Rational Proofs with Multiple Provers*

Jing Chen Samuel McCauley Shikha Singh

Computer Science Department, Stony Brook University
Stony Brook, NY 11794
{*jingchen, smccauley, shiksingh*}@*cs.stonybrook.edu*

Abstract

Interactive proofs model a world where a verifier delegates computation to an untrustworthy prover, verifying the prover's claims before accepting them. These proofs have applications to delegation of computation, probabilistically checkable proofs, crowdsourcing, and more.

In some of these applications, the verifier may pay the prover based on the quality of his work. Rational proofs, introduced by Azar and Micali (2012), are an interactive proof model in which the prover is *rational* rather than untrustworthy—he may lie, but only to increase his payment. This allows the verifier to leverage the greed of the prover to obtain better protocols: while rational proofs are no more powerful than interactive proofs, the protocols are simpler and more efficient. Azar and Micali posed as an open problem whether multiple provers are more powerful than one for rational proofs.

We provide a model that extends rational proofs to allow multiple provers. In this model, a verifier can cross-check the answers received by asking several provers. The verifier can pay the provers according to the quality of their work, incentivizing them to provide correct information.

We analyze rational proofs with multiple provers from a complexity-theoretic point of view. We fully characterize this model by giving tight upper and lower bounds on its power. On the way, we resolve Azar and Micali's open problem in the affirmative, showing that multiple rational provers are strictly more powerful than one (under standard complexity-theoretic assumptions). We further show that the full power of rational proofs with multiple provers can be achieved using only two provers and five rounds of interaction. Finally, we consider more demanding models where the verifier wants the provers' payment to decrease significantly when they are lying, and fully characterize the power of the model when the payment gap must be noticeable (i.e., at least $1/p$ where p is a polynomial).

*The conference version of this paper has been accepted to appear in ITCS 2016.

1 Introduction

Multi-prover interactive proofs (MIP) [10] and rational interactive proofs (RIP) [5] are two important extensions of interactive proof systems.

In a multi-prover interactive proof, several computationally unbounded, potentially dishonest provers interact with a polynomial time, randomized verifier. The provers can pre-agree on a joint strategy to convince the verifier of the truth of a proposition. However, once the protocol starts, the provers cannot communicate with each other. If the proposition is true, the verifier should be convinced with probability 1; otherwise the verifier should reject with some non-negligible probability. As shown by Babai, Fortnow and Lund, $\mathsf{MIP} = \mathsf{NEXP}$ [8], which demonstrates the power of multiple provers compared to one-prover interactive proofs (recall that $\mathsf{IP} = \mathsf{PSPACE}$ [34, 37]).

Rational interactive proofs [5] are a variant of interactive proofs, where the verifier makes a payment to the prover at the end of the protocol. The prover is assumed to be *rational*, that is, he only acts in ways that maximize this payment. Thus, in contrast to interactive proofs, the prover does not care whether the verifier is convinced or not. Rational proofs ensure that the prover's payment is maximized if and only if the verifier learns the correct answer to the proposition. Azar and Micali [5] show that, while rational proofs are no more powerful than interactive proofs (i.e., $\mathsf{RIP} = \mathsf{PSPACE}$), the protocols are simpler and more efficient. Previous work on rational proofs [5, 6, 29] considers a *single* rational prover.

Many computation-outsourcing applications have ingredients of both of these models: the verifier pays a *team* of provers based on their responses. For example, in internet marketplaces such as *Amazon's Mechanical Turk* [1] and *Proof Market* [3], the requesters (verifiers) post labor-intensive tasks on the website along with a monetary compensation they are willing to pay. The providers (provers) accept these offers and perform the job. In these internet marketplaces and crowdsourcing games [39] correctness is often ensured by verifying one provider's answers against another [2, 38]. Thus, the providers collaborate as a team—their answers need to match, even though they are likely to not know each other and cannot communicate with each other [32].

Inspired by these applications and previous theoretical work, we introduce *multi-prover rational interactive proofs*, which combines elements of rational proofs and classical multi-prover interactive proofs. This model aims to answer the following question: what problems can be solved by a team of rational workers who cannot communicate with each other and get paid based on the joint-correctness of their answers? One of our main contributions is to completely characterize the power of this model.

Previous Discussions of Multiple Provers in Rational Proofs. The notion of rational proofs with multiple provers has appeared several times in previous work [5, 6, 29]. However, the authors only use multiple provers to simplify the analysis of single-prover protocols, without formalizing the model. They show that multiple provers in their protocols can be simulated by a single prover by scaling the payments appropriately. Azar and Micali [5] discuss one of the fundamental challenges of using multiple rational provers: in a cooperative setting, one prover may lie to give subsequent provers the opportunity to obtain a larger payment. They pose the following open problem: are multiple provers more powerful than one in rational proofs? In this paper, we show that, in general, a protocol with multiple rational provers cannot be simulated by a single-prover protocol under standard complexity-theoretic assumptions.

The Model of Multi-Prover Rational Proofs We briefly summarize our model in order to compare it with the literature and discuss our results. The model is formally defined in Section 2.

In a multi-prover rational interactive proof, several computationally-unbounded provers communicate with a polynomial-time randomized verifier who wants to determine the membership of an input string in a language. The provers can pre-agree on how they plan to respond to the verifier's queries. However, they cannot communicate with each other once the protocol begins. At the end of the protocol, the verifier outputs the answer and computes a total payment for the provers based on the input, his own randomness, and the messages exchanged. This total payment may be distributed in any pre-determined way by the verifier or the provers themselves.

A protocol is an MRIP protocol if any strategy of the provers that maximizes their expected payment leads the verifier to the correct answer. The class of languages having such protocols is denoted by MRIP.

Distribution of Payments. In classical MIP protocols, the provers work collaboratively to convince the verifier of the truth of a proposition and their goal is to maximize the verifier's acceptance probability. Similarly, the rational provers in MRIP work as a team to maximize the total payment received from the verifier. Any pre-specified distribution of this payment is allowed, as long as it does not depend on the transcript of the protocol (i.e., the messages exchanged, the coins flipped, and the amount of the payment). For instance, the division of the payment can be pre-determined by the provers themselves based on the amount of work each prover must perform, or it can be pre-determined by the verifier based on the reputation of each prover.[1] We ignore the choice of division in our model and protocols, as it does not affect the choices made by the provers in deciding their strategy.

1.1 Results and Discussions

We state our main results and discuss several interesting aspects of our model.

The Power of Multi-Prover Rational Proofs. We give tight upper and lower bounds on the power of MRIP protocols. As a warm up, we show that MRIP contains NEXP, using an MIP protocol as a black box and paying the provers appropriately. A similar technique shows that MRIP also contains coNEXP. In fact, an important property of MRIP is that it is closed under complement (Lemma 1). Thus, MRIP is strictly more powerful than RIP (assuming $\mathsf{PSPACE} \neq \mathsf{NEXP}$), resolving the question raised in [5]. Furthermore, MRIP is also more powerful than MIP (assuming $\mathsf{NEXP} \neq \mathsf{coNEXP}$), in contrast to the single-prover case in which the classical and rational proofs have the same power.

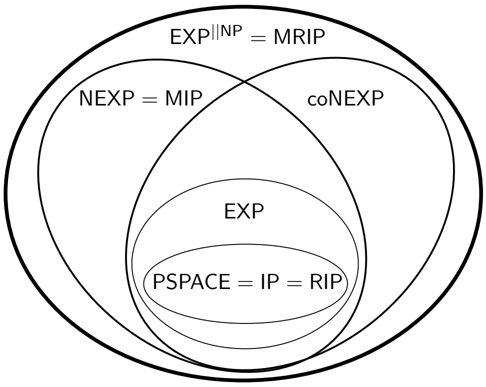

Figure 1: The relative power of rational and classical interactive proof systems. Note that it is widely believed that $\mathsf{PSPACE} \neq \mathsf{EXP}$, $\mathsf{EXP} \neq \mathsf{NEXP}$, and $\mathsf{NEXP} \neq \mathsf{coNEXP}$.

We exactly characterize the class MRIP by showing that a language has a multi-prover rational interactive proof if and only if it is decidable by a deterministic exponential-time oracle Turing machine with non-adaptive access to an NP oracle. That is,

Theorem 1. $\mathsf{MRIP} = \mathsf{EXP}^{||\mathsf{NP}}$.

To prove Theorem 1, we (a) provide a new characterization for $\mathsf{EXP}^{||\mathsf{NP}}$, and (b) decompose the circuit for $\mathsf{EXP}^{||\mathsf{NP}}$ obtained from this characterization into three stages, construct MRIP protocols for each stage and combine them together appropriately. A similar 3-stage decomposition was used in [6], but their technique results in an exponential blow-up in the number of messages when applied to multi-prover protocols.

It is known that any MIP protocol can be simulated using only two provers and one round of communication between the provers and the verifier [18]. Similarly, we show that only two provers and five rounds are sufficient to capture the full power of MRIP. That is, denoting by $\mathsf{MRIP}(p, r)$ the class of languages that have p-prover rational proofs with r rounds, we have:

Theorem 2. $\mathsf{MRIP} = \mathsf{MRIP}(2, 5)$.

[1] Note that unbalanced divisions are allowed: for example, it may be that one particular prover always receives half of the total payment, and the other provers split the remainder equally.

Note that we count the number of rounds as the total number of interactions (prover's messages and verifier's queries are separate rounds), in contrast to the number of *pairs* of back-and-forth interaction of MIP [18]. We use this convention to simplify our technical discussion.[2] It remains open whether or not the number of rounds in our protocol can be further reduced (see Section 4 for further discussion).

Utility Gaps. Rational proofs assume that the provers always act to maximize their payment. However, how much do they lose by lying? The notion of *utility gaps*, first introduced in [6], measures the loss in payment (or utility) incurred by a lying prover. Notice that a lying prover may (a) deviate (even slightly) from the truthful protocol but still lead the verifier to the correct answer or (b) deviate and mislead the verifier to an incorrect answer. The authors of [6] demanded their protocols to be robust against provers of type (a): that is, any deviation from the prescribed strategy that the verifier intends the provers to follow results in a significant decrease in the payment. This ideal requirement on utility gaps is quite strong, and even the protocol in [6] fails to satisfy it, as pointed out by Guo, Hubáček, Rosen and Vald [29]. In this paper, we consider multi-prover rational proofs robust against cheating provers of type (b): that is, the provers may respond to some messages of the verifier incorrectly and incur a small payment loss, but if the verifier learns the answer to the membership question of the input string incorrectly, the provers must suffer a significant loss in payment. Note that [29] also considers type (b) utility gap, but for single-prover protocols. Our utility gaps are formally defined in Section 5.

We show that requiring a noticeable utility gap results in protocols for a different, possibly smaller, complexity class. In particular, let $\mathsf{poly(n)\text{-}gap\text{-}MRIP}$ be the class of languages that have MRIP protocols with $1/\alpha(n)$ utility gap for lying provers, where $\alpha(n)$ is a polynomial in n. We completely characterize this class as the class of languages decidable by a polynomial-time oracle Turing machine with non-adaptive access to an NEXP oracle. That is,

Theorem 3. $\mathsf{poly(n)\text{-}gap\text{-}MRIP} = \mathsf{P}^{||\mathsf{NEXP}}$.

Simple and Efficient MRIP protocol for NEXP. Classical multi-prover interactive proofs for languages in NEXP rely on the MIP protocol for the NEXP-complete language $\mathsf{Oracle\text{-}3SAT}$ [8]. This MIP protocol is (a) complicated, involving techniques such as multilinearity test of functions and arithmetization, and (b) requires polynomial computation and polynomial rounds of interaction from the verifier. We construct a simple two prover, three round MRIP protocol for $\mathsf{Oracle\text{-}3SAT}$ using *scoring rules*.[3] Our protocol is very efficient: the verifier performs linear computation along with constant number of basic arithmetic operations to calculate the reward.

Contrasting MRIP with Other Relevant Models. Existing models, such as *refereed games*, MIP, and single-prover rational proofs all differ from MRIP in distinct and potentially interesting ways.

Refereed games [20] are interactive proof models consisting of two competing provers, who try to convince the verifier of the membership (non-membership) of a given input string in a language. However, one of them is honest and the other is not. The model of refereed games reflects the strategic nature of the provers, but does not allow collaboration between them.

Classical multi-prover interactive proofs are robust against arbitrary collaborative provers, who may be irrational or even malicious. While MIP protocols provide a stronger guarantee, they are often complicated and computationally-intensive (require polynomial-work from the verifier). In contrast, MRIP restricts provers to be rational, a reasonable assumption in a "mercantile world", as pointed out in [5]. This restriction leads to simple and efficient protocols for the single-prover case [6, 29], making rational proofs a meaningful and interesting model to study.

Finally, MRIP achieves its full power with only five rounds of interaction between the verifier and the provers. In contrast, RIP is less powerful when restricted to constant rounds.[4]

[2] Using the convention of [18], our simulation of any MRIP protocol with two provers would require only three rounds.

[3] Recalled in Section 2, scoring rules are powerful tools to elicit information about probabilistic distributions from experts.

[4] As shown in [5], $\mathsf{RIP\text{-}O(1)} = \mathsf{CH}$, where $\mathsf{RIP\text{-}O(1)}$ is the class of languages having constant-round rational proofs and CH is the counting hierarchy.

1.2 Additional Related Work

Interactive Proofs First introduced by Goldwasser, Micali and Rackoff [26] and in a different form by Babai and Moran [7], interactive proofs (IP) have been extensively studied (see, e.g., [8, 9, 10, 22, 23, 25, 28]) and precisely characterized: that is, $\mathsf{IP} = \mathsf{PSPACE}$ [34, 37]. Ben-Or et al. [10] introduced multi-prover interactive proofs (MIP), which were shown to be exactly equal to NEXP [8]. In fact, two provers and one round is sufficient; in other words, $\mathsf{NEXP} = \mathsf{MIP}(2, 1)$ [18] .

Recently, interactive proofs have been studied from the perspective of *computation delegation* [27, 30, 31, 36], and also in the *streaming* setting [12, 14, 15, 16].

Rational Proofs Azar and Micali [5] use scoring rules in a novel way to construct simple and efficient single-prover rational protocols. The prover is assumed to be sensitive to exponentially small losses in payment, that is, a negligible utility gap is a sufficient punishment for the prover. In [6], the same authors proposed the idea of utility gaps and constructed super-efficient rational proofs, where the verifier performs only logarithmic computation. Guo, Hubáček, Rosen and Vald [29] studied *rational arguments*, which are rational proofs with a single computationally-bounded rational prover. They constructed efficient (i.e., sub-linear verification) rational arguments for single-round computation delegation. They also mentioned that a rational proof model with multiple provers may have interesting implications in computation delegation schemes, but did not define such a model. How to scale down the computation power of the multiple provers in our model to achieve super efficient computation delegation is an interesting problem for future study.

Game-Theoretic Characterization of Complexity Classes Game-theoretic characterization of complexity classes has been largely studied in the form of *refereed games* [13, 17, 19, 20, 21, 33, 35].

The classic work of Chandra and Stockmeyer [13] proved that any language in PSPACE is refereeable by a game of perfect information. Feige and Kilian [17] show this is tight for single-round refereed games and that the class of languages with polynomial-round refereed games is exactly EXP.

Feigenbaum, Koller and Shor [21] study a related complexity class $\mathsf{EXP}^{\mathsf{NP}}$ and show that it can be simulated as a zero-sum refereed game between two computationally unbounded provers with *imperfect recall*. Note that imperfect recall is a very strong assumption and makes the computationally unbounded provers essentially act as oracles. By contrast, MRIP consists of collaborative provers having imperfect information (since a prover does not see the messages exchanged between the verifier and other provers) and perfect recall (since a prover remembers the history of messages exchanged between himself and the verifier). Notice that imperfect information is necessary for multi-prover protocols: if all provers can see all messages exchanged in the protocol, then the model degenerates to a single-prover case. Moreover, perfect recall gives the provers the ability to cheat adaptively across messages. With these differences, MRIP is equivalent to $\mathsf{EXP}^{||\mathsf{NP}}$. To the best of our knowledge, this is the first game-theoretic characterization of this complexity class.

It is worth pointing out that $\mathsf{EXP}^{\mathsf{NP}}$ is also an important class in the study of circuit lower bounds [40]. It would be interesting to see if the related complexity class $\mathsf{EXP}^{||\mathsf{NP}}$ emerges in similar contexts.

1.3 Outline of the paper

The paper is organized as follows. In Section 2 we define the class of languages that have multi-prover rational interactive proofs, MRIP, and discuss some of its properties. We construct MRIP protocols for NEXP in the same section. We characterize the class MRIP in Section 3. In Section 4 we show how to simulate any MRIP protocol with two provers and five rounds. In Section 5 we define the notion of *utility gap* and characterize the power of MRIP protocols with a polynomial utility gap. For readability, many of the longer proofs are moved to Appendix A. Finally, Appendix B provides a glossary of standard terms and definitions used throughout the paper.

2 Multi-Prover Rational Interactive Proofs

In this section we define the model of multi-prover rational interactive proofs and demonstrate several important properties of the class of languages recognized by these proofs.

First, we describe how the verifier and the provers interact, and introduce necessary notations on the way. Let L be a language, x a string whose membership in L is to be decided, and $n = |x|$. An interactive protocol is a pair $(V, \vec{P})$, where V is the *verifier* and $\vec{P} = (P_1, \ldots, P_{t(n)})$ is the vector of *provers*, with $t(n)$ a polynomial.[5] The verifier runs in polynomial time and flips private coins, whereas each P_i is computationally unbounded. The verifier and provers all know x. The verifier can communicate with each prover privately, but no two provers can communicate with each other. In a *round*, either each prover sends a message to the verifier, or the verifier sends a message to each prover, and these two cases alternate. Without loss of generality we assume the first round of messages are sent by the provers, and the first bit sent by P_1, denoted by c, indicates whether $x \in L$ (corresponding to $c = 1$) or not (corresponding to $c = 0$). Notice that c does not depend on the randomness used by V.

The length of each message and the number of rounds are polynomial in n. Let $p(n)$ be the number of rounds and r the random string used by V. For each $j \in \{1, 2, \ldots, p(n)\}$, let m_{ij} be the message exchanged between V and P_i in round j. In particular, the first bit of m_{11} is c. The transcript that each prover P_i has seen at the beginning of each round j is $(m_{i1}, m_{i2}, \ldots, m_{i(j-1)})$. Let $\vec{m}$ be the vector of all messages exchanged in the protocol (therefore $\vec{m}$ is a random variable depending on r).

At the end of the communication, the verifier computes a payment function R based on x, r, and $\vec{m}$ as the total payment to give to the provers. We restrict $R(x, r, \vec{m}) \in [-1, 1]$ for convenience.[6]

The protocol followed by V, including the payment function R, is public knowledge.

The verifier outputs c as the answer for the membership of x in L: that is, V does not check the provers' answer and follows it blindly. As will become clear from our results, this requirement for the verifier does not change the set of languages that have multi-prover rational interactive proofs. Indeed, we could have allowed V to compute his answer based on x, r, and $\vec{m}$ as well, but the current model eases later discussion on the payment loss of the provers caused by "reporting a wrong answer".

Each prover P_i can choose a *strategy* $s_{ij} : \{0,1\}^* \to \{0,1\}^*$ for each round j, which maps the transcript he has seen at the beginning of round j to the message he sends in that round.[7] Let $\tilde{s}_i = (s_{i1}, \ldots, s_{ip(n)})$ be the vector of strategies P_i uses in rounds $1, \ldots, p(n)$, and $\tilde{s} = (\tilde{s}_1, \ldots, \tilde{s}_{t(n)})$ be the strategy profile of the provers. Given any input x, randomness r, and strategy profile $\tilde{s}$, we denote by $(V, \vec{P})(x, r, \tilde{s})$ the vector of all messages exchanged in the protocol.

The provers are *cooperative* and jointly act to maximize the (total) expected payment received from the verifier. Note that this is equivalent to the provers maximizing their own expected payment when each prover receives a pre-specified division of the payment, that is, for any function γ_i fixed at the beginning of the protocol with $\sum_{i=1}^{n} \gamma_i = 1$, P_i receives $\gamma_i R$.

Thus, before the protocol starts, the provers pre-agree on a strategy profile $\tilde{s}$ that maximizes

$$u_{(V,\vec{P})}(\tilde{s}; x) \triangleq \mathop{\mathbb{E}}_{r} R(x, r, (V, \vec{P})(x, r, \tilde{s})).$$

When $(V, \vec{P})$ and x are clear from the context, we write $u(\tilde{s})$ for $u_{(V,\vec{P})}(\tilde{s}; x)$. Using the above notions, we define MRIP as follows.

Definition 1 (MRIP). *For any language L, an interactive protocol $(V, \vec{P})$ is a* multi-prover rational interactive proof (MRIP) *protocol for L if, for any $x \in \{0,1\}^*$ and any strategy profile $\tilde{s}$ of the provers such that $u(\tilde{s}) = \max_{\tilde{s}'} u(\tilde{s}')$, we have*

1. $u(\tilde{s}) \geq 0$;

2. $c = 1$ if and only if $x \in L$.

We denote by MRIP *the set of languages that have MRIP protocols, and by* $\mathsf{MRIP}(k, p)$ *the set of languages that have MRIP protocols with k provers and p rounds.*

MRIP possesses the following important property.

Lemma 1. MRIP *is closed under complement.*

[5] That is, we allow polynomially many provers.

[6] Note that the payment can be shifted and scaled so that it is in $[0, 1]$. We use both positive and negative payments in our model to better reflect the intuition behind our protocols: the former are rewards while the latter are punishments.

[7] P_i does not send any message s_{ij} for an even-numbered round j, and these s_{ij}'s can be treated as constant functions.

Proof. Let L be a language in MRIP, $(V, \vec{P})$ an MRIP protocol for L, and R the payment function computed by V. We construct a verifier V' and thus an MRIP protocol $(V', \vec{P})$ for $\overline{L}$, as follows.

- V' runs V to compute the messages he should send in each round, except that, whenever V' gives V as an input the first message m'_{11} sent by P_1, he flips the first bit.
- At the end of the communication, V' computes a payment function R': for any x, r, and $\vec{m}'$, $R'(x, r, \vec{m}') = R(x, r, \vec{m})$, where $\vec{m}$ is $\vec{m}'$ with the first bit flipped.
- V' outputs the first bit sent by P_1.

For each strategy profile $\tilde{s}$ of the provers in the protocol $(V, \vec{P})$, consider the following strategy profile $\tilde{s}'$ in the protocol $(V', \vec{P})$:

- $\tilde{s}'_i = \tilde{s}_i$ for each $i \neq 1$.
- In round 1, $\tilde{s}'_1$ outputs the same message as $\tilde{s}_1$, except that the first bit is flipped.
- For any odd number $k > 1$ and any transcript m'_1 for P_1 at the beginning of round k, $\tilde{s}'_1(m'_1)$ is the same as $\tilde{s}_1(m_1)$, where m_1 is m'_1 with the first bit flipped.

It is easy to see that for any x and r, $(V', \vec{P})(x, r, \tilde{s}')$ is the same as $(V, \vec{P})(x, r, \tilde{s})$ except the first bit. Thus $R'(x, r, (V', \vec{P})(x, r, \tilde{s}')) = R(x, r, (V, \vec{P})(x, r, \tilde{s}))$, which implies

$$u_{(V', \vec{P})}(\tilde{s}'; x) = u_{(V, \vec{P})}(\tilde{s}; x).$$

Also, it is easy to see that the mapping from $\tilde{s}$ to $\tilde{s}'$ is a bijection. Accordingly, arbitrarily fixing a strategy profile $\tilde{s}'$ that maximizes $u_{(V', \vec{P})}(\tilde{s}'; x)$, we have that the corresponding $\tilde{s}$ maximizes $u_{(V, \vec{P})}(\tilde{s}; x)$ as well. Thus

$$u_{(V', \vec{P})}(\tilde{s}'; x) = u_{(V, \vec{P})}(\tilde{s}; x) \geq 0,$$

where the inequality is by Definition 1. Furthermore, $x \in L$ if and only if the first bit sent by $\tilde{s}_1$ is 1, if and only if the first bit sent by $\tilde{s}'_1$ is 0. That is, $x \in \overline{L}$ if and only if the first bit sent by $\tilde{s}'_1$ is 1. Therefore $(V', \vec{P})$ is an MRIP protocol for $\overline{L}$, and Lemma 1 holds. □

To demonstrate the power of multi-prover rational proofs, we show that MRIP contains NEXP. With Lemma 1, this implies that MRIP contains coNEXP as well. We show this in two different ways.

First, we construct an MRIP protocol for any language in NEXP using a standard MIP protocol as a subroutine. Given existing MIP protocols, this method is intuitive and its correctness is easy to see. However, the computation and communication complexity of the protocol depends on the MIP protocol.

Second, we construct an MRIP protocol for an NEXP-complete language without relying on MIP protocols, instead by exploiting the rational nature of the provers. This protocol uses *proper scoring rules* to incentivize the provers. In contrast to MIP, this MRIP protocol is very efficient: it only requires the verifier to perform linear amount of computation and communication, along with computing the payment using constant number of arithmetic operations.

2.1 An MRIP Protocol for any Language in NEXP, Based on MIP.

An MIP protocol (see, e.g., [8, 18]) for a language $L \in$ NEXP first reduces L to the NEXP-complete problem Oracle-3SAT (defined below), and then runs an MIP protocol for Oracle-3SAT.

Definition 2 (Oracle-3SAT [8]). *Let w be a binary string of length $r + 3s$. Let B be a 3-CNF of $r + 3s + 3$ variables. A Boolean function $A : \{0, 1\}^s \to \{0, 1\}$ is a* 3-satisfying oracle *for B if $B(w, A(b_1), A(b_2), A(b_3))$ is satisfied for all w, where $b_1 b_2 b_3$ are the last $3s$ bits of w. The* Oracle-3SAT *problem is to decide, for a given B, if there is a 3-satisfying oracle for B.*

Our MRIP protocol for NEXP uses a black-box MIP protocol and an appropriate payment scheme.

Lemma 2. NEXP $\subseteq$ MRIP.

Proof. The MRIP protocol $(V, \vec{P})$ for L is shown in Figure 2. By construction, the payment to the provers is always non-negative.

For any input string x, the protocol $(V, \vec{P})$ works as follows:

1. P_1 sends a bit c to V. V outputs c at the end of the protocol.
2. If $c = 0$ then the protocol ends and the payment given to the provers is $R = 1/2$;
3. Otherwise, V and $\vec{P}$ run an MIP protocol for proving $x \in L$. If the verifier accepts then $R = 1$, else $R = 0$.

Figure 2: A simple MRIP protocol for NEXP.

Now we show that V outputs 1 if and only if $x \in L$. On the one hand, for any $x \in L$, if the provers send $c = 1$ and execute the MIP protocol with V, then their payment is $R = 1$;[8] while if they send $c = 0$, the payment is $R = 1/2 < 1$. Accordingly, their best strategy profile is to send $c = 1$ and run the MIP protocol correctly. On the other hand, for any $x \notin L$, if the provers send $c = 1$ and run the MIP protocol, then by the definition of MIP, the probability that V accepts is at most $1/3$, and the expected payment is at most $1/3$; while if they send $c = 0$, the payment is $1/2 > 1/3$. Accordingly, their best strategy profile is to send $c = 0$.

In sum, $(V, \vec{P})$ is an MRIP protocol for L and Lemma 2 holds. □

Remark 1. *Since any language $L \in$ NEXP has a 2-prover 1-round MIP protocol [18], we automatically obtain an MRIP protocol for L with 2 provers and 3 rounds of interaction (using our convention, executing the MIP protocol takes 2 rounds, plus one round for the answer bit c). Note that this is the best possible. However, the computation and communication complexity of the protocol are both polynomial (in addition to the reduction to* Oracle-3SAT*) and involve complex techniques such as arithmetization and multi-linearity test [8].*

By Lemmas 1 and 2, we immediately have the following corollary.

Corollary 1. coNEXP $\subseteq$ MRIP.

Also note that we can obtain a 2-prover 3-round MRIP protocol for coNEXP directly by modifying the protocol of Lemma 2.

2.2 An MRIP Protocol for any Language in NEXP, Using Scoring Rules.

We now construct an MRIP protocol for any language in NEXP without relying on MIP protocols. Instead, we use *proper scoring rules* to compute the payment that should be given to the provers so as to ensure truthful answers. To begin, we recall the definitions of proper scoring rules in general and the *Brier's scoring rule* in particular, which has been an essential ingredient in the construction of rational proofs [5, 6, 29].

Proper Scoring Rules. Scoring rules are tools to assess the quality of a probabilistic forecast by assigning a numerical score (that is, a payment to the forecaster) to it based on the predicted distribution and the sample that materializes. More precisely, given any probability space Σ, letting $\Delta(\Sigma)$ be the set of probability distributions over Σ, a *scoring rule* is a function from $\Delta(\Sigma) \times \Sigma$ to $\mathbb{R}$, the set of reals. A scoring rule S is *proper* if, for any distribution D over Σ and distribution $D' \neq D$, we have

$$\sum_{\omega \in \Sigma} D(\omega) S(D, \omega) \geq \sum_{\omega \in \Sigma} D(\omega) S(D', \omega),$$

where $D(\omega)$ is the probability that ω is drawn from D. A scoring rule S is *strictly proper* if the above inequality is strict. Notice that, when the true distribution is D, the forecaster maximizes the expected payment under a strictly proper scoring rule by reporting $D' = D$. For a comprehensive survey on scoring rules, see [24].

[8] If the MIP protocol does not have perfect completeness and accepts x with probability at least $2/3$, then the expected payment is at least $2/3$. However, this does not affect the correctness of the MRIP protocol.

Brier's Scoring Rule [11]. This classic scoring rule, denoted by BSR, is defined as follows: for any distribution D and $\omega \in \Sigma$,

$$\mathsf{BSR}(D, \omega) = 2D(\omega) - \sum_{\omega \in \Sigma} D(\omega)^2 - 1.$$

It is well known that BSR is strictly proper.

Notice that BSR requires the computation of $\sum_{\omega \in \Sigma} D(\omega)^2$, which can be hard when $|\Sigma|$ is large. However, similar to [5, 29], in this paper we only consider $\Sigma = \{0, 1\}$. Also notice that, BSR has range $[-2, 0]$ and can be shifted and scaled so that (1) the range is non-negative and bounded, and (2) the resulting scoring rule is still strictly proper. In particular, we shall add 2 to the function when using it.

Next, we construct a simple and efficient MRIP protocol for $\mathsf{Oracle\text{-}3SAT}$. Similar to the classical multi-prover case, an MRIP protocol for any language $L \in \mathsf{NEXP}$ can be obtained by first reducing L to $\mathsf{Oracle\text{-}3SAT}$, and then using our efficient MRIP protocol for $\mathsf{Oracle\text{-}3SAT}$. The complexity of the overall protocol for L is thus the same as the reduction. Our protocol is in Figure 3 and we have the following lemma, which is proved in the appendix.

Lemma 3. $\mathsf{Oracle\text{-}3SAT}$ *has an MRIP protocol with 2 provers and 3 rounds where, for any instance B of length n, the randomness used by the verifier, the computation complexity, and the communication complexity of the protocol are all $O(n)$, and the computation of the payment function consists of constant number of arithmetic operations over $O(n)$-bit numbers.*

Remark 2. *Although scoring rules have been used in rational proofs previously, our protocol makes use of an interesting property of Brier's scoring rule which, to our best knowledge, has never been used before. See the analysis of Lemma 3 for more details.*

For any instance B, the protocol $(V, \vec{P})$ works as follows:

1. P_1 sends $c \in \{0, 1\}$ and $a \in \{0, \ldots, 2^{r+3s}\}$ to V. V always outputs c at the end of the protocol.
2. If $c = 1$ and $a < 2^{r+3s}$, or if $c = 0$ and $a = 2^{r+3s}$, the protocol ends with payment $R = -1$.
3. Otherwise, V randomly chooses two binary strings of length $r + 3s$, $w = (z, b_1, b_2, b_3)$ and $w' = (z', b_4, b_5, b_6)$, as well as a number $k \in \{1, 2, 3, 4, 5, 6\}$.
   V sends $b_1, b_2, b_3, b_4, b_5, b_6$ to P_1 and b_k to P_2.
4. P_1 sends to V six bits, $A(b_i)$ with $i \in \{1, 2, ..., 6\}$, and P_2 sends one bit, $A'(b_k)$.
5. The protocol ends and V computes the payment R as follows.
   (a) If $A(b_k) \neq A'(b_k)$ then $R = -1$.
   (b) Otherwise, if $B(z, b_1, b_2, b_3, A(b_1), A(b_2), A(b_3)) = 0$ then $R = 0$.
   (c) Else, let $b = B(z', b_4, b_5, b_6, A(b_4), A(b_5), A(b_6))$, $p_1 = a/2^{r+3s}$, and $p_0 = 1 - p_1$.
   V computes R using Brier's scoring rule:
   if $b = 1$ then $R = \frac{2p_1 - (p_1^2 + p_0^2) + 1}{11}$; otherwise $R = \frac{2p_0 - (p_1^2 + p_0^2) + 1}{11}$.

Figure 3: An efficient MRIP protocol for $\mathsf{Oracle\text{-}3SAT}$.

3 Characterizing Multi-Prover Rational Interactive Proofs

In this section we prove Theorem 1, that is, $\mathsf{MRIP} = \mathsf{EXP}^{||\mathsf{NP}}$. We first show that MRIP is the same as another complexity class, $\mathsf{EXP}^{||\mathsf{poly-NEXP}}$, which we define below. We complete the proof of Theorem 1 by showing $\mathsf{EXP}^{||\mathsf{NP}} = \mathsf{EXP}^{||\mathsf{poly-NEXP}}$.

Definition 3. $\mathsf{EXP}^{||\mathsf{poly-NEXP}}$ *is the class of languages decidable by a deterministic exponential-time Turing machine with non-adaptive access to an* NEXP *oracle, such that the length of each oracle query is polynomial in the length of the input of the Turing machine.*

We use two intermediate lemmas to prove the lower bound. Lemma 4 is from [4] and gives a circuit characterization of EXP. Our Lemma 5 uses this characterization to provide an MRIP protocol for EXP.

The full proof of Lemma 5, along with an exposition of DC uniform circuits is in Appendix A.

Lemma 4 ([4])**.** *For any language L, $L \in \mathsf{EXP}$ if and only if it can be computed by a DC uniform circuit family of size $2^{n^{O(1)}}$.*

Lemma 5. *Every language L in EXP has an MRIP protocol with two provers and five rounds.*

Using these lemmas, Lemma 6 proves the lower bound on MRIP. We describe the main ideas in the proof sketch below.

Lemma 6. $\mathsf{EXP}^{||\mathsf{poly-NEXP}} \subseteq \mathsf{MRIP}$.

Proof Sketch Using the characterization of EXP in terms of DC uniform circuits from Lemma 4, we have an MRIP protocol for EXP with 2 provers and 5 rounds as in Lemma 5 (see Figure 5 in Appendix A). We then combine this protocol with the MRIP protocol for Oracle-3SAT in Figure 2 to obtain the desired MRIP protocol for $\mathsf{EXP}^{||\mathsf{poly-NEXP}}$.[9] In particular, we use the protocol for Oracle-3SAT to answer NEXP oracle queries. A similar structure has been used by [6] for single-prover rational proofs. However, the prover in [6] can send the entire proof of a circuit satisfiability problem to the verifier, which is not feasible for us because our circuit may be exponentially large while our verifier is still of polynomial time. We overcome this problem by using a second prover to cross-check whether the first prover is answering questions about the circuit correctly. See Appendix A for the full proof. □

Lemma 7 then shows that the above lower bound is tight, leading to an exact characterization.

Lemma 7. $\mathsf{MRIP} \subseteq \mathsf{EXP}^{||\mathsf{poly-NEXP}}$.

Proof Sketch For any language in MRIP, our EXP machine considers each possible payment to the provers. It uses the poly-NEXP oracle to verify that a payment can be generated by some strategy profile of the provers. Thus, it can find out the maximum expected payment the provers can get, and whether the corresponding strategy profile causes the verifier to accept. See Appendix A for the full proof. □

We have now established that $\mathsf{MRIP} = \mathsf{EXP}^{||\mathsf{poly-NEXP}}$. To finish the proof of Theorem 1, we show that $\mathsf{EXP}^{||\mathsf{poly-NEXP}}$ equals $\mathsf{EXP}^{||\mathsf{NP}}$, by a padding argument.

Lemma 8. $\mathsf{EXP}^{||\mathsf{poly-NEXP}} = \mathsf{EXP}^{||\mathsf{NP}}$.

Proof of Theorem 1. Theorem 1 follows immediately from Lemmas 6, 7, and 8. □

4 Simulating MRIP Using Two Provers and Five Rounds

In this section we prove Theorem 2, that is,

Theorem 2. $\mathsf{MRIP} = \mathsf{MRIP}(2, 5)$.

In particular, in Figure 4, we show how to simulate any MRIP protocol $(V, \vec{P})$ with $t(n)$ provers and $p(n)$ rounds using a verifier V' and two provers P_1' and P_2'. Recall that m_{ij} denotes the message exchanged between V and prover P_i in round j of their protocol. Essentially, V' asks one prover to simulate all other provers in the original protocol, and uses a second prover to cross-check his answers.

Proof of Theorem 2. Let $(V, \vec{P})$ be the MRIP protocol for a language L using $\vec{P} = (P_1, \ldots, P_{t(n)})$ provers and $p(n)$ rounds of communication. Let $\vec{m}$ denote the complete transcript of the protocol and m_{ij} be the message of prover P_i in round j of the protocol. We simulate this protocol using two provers P_1' and P_2' and verifier V' in the protocol $(V', \vec{P'})$ in Figure 4. Let r denote the random coin flips used by V in protocol $(V, \vec{P})$.

To see the correctness of the protocol, notice that,

(a) Even though V' sends the random string r of V to P_1', he uses a different random string $r' \neq r$ to select a message in Step 4. Thus, the expected payment to the provers depends on r', which is not known to them.

(b) P_2' does not know r and essentially commits to a transcript $\vec{m}'$ of $(V, \vec{P})$ in Step 5.

[9] The MRIP protocol for Oracle-3SAT in Figure 3 can also be used, with appropriate modifications in the computation of the payment. We use the protocol in Figure 2 to simplify the analysis.

Given any string x of length n,

1. P'_1 sends c, which is the first bit of m_{11}, to V'. V' outputs c at the end of the protocol.
2. V' calculates the random bits r used by V in protocol $(V, \vec{P})$. V' sends r to P'_1.
3. P'_1 uses r to simulate the entire protocol $(V, \vec{P})$ and sends $\vec{m}$ to V'.
4. V' chooses a round j at random from $\{1, \ldots, p(n)\}$ and a prover index k from $\{1, \ldots, t(n)\}$.
5. V' sends (j, k) and $\vec{m}_k = (m_{k1}, \ldots, m_{k(j-1)})$ (that is, message transcript till $j-1$ rounds of V with P_k) to P'_2.
6. P'_2 simulates P_k on the round j and sends m'_{kj} to V.
7. V computes the payment R' as follows,
   (a) If $j = 1$ and $k = 1$, then if the first bit of m'_{kj} is not c, set $R' = -1$.
   (b) If $m_{kj} \neq m'_{kj}$, set $R' = -1$.
   (c) Else, V' computes the payment R given by V in the protocol $(V, \vec{P})$, using $\vec{m}$.
   V' then sets $R' = \frac{R}{2p(n)t(n)}$.

Figure 4: Simulating MRIP with 2 provers and 5 rounds.

The provers can lie in the following ways, with the corresponding payments:

1. P'_1 and P'_2 do not commit to the same transcript $\vec{m}$. Without loss of generality assume that, P'_1 lies on some message in $\vec{m}$. V catches this lie in Step 6a with probability $\frac{1}{p(n)t(n)}$ and gives a payment $R' = -1$.
2. P'_1 and P'_2 agree on the transcript $\vec{m}$, but $\vec{m}$ does not correspond to the best strategy profile in protocol $(V, \vec{P})$. In this case, they get a payment $\frac{R}{2p(n)t(n)}$, where R is the payment that the original protocol generates on this suboptimal transcript $\vec{m}$.

First of all, we can rule out Case 2 as a possible strategy of rational provers P'_1 and P'_2. This is because V' pays them according to the original protocol $(V, \vec{P})$. Committing to any dishonest transcript earns them a payment strictly less than the best possible. This follows from the correctness of the original MRIP protocol $(V, \vec{P})$.

Now suppose P'_1 lies on $y > 0$ messages in $\vec{m}$ —this corresponds to Case 1. The expected payment to P'_1 would then be,

$$-1\left(\frac{y}{p(n)t(n)}\right) + \frac{R}{2p(n)t(n)}\left(\frac{p(n)t(n) - y}{p(n)t(n)}\right) \leq \frac{1}{p(n)t(n)}\left(\frac{R}{2} - y\right) < 0,$$

where the last inequality is because $R \leq 1$ and $y \geq 1$. Thus, it is not rational for P'_1 to lie on any message in $\vec{m}$ and the result follows. $\square$

Remark 3. *If we allow 3 provers instead of 2, then the verifier only needs to interact with each prover exactly once. In particular, V' sends r to P'_3 instead of P'_1. This property may be desirable for applications that do not allow repeated interactions (e.g., each prover may be available only for a limited amount of time).*

Optimal number of rounds. We conclude this section with a discussion on the optimal number of rounds required to capture the full power of MRIP. It is well known that MIP can be simulated using two provers and one round [18], which is clearly optimal. It would be interesting to obtain a similar result for MRIP. Note that, because we count the number of rounds differently (provers' messages and verifier's queries are considered to be separate rounds) and we assume the first round of messages are always from the provers, any non-trivial MRIP protocol with at least two provers requires at least three rounds. Indeed, in less than three rounds, the verifier cannot cross-check answers with the second prover (beyond the initial messages received from them), in which case the protocol reduces to a single-prover protocol.

Currently, the protocol in Figure 4 requires two extra rounds because the verifier's queries to each prover are not parallel. The verifier uses the answers from its query to Prover 1 to form its query to Prover 2. Getting rid of this dependency may result in a 2-prover 3-round protocol for MRIP. Interestingly, note that our protocols for NEXP and coNEXP only require 3 rounds (Section 2.1), so we know NEXP$\cup$

$\mathsf{coNEXP} \subseteq \mathsf{MRIP}(2,3)$. We leave the following closely related problems open for future study: (1) what is the optimal number of rounds required to capture the full power of MRIP, and (2) what is the exact characterization of $\mathsf{MRIP}(2,3)$?

5 Utility Gaps in MRIP

In our MRIP protocols until now, the provers are sensitive to arbitrarily small losses in the payment (similar to RIP in [5]). In [6] the authors strengthen their model by requiring the prover deviating from the honest protocol to suffer a non-negligible loss in their received payment (either constant or polynomial). This loss is demanded for *any* deviation from the prescribed behavior (i.e., the optimal strategy) and not just for reporting an incorrect answer to the membership of the input.

Formally, let $\tilde{s}$ be the optimal strategy profile and $\tilde{s}^*$ a suboptimal strategy profile of P. Then the *ideal* utility gap requires that $u(\tilde{s}) - u(\tilde{s}^*) > 1/\alpha(n)$, where $\alpha(n)$ is constant or polynomial in n. Although an ideal utility gap strongly guarantees that the prover uses his optimal strategy, as pointed out by [29], such a utility gap appears to be too strong to hold for many meaningful protocols.

To provide some intuition about the ideal utility gap, consider the MRIP protocol for NEXP given in Figure 2. This protocol does not satisfy the ideal utility gap condition, even though at first glance the gap appears to be significant. Indeed, the provers who *report the incorrect answer* always have a constant utility gap. However, a prover may tell the truth about the membership of x but deviate slightly from the optimal strategy. For example, the prover could lie so that the verifier accepts with probability $1-\varepsilon$, where ε is exponentially small in $|x|$. In this situation, the verifier may have an exponentially small probability of detecting the deviation, and the expected payment would decrease by an exponentially small amount.

The difficulty in achieving the ideal gap can be seen in previous work as well. For example, the rational proof for MA in [6] fails to satisfy this constraint. Guo, Hubáček, Rosen and Vald in [29] also echo the concern about the ideal utility gap being strong and define a weaker notion of utility gap. However, they impose it on *rational arguments* rather than rational proofs, and they still consider a single prover. They also require that *noticeable deviation* leads to *noticeable loss*: if under a strategy $\tilde{s}'$ of the prover, the probability for the verifier to output the correct answer is noticeably smaller than 1, then the expected payment to the prover under $\tilde{s}'$ is also noticeably smaller than the optimal expected payment. While we do not impose this requirement, our notion of utility gap as defined below implies it, but not vice-versa.

In particular, our notion allows us to encompass both the protocol in [6] and the NEXP protocol in Figure 2. Intuitively, we require that the provers who report the membership of the input incorrectly suffer noticeable loss in their received payment.

Definition 4 (Utility Gap)**.** *Let L be a language in* MRIP*, $(V,\vec{P})$ an MRIP protocol for L, $\tilde{s}$ the strategy profile of $\vec{P}$ that maximizes the expected payment, and c the first bit sent by P_1 according to $\tilde{s}$. For any other strategy profile $\tilde{s}'$, let c' be the first bit sent by P_1 according to $\tilde{s}'$. We say that $(V,\vec{P})$ has an* $\alpha(n)$-utility gap *if, whenever $c' \neq c$ we have*

$$\mathbb{E}_r R(x,r,(V,\vec{P})(x,r,\tilde{s})) - \mathbb{E}_r R(x,r,(V,\vec{P})(x,r,\tilde{s}')) > 1/\alpha(n).$$

We denote an MRIP protocol with an $\alpha(n)$-utility gap as an $\alpha(n)$-gap-MRIP protocol, and we shorten "polynomial in n" as $poly(n)$. It is easy to see that the protocol for NEXP in Figure 2 has an $O(1)$-utility gap. Following the analysis of Lemma 1 we get the same for coNEXP. That is,

Lemma 9. $\mathsf{NEXP} \subseteq \mathsf{O(1)\text{-}gap\text{-}MRIP}$ *and* $\mathsf{coNEXP} \subseteq \mathsf{O(1)\text{-}gap\text{-}MRIP}$.

We have shown that with an exponential utility gap, $\mathsf{MRIP} = \mathsf{EXP}^{||\mathsf{NP}}$. We now characterize MRIP protocols with a polynomial utility gap as exactly the class of languages decided by a polynomial-time oracle Turing machine with non-adaptive access to an NEXP oracle. That is,

Theorem 3. $\mathsf{poly(n)\text{-}gap\text{-}MRIP} = \mathsf{P}^{||\mathsf{NEXP}}$.

The proof of Theorem 3 uses similar ideas as that of Lemma 7 and can be found in Appendix A.

Remark 4. *There is a tradeoff between the utility gap and the computational efficiency in the two MRIP protocols we constructed for* NEXP*: the protocol in Figure 2 has a constant utility gap, but it relies on the standard* MIP *protocol; the protocol in Figure 3 is very efficient, but it only has an exponential utility gap. It would be interesting to see if there exists an MRIP protocol for* NEXP *that has a noticeable utility gap and is (almost) as efficient as the protocol in Figure 3.*

Acknowledgments

We thank several anonymous reviewers for their valuable feedback that helped improve this paper. We also thank Andrew Drucker, Silvio Micali, Rafael Pass and Sanjoy Das for their helpful comments.

References

[1] Amazon mechanical turk. Online at `https://www.mturk.com/mturk`.

[2] Effective use of amazon mechanical turk (mturk). Online at `http://neerajkumar.org/writings/mturk/`.

[3] Proof market. Online at `https://proofmarket.org`.

[4] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[5] Pablo Daniel Azar and Silvio Micali. Rational proofs. In *Proceedings of the Forty-Fourth Annual Symposium on Theory of Computing (STOC)*, pages 1017–1028, 2012.

[6] Pablo Daniel Azar and Silvio Micali. Super-efficient rational proofs. In *Proceedings of the Fourteenth Annual ACM conference on Electronic Commerce (EC)*, pages 29–30, 2013.

[7] László Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth annual ACM symposium on Theory of Computing (STOC)*, pages 421–429, 1985.

[8] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational complexity*, 1(1):3–40, 1991.

[9] László Babai and Shlomo Moran. Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36(2):254–276, 1988.

[10] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 113–131, 1988.

[11] Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[12] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and arthur-merlin communication. In *Proceedings of the 30th Conference on Computational Complexity*, pages 217–243, 2015.

[13] Ashok K Chandra and Larry J Stockmeyer. Alternation. In *17th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 98–108, 1976.

[14] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112, 2012.

[15] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment*, 5(1):25–36, 2011.

[16] Samira Daruki, Justin Thaler, and Suresh Venkatasubramanian. Streaming verification in data analysis. In *Algorithms and Computation*, pages 715–726. 2015.

[17] Uriel Feige and Joe Kilian. Making games short. In *Proceedings of the Twenty-Ninth Annual ACM Symposium On Theory of Computing (STOC)*, pages 506–516, 1997.

[18] Uriel Feige and László Lovász. Two-prover one-round proof systems: their power and their problems. In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing (STOC)*, pages 733–744, 1992.

[19] Uriel Feige and Adi Shamir. Multi-oracle interactive protocols with constant space verifiers. *Journal of Computer and System Sciences*, 44(2):259–271, 1992.

[20] Uriel Feige, Adi Shamir, and Moshe Tennenholtz. The noisy oracle problem. In *Proceedings of the Tenth Annual Conference on Advances in Cryptology (CRYPTO)*, pages 284–296, 1990.

[21] Joan Feigenbaum, Daphne Koller, and Peter Shor. A game-theoretic classification of interactive complexity classes. In *Proceedings of Tenth Annual IEEE Structure in Complexity Theory Conference*, pages 227–237, 1995.

[22] Lance Fortnow, John Rompel, and Michael Sipser. On the power of multi-prover interactive protocols. *Theoretical Computer Science*, 134(2):545–557, 1994.

[23] Lance Fortnow and Michael Sipser. Are there interactive protocols for co-np languages? *Information Processing Letters (IPL)*, 28(5):249–251, 1988.

[24] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[25] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *Journal of the ACM (JACM)*, 38(3):690–728, 1991.

[26] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1), 1989.

[27] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 113–122, 2008.

[28] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[29] Siyao Guo, Pavel Hubáček, Alon Rosen, and Margarita Vald. Rational arguments: single round delegation with sublinear verification. In *Proceedings of the Fifth Annual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 523–540, 2014.

[30] Tom Gur and Ron D Rothblum. Non-interactive proofs of proximity. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 133–142, 2015.

[31] Yael Tauman Kalai and Ron D Rothblum. Arguments of proximity. In *Advances in Cryptology–CRYPTO 2015*, pages 422–442. 2015.

[32] Aniket Kittur. Crowdsourcing, collaboration and creativity. *ACM Crossroads*, 17(2):22–26, 2010.

[33] Daphne Koller and Nimrod Megiddo. The complexity of two-person zero-sum games in extensive form. *Games and economic behavior*, 4(4):528–552, 1992.

[34] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM (JACM)*, 39(4):859–868, 1992.

[35] John H Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Sciences*, 29(2):274–301, 1984.

[36] Guy N Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 793–802, 2013.

[37] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[38] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326, 2004.

[39] Luis Von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[40] Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

A Omitted Proofs

A.1 Multi-Prover Rational Interactive Proofs

An MRIP for any Language in NEXP, Using Scoring Rules

Proof of Lemma 3. For any instance B with $r + 3s + 3$ variables (thus $n \geq r + 3s + 3$), the provers can, with their unbounded computation power, find an oracle A^* that maximizes the number of satisfying w's for B. Denote this number by a^*. If $B \in \mathsf{Oracle\text{-}3SAT}$ then $a^* = 2^{r+3s}$, otherwise $a^* < 2^{r+3s}$.

Roughly speaking, in our MRIP protocol $(V, \vec{P})$, the verifier incentivizes the provers to report the correct value of a^*, so that the membership of B can be decided. The protocol is shown in Figure 3.

To see why this protocol works, first notice that, if the provers send $c = 0$ and $a = 0$ in Step 1 and always send 0's in Step 4, then the protocol always ends in Steps 5b or 5c, and the provers' expected payment is 0. Accordingly, the best strategy profile $\tilde{s}^*$ of the provers gives them expected payment at least 0, and Condition 1 of Definition 1 holds. Moreover, $\tilde{s}^*$ must be such that

$$\text{either } c = 1 \text{ and } a = 2^{r+3s}, \text{ or } c = 0 \text{ and } a < 2^{r+3s}, \tag{1}$$

since otherwise the provers' expected payment is -1.

It remains to show that Condition 2 of Definition 1 holds. Notice that P_2 only answers one query of the verifier (in step 4), thus under any strategy $\tilde{s}_2$ and given any c and a, P_2 de facto commits to an oracle $A' : \{0,1\}^s \to \{0,1\}$. Assume that P_1, using a strategy $\tilde{s}_1$ and seeing $(b_1, ..., b_6)$, sends V six bits in step 4 that are not consistent with A' —that is, there exists $i \in \{1, \dots, 6\}$ such that $A(b_i) \neq A'(b_i)$. Let q be the probability that, conditioned on $(b_1, ..., b_6)$, the verifier chooses a k that catches the provers in step 5a. We have $q \geq 1/6$. Let R be the payment to the provers conditioned on $(b_1, ..., b_6)$ and on the event that they are not caught in step 5a. Note that $R \leq \frac{2}{11}$ by the definition of Brier's scoring rule (after shifting and scaling). Thus the expected payment to the provers conditioned on $(b_1, ..., b_6)$ is $-q + (1 - q)R < 0$. However, if P_1 answers the verifier's queries consistently with A', their expected payment conditioned on $(b_1, ..., b_6)$ is non-negative. Accordingly, the best strategy profile $\tilde{s}^*$ must be such that, for any c, a, and the oracle commited by P_2, P_1's answers for any $(b_1, ..., b_6)$ are always consistent with A'. Thus under $\tilde{s}^*$ the payment is never computed in step 5a.

Furthermore, given b_1, b_2, b_3 and A', whether B evaluates to 0 in step 5b or not is totally determined. If B evaluates to 0, then it does not matter what a or c is and the provers' received payment is 0. If B does not evaluate to 0 in step 5b, then the expected payment to the provers in step 5c is defined by Brier's scoring rule: the true distribution of b, denoted by D, is such that $D(1) = a'/2^{r+3s}$, with a' being the number of satisfying w's for B under oracle A'; and the realized value is $b = B(z', b_4, b_5, b_6, A(b_4), A(b_5), A(b_6))$. Indeed, since b_4, b_5, b_6 are independent from b_1, b_2, b_3, we have that w' is a uniformly random input to B, and the probability for b to be 1 is exactly $a'/2^{r+3s}$. Since Brier's scoring rule is strictly proper, conditioned on A' the provers maximize their expected utility by reporting

$$a = a', \tag{2}$$

which implies $(p_1, p_0) = (D(1), D(0))$.

If $B \notin \mathsf{Oracle\text{-}3SAT}$, then no matter which oracle A' is committed under $\tilde{s}^*$, we have $a' < 2^{r+3s}$. By Equations 1 and 2, we have $a < 2^{r+3s}$ and $c = 0$, as desired.

If $B \in \mathsf{Oracle\text{-}3SAT}$, we show that under $\tilde{s}^*$ the prover P_2 commits to the desired 3-satisfying oracle A^* (so that $a' = 2^{r+3s}$ and $D(1) = 1$). To see why this is true, denote by $\mathsf{BSR}(D)$ the expected score for reporting D under Brier's scoring rule, when the true distribution is D. We have

$$\begin{aligned} \mathsf{BSR}(D) &= D(1)[2D(1) - D(1)^2 - (1 - D(1))^2 - 1] \\ &\quad +(1 - D(1))[2(1 - D(1)) - D(1)^2 - (1 - D(1))^2 - 1] \\ &= 2(D(1)^2 - D(1)). \end{aligned}$$

Thus $\mathsf{BSR}(D)$ is symmetric at $D(1) = 1/2$, strictly decreasing on $D(1) \in [0, 1/2]$, strictly increasing on $D(1) \in [1/2, 1]$, and maximized when $D(1) = 1$ or $D(1) = 0$. Notice that the shifting and scaling of BSR in step 5c do not change these properties, but make $\mathsf{BSR}(D)$ strictly positive when $D(1) = 1$ or $D(1) = 0$. Accordingly, to maximize their expected payment conditioned on the event that step 5c is reached, P_2 should commit to either an oracle A' such that $D(1)$ is as small as possible, or an A' such that $D(1)$ is as large as possible, whichever makes $D(1)$ further from $1/2$.

If there is no oracle A' such that $a' = 0$ given A', then the only way for the provers to maximize their expected payment is to commit to the 3-satisfying oracle A^* (thus $a' = 1$), under which step 5c is reached with probability 1. Again by Equations 1 and 2, we have $c = 1$ and $a = 2^{r+3s}$ as desired.

If there is both a 3-satisfying oracle A^* and an oracle A' such that $a' = 0$, we need to make sure that P_2 does not commit to A': indeed, committing to any other oracle results in an expected payment strictly smaller than that by committing to A^*, since it increases the probability that the protocol ends at step 5b with $R = 0$, and strictly decreases the expected payment conditioned on step 5c being reached. If P_2 commits to A', then B *always* evaluates to 0 in step 5b, and step 5c is actually never reached. Thus, even though by committing to A' the provers maximize their expected payment in step 5c, their actual expected payment is 0. Instead, by committing to A^*, step 5c is reached with probability 1 and the provers get positive payment. Accordingly, the strategy profile $\tilde{s}^*$ must be such that P_2 commits to A^* and P_1 sends $a = 2^{r+3s}$ and $c = 1$, as desired. Notice that if there are multiple 3-satisfying oracles for B, then the provers can pre-agree on any one of them.

In sum, Condition 2 of Definition 1 also holds and $(V, \vec{P})$ is an MRIP protocol for $\mathsf{Oracle\text{-}3SAT}$. Moreover, since $n \geq r + 3s + 3$, it is easy to see that the number of coins flipped by V is at most $2n$ (for sampling w, w', and k) and the number of bits exchanged between V and $\vec{P}$ is at most $3n + 3$. Finally, given an input string $w = (z, b_1, b_2, b_3)$ for B and the 3-bit answers of the oracle for b_1, b_2, b_3, B can be evaluated in linear time, thus the running time of V is $O(n)$ plus constant number of arithmetic operations to compute the payment in step 5c. Therefore Lemma 3 holds. □

A.2 Characterizing Multi-Prover Rational Interactive Proofs

MRIP protocol for EXP**.**

Proof of Lemma 5. To begin, we recall some definitions and results from the literature that we use in our proof. First of all, a *circuit family* $\{C_n\}_{n=1}^{\infty}$ is a sequence of boolean circuits such that $C_n : \{0,1\}^n \to \{0,1\}$. The gates in the circuits are of type AND, OR, and NOT, with fan-ins 2, 2, and 1 respectively. The input string to a circuit is connected to a special set of "input gates", one for each bit of the input, whose output value is always the value of the corresponding bit. The *size* of a circuit C is the number of gates (including the input gates) in C. For a circuite C of size g, the set of gates in it is denoted by $\{1,2,...,g\}$. Without loss of generality we assume that gate g is the output gate of the whole circuit. Moreover, if C has input length n, without loss of generality we assume that gates $1,2,..,n$ are the input gates. Notice that the number of wires in C is at most $2g$, since each gate has at most 2 fan-ins. The set of wires is denoted by $\{1,2,...,2g\}$.

Definition 5 (DC uniform circuits [4]). *A circuit family $\{C_n\}_{n=1}^{\infty}$ is a* Direct Connect uniform (DC uniform) *family if the following questions can be answered in polynomial time:*

1. *$SIZE(n)$: what is the size of C_n?*
2. *$INPUT(n,h,i)$: is wire h an input to gate i in C_n?*
3. *$OUTPUT(n,h,i)$: is wire h the output of gate i in C_n?*
4. *$TYPE(n,i,t)$: is t the type of gate i in C_n?*

That is, the circuits in a DC uniform family may have exponential size, but they have a succinct representation in terms of a polynomial-time Turing machine that can answer all the questions in Definition 5. The class EXP can be characterized by the class of DC uniform circuit families [4]; this is stated formally in Lemma 4.

Now we are ready to prove Lemma 5.

Following Lemma 4, there exists a DC uniform circuit family $\{C_n\}_{n=1}^{\infty}$ that computes L. Let $g = 2^{n^k}$ be the size of each C_n, where k is a constant that may depend on L. For any input string x of length n and any gate $i \in \{1,2,...,g\}$ in C_n, let $v_i(x) \in \{0,1\}$ be the value of i's output on input x. In particular, $v_i(x) = x_i$ for any $i \in \{1,2,...,n\}$. We call a gate i' in C_n an *input gate* of i if there is a directed wire from i' to i. The MRIP protocol $(V, \vec{P})$ is shown in Figure 5.

Given any string x of length n,

1. P_1 sends one bit $c \in \{0,1\}$ to V. V always outputs c at the end of the protocol.
2. V computes $g = SIZE(n)$, picks a gate $i \in \{1,2,...,g\}$ uniformly at random, and sends i to P_1. That is, V queries P_1 for:
   (a) the type of gate i,
   (b) the input gates of i and corresponding input wires, and
   (c) the values of gate i and its input gates.
3. P_1 sends to V the concatenation of the following strings: type $t_i \in \{AND, OR, NOT\}$; gates $i_1, i_2 \in \{1,2,...,g\}$; wires $h_1, h_2 \in \{1,2,...,2g\}$; and values $v_i(x), v_{i_1}(x), v_{i_2}(x) \in \{0,1\}$.
4. V picks a gate $i' \in \{i, i_1, i_2\}$ uniformly at random and sends i' to P_2.
5. P_2 sends $v_{i'}(x) \in \{0,1\}$ to V.
6. The protocol ends and V computes the payment R by verifying the following statements:
   (a) If $i \notin \{1,2,...,n\}$ (that is, i is not an input gate), then $TYPE(n,i,t_i) = 1$, $INPUT(n,h_1,i) = OUTPUT(n,h_1,i_1) = 1$, and, if $t_i \neq NOT$, $INPUT(n,h_2,i) = OUTPUT(n,h_2,i_2) = 1$.
   (b) If $i \notin \{1,2,...,n\}$, then $v_i(x)$ is computed correctly based on type t_i from $v_{i_1}(x)$ and (when $t_i \neq NOT$) $v_{i_2}(x)$.
   (c) If $i \in \{1,2,...,n\}$ then $v_i(x) = x_i$.
   (d) If $i = g$ (that is, the output gate of the circuit) then $v_i(x) = c$.
   (e) The answers of P_1 and P_2 on the value of gate i' are consistent.

   If any of these verifications fails then $R = 0$; otherwise $R = 1$.

Figure 5: An MRIP protocol for EXP.

Clearly this protocol has two provers and five rounds. To see why it is an MRIP protocol, notice that if P_1 and P_2 always send the correct c and answer V's queries correctly according to C_n, their received payment is always $R = 1$, irrespective of V's coin flips. Thus the expected received payment is 1. Below we show that any other strategy profile gives expected payment strictly less than 1.

First of all, when the gate i chosen by the verifier in step 2 is not an input gate, if any of P_1's answers in step 3 to queries 2a and 2b (namely, about i's type, input gates and input wires) is incorrect, then the verification in step 6a fails, giving the provers a payment $R = 0$. Accordingly, if there exists such a gate then the expected payment to the provers is at most $1 - 1/g < 1$. Similarly, if there exists a non-input gate i such that P_1 answers queries 2a and 2b correctly but the values $v_i(x), v_{i_1}(x), v_{i_2}(x)$ are inconsistent with i's type, then, when i is chosen, step 6b fails, thus the expected payment to the provers is at most $1 - 1/g < 1$. Moreover, if there exists an input gate i such that $v_i(x) \neq x_i$ when i is chosen, or if $v_g(x) \neq c$ when gate g is chosen, then the expected payment to the provers is again at most $1 - 1/g < 1$.

Next, similar to the analysis of Lemma 3, P_2 is only queried once (in step 5). Thus P_2 de facto commits to an oracle $A : \{1, \ldots, g\} \to \{0, 1\}$, which maps any gate to its value under input x. If there exists a gate i such that the values $v_i(x), v_{i_1}(x), v_{i_2}(x)$ in step 3 are not consistent with A, then, conditioned on i being chosen in step 2, with probability $1/3$ step 6e fails. Since i is chosen with probability $1/g$, the expected payment is at most $1 - \frac{1}{3g} < 1$.

Accordingly, the only strategy profile $\tilde{s}$ that can have expected payment equal to 1 is exactly what we want: that is,

(1) P_1 and P_2 answer queries to the values of gates using the same oracle $A : \{1, \ldots, g\} \to \{0, 1\}$,
(2) $A(i) = x_i$ for any input gate i,
(3) $A(g) = c$, and
(4) for any non-input gate i, $A(i)$ is computed correctly according to i's type and the values of its input gates in C_n.

Thus $A(g)$ is computed according to C_n with input x, and $A(g) = 1$ if and only if $x \in L$. Since $c = A(g)$, we have that $c = 1$ if and only if $x \in L$, implying that $(V, \vec{P})$ is an MRIP protocol for L. Therefore Lemma 5 holds. □

Lower bound on MRIP**.**

Proof of Lemma 6. For any language $L \in \mathsf{EXP}^{||\mathsf{poly-NEXP}}$, let M be the exponential-time oracle Turing machine deciding it and O be the oracle used by M. Without loss of generality we assume $O = \mathsf{Oracle\text{-}3SAT}$. Let $q(n)$ be the number of oracle queries made by M on any input x of length n, and $p(n)$ be the length of each query. Notice that $p(n)$ is polynomial in n while $q(n)$ can be exponential. Let $\ell(n) = p(n)q(n)$. Below we may refer to $\ell(n)$, $p(n)$, $q(n)$ as ℓ, p, q when the input length n is clear from the context.

Since the oracle queries are non-adaptive, there exists an exponential-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that, for any $x \in \{0,1\}^n$, $f(x) \in \{0,1\}^\ell$ and $f(x)$ is the vector of oracle queries made by M given x. Indeed, we can run M on x until it outputs all queries. Similar to Lemma 4 (or rather, the way how it is proved), there exists a DC uniform circuit family $\{C_n\}_{n=0}^{\infty}$ of size $2^{n^{O(1)}}$ that computes f, where for any n, C_n has n-bit input and ℓ-bit output.

Moreover, given $b \in \{0,1\}^q$, the vector of oracle answers corresponding to M's queries, the membership of x can be decided in time exponential in n, by continuing running M. Accordingly, let $f' : \{0,1\}^* \to \{0,1\}$ be a function such that, given any $n + q$-bit input (x, b) where $|x| = n$ and b is the vector of oracle answers M gets with input x, $f'(x, b)$ is the output of M. Again similar to Lemma 4 and its analysis, f' is computable by a DC uniform circuit family $\{C'_n\}_{n=1}^{\infty}$ of size $2^{n^{O(1)}}$, where each C'_n has $n + q$-bit input and 1-bit output,[10] and the Turing machine that answers questions $SIZE, INPUT, OUTPUT, TYPE$ for C'_n runs in time polynomial in n rather than $n + q$.

[10]The size of C'_n is exponential in n but may not be exponential in its own input length, since q may be exponential in n.

Thus the membership of x in L can be computed by the following three-level "circuit" where, besides the usual AND, OR, NOT gates, there are q "NEXP" gates that have p bits input and 1 bit output, simulating the oracle.

- Level 1: Circuit C_n from the circuit family computing f. We denote its output by $(\phi_1, \phi_2, ..., \phi_q)$, where each ϕ_i is of p bits and is an instance of Oracle-3SAT. Let $g = 2^{n^k}$ be the size of C_n, where k is a constant. Similar as before, the set of gates is $\{1, 2, ..., g\}$, the set of input gates is $\{1, 2, ..., n\}$, and the set of output gates is $\{n+1, n+2, ..., n+\ell\}$, where the input and the output gates correspond to x and $(\phi_1, \phi_2, ..., \phi_q)$ in the natural order.
- Level 2: q NEXP gates, without loss of generality denoted by gates $g+1, g+2, ..., g+q$. For each $i \in \{1, 2, ..., q\}$, gate $g+i$ takes input ϕ_i and output 1 if and only if $\phi_i \in$ Oracle-3SAT.
- Level 3: Circuit C'_n from the circuit family computing f'. Let $g' = 2^{n^{k'}}$ be the size of C'_n, where k' is a constant. Following our naming convention, the set of gates is $\{g+q+1, g+q+2, ..., g+q+g'\}$, the set of input gates is $\{g+q+1, ..., g+q+n, g+q+n+1, ..., g+q+n+q\}$, and the output gate is gate $g+q+g'$. The first n of the input gates connect to x, and the remaining ones connect to the outputs of the NEXP gates of Level 2. The output of C'_n is the final output of the whole three-level circuit.

Given the circuit above, we can compute Level 1, Level 3, and each NEXP gate in Level 2 by using the protocols in Figure 2 and Figure 5, but we need to show that there exists a protocol $(V, \vec{P})$ where the verifier can get a consistent answer to all of them simultaneously. The protocol is specified in Figure 6. Note that the verifier needs to compute $q(n)$ and $p(n)$. Without loss of generality we can assume $q(n) = 2^{n^d}$ for some constant d, so that its binary representation can be computed in time polynomial in n. Since $p(n)$ is a polynomial in n, it can be computed by V.

To see why this is an MRIP protocol, first notice that for any input string x, conditioned on the gate i chosen by V in step 2, if the provers always give correct answers according to the computation of C_n, C'_n, and the NEXP gates, then the payment is at least $R = \frac{1}{p+1} \geq 0$. Indeed, it comes from the case where i is an NEXP gate and its input vector x' (step 9) is not in Oracle-3SAT, so that the protocol $(V, (P_3, P_4))$ outputs $c' = 0$ and $R' = 1/2$ as specified in step 2 of Figure 2, and the final payment is $R = \frac{2R'}{p+1} = \frac{1}{p+1}$ (step 10).

Conditioned on the same gate i, if P_1's answers in step 3 cause some verifications in step 4 to fail, then their received payment is $R = 0 < \frac{1}{p+1}$. Thus the best strategy profile of the provers when i is chosen is to make the verifications in step 4 go through. Since V chooses i uniformly at random, we have the following:

Claim 1. Under the provers' best strategy profile, P_1 makes the verifications in step 4 go through for every gate i.

Moreover, similar as before, P_2 is queried only once (step 6), thus any strategy of P_2 in $(V, \vec{P})$ de facto commits to an oracle $A : \{1, 2, ..., g+q+g'\} \to \{0, 1\}$, mapping each gate in the three-level circuit to its value under input x. For any gate i that is not an NEXP gate, conditioned on i being chosen in step 2, when P_1's answers to the values of i and its input gates (step 3) are all consistent with oracle A, the expected payment is $R = 1$ (step 8). When there exists an answer that is inconsistent with A, since i' is chosen uniformly at random from i and its input gates, and since i has at most 2 fan-ins, with probability $r \geq 1/3$ the protocol ends in step 7 with payment -1. Thus the expected payment is $R \leq -1/3 + 2/3 = 1/3 < 1$. That is, the best strategy profile of the provers conditioned on i being chosen is to answer the values of i and its input gates consistently with A. Since i is chosen uniformly at random, we have the following:

Claim 2. Under the provers' best strategy profile, P_1 always answers the value queries to i and its input gates consistently with A, for every i that is not an NEXP gate.

Combining the above two claims and the construction of step 4 (in particular, step 4c) we have that, under the provers' best strategy profile, the oracle A corresponds to the correct computation of C_n with input x, and the correct computation of C'_n with input *string x and A's own outputs for the* NEXP *gates*

Given any string x of length n,

1. P_1 sends one bit $c \in \{0, 1\}$ to V. V always outputs c at the end of the protocol.
2. V computes $g = SIZE(n)$ for C_n, $g' = SIZE(n)$ for C'_n, and $q(n)$.
   V picks a gate $i \in \{1, 2, ..., g + q + g'\}$ uniformly at random, and sends i to P_1. That is, V queries P_1 for:
   (a) the type of gate i, which can be AND, OR, NOT, NEXP, or $input$.
   (b) the input gates of i and corresponding input wires, and
   (c) the values of gate i and its input gates.
3. P_1 sends to V the following:
   (a) type $t_i \in \{AND, OR, NOT, \mathsf{NEXP}, input\}$;
   (b) input gates $i_1, i_2, ..., i_{p(n)}$ and wires $h_1, h_2, ..., h_{p(n)}$ (only the correct number of input gates and wires are used based on i's type);
   (c) values of i and its input gates: $v_i(x)$, $v_{i_1}(x), v_{i_2}(x), ..., v_{i_p}(x)$ (again only the correct number of input gates are used).
4. V verifies the following:
   (a) t_i is the correct type of i, by either using the DC uniformity of C_n and C'_n or using the naming convention when defining the three-level circuit (e.g., if $i \in \{g + 1, ..., g + q\}$ then $t_i = \mathsf{NEXP}$, etc);
   (b) the set of input gates of i is correct, again by using DC uniformity or the naming convention;
   (c) if $t_i \neq \mathsf{NEXP}$ then $v_i(x)$ is computed correctly from the values of i's input gates: that is, if i is an input gate connected to one bit of x then $v_i(x)$ equals the value of that bit, if i is an input gate of C'_n connected to an NEXP gate then $v_i(x) = v_{i_1}(x)$, otherwise $v_i(x)$ follows the Boolean logic between i and its input gates;
   (d) if $i = g + q + g'$ (that is, i is the output gate of the whole circuit), then $v_i(x) = c$.
   If any of the verifications fails, the protocol ends and $R = 0$.
5. V picks a gate i' uniformly at random from the union of $\{i\}$ and the set of its input gates, and sends i' to P_2.
6. P_2 sends $v_{i'}(x) \in \{0, 1\}$ to V.
7. V verifies that the answers of P_1 and P_2 on the value of gate i' are consistent. If not then the protocol ends and $R = -1$.
8. If $t_i \neq \mathsf{NEXP}$ then the protocol ends and $R = 1$.
9. If $t_i = \mathsf{NEXP}$, then V sends $x' = (v_{i_1}(x), ..., v_{i_p}(x))$ to P_3 and P_4 and runs the MRIP protocol for NEXP in Figure 2 with them, to decide whether $x' \in$ Oracle-3SAT or not, with the result denoted by c' and the payment by R'
10. If $c' = v_i(x)$ then $R = \frac{2R'}{p+1}$; otherwise $R = -1$.

Figure 6: An MRIP protocol for $\mathsf{EXP}^{||\mathsf{poly}-\mathsf{NEXP}}$.

in Level 2. Accordingly, it is left to show that under the provers' best strategy profile,

(1) A's own outputs for the NEXP gates are all correct,

(2) for any NEXP gate i, conditioned on i being chosen in step 2, P_1's answers to the values of i and its input gates are all consistent with A, and

(3) for any NEXP gate i, conditioned on i being chosen in step 2, P_3 and P_4 use their best strategy profile to compute the correct c'.

To see why this is the case, notice that if the provers' strategy profile are such that (1), (2), and (3) all hold, then conditioned on any NEXP gate i being chosen in step 2, the expected payment is as follows:

- $R = \frac{2}{p+1}$ when $x' \in \mathsf{Oracle\text{-}3SAT}$ ($R' = 1$ following step 3 of Figure 2);
- $R = \frac{1}{p+1}$ when $x' \notin \mathsf{Oracle\text{-}3SAT}$ ($R' = 1/2$ following step 2 of Figure 2).

If (1) and (2) hold but (3) does not hold for some NEXP gate i, then conditioned on i being chosen, either $c' \neq v_i(x)$ and the payment is $R = -1$, or $c' = v_i(x)$ but the payment R' is smaller than what P_3 and P_4 could have gotten (that is, 1 when $x' \in \mathsf{Oracle\text{-}3SAT}$ and $1/2$ otherwise). Either way the provers' received payment conditioned on i being chose is smaller than if (1), (2) and (3) all hold.

If (1) holds but (2) does not hold for some NEXP gate i, then conditioned on i being chosen, with probability $r \geq \frac{1}{p+1}$ the protocol ends in step 6 with $R = -1$. Since $R' \in [0, 1]$, the expected payment conditioned on i being chosen is

$$R \leq -\frac{1}{p+1} + (1 - \frac{1}{p+1}) \cdot \frac{2R'}{p+1} \leq -\frac{1}{p+1} + (1 - \frac{1}{p+1}) \cdot \frac{2}{p+1} = \frac{1}{p+1} - \frac{2}{(p+1)^2} < \frac{1}{p+1},$$

which is strictly smaller than if (1), (2) and (3) all hold.

If (1) does not hold for some NEXP gate i, then conditioned on i being chosen, we distinguish the following two cases:

Case 1 (2) does not hold for i. Similar to the analysis above, in this case the expected payment is

$$R \leq -\frac{1}{p+1} + (1 - \frac{1}{p+1}) \cdot \frac{2}{p+1} < \frac{1}{p+1},$$

strictly smaller than if (1), (2) and (3) all hold.

Case 2 (2) holds for i. In this case, x' is the correct input to gate i, but $v_i(x)$ is not the correct answer for whether $x' \in \mathsf{Oracle\text{-}3SAT}$ or not. Therefore no matter what P_3 and P_4 do, if c' is the correct answer for x' then $c' \neq v_i(x)$ and $R = -1$, or if c' is wrong then R' is strictly smaller than what P_3 and P_4 could have gotten under their best strategy profile for computing the correct answer. Either way, the expected payment is strictly smaller than if (1), (2) and (3) all hold.

In sum, under the provers' best strategy profile we have that (1), (2), and (3) all hold, and thus A's output for any gate i in the three-level circuit corresponds to the value of i under input x. In particular, $A(g + q + g')$ is the correct answer for whether $x \in L$ or not. Notice that if $c \neq A(g + q + g')$, then conditioned on gate $g + q + g'$ being chosen in step 2, the verification in step 4d fails. Thus under the provers' best strategy profile we have $c = A(g+q+g')$: that is, $c = 1$ if and only if $x \in L$. Accordingly, $(V, \vec{P})$ is an MRIP protocol for L and Lemma 6 holds. □

Upper Bound on MRIP.

Proof of Lemma 7. To prove Lemma 7, arbitrarily fix a language $L \in \mathsf{MRIP}$ and let $(V, \vec{P})$ be an MRIP protocol for L. Since V runs in polynomial time, there exists a constant k such that, for any two payments R and R' that can be output by V under some input of length n and some randomness, we have

$$R \neq R' \Rightarrow |R - R'| \geq \frac{1}{2^{n^k}}.$$

For example, n^k can be an upper bound of V's running time. Moreover, since V uses polynomially many random coins, there exists another constant k' such that, when a payment appears with positive probability under some input of length n, it must appear with probability at least $\frac{1}{2^{n^{k'}}}$. Accordingly, for any input x of length n and any two strategy profiles $\tilde{s}$ and $\tilde{s}'$ of the provers, if the expected payments $u(\tilde{s};x)$ and $u(\tilde{s}';x)$ are different, then

$$|u(\tilde{s};x) - u(\tilde{s}';x)| \geq \frac{1}{2^{n^{k+k'}}}. \tag{3}$$

Consider the following deterministic oracle Turing machine M: Given any input x of length n, it divides the interval $[0,1]$ to $2 \cdot 2^{n^{k+k'}}$ sub-intervals of length $\frac{1}{2 \cdot 2^{n^{k+k'}}}$. For any $i \in \{1, ..., 2 \cdot 2^{n^{k+k'}}\}$, the i-th interval is $[2(i-1) \cdot 2^{n^{k+k'}}, 2i \cdot 2^{n^{k+k'}}]$. M then makes $4 \cdot 2^{n^{k+k'}}$ oracle queries of the form (i,j), where $i \in \{1, ..., 2 \cdot 2^{n^{k+k'}}\}$ and $j \in \{0,1\}$. Notice that the lengths of the queries are upper bounded by $2 + n^{k+k'}$, which is a polynomial as required by the class $\mathsf{EXP}^{||\mathsf{poly}-\mathsf{NEXP}}$.

For each query (i,j), if $j = 0$ then the corresponding question is "whether there exists a strategy profile $\tilde{s}$ of the provers such that $u(\tilde{s};x)$ is in the i-th interval"; and if $j = 1$ then the corresponding question is "whether there exists a strategy profile $\tilde{s}$ such that $u(\tilde{s};x)$ is in the i-th interval *and the first bit sent by* P_1 *is* $c = 1$". Notice that all the queries are indeed non-adaptive. We say that interval i is *non-empty* if the query $(i,0)$ is answered 1, and *empty* otherwise.

Assume for now all the queries are answered correctly. M finds the highest index i^* such that the interval i^* is non-empty. It accepts if $(i^*,1)$ is answered 1, and rejects otherwise. M clearly runs in exponential time. To see why it decides L (given correct oracle answers), notice that by Definition 1, there exists a strategy profile whose expected payment is non-negative and thus in $[0,1]$ (since the payment is always in $[-1,1]$). Thus there exists an interval i such that $(i,0)$ is answered 1. Also by definition, the best strategy profile $\tilde{s}$ has the highest expected payment, and thus $u(\tilde{s};x)$ falls into interval i^*.

By Inequality 3, any strategy profile $\tilde{s}'$ with $u(\tilde{s}';x) < u(\tilde{s};x)$ has $u(\tilde{s}';x)$ not in interval i^*, since the difference between $u(\tilde{s}';x)$ and $u(\tilde{s};x)$ is larger than the length of the interval. Accordingly, all strategy profiles $\tilde{s}'$ with $u(\tilde{s}';x)$ in interval i^* satisfies $u(\tilde{s}';x) = u(\tilde{s};x)$: they are all the best strategy profiles of the provers. Again by Definition 1, P_1 sends the same first bit c under all these strategy profiles, $c = 1$ if and only if $x \in L$, and there does not exist any other strategy profile whose expected payment falls into interval i^* but the first bit sent by P_1 is different from c. Thus the answer to $(i^*,1)$ always equals c, and M accepts if and only if $c = 1$, if and only if $x \in L$, as desired.

It remains to show that the oracle queries can be answered by an NEXP oracle. Recall that in the protocol $(V, \vec{P})$ a strategy s_{ij} of each prover P_i for each round j is a function mapping the transcript P_i has seen at the beginning of round j to the message he sends in that round. Since the protocol has polynomially many provers and polynomially many rounds, by definition a strategy profile consists of polynomially many functions from $\{0,1\}^*$ to $\{0,1\}^*$, where for each function both the input length and the output length are polynomial in n (since otherwise the messages cannot be processed by V). Accordingly, it takes exponentially many bits to specify each function: if the input length is $p(n)$ and the output length is $q(n)$, then $2^{p(n)}q(n)$ bits are sufficient to specify the truth table of the function. Therefore a strategy profile can also be specified by exponentially many bits. Below we construct an exponential-time non-deterministic Turing machine M' that decides the questions corresponding to the queries.

Given any input $(i,0)$, M' non-deterministically chooses a strategy profile $\tilde{s}$, in exponential time. It then goes through all the realizations of V's random string, and for each realization r it runs V with x and r, and generates the provers' messages by looking them up in the corresponding truth table in $\tilde{s}$. M' computes for each r the payment output by V at the end of the protocol, and by combining these payments with corresponding probabilities M' computes the expected payment $u(\tilde{s};x)$. If $u(\tilde{s};x)$ is in interval i then M' accepts, otherwise it rejects. Since V's random string is polynomially long, there are exponentially many realizations, and each one of them takes exponential time to run: V runs in polynomial time and it takes M' exponential time to look up the truth tables in $\tilde{s}$ to generate a prover's message. Thus M' runs in non-deterministic exponential time. Also, if interval i is non-empty then there exists a strategy profile $\tilde{s}$ that makes M' accept, and if interval i is empty then M' always rejects.

Similarly, given any input $(i, 1)$, M' non-deterministically chooses a strategy profile $\tilde{s}$ and computes its expected payment $u(\tilde{s}; x)$. If $u(\tilde{s}; x)$ is not in interval i then M' rejects; otherwise, M' accepts if and only if the first bit sent by P_1 is 1. The correctness and the running time of this part follows immediately.

In sum, there exists an NEXP oracle that answers M's queries correctly and Lemma 7 holds. $\square$

Equivalence of $\mathsf{EXP}^{||\mathsf{poly-NEXP}}$ and $\mathsf{EXP}^{||\mathsf{NP}}$.

Proof of Lemma 8. First, we show $\mathsf{EXP}^{||\mathsf{poly-NEXP}} \subseteq \mathsf{EXP}^{||\mathsf{NP}}$ using a padding argument. Let M_1 be an exponential-time oracle Turing machine with non-adaptive access to an oracle O_1 for an NEXP language, where the lengths of the oracle queries are polynomial in the input length. Let O_1 be decided by a non-deterministic Turing machine M_{O_1} with time complexity $2^{|q|^{k_1}}$, where k_1 is a constant and q is the query to the oracle (thus the input to M_{O_1}). We simulate $M_1^{O_1}$ using another exponential-time oracle Turning machine M_2 and another oracle O_2, as follows.

Given any input x of length n, M_2 runs M_1 to generate all the oracle queries. For each query q, M_2 generates a query q' which is q followed by $2^{|q|^{k_1}}$ bits of 1. It then gives all the new queries to its own oracle O_2. Given the oracle's answers, M_2 continues running M_1 to the end, and accepts if and only if M_1 does. Since $|q|$ is polynomial in n, $2^{|q|^{k_1}}$ is exponential in n. Furthermore, since there are exponentially many queries and M_1 runs in exponential time, we have that M_2 runs in exponential time as well. It is clear that (1) M_2 makes non-adaptive oracle queries, and (2) $M_2^{O_2}$ decides the same language as $M_1^{O_1}$ if O_2's answer to each query $q' = (q, 1^{2^{|q|^{k_1}}})$ is the same as O_1's answer to the corresponding query q.

We define O_2 by constructing a non-deterministic Turing machine M_{O_2} that simulates M_{O_1}. O_2 is the language decided by M_{O_2}. More specifically, given a query $q' = (q, 1^{2^{|q|^{k_1}}})$, M_{O_2} runs M_{O_1} on input q, makes the same non-deterministic choices as M_{O_1}, and outputs whatever M_{O_1} outputs. Since M_{O_1} runs in time $2^{|q|^{k_1}}$, M_{O_2}'s running time is polynomial (in fact, linear) in its own input length. Thus the language decided by M_{O_2} is in NP, and $q' = (q, 1^{2^{|q|^{k_1}}})$ is in this language if and only if q is in the language decided by M_{O_1}.

In sum, $M_2^{O_2}$ decides the same language as $M_1^{O_1}$, and we have $\mathsf{EXP}^{||\mathsf{poly-NEXP}} \subseteq \mathsf{EXP}^{||\mathsf{NP}}$.

Now we show $\mathsf{EXP}^{||\mathsf{NP}} \subseteq \mathsf{EXP}^{||\mathsf{poly-NEXP}}$. The analysis is somewhat symmetric to what we have seen above. Let M_2 be an exponential-time oracle Turing machine with non-adaptive access to an oracle O_2 for an NP language. Notice that the queries made by M_2 can be exponentially long. Let O_2 be decided by a non-deterministic Turing machine M_{O_2} with time complexity $|q|^{k_2}$, where k_2 is a constant and q is the query to the oracle (thus the input to M_{O_2}). We simulate $M_2^{O_2}$ using another exponential-time oracle Turning machine M_1 and another oracle O_1, as follows.

Given any input x of length n, M_1 runs M_2 to compute the *number of oracle queries* made by M_2, denoted by Q. M_1 generates Q oracle queries, with the i-th query being x followed by the binary representation of i. Since M_2 makes at most exponentially many queries, the length of each query made by M_1 is (at most) polynomial in n.

Query i of M_1 corresponds to the following question: *is the i-th query made by M_2 given input x in the language O_2?* M_1 then gives all its queries to its own oracle O_1. Given the oracle's answers, M_1 uses them to continue running M_2, and accepts if and only if M_2 does. Since M_2 runs in exponential time, M_1 runs in exponential time as well. It is clear that (1) M_1 makes non-adaptive oracle queries, and (2) $M_1^{O_1}$ decides the same language as $M_2^{O_2}$ if O_1 answers each query correctly.

We define O_1 by constructing a non-deterministic Turing machine M_{O_1} that simulates M_{O_2}. O_1 is the language decided by M_{O_1}. More specifically, given an input string of the form (x, y), M_{O_1} interprets the second part as the binary representation of an integer i. It runs M_2 on input x to compute its i-th query, denoted by q. It then runs M_{O_2} on input q, makes the same non-deterministic choices as M_{O_2}, and outputs whatever M_{O_2} outputs. Since q is at most exponentially long in $|x|$ and M_{O_2}'s running time is $|q|^{k_2}$, the running time of M_{O_1} is (at most) exponential in its input length. Thus the language decided by M_{O_1} is in NEXP. Moreover, if q is in the language O_2 then there exist non-deterministic choices that make M_{O_2} and thus M_{O_1} accept; otherwise M_{O_2} and thus M_{O_1} always reject. That is, O_1's answers to M_1's queries under input x are the same as O_2's answers to M_2's queries under the same input.

In sum, $M_1^{O_1}$ decides the same language as $M_2^{O_2}$, and we have $\mathsf{EXP}^{||\mathsf{NP}} \subseteq \mathsf{EXP}^{||\mathsf{poly-NEXP}}$. $\square$

A.3 Utility Gaps in MRIP

Proof of Theorem 3. First, we show the lower bound: $\mathsf{P}^{||\mathsf{NEXP}} \subseteq \mathsf{poly(n)\text{-}gap\text{-}MRIP}$. We note that every language in P can be decided by a DC-uniform circuit family $\{C_n\}_{n=1}^{\infty}$ of polynomial size. We run the protocol from Lemma 6, noting that the Level 1 and Level 3 circuits are polynomially sized. Thus p, g and g' are polynomial. A case-by-cases analysis shows that the protocol from Lemma 6 achieves a polynomial gap.

Now we show $\mathsf{poly(n)\text{-}gap\text{-}MRIP} \subseteq \mathsf{P}^{||\mathsf{NEXP}}$. For $L \in \mathsf{poly(n)\text{-}gap\text{-}MRIP}$, let $\alpha(n)$ be the polynomial in n which gives the utility gap in the protocol $(V, \vec{P})$ for L. We simulate the protocol using a $\mathsf{P}^{||\mathsf{NEXP}}$ Turing machine.

Divide the interval $[0, 1]$ into evenly-sized intervals of size $2\alpha(n)$. In other words, consider each interval $[i/2\alpha(n), (i+1)/2\alpha(n)]$ for each $i \in \{0, \ldots, 2\alpha(n) - 1\}$. Let $\tilde{S}_i$ be the set of strategy profiles of the provers $\vec{P}$ with expected payment in the range $[i/\alpha(n), (i + 1)/\alpha(n)]$. By Definition 4, if the strategy profile maximizing the expected payment has $c = 1$, then all strategy profiles in the same interval have $c = 1$.

We modify the ideas in Lemma 7, but we only iterate over a polynomial number of expected-payment intervals, one for each i. For each interval $[i/2\alpha(n), (i+1)/2\alpha(n)]$, a polynomial turing machine makes the following two queries to an NEXP oracle:

1. Is $\tilde{S}_i$ empty (that is, is there a stategy with expected payment in $[i/2\alpha(n), (i + 1)/2\alpha(n)]$)?
2. Is there a strategy profile with $c = 1$ in $\tilde{S}_i$?

Each of these queries can be solved using an NEXP oracle. The polynomial Turing machine finds the largest i with a nonempty interval—the first query returned that there is a strategy in i. If the second query on interval i finds a strategy with $c = 1$, all strategies in i must have $c = 1$ by assumption, so we must have $x \in L$. On the other hand, if the second query does not find a strategy in the interval with $c = 1$, the optimal protocol must have $c = 0$, and $x \notin L$.

Note that some queries made above may be unnecessary, but we make all queries to preserve non-adaptivity. $\square$

B Glossary of Definitions and Notions

For completeness, here we recall various terms and definitions used throughout the paper.

Definition 6. *The complexity classes used in our paper are as follows.*

- $\mathsf{EXP} = \bigcup_{k\in\mathbb{N}} \mathsf{DTIME}(2^{n^k})$.
- $\mathsf{NEXP} = \bigcup_{k\in\mathbb{N}} \mathsf{NTIME}(2^{n^k})$.
- $\mathsf{PSPACE} = \bigcup_{k\in\mathbb{N}} \mathsf{SPACE}(n^k)$.
- $\mathsf{P}^{||\mathsf{NEXP}}$ *is the set of languages decided by a deterministic polynomial-time Turing machine with non-adaptive access to an* NEXP *oracle.*
- $\mathsf{EXP}^{||\mathsf{NP}}$ *is the set of languages decided by a deterministic exponential-time Turing machine with non-adaptive access to an* NP *oracle.*

Interactive Proofs Interactive proof protocols consist of computational unbounded provers, denoted by $P_1, P_2, \ldots$, and a polynomial-time randomized verifier V. With an underlying language L and given an input string x, the provers try to convince the verifier of the membership of x in L. This involves an exchange of messages between them, in which the verifier uses his coin flips and the transcript of messages to decide whether he should accept or reject. In multi-prover interactive proofs the provers cannot communicate with each other during the protocol, nor can a prover learn the messages exchanged between the verifier and another prover. They can, however, pre-agree on how to respond to particular queries of the verifier. We define the two kinds of interactive proofs formally below. Let $(V, P)(x)$

denote the (randomized) transcript of messages exchanged during the protocol on input x when there is a single prover, and $(V, \vec{P})(x)$ the transcript when there are multiple provers $\vec{P} = (P_1, \dots, P_t)$, where t is polynomial in $|x|$.

Definition 7 (Interactive Proofs (IP))**.** *Let $k : \mathbb{N} \to \mathbb{N}$ be a function with $k(n)$ polynomial in n. A language L has an interactive proof (IP) protocol with k rounds (that is, $L \in \mathsf{IP}[\mathsf{k}]$) if there is a Turning machine V such that, on any input x of length n, V runs in time polynomial in n and,*

- *(Completeness) $x \in L \implies \exists P$ s.t. Pr[V accepts $(V, P)(x)$] $\geq 2/3$.*
- *(Soundness) $x \notin L \implies \forall P$ Pr[V accepts $(V, P)(x)$] $\leq 1/3$.*

We denote by IP *the set of languages having polynomial-round IP protocols.*

Definition 8 (Multi-Prover Interactive Proofs (MIP))**.** *Let $k : \mathbb{N} \to \mathbb{N}$ be a function with $k(n)$ polynomial in n. A language L has a multi-prover interactive proof (MIP) protocol with k rounds (that is, $L \in \mathsf{MIP}[\mathsf{k}]$) if there is a Turning machine V such that, on any input x of length n, V runs in time polynomial in n and,*

- *(Completeness) $x \in L \implies \exists \vec{P}$ s.t. Pr[V accepts $(V, \vec{P})(x)$] $\geq 2/3$.*
- *(Soundness) $x \notin L \implies \forall \vec{P}$ Pr[V accepts $(V, \vec{P})(x)$] $\leq 1/3$.*

We denote by MIP *the set of languages having polynomial-round MIP protocols.*

Alternatively, one can define IP and MIP with perfect completeness: that is, V accepts with probability 1 when $x \in L$.

Rational Interactive Proofs In rational interactive proof (RIP) protocols, a single computationally unbounded prover P interacts with a polynomial-time randomized verifier. P can choose a strategy $s_j : \{0,1\}^* \to \{0,1\}^*$ in each round j of the protocol, based on the transcript of messages he has seen so far. Let $\tilde{s} = (s_1, \dots, s_k)$ be the vector of strategies P uses in rounds $1, \dots, k$, where $k : \mathbb{N} \to \mathbb{N}$ is polynomial in the length of input x. At the end of the protocol, the verifier computes an output and a payment function $R(x, r, (V, P)(x, r, \tilde{s}))$, based on the input x, his own randomness r, and the transcript $(V, P)(x, r, \tilde{s})$. P is *rational* and chooses a strategy that maximizes his utility,

$$u_{(V,P)}(\tilde{s}; x) \triangleq \mathop{\mathbb{E}}_{r} R(x, r, (V, P)(x, r, \tilde{s})).$$

For simplicity, we may write $u(\tilde{s})$ for $u_{(V,P)}(\tilde{s}; x)$. We can now define the class RIP.

Definition 9 (Rational Interactive Proofs (RIP))**.** *For any language L, an interactive protocol (V, P) is a* rational interactive proof (RIP) *protocol for L if, for any $x \in \{0,1\}^*$ and any strategy $\tilde{s}$ of P such that $u(\tilde{s}) = \max_{\tilde{s}'} u(\tilde{s}')$, we have*

1. *$u(\tilde{s}) \geq 0$;*
2. *V outputs 1 if and only if $x \in L$.*

We denote by RIP *the set of languages that have RIP protocols.*