbols in Ξ.

It may be seen that a MALL sequent $\vdash \Xi$ is provable iff $\rho(\Xi) \geq 0$. Furthermore, if $\vdash \Xi$ is unprovable and does not contain the distinguished propositional atom d, then $\rho(\Xi) \leq -1$.

Game score functions considered here are intrinsic to the proof system MALL. In particular, they are invariant with respect to certain *permutability* properties, important "structural" properties of MALL. That is, our game score functions are invariant with respect to invertible inference rules of MALL. The following theorem is proved by induction on the length of Γ.

THEOREM 6.1. *The following equalities hold, with n the length of the longest sequent in each equality*:

- $\rho_n(\Gamma, A \parr B) = \rho_n(\Gamma, A, B)$,
- $\rho_n(\Gamma, A\&B) = \frac{1}{2}(\rho_n(\Gamma, A) + \rho_n(\Gamma, B))$,

- *If* Γ *does not contain* &, *then* $\rho_n(\Gamma, A \oplus B) = \max\{\rho_n(\Gamma, A), \rho_n(\Gamma, B)\}$, *and similarly for* μ.

Let us also observe that many isomorphisms of linear logic are respected by the score functions we consider, *e.g.,* $A \parr (B \parr C) \cong (A \parr B) \parr C$, $A \parr (B \& C) \cong (A \parr B) \& (A \parr C)$, $A \parr \bot \cong A$, $A \parr B \cong B \parr A$, $A \otimes B \cong B \otimes A$, $A \otimes 1 \cong A$, $A \otimes (B \otimes C) \cong (A \otimes B) \otimes C$, etc. Notable exceptions are $A \& \top \cong A$ and $A \& (B \& C) \cong (A \& B) \& C$.

§7. **Representing games in the linear logic proof game.** In this section, stochastic (generalized) formula games discussed in Section 2 will be represented by linear logic games introduced in Section 6. The simple game on formulas will be represented by the simple version of the linear logic proof game. The clause-counting games will be represented by means of the advanced version of the linear logic proof game. Of course, the representations themselves will be computable in polynomial time.

In the instances we consider, a natural way to define game representations is to emphasize optimal strategies, that is, optimal with respect to score, rather than winning strategies:

DEFINITION 7.1. A game representation is an assignment that maps
- inputs to inputs,
- proponent's moves to sequences of proponent's moves,
- opponent's moves to sequences of moves containing exactly one opponent's move,
- proponent's strategies to proponent's strategies, and
- proponent's optimal strategies to proponent's optimal strategies.

In the instances we consider, these conditions imply that the game score is preserved.

The reader will observe that the proof of the PSPACE-hardness of MALL in [21] in fact establishes a polynomial-time representation of the simple evaluation game on classical formulas (discussed here at the beginning of Section 2) by means of the simple linear logic proof game with a *universal* opponent, who may make clever choices in blocking one of the premises of a & rule instance.

Let us summarize the main results.

THEOREM 7.2. *There exists a polynomial-time representation of the* PROB-STOC *game by the simple linear logic proof game with the randomized opponent. In particular, the associated polynomial-time encoding* $[\![\cdot]\!]$ *of stochastic formulas by* MALL *sequents satisfies the condition* PROB-STOC$(\phi) = \mu([\![\phi]\!])$ *for all closed* ϕ.

THEOREM 7.3. *There exists a polynomial-time representation of the* MAX-STOC *game by the advanced linear logic proof game with the randomized opponent. In particular, the associated polynomial-time encoding* $[\![\cdot]\!]$ *of generalized stochastic formulas by* MALL *sequents satisfies the condition* MAX-STOC$(\phi) = \rho([\![\phi]\!])$ *for all closed* ϕ.

7.1. Lower bounds for linear logic proof game. The existence of the game representations just discussed implies that, for each version of the linear logic game defined in Section 6, approximating the optimal score within a certain multiplicative ratio is PSPACE-hard.

Regarding the simple linear logic game against a stochastic opponent, Theorem 7.2 and the result from [9] cited here as Theorem 5.4 yield the following lower bound.

THEOREM 7.4. *There exists a positive real number* c *such that the function* μ *is* PSPACE-*hard to approximate within the multiplicative ratio* 2^{-n^c}, *where* n *is the length of a sequent.*

In other words, computing any function g that, for every MALL formula F, satisfies

$$2^{-|F|^c} \leq \frac{g(F)}{\mu(F)} \leq 2^{|F|^c}$$

would amount to the ability to recognize every PSPACE language. As an illustration of the meaning of the condition, note that even a very large constant g fails to satisfy the condition, since the values of μ can be very close to zero. On the other hand, the theorem implies that the function μ is PSPACE-hard to approximate up to any constant factor.

Note that the similar result in the case of the simple linear logic game against a universal opponent is a trivial consequence of the PSPACE-completeness of MALL known from [21]. Indeed, the score v when proponent plays optimally is discrete ($v(\Xi) = 1$ if Ξ is provable, else $v(\Xi) = 0$), and thus approximating v within a multiplicative ratio amounts to computing v exactly.

Let us close by considering lower bounds for the advanced version of the linear logic game, in which proponent's goal is to maximize the number of occurrences of the distinguished identity axiom in a cut-free MALL proof.

THEOREM 7.5. *The function* ρ *is* PSPACE-*hard to approximate within any positive multiplicative ratio on provable* MALL *sequents.*

This theorem may be obtained directly from the PSPACE-completeness of MALL known from [21]. However, the set of provable MALL sequents on which the function ρ is PSPACE-hard to approximate may be further restricted in several ways by using Theorems 5.5 and 7.3, and the result from [9] cited here as Theorem 5.3. Let us say that a *constraint* is a MALL formula of the form $(A \otimes z_1) \parr (z_1^{\perp} \otimes z_2) \dots \parr (z_n^{\perp} \otimes \star)$, where & does not occur in A, each z_i is a literal, and where $\star$ abbreviates the formula $d \parr d^{\perp}$ related to the distinguished axiom $\vdash d, d^{\perp}$. Note that the *size* of a constraint includes the size of the formula A. Let us say that a sequent is *constrained* if the distinguished formula $\star$ occurs only in constraints. Note that provability of constraints is in NP, in contrast with the general MALL provability, which is PSPACE-complete.

THEOREM 7.6. *There exists a positive real number c such that the function ρ is* PSPACE-*hard to approximate within multiplicative ratio n^{-c} on provable constrained* MALL *sequents, where n is the length of the sequent. Furthermore, there exists a positive real number $\varepsilon < 1$ such that the function ρ is* PSPACE-*hard to approximate within multiplicative ratio ε on provable constrained* MALL *sequents where every constraint is of constant size.*

On the other hand, proof-theoretic properties of linear logic allow us to restrict provable MALL sequents to those that contain only formulas of depth at most 2.

Theorems 7.3, 7.5, and 7.6 have analogs in the case in which the game is played against a universal opponent, but that case must be omitted here. The reader is referred to [23]. Our results also have implications for proof search heuristics in linear logic. This is discussed in [24].

Acknowledgments. We would like to thank Madhu Sudan for directing us to [10, 9], and Sanjeev Arora and Joan Feigenbaum for their informative conversations and email correspondence and for their help with the literature. Patrick Lincoln would like to thank the University of Pennsylvania for hosting him during his visit in the spring of 1996. Scedrov would like to thank Stanford University and SRI International for hosting him during his visit in the spring of 1995. All the authors would like to thank the Isaac Newton Institute for Mathematical Sciences in Cambridge, England for hosting them at various times in the fall of 1995.

REFERENCES

[1] S. ABRAMSKY and R. JAGADEESAN, *Games and full completeness for multiplicative linear logic*, this JOURNAL, vol. 59 (1994), pp. 543–574.

[2] S. ABRAMSKY, R. JAGADEESAN, and P. MALACARIA, *Full abstraction for PCF (extended abstract)*, ***Proceedings of the TACS '94***, Springer LNCS, vol. 789, 1994, pp. 1–15.

[3] S. ARORA, ***Probabilistic checking of proofs and hardness of approximation problems***, Technical Report CS-TR-476-94, Department of Computer Science, Princeton University, 1994, revised version of a Ph.D. Dissertation, University of California, Berkeley, 1994.

[4] S. ARORA and C. LUND, *Hardness of approximations*, ***Approximation algorithms for NP-hard problems***, PWS Publishing, 1996.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and hardness of approximation problems*, ***Proceedings of the 33-rd Annual IEEE Symposium on Foundations of Computer Science***, IEEE Computer Society Press, Los Alamitos, California, 1992, pp. 14–23.

[6] S. Arora and S. Safra, *Probabilistic checking of proofs; a new characterization of NP*, ***Proceedings of the 33-rd Annual IEEE Symposium on Foundations of Computer Science***, IEEE Computer Society Press, Los Alamitos, California, 1992, pp. 2–13.

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

[8] A. Blass, *A game semantics for linear logic*, ***Annals of Pure and Applied Logic***, vol. 56 (1992), pp. 183–220.

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

[10] ———, *Probabilistically checkable debate systems and nonapproximability of PSPACE-hard functions*, ***Chicago Journal of Theoretical Computer Science*** (1995), no. 4, http://www.cs.chicago.edu/publications/cjtcs/articles/1995/4/contents.html, extended abstract in ***Proceedings of the 25-th ACM Symposium on Theory of Computing***, ACM, New York, 1993, pp. 305–314.

[11] V. Danos, H. Herbelin, and L. Regnier, *Game semantics and abstract machines*, ***Proceedings of the 11-th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, NJ***, July 1996.

[12] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, *Approximating clique is almost NP-complete*, ***Proceedings of the 32-nd Annual IEEE Symposium on Foundations of Computer Science*** (Los Alamitos, California), IEEE Computer Society Press, 1991, pp. 2–12.

[13] M. R. Garey and D. S. Johnson, ***Computers and intractability: A guide to the theory of NP-completeness***, W. H. Freeman and Co., 1979.

[14] J. Y. Girard, *Linear logic*, ***Theoretical Computer Science***, vol. 50 (1987), pp. 1–102.

[15] ———, *Linear logic: its syntax and semantics*, ***Advances in linear logic*** (J. Y. Girard, Y. Lafont, and L. Regnier, editors), London Mathematical Society Lecture Note Series, vol. 222, 1995.

[16] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, ***SIAM Journal on Computing***, vol. 18 (1989), pp. 186–208.

[17] J. M. E. Hyland and C. H. L. Ong, *Pi-calculus, dialogue games and PCF*, ***Proceedings of the 7th ACM Conference on Functional Programming Languages and Computer Architecture***, ACM Press, 1995, pp. 96–107.

[18] H. B. Hunt III, M. V. Marathe, and R. E. Stearns, *Generalized CNF satisfiability problems and non-efficient approximability*, ***Proceedings of the 9-th Annual IEEE Conference on Structure in Complexity Theory***, IEEE Computer Society Press, Los Alamitos, California, 1994, pp. 356–366.

[19] A. Joyal, *Free lattices, communication, and money games*, to appear in ***Proceedings of the 10-th international congress of logic, methodology, and philosophy of science, Firenze***.

[20] F. Lamarche, *Games semantics for full propositional linear logic*, ***Proceedings of the 10-th Annual IEEE Symposium on Logic in Computer Science, San Diego, California***, June 1995, pp. 464–473.

[21] P. D. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar, *Decision problems for propositional linear logic*, ***Annals of Pure and Applied Logic***, vol. 56 (1992), pp. 239–311,

extended abstract in ***Proceedings of the 31-st Annual IEEE Symposium on Foundations of Computer Science***, 1990, pp. 662–671.

[22] P. D. Lincoln, J. C. Mitchell, and A. Scedrov, *Stochastic interaction and linear logic*, ***Advances in linear logic*** (J. Y Girard, Y. Lafont, and L. Regnier, editors), London Mathematical Society Lecture Notes, vol. 222, Cambridge University Press, 1995, pp. 147–166.

[23] ———, *Linear logic proof games and optimization*, Preliminary report, January 1996, available by anonymous ftp from ftp.cis.upenn.edu as pub/papers/scedrov/approx.[dvi,ps].Z.

[24] ———, *The complexity of local proof search in linear logic. extended abstract*, ***Proceedings linear logic '96, Tokyo meeting***, Electronic Notes in Theoretical Computer Science, 1996, to appear, p. http://www.elsevier.nl:80/mcs/tcs/pc/Menu.html.

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

[26] R. Motwani and P. Raghavan, ***Randomized algorithms***, Cambridge University Press, 1995.

[27] C. Papadimitriou, ***Computational complexity***, Addison-Wesley, 1994.

[28] A. Scedrov, *A brief guide to linear logic*, ***Current trends in theoretical computer science*** (G. Rozenberg and A. Salomaa, editors), World Scientific Publishing Co., 1993, pp. 377–394.

[29] ———, *Linear logic and computation: A survey*, ***Proof and computation, proceedings Marktoberdorf summer school 1993*** (H. Schwichtenberg, editor), vol. 139, NATO Advanced Science Institutes, Series F, Springer-Verlag, Berlin, 1995, pp. 379–395.

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

[31] M. Sudan, *Efficient checking of polynomials and proofs and the hardness of approximation problems*, ACM Distinguished Dissertation Series, Springer-Verlag, 1996, revised version of a Ph.D. Dissertation, University of California, Berkeley, 1993.

SRI INTERNATIONAL COMPUTER SCIENCE LABORATORY
MENLO PARK CA 94025, USA

E-mail: lincoln@csl.sri.com

DEPARTMENT OF COMPUTER SCIENCE, STANFORD UNIVERSITY
STANFORD, CA 94305-9045, USA

E-mail: mitchell@cs.stanford.edu
http://theory.stanford.edu/people/jcm/home.html

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF PENNSYLVANIA
PHILADELPHIA, PA 19104-6395, USA

E-mail: scedrov@cis.upenn.edu
http://www.cis.upenn.edu/˜scedrov