The Mechanism Design Approach to Interactive Proofs

A Dissertation presented

by

Shikha Singh

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

August 2018

Stony Brook University

The Graduate School

Shikha Singh

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation

Michael Bender - Dissertation Advisor
Professor, Computer Science

Jing Chen - Dissertation Advisor
Assistant Professor, Computer Science

Steven Skiena - Chairperson of Defense
Professor, Computer Science

Jie Gao
Associate Professor, Computer Science

Martin Farach-Colton
Professor, Computer Science, Rutgers University

This dissertation is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

The Mechanism Design Approach to Interactive Proofs

by

Shikha Singh

Doctor of Philosophy

in

Computer Science

Stony Brook University

2018

Most computation today is not done locally by a client but rather outsourced to third-party service providers in exchange for money. Trading computation for money brings up two problems—(a) how can the client guarantee correctness of the outsourced computation efficiently, and (b) how to design an effective payment scheme. The two problems are closely related—ideally, we want the payment scheme to incentivize the service providers to perform the computation correctly.

Interactive proofs (IP) are a fundamental theoretical framework used to study verifiable computation outsourcing. In an IP, the weak client (or verifier) interacts with powerful service providers (or provers) to determine the truthfulness of their claim. IPs give strong guarantees; however they do not admit a non-trivial payment scheme.

This dissertation aims to answer the following question: *if computation is being traded for money in financially-driven marketplaces, how do we use payments to leverage correctness?* Rational proofs are payment-based interactive proofs for computation outsourcing which incentivize the service provider to do the computation correctly. In a rational proof, the prover is rational in the economic sense and acts to maximize its payment.

In this dissertation, we introduce and analyze the model of rational proofs with multiple service providers. Multiple rational agents pose new game-theoretic challenges: how do

we design and analyze their joint incentive structure? Provers may be cooperative (work together as a team to maximize their total payment) or non-cooperative (act to maximize their own individual payment). Using principles from game theory and mechanism design, we show how the incentives of multiple rational provers can used against each other to design simple, efficient and extremely powerful proof systems.

Besides the main work on rational proofs, this work also addresses an important algorithmic problem faced by memory-constrained devices that store most of their data on external storage (accessing which is expensive). In particular, we study the following problem: how do we efficiently test membership and perform updates on a set that is stored remotely? We present improved variants of a Bloom filter, a data structure that is widely used to speed up membership queries to a remote set.

Dedicated to my parents, Nirupma and Subhash Singh, for teaching me to work hard and to dream big.

Table of Contents

List of Figures

Acknowledgements

I have been fortunate to have had amazing teachers and mentors in my life, and I am grateful to each one of them even if they are not explicitly mentioned here.

I would like to start by acknowledging my fantastic advisors, Jing Chen and Michael Bender, for their unwavering support and guidance during the last five years. From Jing I learnt the value of patience and rigor when attacking challenging research problems. From Michael I learnt to appreciate the elegance in simple proofs and the importance of communicating research ideas well. It has been an immensely rewarding experience to observe Michael and Jing at work and it is to them that I owe my research aesthetics, my writing and teaching style and my love for collaboration.

I am also grateful to Steven Skiena, Martin Farach-Colton and Rob Johnson for their mentorship and time; to Steve, for trusting me with his classes and helping me figure out what I wanted to do with my life; and to Martin and Rob, for making my ideas feel valued in meetings and for making research fun with their endless repertoire of puns and sarcasm.

I was fortunate to have the opportunity to visit several research labs during my PhD. Working in these new environments with new people has helped me grow as a researcher. I would like to thank Manoj Gupta, Mayank Goswami and Nguyen Kim Thang for hosting me in their labs, introducing me to new research problems and advising me during my visit.

I want to thank the Algorithms Reading Group, especially Estie Arkin, Joe Mitchell, Rezaul Chowdhury, Jie Gao, Rob Patro, Michael Bender, Jing Chen, and Steve Skiena, for sharing fun problems and creating a vibrant algorithms culture at Stony Brook. A special mention to our admin team, Kathy Germana, Betty Knitweiss and Cindy SantaCruz-Scalzo; thanks for all the paperwork, and making sure that I get paid and that I graduate.

I want to thank everyone I have interacted with at workshops and conferences, as these discussions have enriched my grad school experience. Among them, I would especially like to mention Suresh Venkatasubramanian, Jeremy Fineman, Ben Moseley, Cindy Phillips, Andrew McGregor, Tsvi Kopelowitz, Nodari Sitchinava, and Don Sheehy. I would also like to thank Samir Khuller for helping me navigate the US grad-school application process.

I want to thank my family back in India (Ma, Papa, Shweta di, Prateek jeeju, Saurabh bhaiyya, Anshu bhabhi, Tvisha and Avik) for being supportive of my move to this faraway land, and for their love and time-difference-immune phone calls. I want to thank my new family here (Jamie, Tom, Spencer, Max, Tanya and Arya) for welcoming me into their lives,

feeding me real food and taking me hiking along the Norwegian fjords, and to my husband Sam for being the perfect partner in work and life. I am also grateful to my host family (Kathy and Kevin) for opening their homes to international students like me each year, taking care of us, and giving us real Thanksgivings, Christmases, and BBQs.

I was lucky to be surrounded by wonderful friends who kept me sane through the stresses of grad school and forced me to stop and smell the roses. I want to give special thanks to Rishab, for always being around even after graduating, to Zoya, for being the most spunky lady in my life from Allahabad to Stony Brook, the wonderfully spirited Barnum clan (Ainara, Inigo, Julia, Cagla and Michael), the past and present fellow inmates of the Algorithms lab (Dzejla, Roozbeh, Golnaz, Jiemen, Sam, Tyler, Prashant and Rithesh) and the many friends who made grad life enjoyable (Potu, Emma, Petrina, Abhishodh, Brian, Bill, Vivek, and Amogh), and finally, the friends from my previous life, who have kept in touch and roamed with me the streets of New York city, Florence and Chandi Chowk (Indra, Ayushi, Abhilash, Pulkit, Kovid, Niti, Anvita, Ram, Maldy, and Sanjoy).

Collaboration has been an integral part of my research and I thank all my coauthors, working with whom has taught me so much: David Zage, Frédéric Vivien, Hoa Vu, Bertrand Simon, Cindy Phillips, Nguyen Kim Thang, Andrew McGregor, Sam McCauley, Tom Kroger, Rob Johnson, Mayank Goswami, Pramod Ganapathi, Martin Farach-Colton, Alex Conway, Rezaul Chowdhury, Jing Chen, Jon Berry, Michael Bender, and Eric Angel. I would like to thank the National Science Foundation and Sandia National Laboratories for funding my research, the Office for Science & Technology of the Embassy of France in the US for giving me the Chateaubriand Fellowship to do research in France, and the Max Planck Institute for Informatics Germany for funding my research visit.

Lastly, I want to thank all the awesome women in my life for serving as trailblazers, starting from my formidable mom who never let society's rigid boundaries define her, to the inspiring women in computer science who have motivated and mentored me (Cindy, Aruna, Jing, Estie, Jie, Phillipa, Ada, and Darakshan), and finally the talented women of my GWISE executive board (Anusha, Alexa, Alyssa, Aniska, Ashley, Lupe, Pamela, Sonali, Coryn and Emily), serving with whom has made my time in grad school more meaningful.

Chapter 1

Introduction

1.1 Motivation and Dissertation Overview

Modern computational devices are getting increasingly small and weak, while they are expected to perform increasingly complex tasks. As a result, most computation today is not done locally and instead is outsourced to powerful service providers, usually in exchange for money. Similarly, most of our data today is not stored in local memory but stored remotely on external servers or cloud, and accessing it is expensive.

Outsourcing computation and storage bring up several algorithmic challenges.

When computation is outsourced to untrusted external service providers, how can the weak client guarantee that the outsourced computation is performed correctly (without having to redo the computation)? Furthermore, since computation is being performed by a third party as a service, how should the client design a payment scheme for this service?

On the other hand, if the data is stored on remote servers or cloud, how can a memory-constrained client query and update this data while minimizing expensive remote accesses?

The main focus of this dissertation is to design and analyze efficient payment-based protocols for verifying the correctness of outsourced computation. We take a mechanism design approach and design *rational interactive-proof protocols* that incentivize multiple financially-driven service providers to perform the computation correctly.

Apart from the above main line of work, this dissertation also addresses an important algorithmic problem posed by storage outsourcing: if a set is stored remotely and accessing it is expensive, how can we efficiently query and update it? We design adaptive variants of a *Bloom filter*, a data structure widely used to minimize remote queries.

1.2 Rational Interactive Proofs

How can a weak client verify the correctness of computation that has been outsourced to powerful service providers without having to reexecute it? Classically this is done by asking the service providers to return along with the result, a "proof," a string of symbols certifying the correctness of the computation that the client can easily verify.

Interactive proofs (IPs). Interactive proofs are a generalization of this idea, where a weak client or *verifier* interacts with powerful service providers or *provers* to determine the truthfulness of their claim, represented as the membership of a string x in a language L. At the end of the interaction, the verifier either accepts or rejects their claim that $x \in L$. This interaction constitutes a proof with the guarantee that if the provers' claim is true then the verifier always accepts, otherwise, the verifier rejects with sufficiently high probability.

Interactive proofs are a fundamental theoretical concept that, once introduced in a security context [12, 17, 76], have become increasingly widely used as a framework to design efficient computation outsourcing protocols [23, 27, 36, 37, 39, 49, 50, 52, 77, 82, 93, 95, 124].

Incentives and Payments. Even though provers in interactive proofs are often described as *arbitrary* or *malicious*, they in fact do have a well-defined objective—given input x and language L, they want to maximize the probability that the verifier accepts the claim that $x \in L$. Thus, if $x \in L$, they are honest; otherwise they are dishonest or malicious. This incentive model of IPs originated in a security context where an adversary's goal is to prove a false statement, and verifier should reject such attempts. In a computation-outsourcing setting, where computation is traded for money in a marketplace, the objective of the service providers is not adversarial, rather it is economic. Furthermore, payments naturally exist in computation-outsourcing applications and need to be incorporated into the model.

Mechanism design and IPs. A possible approach is to incorporate payments in IPs is to give the provers \$1 if the verifier accepts, else \$0. This means that provers who maximize the verifier's acceptance probability also end up maximizing their payment. However, given that service providers are incentivized by the payment they receive, and not the probability of acceptance, this leads us to the question: *can we design more efficient interactive protocols by using sophisticated payment schemes that directly leverage correctness from the provers?*

This is a mechanism-design question—in *mechanism design* [114, 115], the objective is to design the rules of a game, played by strategic self-interested agents, so as to achieve a particular desired outcome. It is sometimes called "reverse game theory." In the context of interactive proof systems, the mechanism is the protocol along with the payments, and the goal is to design it in a way that even strategic provers trying to maximize their payment end up reporting the answer truthfully to the verifier.

In this dissertation, we answer this mechanism-design question for the case of *multiple* provers. For the single-prover case, Azar and Micali [10] introduced the model of payment-driven proofs called *rational interactive proofs* (RIP), where the prover is *rational*—wants to maximize its payment. An RIP protocol is such that if the prover's payment is maximized

then the verifier learns the correct answer. Since its introduction, many simple and efficient single-prover RIP protocols have been designed for computation outsourcing [10, 11, 80, 81].

Many computation outsourcing applications, such as, Amazon's Mechanical Turk [1], and crowdsourcing games [137], however, involve multiple service providers.

Rational proofs with multiple provers. In this dissertation, we introduce the model of rational proofs with multiple provers. There are two natural ways to generalize the single-prover RIP model to multiple provers—*cooperative rational proofs*, where the provers jointly strategize to maximize their total payment and *non-cooperative rational proofs*, where each prover wants to maximize its individual payment given others' strategies. Cooperative provers can jointly collude to mislead the verifier; the challenge in this case is to design protocols that handle such collusion. On the other hand, protocols for non-cooperative provers are more intuitive and natural; the challenge here lies in analyzing the provers' strategic interactions in the resulting game using an appropriate solution concept.

We introduce and analyze both the cooperative and non-cooperative model of rational proofs with multiple provers in this dissertation.

Cooperative rational provers. In a cooperative multi-prover rational proof (MRIP), the provers can jointly agree on a strategy at the beginning, but similar to classical multi-prover IPs, they cannot communicate once the protocol begins. Based on the verifier's randomness and the messages exchanged, the verifier computes a total payment at the end. In a MRIP protocol, a strategy of the prover maximizes their total expected payment if and only the verifier gets the correct answer. Furthermore, if a MRIP protocol has a *utility gap* of u, then the provers lose at least $1/u$ from their expected payment on misreporting the answer.

Non-cooperative rational provers. In a non-cooperative rational interactive proof (ncRIP), the provers act selfishly to maximize their individual payments, given the strategy of other provers. The game resulting from their strategic interactions is an *extensive-form game of imperfect information*. This is because the protocol proceeds in rounds and the provers cannot observe the messages exchanged between the verifier and other provers.

To analyze ncRIP protocols, we define a new solution concept, *strong sequential equilibrium* (SSE), which strengthens the requirements of *sequential equilibrium* (SE), the solution concept usually used to analyze extensive-form games with imperfect information. Unlike an SE, not every game will admit an SSE. However, as a mechanism designer, we can use such a strong solution concept as long as we can design protocols that satisfy its requirements.

We extend the notion of utility gap for non-cooperative provers, which is not as intuitive

as for cooperative provers. This is because provers may not just deviate and lie on the overall answer, they may deviate within subgames of the resulting extensive-form game.

1.2.1 Results and Contributions

Next, we summarize the results and contribution of this dissertation, besides the model of rational interactive proofs with cooperative and non-cooperative provers.

- *Exact characterizations.* We give exact complexity-theoretic characterizations of the class of languages decided by the rational cooperative and non-cooperative proof systems with different utility-gap guarantees. These characterizations are based on the class of languages decided by oracle Turing machines and show that while non-cooperative rational provers can be used to simulate adaptive oracle queries (that depend on each other), cooperative provers can only be used to simulate nonadaptive oracle queries (that can be made in parallel). Thus, non-cooperative provers are more powerful than cooperative when the adaptive queries do not reduce to nonadaptive queries. Both MRIP and ncRIP, are more powerful than all existing interactive proof models, even under constant utility gap.

- *Optimal number of provers and rounds.* We show that the full power of MRIP and ncRIP can be captured by constant provers and rounds. Furthermore, we show how to simulate any MRIP and ncRIP protocol using the optimal the number of provers and rounds.

- *Rational proofs with IP guarantees.* We formalize the relationship between the utility-gap guarantee of rational proofs and the completeness and soundness guarantees of interactive proofs. Furthermore, we show conditions on the expected payment and utility gap, under which the MRIP protocol gives the same guarantee as an MIP protocol.

- *Strong sequential equilibrium.* We prove several important properties of our new solution concept, strong sequential equilibrium, which in turn make it a solution concept of independent interest for general extensive-form mechanisms with imperfect information.

- *Log-time, constant-utility gap ncRIP for* NP. We design an ncRIP protocol for any language in the class NP with constant utility gap, where the verifier is *deterministic* and runs in $O(\log n)$ time. This result is surprising because it seems to be hard to achieve in other IP models, including MRIP. In particular, the well-known results on probabilistically checkable proofs [6, 8, 9, 83, 131] for NP have a polynomial-time verifier. Even when the verifier's running time is improved to $O(\log n)$ in probabilistically checkable proofs of proximity [19, 20, 82, 124], it comes at the cost of weakening the soundness guarantee.[1]

[1]The soundness guarantee of probabilistically checkable proofs of proximity is that the verifier rejects a given $x \notin L$ with probability that is proportional to the Hamming distance of x from L.

- *Scaled-down rational proofs.* We design MRIP and ncRIP protocols that are highly-efficient—$O(\log n)$-time verifiable—for a large class of optimization problems. Even with the verifier's power scaled down, these protocols retain strong utility-gap guarantees.

  Figure 1 summarizes our scaled-down rational proofs, their computational costs and their guarantees, for important complexity classes. These classes are also studied by Wagner [140] from whom we borrow hierarchical structure of the figure.

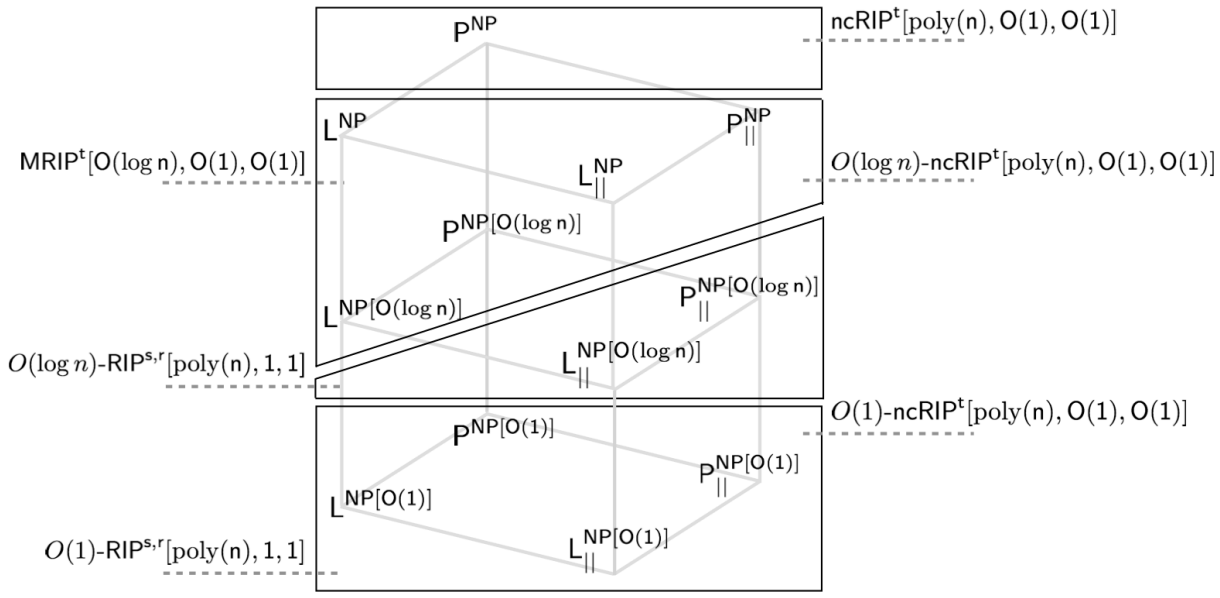

Figure 1: Rational proofs for bounded-query complexity classes studied by Wagner [140]. All classes within the black lines are equivalent. The gray dotted lines link the class of languages to their rational proof. A superscript t is a protocol with $O(\log n)$-time verifier; a superscript s, r is for a protocol with where the verifier uses $O(\log n)$ space and randomness. Utility gap is represented by the prefix, e.g., $\gamma(\mathsf{n})$-ncRIP stands for ncRIP protocols with $\gamma(n)$ utility gap. The suffix $[C(n), p(n), k(n)]$ stands for $C(n)$ communication cost, $p(n)$ provers and $k(n)$ rounds. The complexity classes are log-space (L) and polynomial-time (P) Turing machines that have access to a non-deterministic polynomial time (NP) oracle. A subscript $||$ denotes nonadaptive or parallel queries and the function $\gamma(n)$ next to NP denotes an upper bound on the number of oracle queries; see Chapter 2 for a formal review of these complexity classes. When $\gamma(n)$ is polynomial, we drop it from the utility-gap and number-of-queries notation.

1.3 Minimizing Remote Accesses: Adaptive Bloom Filters

When the input is too big to fit in a device's local memory and is stored remotely, we need algorithms that minimize expensive accesses to remote storage.

In this dissertation, we focus on a fundamental algorithmic problem in this context: the *dictionary problem*. In particular, consider a set $\mathcal{S}$ of keys, from a universe $\mathcal{U}$, such that $\mathcal{S}$ is stored remotely (e.g., on a cloud or disk). A dictionary data structure for set $\mathcal{S}$ supports membership queries (that is, "is x in $\mathcal{S}$"), inserts and deletes to $\mathcal{S}$.

Bloom filters [24, 29]—or, more generally, *approximate membership query* data structures (AMQs)—are widely used to speed up dictionaries that are stored remotely [29, 41, 46, 54, 55, 58, 59, 61, 103, 108, 111, 132, 143, 144, 147]. An AMQ maintains a compact, probabilistic representation of the set $\mathcal{S}$ and supports queries, inserts, and sometimes deletes. A query for an $x \in \mathcal{S}$ is guaranteed to return "present." A query for $x \notin \mathcal{S}$ returns "absent" with probability at least $1 - \varepsilon$, where ε is a tunable *false-positive probability*. If a query returns "present," but $x \notin \mathcal{S}$, then x is a *false positive* of the AMQ. Because AMQs have a nonzero probability of false-positives, they require far less space than explicit set representations.

An AMQ that is stored locally (e.g., in memory) can directly answer most negative queries to the dictionary. The remote dictionary only needs to be accessed when the AMQ indicates that the queried item might be present. Thus, the primary performance metric of an AMQ is how well it enables a dictionary to avoid these expensive remote accesses.

Most AMQs offer weak guarantees on the number of false positives that they will return on a given query workload. The false-positive probability guarantee of ε holds only for a single query. A sequence of queries that contain repeated false-positives can immediately drive the false-positive rate of most AMQs to 1.

In this dissertation, we show the conditions under which a space-efficient AMQ can have strong false-positives guarantees. We say that an AMQs is *adaptive* if it guarantees a false-positive probability of ε for every query, *regardless of answers to previous queries*.

First, we prove that it is impossible to build a space-efficient adaptive AMQ, even when the AMQ is immediately told whenever it returns a false positive.

We then show how to build an adaptive AMQ that partitions its state into a small local and a larger remote component. The local component is itself an AMQ, and the remote component serves as an oracle to help correct false positives. The AMQ accesses its remote state only when the remote dictionary is accessed and thus these accesses are essentially free.

Finally, the local component of our AMQ dominates existing AMQs in all regards. It uses optimal space up to lower-order terms and supports queries and updates in worst-case constant time, with high probability. Thus, we show that adaptivity has no cost.

1.4 Dissertation Outline

We present background on interactive proofs, game theory and other relevant complexity theoretic concepts in Chapter 2. Related work on classical and game-theoretic interactive proofs is also covered in Chapter 2.

In Chapter 3, we introduce the model of multi-prover prover rational proofs (MRIP) with cooperative prover and characterize the power of MRIP protocols with constant, polynomial and negligible utility gap. This chapter is an extended version of [43].

In Chapter 4, we scale down the power of the MRIP verifier to $O(\log n)$ time and fully characterize the power of such proofs under different communication costs. We also design and characterize the power MRIP protocols where the verifier uses $O(\log n)$ space and randomness with different utility gap, for example, constant, and logarithmic.

In Chapter 5, we closely compare the utility-gap guarantee of rational proofs to completeness and soundness conditions of IPs. Furthermore, we show when and how an MRIP protocol can be converted to an MIP protocol.

In Chapter 6, we introduce and formalize the model of non-cooperative rational interactive proofs (ncRIP). As part of the model, we define strong sequential equilibrium (SSE) and prove several important properties about it. Finally, we define a recursive-maximum SSE, the solution concept for ncRIP, and define utility gap. This chapter is based on [44]

In Chapter 7, we exactly characterize the power of ncRIP protocols under constant, polynomial and negligible utility gap. This chapter is also based on [44].

We scale down the verifier's running time in ncRIP protocols to $O(\log n)$ in Chapter 8 and design a protocol for NP with constant utility gap. We give a lower and upper bound for $O(\log n)$-time ncRIP protocols but there is communication-cost gap between them.

In Chapter 9, we shift our focus to storage outsourcing and design adaptive approximate membership query data structures to minimize expensive accesses to a remotely stored set. This chapter is a revised version of [21].

Chapter 2

Rational Proofs: Background and Related Work

2.1 Interactive Proofs

First introduced by Goldwasser, Micali and Rackoff [76] and in a different form by Babai and Moran [12], interactive proofs (IP) are the now most well-studied and widely-used theoretical framework to verify correctness of outsourced computation. In an interactive proof, a weak client (or *verifier*) interacts with powerful service providers (or *provers*) to determine the correctness of their claim. At the end, the verifier probabilistically accepts or rejects the claim. Interactive proofs guarantee that, roughly speaking, the verifier accepts a truthful claim with probability at least 2/3 (*completeness*) and no strategy of the provers can make the verifier accept a false claim with probability more than 1/3 (*soundness*).

Interactive proofs (IP) have been extensively studied (see, e.g., [13,14,17,68,69,75,78,79]) and precisely characterized as IP = PSPACE [105, 129]. Ben-Or et al. [17] introduced multi-prover interactive Proofs (MIP), which were later proved by Babai et al. [13] to be exactly equal to NEXP, the class of languages that can be decided by non-deterministic exponential-time Turing machines. Feige and Lovasz [64] showed that two provers and one round are sufficient to capture the full power of MIP.

The MIP characterization later led to the important area of probabilistic checkable proofs; see e.g. [6, 8, 9, 83, 131]. More recently, the study of IPs has resulted in extremely efficient (e.g., near linear or even logarithmic time) protocols for delegation of computation [23, 27, 36, 37, 39, 49, 50, 52, 77, 82, 93, 95, 124]. Such super-efficient IPs have brought theory closer to practice, resulting in "nearly-practical" systems [26, 35, 120, 126–128, 133, 134, 138, 141].

Thus interactive proofs are not only a fundamental theoretical concept but an indispensable framework to design efficient computation outsourcing protocols.

Model of interactive proofs. 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 $\vec{P} = (P_1, \ldots, P_{p(n)})$ is the vector of *provers*, and $p(n)$ a polynomial in n. The verifier runs in polynomial time and flips private coins, whereas each prover P_i is computationally unbounded. The verifier and provers know x. The verifier can communicate with each prover privately, but no two provers can communicate with each other. A *round* in the classical

model of interactive proofs consists of pair of messages, that is, messages sent by the verifier to all or some of the provers and the provers' response to those messages. Furthermore, in classical proofs the verifier always sends the first message.

In our model of rational interactive proofs, we count each round of messages (from verifier to provers or vice versa) separately and the first message is always by the provers reporting whether or not x is in the language L; see Chapter 3.2 for more details.

The length of each message and the number of rounds are polynomial in n. Each prover P_i chooses a *strategy* $s_{ij} : \{0,1\}^* \to \{0,1\}^*$ for each round j, which maps the messages it has seen up until the beginning of round j to the message it sends in round j. Given any input x, randomness r and strategy profile s, let the transcript of messages exchanged in the protocol be denoted as $(V, \vec{P})(x, r, s)$.

At the end of the communication, the verifier either *accepts* or *rejects*.

This interaction constitutes an interactive proof if it satisfies the following definition.

Definition 2.1 (Single and Multi-prover Interactive Proofs [7])**.** *A language L has an interactive proof (IP) protocol if there exists a Turning machine V such that, on any input x of length n, V runs in time polynomial in n and,*

- *(Completeness) If $x \in L$, then there exists provers $\vec{P}$ and strategy s such that* $\Pr(V$ *accepts* $(V, \vec{P})(x, r, s)) \geq 2/3$.
- *(Soundness) If $x \notin L$, then for all $\vec{P}$ and strategy s,* $\Pr(V$ *accepts* $(V, \vec{P})(x, r, s)) \leq 1/3$.

We denote by IP *the set of languages that have single-prover IP protocols. We denote by* MIP *the set of languages that have multi-prover IP protocols.*

Note that it is possible to define define the classes IP and MIP with perfect completeness: that is, V accepts with probability 1 when $x \in L$.

IP protocols used as building blocks. We now review the classic IP protocols and concepts that we will use in this dissertation.

We use the MIP protocols for the class NEXP (e.g., [13, 64]) as a blackbox in Chapter 3 and Chapter 6. These protocols proceed by first reducing a language $L \in$ NEXP to the NEXP-complete problem Oracle-3SAT, and then run an MIP protocol for Oracle-3SAT. We recall the definition of Oracle-3SAT below.

Definition 2.2 (Oracle-3SAT [13])**.** *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 binary strings w of length $r + 3s$, 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, whether there is a 3-satisfying oracle for it.*

In Chapter 4, we use the interactive protocol for NP where the verifier only uses logarithmic space and randomness given by Condon and Ladner [47] as a blackbox

While there exist rational protocols for NEXP and NP that do not need to use classical protocols as blackbox, the main purpose of using classical protocols when it comes to cooperative rational proofs is that they lead to strong (constant) utility-gap guarantees.

2.2 Review of Game Theoretic Concepts

We review game-theoretic concepts that we will build upon in this dissertation. We refer the readers to [117] for a more detailed overview.

A *game* is a formal description of strategic interaction between a set of self-interested players. A players' strategy is a complete plan of action that player will take at any point in the game. In a *cooperative* game the players work together and jointly choose their strategy. Single-prover rational proof protocols and cooperative multi-prover rational proof protocols (in Chapter 3 and Chapter 4) result in a cooperative game.

In a non-cooperative game, each player individually chooses its strategy, based on other players' strategies. Rational proof protocols with non-cooperative provers (in Chapter 6 and Chapter 8) result in a non-cooperative game.

In a *strategic game*, each player chooses its complete strategy before the game starts and the decisions of all players are simultaneous. In an *extensive-form game* or an *extensive game*, the game proceeds in rounds and each player can make a decision on what action to take at every turn in the game, based on the actions the other players have taken as far. Since interactive protocols proceed in rounds, all the protocols studied in this work result in an extensive-form game.

An extensive-form game is said to have *perfect information*, if at each turn the players are fully informed of other players' moves and the history of the game that led to their turn. If the players are not fully informed of the actions of other players and the history that led to their turn, then the game is said to *imperfect information*. Classical interactive proof and rational proof protocols form extensive-form games of imperfect information because the provers do not see the messages exchanged between the verifier and the other provers.

Finally, the players in a game have *perfect recall* if they remember the previous actions they took and *imperfect recall* otherwise. The provers in this work remember the history of messages they send and thus have perfect recall.

Equilibrium concepts for extensive-form games. To analyze non-cooperative rational proofs in Chapter 6, we need a meaningful equilibrium concept (from the mechanism designer's point of view) for extensive form games with imperfect information that result from

the interactive protocols. We introduce a new solution for such protocols in Chapter 6. Here, we review existing solution concepts such as Nash equilibrium, subgame perfect equilibrium, and sequential equilibrium and discuss why they are not suitable for our purpose.

Equilibrium selection through maximization. Given a specific equilibrium concept, there can be many equilibria such that some players' utilities are higher under an equilibrium s than under another equilibrium s', while some other players' utilities are higher under s'. Due to the players' non-cooperative nature and their lack of coordination, different players may hold different beliefs about which equilibrium will be played out by the others, and their actual strategies may mismatch—possibly failing to reach an equilibrium at all.

This is the well-known *equilibrium-selection* problem and can be resolved by requiring a *maximum equilibrium*, an equilibrium simultaneously maximizes all players' utilities among all equilibria. If a game enjoys a maximum equilibrium, it is likely that rational players will agree to play it. There are other remedies to the equilibrium-selection problem in mechanism design, such as requiring a dominant-strategy equilibrium. However, we need to balance meaningfulness and feasibility when designing complex extensive-form games such as multi-prover interactive proofs. And while many normal-form mechanisms have been successfully designed under dominant-strategy equilibrium, requiring such an equilibrium for extensive-form games in general seems difficult.

Nash equilibrium. A strategy profile s is a *Nash equilibrium* if for each player i and strategy s_i' of i, $u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i})$, where s_{-i} denotes the strategy of all players except i and $u_i(s)$ denotes the utility of player i under strategy s.

Nash equilibrium is not a suitable solution concept for extensive-form games in general because it is widely known to be susceptible to *empty threats* or *non-credible threats* [85]. An empty threat constitutes a situation where a player threatens to play a suboptimal or irrational strategy in a part of the game so as to obtain a higher utility when the game is actually played. Nash equilibria cannot rule out the occurrence of empty threats, which may lead the players to counter-intuitive or undesirable equilibria. This problem occurs even if we require a maximum Nash equilibrium.

Subgame perfect equilibrium. Subgame perfect equilibrium is used in classical game theory to overcome empty threats. A *subgame* in an extensive-form game is a subtree of the game tree starting at a particular decision node d, such that all successors of d cannot be reached from any other decision node outside of their subtree. Thus, a subgame can be isolated and played as a separate game, and a strategy profile of the original game naturally

induces a strategy profile in a subgame. A strategy profile s is a *subgame perfect equilibrium* (SPE) if it induces a Nash equilibrium in every subgame of the original game. In extensive-form games of perfect information, SPE is a powerful solution concept. Thus, under such an equilibrium, each player best-responds to the other players' strategies in every subgame, whether or not the subgame is reached.

In extensive-form games of imperfect information, however, SPE is not powerful enough, as there may be too few subgames in the game tree. At an extreme, the only subgame is the original game tree and SPE is just Nash equilibrium.

Sequential equilibrium. Sequential equilibrium is a solution concept used to analyze extensive-form games with imperfect information. At a high level, sequential equilibrium imposes the condition that provers should act rationally on every turn, however since the provers have imperfect information, it requires a belief system and consistency conditions.

A strategy profile s and belief system μ together form an *assessment*. We give the formal definition of consistency below.

Definition 2.3 (Consistency [117])**.** *An assessment (s, μ) is consistent if there is a sequence $((s^t, \mu^t))_{t=1}^{\infty}$ of assessments that converges to (s, μ) such that s^t is completely mixed and that each belief system μ^t is derived from s^t using Bayes' rule.*

The main argument against sequential equilibrium is the artificiality of the above consistency condition which requires computing the limit of sequences. In fact, to quote Kreps [100], "rather a lot of bodies are buried in this definition".

2.3 Previous Work on Game-theoretic Interactive Proofs

Next, we review game-theoretic models of interactive proofs that have been studied prior to the work presented in this dissertation.

2.3.1 Refereed Games

The model of *refereed games* [40, 63, 65–67, 98, 123] is a multi-prover game-theoretic model of interactive proofs, that has been used to characterize several complexity classes. Refereed games consist of a verifier interacting with two competing provers, one claiming that $x \in L$ and the other claiming $x \notin L$. Thus refereed games require at least one prover to be honest. The objective of the provers is to maximize the probability that the verifier accepts their claim, and thus the provers are involved in a *zero-sum* game.

The model of refereed games was first introduced by Fiege, Shamir and Tennenholtz [66]. In Chandra and Stockmeyer [40] prove that any language in PSPACE is refereeable, by a

game of perfect information. Feige and Kilian [63] show that the class of languages with single-round refereed games is exactly PSPACE, and the class of languages with polynomial-round refereed games is exactly EXP. Feigenbaum, Koller and Shor [67] study the effect of varying game theoretic properties on the class of languages recognized by two player games. They consider games of imperfect information and imperfect recall, and show that every language in $\mathsf{EXP}^{\mathsf{NP}}$ has a polynomially definable game in which both players have imperfect recall. The question whether the bound is tight is left as an open problem.

We note that imperfect recall is a very strong assumption and makes the computationally unbounded provers essentially act as oracles. By contrast, the provers in this dissertation have imperfect information and perfect recall. 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.

In Chapter 6, we show that non-cooperative rational proofs are more powerful than refereed games with competing provers (under imperfect information and perfect recall).

Non-zero sum games have also been studied in the context of complexity classes. In particular, Peterson and Reif [121] show that NEXP can be described as a game between three players, where two of them make existential moves and cannot communicate with each other, and a third makes universal moves and can communicate with the others. Simon [130] and Orponen [116] consider games between an existential oracle and a universal player, and prove their equivalence to NEXP.

2.3.2 Single-Prover Rational Proofs

Azar and Micali [10] introduced the model of *single-prover rational interactive proofs*, a payment-based interactive proof system. In a rational proof protocol, the verifier computes a payment for the prover at the end of the interaction. The prover is neither dishonest nor malicious, only *rational*—that is, the prover only acts in ways that maximize its payment.

Azar and Micali showed that adding rationality resulted in simpler, faster and more efficient proofs. In particular they give a single-round rational proof for #P. They characterize constant-round rational proofs as CH, the counting hierarchy [139], and polynomial-round rational proofs as PSPACE. They posed the problem of whether multiple rational provers are more powerful than one as an open problem. In Chapter 3, we resolve this open problem and show that multiple provers are indeed more power than one in rational proofs.

In a follow up paper [11], the authors consider the notion of *super efficient* rational proofs where the verifier runs in logarithmic time. They construct such proofs for the complexity classes Uniform TC^0 and $\mathsf{P}^{\|\mathsf{NP}}$. Their characterization of $\mathsf{P}^{\mathsf{NP}}_{\|}$ requires polynomial commu-

nication, which we improve to logarithmic using a second prover. We also note that all protocols in [11] have a polynomial utility gap (under a constant budget). We improve and extend their results on logarithmic-time verifiable multi-prover rational proofs in Chapter 4.

Guo, Hubáček, Rosen and Vald [80] extend the notion of super efficient rational proofs to *rational arguments* where the prover is computationally bounded, and give efficient rational arguments for the class of search problems computable by threshold circuits of $o(n)$ depth. We stick to the original model of [10] where the provers are computationally unbounded.

Campanelli and Rosario [34] study sequentially composable rational proofs. Zhang and Blanton [145] design protocols to outsource matrix multiplications to a rational cloud.

Different variants of the rational-proof models have also been studied. Inasawa and Kenji [91] consider rational proofs where the verifier is also rational and wants to minimize the payment to the provers.

In this dissertation, we introduce and analyze the following generalizations and variations of the model of rational proofs.

- Rational proofs with multiple cooperative provers [43], where the provers work together to maximize their total expected payment; see Chapter 3.
- Scaled-down cooperative rational proofs, where the verifier is highly efficient: runs in logarithmic time or uses logarithmic space; see Chapter 4.
- Rational proofs with multiple non-cooperative provers [44], where the provers want to selfishly maximize their own payment given others' strategies; see Chapter 6.
- Scaled-down non-cooperative rational proofs, where the verifier is highly efficient and runs in logarithmic time; see Chapter 8.

Model of single-prover rational 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 it has seen so far. Let $s = (s_1, \ldots, s_k)$ be the vector of strategies P uses in rounds $1, \ldots, k$, 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 $R(x, r, (V, P)(x, r, s))$, based on the input x, its own randomness r, and the transcript $(V, P)(x, r, s)$. P is *rational* and chooses a strategy that maximizes its utility,

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

The class RIP is defined below.

Definition 2.4 (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 $u(\tilde{s}) = \max_{\tilde{s}'} u(\tilde{s}')$, then V outputs 1 if and only if $x \in L$.*

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

Proper scoring rules. *Scoring rules* are an important tool used to design rational interactive proofs. Scoring rules let us assess the quality of a probabilistic forecast by assigning a numerical score (that is, a reward 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 its expected reward under a strictly proper scoring rule by reporting $D' = D$. For a comprehensive survey on scoring rules, see [73].

Brier's scoring rule. The *Brier's scoring rule* [28], denoted by BSR, is defined as follows: for any distribution D and $\omega \in \Sigma$,

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

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 [10, 80], in this dissertation we shall only evaluate BSR for $D \in \Delta(\{0,1\})$. Also notice that BSR has range $[-2, 0]$ and can be shifted and scaled so that the range is non-negative and bounded. In particular, we shall add 2 to the function.

In Chapter 3, we show how to design an efficient protocol for NEXP using scoring rules. In the analysis of our scoring-rule based protocol we use a property of Brier's scoring rule that, to the best of our knowledge, has not been used before. All existing uses of proper scoring rules are with respect to a fixed distribution and have the expert report the truth about that distribution. In contrast, we show how to compare the expected scores of experts across *different* distributions; see Chapter 3.3 for more details.

2.4 Review of Uniform Circuits and Complexity Classes

Next we review complexity theoretic concepts that we use in this dissertation.

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 *size* of a circuit C is the number of gates in C.

DC uniform circuits. Next we define DC uniform and DLOGTIME uniform circuits.

Definition 2.5 (DC uniform circuits)**.** *Let $\{C_n\}_{n=1}^{\infty}$ be a circuit family with gates of type AND, OR, and NOT, whose indegrees are 2, 2, and 1 respectively. We say this family is a* Direct Connect uniform (DC uniform) *family if the following queries can be answered in polynomial time in the size of the input:*

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?*

DLOGTIME *uniform circuits are defined analogously for which the above queries can be answered in logarithmic time in the size of the input.*

We use the following characterization of EXP and P in our protocols and analyses.

Lemma 2.6 ([7])**.** *The set of languages decidable by exponential-time Turing machines,* EXP*, is exactly the set of languages with DC uniform circuit families of size 2^{n^k}, where k is a constant that may depend on the language. Similarly, the set of languages decidable by polynomial-time Turing machines,* P*, is exactly the set of languages with DC uniform circuit families of size n^k, where k is a constant that may depend on the language.*

Oracle Turing machines. We frequently use oracle Turing machine which are Turing machines that have a blackbox, called the oracle, such that with the help of the oracle the Turing machine can solve certain problems in a single step.

Definition 2.7. *An oracle Turing machine M is a multi-tape Turing machine with access to an oracle O for a language L and special states $O_?, O_y$, and O_n. It runs as a normal Turing machine and whenever the head moves to state $O_?$, oracle O is consulted with a query y (on a separate query tape) and if $x \in L$, then restarted at state O_y, otherwise at state O_n.*

Queries to an oracle in an oracle Turing machine can be *adaptive* or *non adaptive*.

A set of queries are non adaptive if they can be issued by the Turing machine in parallel, that is, all queries must be decided before any one query is made. On the other hand, adaptive queries are queries where the answer to some queries may depend on the others. In this dissertation, we study the power of adaptive vs nonadaptive in the context of rational provers. In particular, we show that rational proofs with non-cooperative provers can simulate adaptive oracle queries, while those with cooperative provers cannot (unless the adaptive queries can be reduced to nonadaptive ones).

We also study complexity classes with bounded-query complexity. In particular, complexity classes where the number of oracle queries that can be issues are restricted. We prove tight characterizations of complexity classes using rational proofs, relating the query complexity of the former to the utility gap of the latter.

In this dissertation, we provide characterizations of various complexity classes defined by oracle Turing machines. We list them here.

1. **Nonadaptive complexity classes**[2]

   (a) $\mathsf{EXP}_{||}^{\mathsf{NP}}$: the class of languages decidable by an exponential-time Turing machine that can make nonadaptive queries to an NP oracle.

   (b) $\mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(n)]}$: the class of languages decidable by a polynomial-time Turing machine that can make $O(\gamma(n))$ nonadaptive queries to an NEXP oracle. When $\gamma(n)$ is polynomial in n, $\mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(n)]} = \mathsf{P}_{||}^{\mathsf{NEXP}}$.

   (c) $\mathsf{P}_{||}^{\mathsf{NP}[\gamma(n)]}$: the class of languages decidable by a polynomial-time Turing machine that can make $O(\gamma(n))$ nonadaptive queries to an NP oracle. When $\gamma(n)$ is polynomial in n, $\mathsf{P}_{||}^{\mathsf{NP}[\gamma(n)]} = \mathsf{P}_{||}^{\mathsf{NP}}$. This is a well-studied class (e.g., [31,45,92,99,106,140]) and includes important optimization problems such as maximum clique, longest paths, and variants of the traveling salesman problem even when $\gamma(n) = O(1)$.

   (d) $\mathbf{L}_{||}^{\mathsf{NP}[\gamma(n)]}$: the class of languages decidable by a logarithmic space Turing machine that can make $O(\gamma(n))$ nonadaptive queries to an NP oracle. When $\gamma(n)$ is polynomial in n, $\mathbf{L}_{||}^{\mathsf{NP}[\gamma(n)]} = \mathbf{L}_{||}^{\mathsf{NP}}$. (Wagner [140] showed that $\mathbf{L}_{||}^{\mathsf{NP}[\gamma(n)]} = \mathsf{P}_{||}^{\mathsf{NP}[\gamma(n)]}$.)

2. **Adaptive complexity classes**

   (a) $\mathsf{EXP}^{\mathsf{poly-NP}}$: the class of languages decidable by an exponential-time Turing machine that can make polynomial length adaptive queries to an NEXP oracle. (We

[2]For parallel oracle queries, notations with $||$ as subscript and as superscript are used in literature, for example $\mathsf{P}_{||}^{\mathsf{NP}}$ [140] and $\mathsf{P}^{||\mathsf{NP}}$ [11]. We follow the subscript notation in this dissertation.

note that $\mathsf{EXP}^{\mathsf{poly-NP}} = \mathsf{EXP}^{\mathsf{NP}}_{||}$). A related complexity class $\mathsf{EXP}^{\mathsf{NP}}$ is important in the the study of circuit lower bounds [142].

(b) $\mathsf{P}^{\mathsf{NEXP}[\gamma(\mathsf{n})]}$: the class of languages decidable by a polynomial-time Turing machine that can make $O(\gamma(n))$ adaptive queries to an NEXP oracle. When $\gamma(n)$ is polynomial in n, $\mathsf{P}^{\mathsf{NEXP}[\gamma(\mathsf{n})]} = \mathsf{P}^{\mathsf{NEXP}}$.

(c) $\mathsf{P}^{\mathsf{NP}[\gamma(\mathsf{n})]}$: the class of languages decidable by a polynomial-time Turing machine that can make $O(\gamma(n))$ adaptive queries to an NP oracle. When $\gamma(n)$ is polynomial in n, $\mathsf{P}^{\mathsf{NP}[\gamma(\mathsf{n})]} = \mathsf{P}^{\mathsf{NP}}$. This is a well-studied class (e.g., [31, 45, 92, 99, 106, 140]) and includes important optimization problems such as maximum clique, longest paths, and variants of the traveling salesman problem even when $\gamma(n) = O(1)$.

(d) $\mathbf{L}^{\mathsf{NP}[\gamma(\mathsf{n})]}$: the class of languages decidable by a logarithmic space Turing machine that can make $O(\gamma(n))$ adaptive queries to an NP oracle. When $\gamma(n)$ is polynomial in n, $\mathbf{L}^{\mathsf{NP}[\gamma(\mathsf{n})]} = \mathbf{L}^{\mathsf{NP}}$. (Wagner [140] showed that $\mathbf{L}^{\mathsf{NP}[\mathsf{O}(\log \mathsf{n})]}_{||} = \mathsf{P}^{\mathsf{NP}[\mathsf{O}(\log \mathsf{n})]}_{||}$.)

The query complexity of oracle Turing machines has been widely studied in the literature [15,32,140]. We also study $\mathsf{UniformTC_0}$, which is the class of constant depth, polynomial size uniform threshold circuits. It is important complexity class that includes problems such as integer division, iterated multiplication and radical summations [3, 4, 84, 89, 90].

Chapter 3

Rational Proofs with Cooperative Provers

3.1 Introduction

In Chapter 2, we reviewed the model of single-prover interactive rational proofs, introduced by Azar and Micali [10].

In this chapter, we present an extended version of our paper [43], and introduce the model of *multi-prover rational interactive proofs* (MRIP), an extension of classical multi-prover interactive proofs (MIP) and single-prover rational interactive proofs (RIP).

Many computation-outsourcing applications have ingredients of both of MIP and RIP models: the verifier pays a *team* of provers based on their responses. For example, in internet marketplaces such as *Amazon's Mechanical Turk* [1], 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 [137] correctness is often ensured by verifying one provider's answers against another [2,136]. 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 [96].

Inspired by these applications and previous theoretical work, we study rational proofs with multiple *cooperative* provers. *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 [10,11,80]. 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 [10] 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? We show that in general a protocol with multiple rational provers cannot be simulated by a single prover under standard complexity-theoretic assumptions.

Cooperative multi-prover rational proofs. In a cooperative multi-prover rational interactive proof, polynomially-many computationally-unbounded provers communicate with a polynomial-time randomized verifier, where the verifier wants to decide the membership of an input string in a language. The provers can pre-agree on how they plan to respond to the verifier's messages, but 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, its own randomness, and the messages exchanged.

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 cooperatively 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 protocols work cooperatively to maximize the total payment received from the verifier.

Any pre-specified way of distributing this payment among them 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 in a marketplace. Unbalanced divisions are allowed: for example, one prover may receive half of the total payment, while the others split the remaining evenly. We ignore the choice of division in our model, as it does not affect the provers' decisions.

Utility gap. Rational proofs assume that the provers always act to maximize their payment. However, how much do they lose by lying? If the payment loss is small, a prover may very well "get lazy" and simply return a default answer without performing any computation. Although the classic notion of rationality in game theory requires a player to always choose the best strategy to maximize its utility, the notion of bounded rationality has also been studied [48, 125].

The notion of *utility gap* measures the payment or utility loss incurred by a deviating prover. A deviating prover may (a) deviate 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. Azar and Micali [11] introduce utility gaps by demanding their protocols be robust against provers of type (a)—any deviation from the prescribed strategy results in a significant decrease in the payment. This ideal requirement on utility gaps is too strong: even the protocol in [11] fails to satisfy it [80].

We consider multi-prover rational proofs robust against provers of type (b), i.e., the provers may send some incorrect messages and only incur a small payment loss, but if they mislead the verifier to the wrong answer to the membership question of the input string, then the provers must suffer a significant loss in the payment.

We strengthen our model by considering MRIP protocols with *constant* as well as *noticeable* (i.e. *polynomial*) utility gaps, where the payment loss suffered by the provers on reporting the incorrect answer is at least $1/k$ and $1/n^k$ respectively, where k is a constant and n is the length of the input string. We say an MRIP protocol has a *negligible* (or *exponential*) utility gap if the payment loss is at least $1/2^{n^k}$. Any MRIP protocol has at least a negligible utility gap, because the rewards are generated by a polynomial-time verifier.

3.1.1 Overview of Results and Contributions

We now present our main results and discuss several interesting aspects of our model. Our characterizations of MRIP protocols are closely related to complexity classes with oracle queries. We reviewed these classes in Chapter 2.

We denote the classes of MRIP protocols with constant, polynomial and exponential utility gap as $\mathsf{O(1)\text{-}MRIP}$, $\mathsf{poly(n)\text{-}MRIP}$ and MRIP respectively. In this work, we fully characterize the computational power of all three MRIP classes exactly.

First, we show that a language has an MRIP protocol with constant utility gap if and only if it can be decided by a polynomial-time Turing machine that makes a constant number of nonadaptive queries to an NEXP oracle. That is, we prove the following.

Theorem 3.1. $\mathsf{O(1)\text{-}MRIP} = \mathsf{P}^{\mathsf{NEXP}[\mathsf{O(1)}]}_{||}$.

Thus, $\mathsf{O(1)\text{-}MRIP}$ contains both NEXP and coNEXP. That is, multi-prover rational proofs with even *constant* utility gaps are strictly more powerful than single-prover rational proofs, assuming $\mathsf{PSPACE} \neq \mathsf{NEXP}$. Furthermore, multi-prover rational proofs (even with constant utility gaps) are strictly more powerful than classical multi-prover interactive proofs, assuming $\mathsf{NEXP} \neq \mathsf{coNEXP}$. The relationship between rational and classical interactive proof systems is illustrated in Figure 2.

Next, we show that a language has an MRIP protocol with polynomial utility gap if and only if it can be decided by a polynomial-time Turing machine with nonadaptive access to an NEXP oracle. That is, we prove the following.

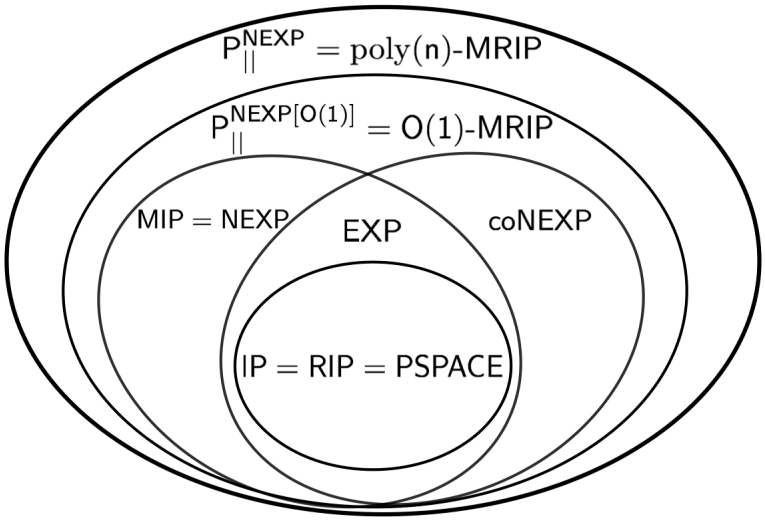

Figure 2: The computation power of MRIP vs. classical interactive proof systems. It is widely believed that PSPACE $\neq$ EXP, EXP $\neq$ NEXP, and NEXP $\neq$ coNEXP.

Theorem 3.2. $\mathsf{poly(n)\text{-}MRIP} = \mathsf{P}^{||\mathsf{NEXP}}$.

Finally, we characterize the full power of MRIP, and show that a language has an MRIP protocol (with exponential utility gap) if and only if it can be decided by an exponential-time Turing machine with nonadaptive access to an NP oracle. That is, we prove the following.

Theorem 3.3. $\mathsf{MRIP} = \mathsf{EXP}^{||\mathsf{NP}}$.

We give MRIP protocols for NEXP, which are used as a building block in our proofs. To prove Theorem 3.1 and Theorem 3.2, we establish a general reduction between the utility gap of MRIP protocols and the query complexity of oracle Turing machines.

Finally, to prove Theorem 3.3, we introduce another complexity class as an intermediate step, and use its circuit characterization to construct the corresponding MRIP protocol. Similar circuit based characterization is also used by Azar and Micali in [11], but their technique results in an exponential blow-up in the number of messages when applied directly to our case. We use multiple provers to avoid this communication blow up; see Chapter 3.5.

Optimal number of provers and rounds. While we allow polynomially-many provers and rounds in MRIP, how many provers and rounds are really needed to capture the full power of the system? In computation outsourcing applications, protocols requiring few provers and rounds are desirable, as it may be hard for the verifier to recruit a large number of provers or to retain the provers for a long period of time to execute many rounds.

Under the classic model of interactive proofs, it is well known that any MIP protocol can be simulated using only two provers and one round of communication between the provers and the verifier [64]. In this chapter, we prove analogous results for all three of our MRIP classes—we show that two provers and three rounds are sufficient to capture their full power.

Specifically, any MRIP protocol using polynomially-many provers and polynomially-many rounds that has a constant, polynomial, or exponential utility gap can be simulated by a 2-prover 3-round MRIP protocol that retains their respective utility gap.

It is worth pointing out that we count the number of rounds in a protocol differently from classic IP and MIP protocols. In the classic protocols, the number of rounds is the number of *pairs* of back-and-forth interactions (see, e.g., [64]); while in our protocols it is the total number of interactions—that is, the provers' messages and the verifier's messages are considered as different rounds. An odd number of rounds is an intrinsic property of multi-prover rational proofs, as an MRIP protocol by default starts with the provers reporting the *answer bit* to the verifier. Thus, the 3-round protocols consist of the first "answer bit round", followed by a single back-and-forth exchange corresponding to a single round in IP or MIP. Indeed, any non-trivial MRIP protocol—that is, any MRIP protocol that cannot be simulated by a single prover—requires at least three rounds. Thus, three rounds are optimal.

Finally, we note that the power of MRIP protocols remains the same even when it is restricted to constant number of rounds, while the power of RIP protocols decreases. In particular, Azar and Micali [10] show that the class of languages having constant-round single-prover rational proofs is exactly the counting hierarchy, while $\mathsf{RIP} = \mathsf{PSPACE}$. This difference between MRIP and RIP is analogous to the difference between MIP and IP.

3.2 Multi-Prover Rational Interactive Proofs (MRIP)

In this section, we first define multi-prover rational interactive proofs (MRIP) in general, and then strengthen the model by imposing proper utility gap.

Notation and definitions. The interactive model of multi-prover rational interactive proofs is similar to interactive proofs described in Chapter 2, except for the following differences. In our model of rational interactive proofs, a *round* of interaction consists of either messages sent in parallel by all or some provers to the verifier or messages sent by the verifier to all or some provers, 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 $c = 0$).

In general, the length of each message and the number of rounds are polynomial in n. Let $k(n)$ be the number of rounds and r be the random string used by V. For each

$j \in \{1, 2, \ldots, k(n)\}$, 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. By definition, $\vec{m}$ is a random variable depending on r.

At the end of the communication, the verifier does not accept or reject, but instead computes the *total payment* of the provers, denoted by a payment function R on x, r, and $\vec{m}$. We restrict $R(x, r, \vec{m}) \in [-1, 1]$ for convenience. Of course, the payment can be shifted so that it is non-negative—that is, the provers do not lose money. We use both positive and negative payments to better reflect the intuition behind our protocols: the former are rewards while the latter are punishments. 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. This requirement for the verifier does not change the set of languages that have multi-prover rational interactive proofs; however, it simplifies our analysis of utility gap (the payment loss incurred by provers that report the 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 it has seen up until the beginning of round j to the message it sends in round j. Note that P_i does not send any message when j is even; in this case s_{ij} can be treated as a constant function. Let $s_i = (s_{i1}, \ldots, s_{ik(n)})$ be the strategy vector of P_i and $s = (s_1, \ldots, s_{p(n)})$ be the strategy profile of the provers. Given any input x, randomness r and strategy profile s, we may write the vector $\vec{m}$ of messages exchanged in the protocol more explicitly as $(V, \vec{P})(x, r, s)$.

The provers are *cooperative* and jointly act to maximize the total expected payment received from the verifier. Note that this is equivalent to each prover maximizing its own expected payment when each P_i receives a pre-specified fraction f_i of the payment, where $\sum_{i=1}^{p(n)} f_i = 1$ and f_i may depend on x but not on r and $\vec{m}$.

Before the protocol starts, the provers pre-agree on a strategy profile s that maximizes

$$u_{(V,\vec{P})}(s, x) \triangleq \mathop{\mathbb{E}}_{r}\left[R\left(x, r, (V, \vec{P})(x, r, s)\right)\right].$$

When $(V, \vec{P})$ and x are clear from the context, we write $u(s)$ for $u_{(V,\vec{P})}(s, x)$. We define multi-prover rational interactive proofs as follows.

Definition 3.4 (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 s of the provers such that $u(s) = \max_{s'} u(s')$, $c = 1$ if and only if $x \in L$. We denote the class of languages that have MRIP protocols by* MRIP*.*

This definition immediately leads to the following property.

Lemma 3.5. MRIP *is closed under complement.*

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

For any input string x of length n, the protocol $(V, \vec{P})$ works as follows. Initially $R_n = 0$.

1. P_1 sends m'_{11}. V' flips the first bit. Denote the new message by m_{11}.
2. V' runs V to compute the messages it should send in each round, except that m'_{11} is replaced by m_{11} in the input to V. Let $\vec{m}'$ be the vector of messages exchanged between V' and $\vec{P}$.
3. 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 m'_{11} replaced by m_{11}.
4. V' outputs the first bit sent by P_1.

Figure 3: An MRIP protocol for $\overline{L}$ using MRIP protocol for L.

We now show that the protocol in Figure 3 is an MRIP protocol for $\overline{L}$. For each strategy profile s of the provers in $(V, \vec{P})$, consider the following strategy profile s' in $(V', \vec{P})$.

1. $s'_i = s_i$ for each $i \neq 1$.
2. In round 1, s'_1 outputs the same message as s_1, except that the first bit is flipped.
3. For any odd $j > 1$ and any transcript m'_1 for P_1 at the beginning of round j, $s'_1(m'_1)$ is the same as $s_1(m_1)$, where m_1 is m'_1 with the first bit flipped.

By induction, for any x and r, $(V', \vec{P})(x, r, s')$ is the same as $(V, \vec{P})(x, r, s)$ except the first bit. Thus $R'(x, r, (V', \vec{P})(x, r, s')) = R(x, r, (V, \vec{P})(x, r, s))$, which implies $u_{(V', \vec{P})}(s', x) = u_{(V, \vec{P})}(s, x)$. Since the mapping from s to s' is a bijection, if we arbitrarily fix a strategy profile s' that maximizes $u_{(V', \vec{P})}(s', x)$, the corresponding strategy profile s maximizes $u_{(V, \vec{P})}(s, x)$. By definition, $x \in L$ if and only if the first bit sent by s_1 is 1; thus, $x \in \overline{L}$ if and only if the first bit sent by s'_1 is 1. Therefore $(V', \vec{P})$ is an MRIP protocol for $\overline{L}$. □

Note that the MRIP protocols for $\overline{L}$ and L have the same number of provers and the same number of rounds. Moreover, the class of languages having classical multi-prover interactive proofs is not closed under complement (assuming $\mathsf{NEXP} \neq \mathsf{coNEXP}$). This is a key distinction between multi-prover rational proofs and multi-prover interactive proofs.

3.2.1 MRIP with Utility Gap

The model of rational proofs and the MRIP model so far assumes the provers are sensitive to arbitrarily small losses in the payment. That is, the provers choose s to just maximize their expected payment—the amount they lose if they use a suboptimal strategy is irrelevant.

However, if the payment loss is small, a prover may very well "get lazy" and simply return a default answer without performing any computation. Although the classic notion of rationality in game theory requires a player to always choose the best strategy to maximize its utility, the notion of bounded rationality has also been studied [48, 125].

The notion of *utility gap* measures the payment or utility loss incurred by a deviating prover. A deviating prover may (a) deviate slightly from the honest protocol but still lead the verifier to the correct answer or (b) deviate and mislead the verifier to an incorrect answer.

In [11], Azar and Micali introduce utility gap by demanding their protocols be robust against provers of type (a)—any deviation from the prescribed strategy results in a non-negligible loss in the payment. Formally, let s be an optimal strategy and s' a suboptimal strategy of the prover P. Then the *ideal* utility gap requires that $u(s) - u(s') > 1/\gamma(n)$, where $\gamma(n)$ is constant or polynomial in n. Although an ideal utility gap strongly guarantees that the prover uses its optimal strategy, such a utility gap appears to be too strong to hold for many meaningful protocols, even the ones in [11].

In [80], Guo et al. define a weaker notion of utility gap and impose it on rational arguments rather than rational proofs. They require that a noticeable deviation leads to a noticeable loss: if under a strategy 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 s' is also noticeably smaller than the optimal expected payment.

Our notion of utility gap is slightly different and we require our protocols to be robust against provers of type (b), i.e., the provers may send some incorrect messages and only incur a small payment loss, but if they mislead the verifier to the wrong answer to the membership question of the input string, then the provers must suffer a significant loss in the payment.

Definition 3.6 (Utility Gap)**.** *Let L be a language in* MRIP*, $(V, \vec{P})$ an MRIP protocol for L, and $\gamma(n) \geq 0$. We say that $(V, \vec{P})$ has an* $\gamma(n)$-utility gap *if for any input x with $|x| = n$, any strategy profile s of $\vec{P}$ that maximizes the expected payment, and any other strategy profile s', where the answer bit c' under s' does not match the answer bit c under s, i.e., $c' \neq c$, then*

$$u(s) - u(s') > \frac{1}{\gamma(n)}.$$

We denote the class of languages that have an MRIP protocol with constant utility gap

by O(1)-MRIP, and the class of languages that have an MRIP protocol with polynomial (or noticeable) utility gap by poly(n)-MRIP. Specifically, poly(n)-MRIP is the union of MRIP classes with $\gamma(n)$ utility gap, where $\gamma(n)$ is a polynomial in n. O(1)-MRIP is defined analogously. We also use the notation γ(n)-MRIP to denote the class of languages which have ncRIP protocol with utility gap $1/O(\gamma)$, where $\gamma(n)$ is any polynomial-time computable function (given 1^n) that is polynomially bounded, e.g., $\gamma(n)$ can be a constant, $\log n$, or $\sqrt{n}$.

Relationship between utility gap and budget. The *budget* is the total expected payment that a verifier can give in a protocol.

Utility gap and budget are closely related. To study utility gap consistently, we maintain a fixed $O(1)$ budget.[3] This is because utility gap scales naturally with the payment—a polynomial utility gap under a constant budget is the same as a constant utility gap under a sufficiently-large polynomial budget.

Following Definition 3.6, it is not hard to see that the MRIP protocol for $\overline{L}$ in the proof of Lemma 3.5 has the same utility gap as the one for L. Thus, we have the following.

Corollary 3.7. O(1)-MRIP *and* poly(n)-MRIP *are both closed under complement.*

3.3 MRIP Protocols for NEXP

To demonstrate the power of multi-prover rational proofs, we start by constructing two different MRIP protocols for NEXP, the class of languages decidable by exponential-time non-deterministic Turing machines.

Constant utility gap MRIP protocol for NEXP using MIP. First, we show that O(1)-MRIP contains NEXP. We construct the desired MRIP protocol using a MIP protocol as a blackbox. Existing MIP protocols (see, e.g., [13, 64]) for a language $L \in$ NEXP first reduce L to the NEXP-complete problem Oracle-3SAT, and then run a MIP protocol for Oracle-3SAT. See Chapter 2 for the definition of Oracle-3SAT.

Lemma 3.8. *Any language $L \in$ NEXP has an MRIP protocol that uses two provers, three rounds and has with constant utility gap.*

Proof. The desired MRIP protocol $(V, \vec{P})$ is defined in Figure 4.

The 2-prover 3-round MRIP protocol is obtained by running the MIP protocol in [64]. Without loss of generality, let the MIP protocol have completeness 1 and soundness 1/3. That

[3] In contrast, Azar and Micali [11] maintain a polynomial-size budget.

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

1. P_1 sends a bit $c \in \{0, 1\}$ 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 a MIP protocol for proving $x \in L$. If the verifier accepts then $R = 1$; else, $R = 0$.

Figure 4: A simple MRIP protocol for NEXP.

is, the verifier accepts every $x \in L$ with probability 1, and every $x \notin L$ with probability at most $1/3$. We show that V outputs 1 if and only if $x \in L$.

For any $x \in L$, if the provers send $c = 1$ and execute the MIP protocol with V, then the payment is $R = 1$ because V accepts with probability 1.[4] If they send $c = 0$, then the payment is $R = 1/2 < 1$.

For any $x \notin L$, if the provers send $c = 1$ and run the MIP protocol, then the probability that V accepts is at most $1/3$ and the expected payment is at most $1/3$. If they send $c = 0$, then the payment is $1/2 > 1/3$.

Thus, V outputs 1 iff $x \in L$, and $(V, \vec{P})$ is an MRIP protocol for L. Since the provers' payment loss when sending the wrong answer bit is at least $1/6$, the utility gap is $O(1)$. □

Combining Corollary 3.7 and Lemma 3.8, we have the following.

Corollary 3.9. *Any language $L \in$ coNEXP has an MRIP protocol that uses two provers, three rounds and has with constant utility gap.*

Remarks. Three rounds of interaction is the best possible for any non-trivial MRIP protocol with at least two provers, because P_1 always sends the answer c in the first round. In particular, if the protocol has only two rounds, then the last round consists of the verifier sending messages to the provers and can be eliminated. A single-round MRIP protocol degenerates into a single-prover rational protocol, as the provers can pre-agree on the messages.

The constant utility gap in our MRIP protocol comes from the constant soundness gap of classical MIP protocols—that is, the gap between the accepting probability for $x \in L$ and $x \notin L$. Using the same construction, any classical interactive proof protocol can be converted into an MRIP protocol where the utility gap is a constant fraction of the soundness gap.

[4] 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 $2/3$. This does not affect the correctness of our MRIP protocol.

MRIP protocol for NEXP using scoring rules. We now construct an MRIP protocol for any language in NEXP without relying on MIP protocols. Instead, we use a *proper scoring rule* to compute the payment for the provers, so as to incentivize them to report the correct answer. However, the way we use the scoring rule is highly non-standard and differs from previous uses of scoring rules (including those in rational proofs [10, 11, 80]). For a review of proper scoring rules, see Chapter 2.3.2.

We construct a simple and efficient MRIP protocol for Oracle-3SAT. As in classical MIP protocols, an MRIP protocol for any language $L \in \mathsf{NEXP}$ can be obtained by first reducing L to Oracle-3SAT and then using our protocol. As our protocol is highly efficiently, the complexity of the overall protocol for L is the same as the reduction. Our protocol for Oracle-3SAT is defined in Figure 5, and we have the following lemma.

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

1. P_1 sends $c \in \{0,1\}$ and $a \in \{0,1,\ldots,2^{r+3s}\}$ to V. V outputs c at the end.
2. If $c = 1$ and $a < 2^{r+3s}$, or if $c = 0$ and $a = 2^{r+3s}$, the protocol ends, and $R = -1$.
3. Otherwise, V uniformly and 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 $k \in \{1, 2, \ldots, 6\}$.
   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 $i \in \{1, 2, \ldots, 6\}$, 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 BSR. If $b = 1$, $R = \frac{2p_1 - (p_1^2 + p_0^2) + 1}{11}$, else $R = \frac{2p_0 - (p_1^2 + p_0^2) + 1}{11}$.

Figure 5: A simple and efficient MRIP protocol for Oracle-3SAT.

Lemma 3.10. Oracle-3SAT *has a 2-prover 3-round MRIP protocol 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 $O(n)$. Moreover, the evaluation of the payment function consists of constant number of arithmetic operations over $O(n)$-bit numbers.*

Proof. 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 $(r+3s)$-bit strings for B. Denote this number by a^*. If $B \in$ Oracle-3SAT then $a^* = 2^{r+3s}$, otherwise $a^* < 2^{r+3s}$.

Roughly speaking, in our MRIP protocol in Figure 5, the verifier incentivizes the provers to report the correct value of a^*, so that the membership of B can be decided. To see why this is the case, let s^* be one of the best strategy profiles of the provers. Then s^* must satisfy

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

Otherwise, the provers' expected payment is -1. Meanwhile, by sending $c = 0$ and $a = 0$ in Step 1 and all 0's in Step 4, their expected payment is 0.

We consider which of the two cases in Equation 1 the provers will report. Note 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 $i \in \{1, \ldots, 6\}$ 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. 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 nonnegative. Thus, the best strategy profile s^* is such that, for any c, a and the oracle committed by P_2, P_1's answers for any $(b_1, ..., b_6)$ is consistent with A'. Thus, under s^*, the payment is never computed in Step 5a.

Whether or not B evaluates to 0 in Step 5b is determined solely by b_1, b_2, b_3 and A'. 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 $(r+3s)$-bit strings for B under oracle A'; the realized value is $b = B(z', b_4, b_5, b_6, A(b_4), A(b_5), A(b_6))$; and the reported distribution is (p_1, p_0). 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 the expected payment by reporting $a = a'$, which implies $(p_1, p_0) = (D(1), D(0))$.

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

If $B \in$ Oracle-3SAT, which is the more interesting part, we show that under s^* prover P_2

commits to the desired 3-satisfying oracle A^* (so that $a' = 2^{r+3s}$ and $D(1) = 1$). Let $\mathsf{BSR(D)}$ denote the expected score for reporting D under BSR, when D is the true distribution.

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

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 $D(1) = 0$. Note 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 $D(1) = 0$. Therefore, 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 $a' = 0$, then the provers can only maximize their expected payment by committing to the 3-satisfying oracle A^* (thus $a' = 1$), under which Step 5c is reached with probability 1. By Equations 1 and $a' = a$, we have $c = 1$ and $a = 2^{r+3s}$.

If there are 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'. To do so, we use w along with Step 5b. In particular, committing to any oracle other than A^* or A' 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. Moreover, 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. The strategy profile s^* must be such that P_2 commits to A^* and P_1 sends $a = 2^{r+3s}$ and $c = 1$, as desired. If there are multiple 3-satisfying oracles for B, then the provers can pre-agree on any one of them (e.g., the lexicographically first one).

Thus, $(V, \vec{P})$ is an MRIP protocol for $\mathsf{Oracle\text{-}3SAT}$. Since $n \geq r + 3s + 3$, the number of coins flipped by V for sampling w, w', and k is $O(n)$, and so the number of bits exchanged is also $O(n)$. Moreover, given an input $w = (z, b_1, b_2, b_3)$ for B and the 3-bit answers of the oracle for b_1, b_2, b_3, B can be evaluated in $O(n)$ time. Thus V's running time is $O(n)$ plus a constant number of arithmetic operations to compute the payment in Step 5c. □

Remarks. There is a tradeoff between the utility gap and the computational efficiency in the two MRIP protocols we have constructed for NEXP. The protocol in Figure 4 has constant utility gap but relies on the MIP protocol, which has high (even though polynomial) communication and computation overheads beyond the reduction to Oracle-3SAT. On the other hand, the protocol in Figure 5 is very efficient, with just linear computation and communication overheads beyond the reduction to Oracle-3SAT, but has exponential utility gap. It would be interesting to see if there exists an MRIP protocol for NEXP that has constant or noticeable utility gap and is highly efficient (e.g., with linear overhead beyond the reduction to Oracle-3SAT).

To the best of our knowledge, our use of the property of BSR in Equation 2 is new. Existing uses of proper scoring rules have the expert report the truth about a fixed distribution. In contrast, our use of scoring rules compares the expected scores across *different* distributions: by committing to different oracles, the expert can choose which distribution is the true distribution, and can tell the truth about that distribution to maximize its corresponding score. The correctness of our protocol depends on the expert committing to the distribution with the highest score under truth-telling.

3.4 Constant and Noticeable Utility Gap

We have shown in Chapter 3.3 that the class of MRIP protocols with constant utility gaps contains both NEXP and coNEXP, making them more powerful than classic MIP protocols. In this section, we prove Theorem 3.1 and Theorem 3.2.

To do so, recall that $\gamma(n)$ is a function of n, which (1) only takes positive integral values, (2) is upper-bounded by a polynomial in n, and (3) is polynomial-time computable.[5] We refer to the class of languages that have an MRIP protocol with $O(\gamma(n))$ utility gaps as $\gamma(\mathsf{n})$-MRIP. Recall that $\mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(\mathsf{n})]}$ is the class of languages decidable by polynomial-time Turing machines making $O(\gamma(n))$ nonadaptive queries to an NEXP oracle. We prove tight upper- and lower-bounds on the power of the class $\gamma(\mathsf{n})$-MRIP.

Lemma 3.11. $\mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(\mathsf{n})]} \subseteq \gamma(\mathsf{n})\text{-MRIP}$.

Proof. Consider any language $L \in \mathsf{P}^{||\mathsf{NEXP}[\gamma(\mathsf{n})]}$. Let M be a polynomial-time Turing machine deciding L, with access to an oracle O for an NEXP language. Without loss of generality, M makes exactly $\gamma(n) \geq 1$ nonadaptive queries to O. The MRIP protocol for L uses our MRIP protocol for NEXP to simulate the oracle, as in Figure 6.

[5]For Theorem 3.1 and Theorem 3.2, we only need $\gamma(n)$ to be constant or polynomial in n. However, the lemmas in this section hold for all $\gamma(n)$'s that are polynomial-time computable (given 1^n) and polynomially bounded. That is, $\gamma(n)$ can be $\log n$, $\sqrt{n}$, etc.

For any input string x of length n, the protocol $(V, \vec{P})$ works as follows. Initially $R_n = 0$.

1. P_1 sends a bit $c \in \{0,1\}$ to V. V outputs c at the end of the protocol.
2. V simulates M on x till M outputs $\gamma(n)$ queries for O, denoted by $q_1, \ldots, q_{\gamma(n)}$.
3. To answer M's oracle queries, for each $i \in \{1, 2, \ldots, \gamma(n)\}$, V does the following:
   (a) V first reduces q_i to an $\mathsf{Oracle\text{-}3SAT}$ instance ϕ_i.
   (b) V sends ϕ_i to P_1 and P_2 and executes the MRIP protocol for NEXP in Figure 4. Let c_i^* and R_i^* be the answer bit and the payment in that protocol respectively. V returns c_i^* as the oracle's answer for q_i, and updates the sum $R_n \leftarrow R_n + R_i^*$.
4. V continues simulating M till the end. If c does not match M's output, then the protocol ends with reward $R = -1$; otherwise the protocol ends with $R = R_n/\gamma(n)$.

Figure 6: An MRIP protocol for $\mathsf{P}^{||\mathsf{NEXP}[\gamma(n)]}$.

To see why this protocol works, first note that reporting the correct answer bit c and answering all $\gamma(n)$ NEXP queries $q_1, \ldots, q_{\gamma(n)}$ correctly leads to a reward $R \geq 1/2$ for the provers. In particular, according to our protocol in Figure 4 and the proof of Lemma 7.7, if the provers use the optimal strategy for each query q_i (which includes sending the correct answer bit c_i^*), the provers get $R_i^* = 1$ if $\phi_i \in \mathsf{Oracle\text{-}3SAT}$ and $R_i^* = 1/2$ if $\phi_i \notin \mathsf{Oracle\text{-}3SAT}$.

Now, suppose the provers report an incorrect answer bit $c' \neq c$ at the beginning. Then, either (a) the output of M in Step 4 does not match c', and thus $R = -1$; or (b) there exists an NEXP query q_i such that the answer bit c_i^* in Step 3b is incorrect.

In case (a), the provers' expected payment loss is at least $1/2 + 1 = 3/2 > 1/\gamma(n)$, as $\gamma(n) \geq 1$. In case (b), since the protocol in Figure 4 has $O(1)$ utility gap, the provers' expected payment loss in the overall protocol is $1/O(\gamma(n))$. Thus, the provers' optimal strategy is to report the correct answer bit c. □

Next, we prove a tight upper-bound for $\gamma(\mathsf{n})$-MRIP.

Lemma 3.12. $\gamma(\mathsf{n})\text{-}\mathsf{MRIP} \subseteq \mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(n)]}$.

Proof. Given any $L \in \gamma(\mathsf{n})$-MRIP, let $(V, \vec{P})$ be the MRIP protocol with $O(\gamma(n))$ utility gap for L. Again without loss of generality, assume the utility gap is exactly $\gamma(n)$. To prove Lemma 3.12, we simulate $(V, \vec{P})$ using a $\mathsf{P}^{||\mathsf{NEXP}[\gamma(n)]}$ Turing machine.

Consider the following deterministic oracle Turing machine M. Given an input x of length n, M divides $[-1, 1]$ into $4\gamma(n)$ intervals, each of length $1/(2\gamma(n))$. That is, the ith interval is $[i/2\gamma(n), (i+1)/2\gamma(n))$ for each $i \in \{-2\gamma(n), \ldots, 2\gamma(n) - 1\}$.[6] For each *interval*

[6] To include 1, interval $2\gamma(n) - 1$ should be closed on both sides; we ignore this for simplicity.

i, that is, $[i/2\gamma(n), (i+1)/2\gamma(n))$, M makes the following queries to an NEXP oracle:

1. Does there exist a strategy profile s with expected payment $u_{(V,\vec{P})}(s,x)$ in interval i?
2. Does there exist a strategy profile s with expected payment $u_{(V,\vec{P})}(s,x)$ in interval i and corresponding answer bit $c=1$?

Note that M makes $O(\gamma(n))$ nonadaptive queries, each of polynomial size: indeed, M only needs to specify x, the value i and the query index. Some of these queries may turn out to be unnecessary in the end, but they are made anyway so as to preserve non-adaptivity.

We now show that the queries made by M can be answered by an NEXP oracle. Recall that in an MRIP protocol, a strategy s_{jk} of each prover P_j for each round k is a function mapping the transcript P_j has seen at the beginning of round k to the message it sends in that round. Since the protocol has polynomially many provers and polynomially many rounds, a strategy profile s consists of polynomially many functions from $\{0,1\}^*$ to $\{0,1\}^*$, and for each function, both the input length and the output length are polynomial in n. Thus it takes at most exponentially many bits to specify a strategy profile: if the input length is at most $p(n)$ and the output length is at most $q(n)$, then $2^{p(n)}q(n)$ bits are sufficient to specify the truth table of a function.

Thus, an NEXP machine can non-deterministically choose a strategy profile s. It then goes through all possible realizations of V's random string and, for each realization, simulates $(V,\vec{P})$ on input x using s, to compute the reward R. Finally, the NEXP machine computes the expected payment $u(s,x)$, checks if $u(s,x)$ is in interval i (and if $c=1$ for query 2), and accepts or rejects accordingly. It is easy to see that if the desired strategy profile s exists then this machine accepts s; otherwise it always rejects.

Since V runs in polynomial time and flips polynomial coins, this machine runs in non-deterministic exponential time, and M's queries can be answered by an NEXP oracle.

Finally, given the oracle's answers to its queries, M finds the highest index i^* such that interval i^* is "non-empty": that is, the oracle has answered 1 for query 1 for this interval. M accepts if the oracle's answer to query 2 for this interval is 1, and rejects otherwise. It is clear that M runs in polynomial time.

The only thing left to show is that M decides L given correct answers to its oracle queries. By definition, for the best strategy profile s^* of the provers in $(V,\vec{P})$ for x, $u(s^*,x)$ falls into interval i^*. Because $(V,\vec{P})$ has $\gamma(n)$ utility gap and each interval is of length $1/(2\gamma(n))$, by Definition 3.6, all strategy profiles whose expected payments are in interval i^* must have the same answer bit c as that in s^*. By the definition of MRIP protocols, $x \in L$ if and only if $c=1$, which occurs if and only if the oracle's answer to query 2 for interval i^* is 1. Thus M decides L and Lemma 3.12 holds. □

Theorem 3.1 and Theorem 3.2 follow from Lemma 3.11 and Lemma 3.12.

3.5 Characterizing the Full Power of MRIP

In this section, we prove Theorem 3.3, that is, we show that a language has an MRIP protocol (with exponential utility gap) if and only if it can be decided by an exponential-time Turing machine with nonadaptive access to an NP oracle.

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 3.3 by showing $\mathsf{EXP}_{||}^{\mathsf{NP}} = \mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$.

Definition 3.13. $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$ *is the class of languages decidable by an exponential-time Turing machine with nonadaptive 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 start by proving the following lower bound.

Lemma 3.14. $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}} \subseteq \mathsf{MRIP}$.

Proof. By Lemma 2.6, 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. We call a gate $i' \in \{1, 2, ..., g\}$ of C_n an *input gate of gate* i if there is a directed wire from i' to i. For any input string x of length n and any gate i 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 $i \in \{1, 2, ..., n\}$. The 2-prover 5-round MRIP protocol $(V, \vec{P})$ for L is given in Figure 7.

To see why it is an MRIP protocol, notice that if P_1 and P_2 send the correct c and always answer V's queries correctly according to C_n, then the payment to them is always $R = 1$, irrespective of V's coin flips. Thus the expected payment is 1. Below we show that any other strategy profile makes the 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 by DC uniformity the verification in Step 6a will fail, giving the provers a payment $R = 0$. Indeed, to verify whether i_1 and i_2 are the input gates of i, it suffices to verify whether h_1 and h_2 are both the input wires of i and the output wires of i_1 and i_2: this is why V queries P_1 about i's input wires. Accordingly, if such a gate i exists then the expected payment to the provers will be 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 Step 6d will fail conditioned on gate i being chosen, and the expected payment to the provers is at most

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

1. P_1 sends one bit $c \in \{0,1\}$ to V. V 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 and input wires of i, and
   (c) the values of gate i and its input gates.
3. P_1 sends to V: type $t_i \in \{$AND, OR, NOT, INPUT$\}$; 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. V computes the payment R by verifying the following statements:
   (a) t_i is the correct type of i and the set of input gates of i is correct using DC uniformity;
   (b) if $i \in \{1, 2, ..., n\}$ (that is, an input gate of the circuit), then $v_i(x) = x_i$;
   (c) if $i = g$ (that is, the output gate of the circuit), then $v_i(x) = c$;
   (d) if $t_i \in \{$AND, OR, NOT$\}$, $v_i(x)$ follows the logic based on t_i and i's inputs.
   (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 7: An MRIP protocol for EXP.

$1 - 1/g < 1$. Moreover, if there exists an input gate i such that $v_i(x) \neq x_i$, or if $v_g(x) \neq c$, then conditioned on gate i being chosen, the expected payment is again at most $1 - 1/g < 1$.

Next, as in the proof of Lemma 3.10, 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 each 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, Step 6e will fail with probability 1/3. Since i is chosen with probability $1/g$, the expected payment will be at most $1 - \frac{1}{3g} < 1$.

Thus, the only strategy profile $\tilde{s}$ with an expected payment equal to 1 is the following:
1. P_1 and P_2 report 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. $A(g) = c$ for the output gate, and
4. for any other gate i, $A(i)$ is computed correctly based on i's type and 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 $x \in L$. □

Lower bound for MRIP. Using the protocol in Figure 7 as a building block, we are now ready to prove Lemma 3.14.

We start by creating some circuit structures for the class $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$. For any language $L \in \mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$, let M be an exponential-time oracle Turing machine that decides L using an oracle O. Without loss of generality, assume O is for Oracle-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. By the definition of $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$, $q(n)$ can be exponential in n, while $p(n)$ is polynomial. Without loss of generality, $p(n) \geq 5$. Let $\ell(n) = p(n)q(n)$. When n is clear from context, we refer to $\ell(n)$, $p(n)$ and $q(n)$ as ℓ, p and q respectively.

Since the oracle queries are nonadaptive, 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. As in Lemma 2.6, 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. Without loss of generality, the gates of C_n can be partitioned into q sets, one for each oracle query, such that the output of a gate only affects the value of the corresponding query. This can be done by duplicating each gate at most exponential times. The resulting circuit family is still DC uniform. Also without loss of generality, the oracle queries are all different. This can be done by including the index $i \in \{1, \ldots, q\}$ in the ith query.

Given the vector of oracle answers corresponding to the q queries of M, $b \in \{0, 1\}^q$, the membership of x can be decided in time exponential in n. 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, 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. 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. That is, the Turing machine that answers questions SIZE, INPUT, OUTPUT, TYPE for C'_n runs in time polynomial in n rather than $n+q$.

Given the two circuit families defined above, the membership of x in L can be computed by the following three-level "circuit:" besides the usual AND, OR, NOT gates, it has q "NEXP" gates, which have a p-bit input and 1-bit output, simulating the Oracle-3SAT oracle.

- Level 1: The circuit C_n for 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 to our naming convention 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 $\{n+1, n+2, ..., n+\ell\}$. The input and the output gates correspond to x and $(\phi_1, \phi_2, ..., \phi_q)$ in the natural order.
- Level 2: We have q NEXP gates, without loss of generality denoted by $g+1, g+2, ..., g+q$. For each $i \in \{1, 2, ..., q\}$, gate $g+i$ takes input ϕ_i and outputs 1 iff $\phi_i \in$ Oracle-3SAT.
- Level 3: The circuit C'_n for computing f'. Let $g' = 2^{n^{k'}}$ be the size of C'_n, where k' is a constant. 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 input gates connect to x, and the remaining ones connect to the NEXP gates of Level 2. The output of C'_n is the final output of the whole circuit.

For the three-level circuit, we can compute each output gate of Level 1 and Level 3 using the protocol in Figure 7, and each NEXP gate in Level 2 using the protocol in Figure 4. Now we show that there exists an MRIP protocol $(V, \vec{P})$ such that the provers cannot lie in C_n in order to change the input to the NEXP queries to gain a higher overall expected payment.

Our protocol is specified in Figure 8. It uses four provers. In this protocol the verifier needs to compute $q(n)$ and $p(n)$. Without loss of generality, we assume $q(n) = 2^{n^d}$ for some constant d, so its binary representation can be computed in time polynomial in n. Since $p(n)$ is a polynomial in n, it can be computed by a polynomial-time verifier.

To prove the correctness of the protocol in Figure 8, first note that for any input string x, no matter which gate i is chosen by V in Step 2, if the provers always give correct answers according to the computation of C_n, the NEXP gates and C'_n, the payment to them is $R \geq \frac{1}{p+1} > 0$. The first inequality is tight when either (a) i is not an NEXP gate, or (b) i is an NEXP gate and the corresponding query ϕ_i is not in Oracle-3SAT (since $R^* = 1/2$ in this case). If i is an NEXP gate and $\phi_i \in$ Oracle-3SAT, then $R = \frac{2}{p+1}$ as $R^* = 1$.

Let s be the strategy profile where the provers always send correct answers as described above. Thus we have $u(s) \geq \frac{1}{p+1}$.

For any input string x of length n,

1. P_1 sends one bit $c \in \{0,1\}$ to V. V outputs c at the end of the protocol.

2. V computes $g = \text{SIZE}(C_n)$, $q(n)$, and $g' = \text{SIZE}(C'_n)$.
   V picks a gate $i \in \{1, 2, ..., g+q+g'\}$ uniformly at random and sends i to P_1.
   By doing so, V queries P_1 for:
   (a) the type t_i of gate i,
   (b) the input gates and input wires of i, and
   (c) the values of gate i and its input gates.

3. P_1 sends to V the following:
   (a) type $t_i \in \{\text{AND, OR, NOT}, \text{INPUT}, \mathsf{NEXP}\}$;
   (b) input gates $i_1, i_2, \ldots, i_{f(i)}$ and input wires $h_1, h_2, \ldots, h_{f(i)}$, where $f(i)$ is the number of input gates of type t_i; and
   (c) values of gate i and its input gates: $v_i(x)$, $v_{i_1}(x), v_{i_2}(x), \ldots, v_{i_{f(i)}}(x)$.

4. V verifies the following using DC uniformity or the naming convention:
   (a) t_i is the correct type of i (in particular, if $i \in \{g+1, ..., g+q\}$ then $t_i = \mathsf{NEXP}$) and $f(i)$ is correct for t_i; and
   (b) the set of input gates of i is correct.

   If any of the verifications fails, the protocol ends and $R = -1$.

5. V picks gate i' uniformly at random from $\{i\} \cup \{i_1 \ldots, i_{f(i)}\}$ and sends it to P_2.

6. P_2 sends $v'_{i'}(x) \in \{0,1\}$ to V.

7. **Consistency.** V verifies $v_{i'}(x) = v'_{i'}(x)$: that is, the answers of P_1 and P_2 on the value of gate i' are consistent. If not, the protocol ends and $R = -1$.

8. **Correctness (Non-NEXP gates).** If $t_i \neq \mathsf{NEXP}$, then V checks if $v_i(x)$ is computed correctly from $v_{i_1}(x), v_{i_2}(x), \ldots, v_{i_{f(i)}}(x)$ as follows:
   (a) if $t_i = \text{INPUT}$ then $v_i(x) = v_{i_1}(x)$, and if i is one of the first n gates in C_n or C'_n, then $v_i(x)$ equals the corresponding bit of x;
   (b) if $t_i \in \{\text{AND, OR, NOT}\}$, $v_i(x)$ follows the logic between i and its inputs.
   (c) if $i = g + q + g'$ (i.e., the output gate of the whole circuit), then $v_i(x) = c$.

   The protocol ends with the following reward: if any of the verifications fails then $R = -\frac{1}{p+1}$, otherwise $R = \frac{1}{p+1}$, where p is the length of each NEXP query.

9. **Correctness (NEXP gates).** If $t_i = \mathsf{NEXP}$, then V first checks if $\phi_i = (v_{i_1}(x), ..., v_{i_p}(x))$ forms a valid Oracle-3SAT instance.[a] If not, the protocol ends with $R = -\frac{2}{p+1}$. If ϕ_i is a valid Oracle-3SAT instance, then V sends ϕ_i to P_3 and P_4 and runs the MRIP protocol for NEXP in Figure 4. Let c^* and R^* respectively be the output and the reward of the NEXP protocol. If $c^* = v_i(x)$ then $R = \frac{2R^*}{p+1}$; otherwise $R = -\frac{2}{p+1}$.

[a] WLOG, we assume that the instances of Oracle-3SAT have a canonical form.

Figure 8: An MRIP protocol for $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$.

The correctness of our protocol. Arbitrarily fix a best strategy profile s^* of the provers, we show that under s^*, $c = 1$ if and only if $x \in L$.

Since P_2 is queried only once (Step 6), as in the proof of Lemma 3.10, any strategy of P_2 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. First, we show that for non-NEXP gates, P_1 answers all queries consistently with A.

Claim 3.15. *Under s^*, for any gate i that is not an* NEXP *gate and is chosen in Step 2, P_1 reports the correct type and input gates of i in Step 3, and reports the values of gate i and its input gates consistently with A.*

Proof. Suppose there exists a non-NEXP gate i such that P_1 does not report its type and input gates correctly. Conditioned on i being chosen by the verifier, some verification in Step 4 is guaranteed to fail, and the payment is -1. Consider the following alternative strategy s_1' of P_1: if i is not chosen by V, then P_1's strategy remains the same; if i is chosen, then P_1 acts "correctly" as specified in Claim 3.15. Under this strategy, when i is chosen the payment is at least $-\frac{1}{p+1} > -1$, and when i is not chosen the payment stays the same. Thus the expected payment gets larger, contradicting the fact that s^* is the provers' best strategy profile.

Similarly, consider the case where P_1 reports i's type and input gates correctly, but the reported values do not match A on some gate $i' \in \{i\} \cup \{i_1, \ldots, i_{f(i)}\}$. Conditioned on gate i being chosen, with probability at least $\frac{1}{f(i)+1} \geq \frac{1}{3}$, V picks i' in Step 5 and the consistency check in Step 7 fails, leading to a payment of -1. If i' is not chosen in Step 5, the payment to the provers is at most $\frac{1}{p+1}$ (in Step 8). Thus the expected payment conditioned on i being chosen is at most

$$-\frac{1}{3} + \frac{2}{3} \cdot \frac{1}{p+1} < -\frac{1}{p+1},$$

where the inequality holds since $p \geq 5$. Again, consider the alternative strategy s_1' of P_1. Under this strategy, conditioned on i being chosen the expected payment is at least $-\frac{1}{p+1}$; and conditioned on i not being chosen it stays the same. Thus the expected payment gets larger, again a contradiction. □

Below we only need to consider cases where P_1 acts according to Claim 3.15. We argue about the correctness of A on non-NEXP gates, and we have the following.

Claim 3.16. *Under s^*, for every gate i that is not an* NEXP *gate, $A(i)$ and the values $A(i_1), \ldots, A(i_{f(i)})$ are such that the verifications in Step 8 succeed.*

Proof. By contradiction, assume this is not the case and compare s^* with the "always correct" strategy profile s previously defined. Recall that, conditioned on i being chosen, for any non-NEXP gate i the payment under s is exactly $\frac{1}{p+1}$, and for any NEXP gate i the payment under s is at least $\frac{1}{p+1}$.

Under s^*, by Claim 3.15, P_1's answers for $v_i(x), v_{i_1}(x), \ldots, v_{i_{f(i)}}(x)$ are consistent with A. If A makes some verification in Step 8 fail, then conditioned on i being chosen, the payment under s^* is $-\frac{1}{p+1}$. That is, the payment under s^* drops by $\frac{2}{p+1}$ compared with that under s.

However, s and s^* may not have the same oracle queries to Oracle-3SAT. For each NEXP gate j where the two queries differ, conditioned on j being chosen, the best case for s^* (and the worst case for the analysis) is that its query ϕ_j^* is in Oracle-3SAT, resulting in payment $\frac{2}{p+1}$, while the query ϕ_j of s is not in Oracle-3SAT, resulting in payment $\frac{1}{p+1}$. That is, the payment under s^* increases by $\frac{1}{p+1}$ compared with that under s.

Fortunately, for each NEXP gate j, in order for the two queries to differ, there exists at least one non-NEXP gate i in the part of the circuit C_n for computing the input to j, where the computation of A (and thus s^*) is incorrect, and $A(i)$ and $A(i_1), \ldots, A(i_{f(i)})$ make some verification in Step 8 fail. Otherwise the queries made by A are computed correctly from the input x and are the same as those under s. Since gate j and the corresponding gate i are chosen with the same probability $\frac{1}{g+q+g'}$, we have

$$u(s) - u(s^*) \geq \frac{1}{g+q+g'} \cdot \frac{2}{p+1} - \frac{1}{g+q+g'} \cdot \frac{1}{p+1} > 0.$$

If there is more than one such j, their corresponding gates i are all different from each other, because the circuits for computing different oracle queries are disjoint from each other—so the gap between $u(s)$ and $u(s^*)$ becomes even larger. This contradicts that s^* is the provers' best strategy, and thus Claim 3.16 holds. □

Now we only need to consider cases where P_1 acts according to Claims 3.15 and 3.16. We prove the correctness of A on NEXP gates.

Claim 3.17. *Under s^*, for every* NEXP *gate i, P_1 reports the correct type and input gates of i in Step 3, and reports the values of gate i and its input gates consistently with A. Moreover, $\phi_i = (A(i_1), \ldots, A(i_p))$ forms a valid* Oracle-3SAT *instance and $A(i) = 1$ iff $\phi_i \in$* Oracle-3SAT*.*

Proof. The fact that ϕ_i forms a valid Oracle-3SAT instance follows from Claims 3.15 and 3.16, because each bit of ϕ_i is the output of a logic gate and thus computed correctly from the input x according to C_n. We again compare s^* with the always-correct strategy profile s.

Note that A and s are both correct on C_n, thus form the same Oracle-3SAT queries. They both evaluate C'_n correctly as well, but it is possible that A has incorrect outputs of the NEXP gates and thus incorrect inputs to C'_n. Nevertheless, for each non-NEXP gate i', conditioned on i' being chosen, s^* makes the verifications in Step 8 succeed, and the payment is $\frac{1}{p+1}$ under both s and s^*.

If P_1 reports i's type and input gates incorrectly under s^*, then the payment is -1 (Step 4) conditioned on i being chosen. However, by reporting the required information correctly and reporting $v_i(x), v_{i_1}(x), \ldots, v_{i_p}(x)$ consistently with A, the corresponding payment is at least $-\frac{2}{p+1} > -1$ and the expected payment increases, contradicting with the fact that s^* is the provers' best strategy profile.

Suppose P_1 reports i's type and input gates correctly, but reports $v_{i'}(x)$ inconsistently with A for some $i' \in \{i\} \cup \{i_1, \ldots, i_p\}$. In this case, with probability at least $\frac{1}{p+1}$ the payment is -1 (Step 7), and with probability at most $1 - \frac{1}{p+1}$ the payment is at most $\frac{2}{p+1}$ (Step 9). Thus the expected payment is

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

The corresponding expected payment under s is at least $\frac{1}{p+1}$. As the two strategy profiles have the same payment $\frac{1}{p+1}$ conditioned on every non-NEXP gate i' being chosen, we have $u(s) > u(s^*)$, a contradiction.

Finally, assume P_1 is consistent with A, but $A(i)$ is not the correct answer of ϕ_i. If the answer bit c^* given by P_3 and P_4 is different from $A(i)$ (i.e., $v_i(x)$), then the payment is $-\frac{2}{p+1} < \frac{1}{p+1}$, less than the payment received under the always-correct strategy profile s. If $c^* = v_i(x)$, then c^* is the wrong answer bit in the MRIP protocol for NEXP, and the resulting payment R^* is strictly less than the payment under s. Thus, again we have that $u(s) > u(s^*)$, which is a contradiction, and Claim 3.17 holds. □

Claims 3.15, 3.16, and 3.17 together imply that the always-correct strategy profile s is the only possibility for the provers' best response; that is, $s^* = s$. Under s, for any gate i, $A(i)$ is the correct value of i under input x, and $c = A(g + q + g')$. Thus $c = 1$ iff $x \in L$.

Upper bound for MRIP. We now give a tight upper-bound on MRIP.

Lemma 3.18. $\mathsf{MRIP} \subseteq \mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$.

Proof. The proof is similar to that of Lemma 3.12. Let L be a language with an MRIP protocol $(V, \vec{P})$. Since V runs in polynomial time, there exists a constant k such that, for any two payments R and R' generated by V on the same input of length n and different

random coins: $R \neq R' \Rightarrow |R - R'| \geq \frac{1}{2^{n^k}}$. For example, n^k can be an upper bound on V's running time. Moreover, since V uses polynomially many random coins, there exists a constant k' such that any payment that appears with positive probability under an input of length n must appear with probability at least $\frac{1}{2^{n^{k'}}}$. Thus, for an input x of length n, and any two strategy profiles s and s', we have, $|u(s,x) - u(s',x)| \geq \frac{1}{2^{n^{k+k'}}}$.

Consider the following deterministic oracle Turing machine M: given any input x of length n, M divides the interval $[-1,1]$ into $4 \cdot 2^{n^{k+k'}}$ sub-intervals of length $\frac{1}{2 \cdot 2^{n^{k+k'}}}$. For any $i \in \{-2 \cdot 2^{n^{k+k'}} + 1, \ldots, 2 \cdot 2^{n^{k+k'}}\}$, the ith interval is $\left[\frac{(i-1)}{2 \cdot 2^{n^{k+k'}}}, \frac{i}{2 \cdot 2^{n^{k+k'}}}\right]$. For each interval i, M makes the following two queries to an NEXP oracle:

1. Does there exist a strategy profile s with expected payment $u_{(V,\vec{P})}(s,x)$ in interval i?
2. Does there exist a strategy profile s with expected payment $u_{(V,\vec{P})}(s,x)$ in interval i and the corresponding answer bit $c = 1$?

Notice that M makes exponentially many nonadaptive queries, and each query has length polynomial in n. Each query can be answered by an NEXP oracle; see the proof of Lemma 3.12.

Given the oracle's answers, M finds the highest index i^* such that interval i^* is non-empty, that is, the oracle's answer to the first query for interval i^* is 1. M accepts if the answer to the second query for interval i^* is 1, and rejects otherwise.

M clearly runs in exponential time. We now show that M decides L. Similar to Lemma 3.12, by Definition 3.4, the best strategy profile s^* has the highest expected payment $u(s^*;x)$, which falls into interval i^*. Any strategy profile s' with $u(s',x) < u(s^*,x)$ has $u(s',x)$ not in interval i^*, since the difference between $u(s',x)$ and $u(s^*,x)$ is larger than the length of the interval. Thus, any strategy profile s' with $u(s',x)$ in interval i^* satisfies $u(s',x) = u(s^*,x)$, i.e, they are all the best strategy profiles of the provers. In particular, the answer bit c is the same under all these strategy profiles, and $c = 1$ if and only if $x \in L$. So the second query for interval i^* is 1 if and only if $x \in L$, and M decides L. □

Final characterization. So far we have established that $\mathsf{MRIP} = \mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$. To finish the proof of Theorem 3.3, we show $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}}$ equals $\mathsf{EXP}_{||}^{\mathsf{NP}}$.

Lemma 3.19. $\mathsf{EXP}_{||}^{\mathsf{poly-NEXP}} = \mathsf{EXP}_{||}^{\mathsf{NP}}$.

Proof. 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 nonadaptive 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_1' with time complexity $2^{|q|^{k_1}}$, where k_1 is a constant and q is the query to the oracle (the input to M_1'). We simulate $M_1^{O_1}$ using exponential-time oracle Turing 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 nonadaptive oracle queries, and (2) $M_2^{O_2}$ decides the same language as $M_1^{O_1}$, as long as O_2's answer to each query q' 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_2' that simulates M_1'. That is, O_2 will be the language decided by M_2'. More specifically, given a query q' (q followed by $2^{|q|^{k_1}}$ 1s), M_2' runs M_1' on q, makes the same non-deterministic choices as M_1', and outputs whatever M_1' outputs. Since M_1' runs in time $2^{|q|^{k_1}}$, $M_{2'}$ runs in time polynomial in its own input size. Thus, the language O_2 decided by M_2' is in NP, and $q' \in O_2$ if and only if $q \in O_1$. Accordingly, $M_2^{O_2}$ 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 proof is similar to the above. Let M_2 be an exponential-time oracle Turing machine with nonadaptive access to an oracle O_2 for an NP language. Note that the queries made by M_2 can be exponentially long. Let O_2 be decided by a non-deterministic Turing machine M_2' that runs in time $|q|^{k_2}$, where k_2 is a constant and q is the query to O_2 (the input to M_2'). We simulate $M_2^{O_2}$ using an exponential-time oracle Turing machine M_1 and an 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 ith 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 is to the following question: *is the ith query made by M_2 given input x in the* NP *language O_2?* M_1 then gives all its queries to its own oracle O_1. Given O_1'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 nonadaptive oracle queries, and (2) $M_1^{O_1}$ decides the same language as $M_2^{O_2}$ as long as O_1 answers each query correctly.

We define O_1 by constructing a non-deterministic Turing machine M_1' that simulates M_2'. That is, O_1 will be the language decided by M_1'. More specifically, given an input string of the form (x, y), M_1' interprets the second part as the binary representation of an integer

i. It runs M_2 on x to compute its ith query, denoted by q. It then runs M_2' on q, makes the same non-deterministic choices as M_2', and outputs whatever M_2' outputs. Since q is at most exponentially long in $|x|$ and M_2' runs in time $|q|^{k_2}$, the running time of M_1' is (at most) exponential in its input length. Thus, the language O_1 decided by M_{O_1} is in NEXP. Moreover, if $q \in O_2$, then there exist non-deterministic choices that cause M_2' and thus M_1' to accept; otherwise both reject. That is, O_1's answers to the queries by M_1 on input x are the same as O_2's answers to the queries by M_2 on the same input.

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

Theorem 3.3 follows immediately from Lemmas 3.14, 3.18, and 3.19.

3.6 Optimal Number of Provers and Rounds

In this section, we prove that MRIP protocols with constant, polynomial or exponential utility gap can be simulated using the optimal number of provers and rounds.

First, we prove that any MRIP protocol with a constant or polynomial utility gap can be simulated by a 2-prover, 3-round MRIP protocol with the respective utility gap.

Theorem 3.20. *Let $\gamma(\mathsf{n})$-$\mathsf{MRIP}(\mathsf{p},\mathsf{k})$ denote the class of languages decided by an MRIP protocol with $O(\gamma(n))$ utility gap using $p(n)$ provers and $k(n)$ rounds. Then,*

$$\mathsf{O(1)\text{-}MRIP}(\mathsf{p(n)},\mathsf{k(n)}) = \mathsf{O(1)\text{-}MRIP}(2,3) \text{ and}$$

$$\mathsf{poly(n)\text{-}MRIP}(\mathsf{p(n)},\mathsf{k(n)}) = \mathsf{poly(n)\text{-}MRIP}(2,3).$$

Proof. Recall from Lemma 3.11 and Lemma 3.12 that $\gamma(\mathsf{n})$-$\mathsf{MRIP} = \mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(\mathsf{n})]}$, for any positive integral function $\gamma(n)$ that is polynomially bounded and polynomial-time computable. We show that 2 provers and 3 rounds are enough to simulate the protocol in Figure 6. Setting $\gamma(n)$ to be a constant or a polynomial in n proves Theorem 3.20.

More precisely, for any language $L \in \gamma(\mathsf{n})$-MRIP, we have $L \in \mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(\mathsf{n})]}$. By definition, there exists a polynomial-time oracle Turing machine M that decides L using $O(\gamma(n))$ nonadaptive queries to an NEXP oracle. Assume without loss of generality that the oracle is $\mathsf{Oracle\text{-}3SAT}$ and M makes exactly $\gamma(n)$ oracle queries. For any input x of length n, let $\phi_1, \ldots, \phi_{\gamma(n)}$ denote the $\gamma(n)$ queries made by M. Consider the following 2-prover 3-round variant of the MRIP protocol in Figure 6 for L.:

1. P_1 sends to V the answer bit c to the membership of x in L, as well as the answer bits to all queries, $c_1^*, c_2^*, \ldots, c_{\gamma(n)}^*$, where c_i^* is the answer to ϕ_i. As P_1 can compute all oracle queries by running M on x, there is no need for V to send $\phi_1, \ldots, \phi_{\gamma(n)}$ to P_1.

2. V computes $\phi_1, \ldots, \phi_{\gamma(n)}$ and distinguishes the following two cases. For each $i \in \{1, \ldots, \gamma(n)\}$ with $c_i^* = 0$, V sets $R_i^* = 1/2$. For all i's such that $c_i^* = 1$, V runs the 2-prover 3-round MRIP protocol in Figure 4 for the ϕ_i's *simultaneously*. That is, for each such i, V uses fresh randomness to compute its messages to P_1 and P_2 in the second round of the MRIP protocol for ϕ_i, denoted by m_{12}^i and m_{22}^i respectively, which are by definition its first messages in the corresponding MIP protocol. In the second round of the overall protocol, V sends the concatenation of the m_{12}^i's to P_1 and the concatenation of the m_{22}^i's to P_2.

3. For each i such that $c_i^* = 1$, P_1 computes its response m_{13}^i to m_{12}^i, and P_2 computes its response m_{23}^i to m_{22}^i. They send the concatenation of their responses to V.

4. For each i such that $c_i^* = 1$, V finishes the MIP protocol following the messages exchanged for ϕ_i. If the MIP protocol accepts then V sets $R_i^* = 1$; otherwise $R_i^* = 0$.

5. Finally, V simulates M till the end using the c_i^*'s. If the answer bit c does not match M's output, then the protocol ends with $R = -1$; otherwise the protocol ends with $R = (\sum_{i=1}^{\gamma(n)} R_i^*)/\gamma(n)$. V outputs c at the end of the protocol.

The correctness of this protocol is similar to Lemma 3.11, except some subtleties caused by the simultaneous execution of the MRIP protocols for the ϕ_i's. First of all, sending c and $c_1^*, \ldots, c_{\gamma(n)}^*$ such that the output of M does not match c cannot be part of the provers' best strategy profile, because it leads to $R = -1$, while sending all messages truthfully leads to $R \geq 1/2$. Second, by linearity of expectation, for any strategy profile of the provers such that c matches the output of M given $c_1^*, \ldots, c_{\gamma(n)}^*$, the expected payment is the sum of the expected payment for each ϕ_i.

Note that for each ϕ_i, V's messages in the corresponding MIP protocol only depends on its randomness, and it uses fresh coins for ϕ_i. Thus, even though the provers also see V's messages for other ϕ_j's, they cannot improve V's marginal accepting probability for ϕ_i. From this, the expected payment for each ϕ_i is still maximized when the provers report the correct c_i^* and, when $c_i^* = 1$, run the corresponding MIP protocol correctly. Therefore, under the provers' best strategy profile, the c_i^*'s are correct answers to M's oracle queries, c is the correct output of M given the c_i^*'s, and $c = 1$ if and only if $x \in L$.

Finally, the utility gap of the protocol is the same as the protocol in Figure 6, that is, the utility gap is $O(\gamma(n))$. Thus, $\mathsf{P}_{||}^{\mathsf{NEXP}[\gamma(\mathsf{n})]} \subseteq \gamma(\mathsf{n})\text{-}\mathsf{MRIP}(2,3) \subseteq \gamma(\mathsf{n})\text{-}\mathsf{MRIP}$. □

Finally, we pove that any general MRIP protocol can be simulated by an MRIP protocol that uses only 2 provers and 3 rounds and has exponential utility gap.

Theorem 3.21. $\mathsf{MRIP} = \mathsf{MRIP}(2,3)$.

Proof. Arbitrarily fix an MRIP protocol $(V, \vec{P})$ for a language L with $p(n)$ provers and $k(n)$ rounds. Without loss of generality, each message in the protocol is of length $\ell(n)$ for any input of length n, where $\ell(n)$ is a polynomial in n. We shift and re-scale the reward function of V, so that the payment is always in $[0,1]$, and the expected payment is strictly larger than 0 under the provers' best strategy profile. The corresponding 2-prover 3-round protocol $(V', (P_1', P_2'))$ is defined in Figure 9.

Essentially, V' asks P_1' to simulate all provers in the original protocol. V' wants to use P_2' to cross-check the transcript provided by P_1', but in parallel: that is, without waiting for P_1''s message. It does so by randomly generating a proxy string of polynomial length and giving it to P_2'. There is an exponentially small probability that this string is consistent with the transcript P_1' sends, and if it turns out to be consistent, V' goes on to match the answers it receives from P_1' and P_2', and to compute the payment as in the 5-round protocol in [43].

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

1. P_1' sends $m_{11}, \ldots, m_{p(n)1}$ to V', where m_{ij} denotes the message sent by prover P_i in round j of $(V, \vec{P})$ according to the best strategy profile s of $\vec{P}$.

   Let c be the first bit of m_{11}. V' outputs c at the end of the protocol.

2. V' generates the random string r used by V and sends it to P_1'. V' selects, uniformly at random, a prover index $i \in \{1, \ldots, p(n)\}$ and a round number $j \in \{2, \ldots, k(n)\}$. V' then generates a random string m_i^* of length $(j-1)\ell(n)$ and sends (i, j, m_i^*) to P_2'.

3. P_1' uses r, $m_{11}, \ldots, m_{p(n)1}$ and s to continue simulating the protocol $(V, \vec{P})$, and sends to V' the messages from round 2 to round $k(n)$ in the resulting transcript $\vec{m}$. P_2' uses m_i^* (and s) to simulate P_i on round j, and sends the resulting message m_{ij}' to V'.

4. If $m_i^* \neq (m_{i1}, \ldots, m_{i(j-1)})$, then the protocol ends with payment $R' = 0$.

5. If $m_{ij} \neq m_{ij}'$, then $R' = -1$. Else, V' computes the payment R in the protocol $(V, \vec{P})$ using x, r and $\vec{m}$, and sets $R' = \frac{R}{p(n)2^{k(n)\ell(n)}}$.

Figure 9: Simulating any MRIP protocol with 2 provers and 3 rounds.

To see why this protocol works, first note that, even though V' sends to P_1' the randomness r used by V, V' itself uses fresh randomness in Step 2 to generate i, j and m_i^*, which are unknown to P_1'. Second, the strategy of P_2' in Step 3 de facto commits to a strategy profile for the provers in $(V, \vec{P})$ except for the first round, which together with the randomness r of V and $m_{11}, \ldots, m_{p(n)1}$ sent by P_1' determines a transcript $\vec{m}^*$ in $(V, \vec{P})$.

We distinguish two cases for the strategy profiles of (P_1', P_2').

Case 1. Suppose there exists randomness r, such that P_1' and P_2' do not agree on the transcript under r: that is, $\vec{m} \neq \vec{m}^*$, where $\vec{m}$ is the transcript sent by P_1'. Suppose $\vec{m}$ disagrees with $\vec{m}^*$ on y messages, with $y \geq 1$. Then the probability that the prover index i and the round number j chosen by V' in Step 2 satisfy $m_{ij}^* \neq m_{ij}$ is $\frac{y}{p(n)(k(n)-1)}$.

When $m_{ij}^* \neq m_{ij}$, if the random string m_i^* generated by V' in Step 2 does not equal $(m_{i1}, \ldots, m_{i(j-1)})$, then the inconsistency between m_{ij}^* and m_{ij} is not caught and the payment is 0; otherwise the payment is -1. When $m_{ij}^* = m_{ij}$, the payment is either 0 or at most $\frac{1}{p(n)2^{k(n)\ell(n)}}$, again depending on whether $m_i^* = (m_{i1}, \ldots, m_{i(j-1)})$ or not.

Finally, as the length of each message in $(V, \vec{P})$ is $\ell(n)$, for any i and j, the probability that $m_i^* = (m_{i1}, \ldots, m_{i(j-1)})$ is $\frac{1}{2^{(j-1)\ell(n)}} \geq \frac{1}{2^{(k(n)-1)\ell(n)}}$. We upper bound the expected payment R' in Case 1 under r as follows.

$$
\begin{aligned}
R' &\leq \sum_{i \leq p(n), 2 \leq j \leq k(n)} \frac{1}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot \left(\mathbb{I}_{m_{ij}^* \neq m_{ij}} \cdot (-1) + \mathbb{I}_{m_{ij}^* = m_{ij}} \cdot \frac{1}{p(n)2^{k(n)\ell(n)}} \right) \\
&\leq -\frac{y}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(k(n)-1)\ell(n)}} \\
&\quad + \sum_{i \leq p(n), 2 \leq j \leq k(n)} \frac{1}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot \mathbb{I}_{m_{ij}^* = m_{ij}} \cdot \frac{1}{p(n)2^{k(n)\ell(n)}} \\
&< -\frac{y}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(k(n)-1)\ell(n)}} + \sum_{2 \leq j \leq k(n)} \frac{1}{k(n)-1} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot \frac{1}{p(n)2^{k(n)\ell(n)}} \\
&< -\frac{y}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(k(n)-1)\ell(n)}} + \frac{1}{(k(n)-1)p(n)2^{k(n)\ell(n)}} = \frac{1-2y}{(k(n)-1)p(n)2^{k(n)\ell(n)}} < 0.
\end{aligned}
$$

On the other hand, if P_1' acts consistently with P_2' in Step 3 under r, and keeps its strategy unchanged under any other randomness of V sent to it by V', then the expected payment under r is at least 0 and the expected payment under any other randomness of V does not change, thus, the expected payment in the whole protocol gets larger. Under the best strategy profile of (P_1', P_2'), Case 1 does not occur for any randomness r of V.

Case 2. In their strategy profile s', P_1' and P_2' agree on the transcript $\vec{m}$ under every randomness r of V, but the strategy profile $\tilde{s}$ committed by them for $(V, \vec{P})$ (that is, by P_1' in Step 1 for round 1 and then by P_2' in Step 3 for the remaining rounds) has the answer bit c incorrect. Thus $\tilde{s}$ is not the best strategy profile s of $\vec{P}$.

In this case, given any randomness r, prover i and round j chosen by V' in Step 2, the expected payment is

$$
R' = \frac{1}{2^{(j-1)\ell(n)}} \cdot \frac{R}{p(n)2^{k(n)\ell(n)}},
$$

where R is the payment of $(V, \vec{P})$ under $\tilde{s}$ and r. Therefore, the expected payment for P_1' and P_2' in the whole protocol is

$$\begin{aligned} u_{(V',\vec{P'})}(s', x) &= \sum_{i \le p(n), 2 \le j \le k(n)} \frac{1}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot \frac{u_{(V,\vec{P})}(\tilde{s}, x)}{p(n)2^{k(n)\ell(n)}} \\ &< \sum_{i \le p(n), 2 \le j \le k(n)} \frac{1}{p(n)(k(n)-1)} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot \frac{u_{(V,\vec{P})}(s, x)}{p(n)2^{k(n)\ell(n)}}, \end{aligned}$$

where the inequality is because $u_{(V,\vec{P})}(\tilde{s}, x) < u_{(V,\vec{P})}(s, x)$. Note that the second line in the equation above is exactly the expected payment for P_1' and P_2' when they commit to s. Thus committing to $\tilde{s}$ is not the best strategy profile for P_1' and P_2'. □

Chapter 4

Scaled-Down Cooperative Rational Proofs

4.1 Introduction

In Chapter 3, we designed and analyzed MRIP protocols where the verifier's running time is polynomial in n (the size of the input) and the communication complexity is also polynomial in n. With a polynomial-time verifier, we were able to ensure that the protocols had a strong utility-gap guarantees, that is, polynomial and even constant utility gap.

While a polynomial-time verifier is considered efficient, and rational proof protocols are simpler than their classical variant, in terms of delegation of computation applications, we need more efficient protocols—a "weak" client is unlikely to be able to spend (say) quadratic time or linear extra space on verification.

In this chapter, we design "scaled-down" rational proof protocols that have very small overheads in terms of verification time, space, communication cost and number of rounds. In particular, we design constant-round rational protocols where the verification time and communication cost are logarithmic in the input size n. We also design single-round rational protocols that have only logarithmic overhead on the verifier's use of space and randomness.

Recall that we measure the quality of the guarantee provided by rational proof protocol by its *utility gap*. If a rational protocol has a utility gap of u, then the provers who mislead the verifier to an incorrect answer are guaranteed to lose at least $1/u$. (This is under a normalized budget of 1; if the budget is scaled up to B, such provers can be made to lose at least B/u.) Thus, protocols with small utility gap are sound even against provers with *bounded rationality*; that is, provers who are only sensitive to large losses.

We show that even though our protocols in this chapter are very efficient, they still retain strong utility-gap guarantees—that is, polynomial, logarithmic, and even constant.

Later in Chapter 5, we show when and how a noticeable utility gap of a rational protocol can be used to achieve the completeness and soundness guarantees of a classical proof.

4.1.1 Overview of Results and Contributions

We summarize our results and contributions.

Super time-efficient rational proofs. We study the effect of different communication costs and an additional prover on the power of rational proofs with a highly time-efficient verifier. The utility gap of these protocols is polynomial.

- **Constant communication.** We show that multiple provers do not add any power when the communication complexity of the protocol is restricted to be extremely small—a constant number of bits. That is, we show that the class of languages that admit a multi-prover rational proof with a $O(\log n)$-time verifier and $O(1)$ communication is exactly $\mathsf{UniformTC_0}$, which is the same as the power of single-prover version under the same costs [11,80]. Recall that $\mathsf{UniformTC_0}$ is the class of constant depth, polynomial size uniform threshold circuits, an important complexity class that includes problems such as integer division, iterated multiplication and radical summations [3, 4, 84, 89, 90].

- **Logarithmic communication.** We show that any rational proof with polynomial communication can be simulated by a rational proof with logarithmic communication that uses an additional prover. Using this property, we improve the communication complexity of Azar and Micali's [11] single-prover rational protocol and show that the class of languages that admit a two-prover rational proof with logarithmic communication is exactly the class of languages decidable by a polynomial time machine that can make polynomially many queries in parallel to an NP oracle, denoted $\mathsf{P}_{||}^{\mathsf{NP}}$. We recall that this is a well-studied class (e.g., [31, 45, 92, 99, 106, 140]) and includes important optimization problems such as maximum clique, longest paths, and variants of the traveling salesman problem.

Super space-efficient rational proofs. We achieve even better utility gap guarantees for the setting where the verifier's use of space and randomness is super-efficient. In particular, we exactly characterize the class of single-round rational proofs with $\gamma(n)$ utility gap and logarithmic space and randomness as the class of languages decidable by a polynomial-time machine that makes $O(\gamma(n))$ queries to an NP oracle, denoted $\mathsf{P}_{||}^{\mathsf{NP}[\gamma(n)]}$. Even when $\gamma(n) = O(1)$ this bounded-query class is still sufficiently powerful and contains many of the optimization problems mentioned above.

Thus, highly space-efficient rational protocols with strong guarantees against imperfectly rational provers can solve many important optimization problems.

Interestingly, the logarithmic-space verifier studied in this chapter also happens to be a *streaming algorithm*, that is, the verifier does not need to look again at any input or message bits out of order. Thus, our space-efficient rational proofs are closely related to the work on streaming interactive proofs [39, 49, 50, 52].

4.2 Preliminaries

We use the model of multi-prover cooperative rational proofs defined in Chapter 3.2.

We denote the class of single-prover rational interactive protocols, that is, when $\vec{P} = P$, as RIP. Multi-prover interactive protocols, where $\vec{P} = (P_1, \ldots, P_{p(n)})$ are denoted by MRIP. We use $\text{poly}(n)$ as a shorthand for a polynomial n^k, for some constant k.

Our primary focus in this chapter is analyzing the various computational costs of rational interactive proofs. The different parameters fall into two categories.

Verification costs. A verifier has three main resources: running time, space usage and its randomness. While in general the model allows for a verifier that runs in time polynomial in n and uses polynomial space and randomness, we focus on more efficient verifiers in this chapter. In particular, in Chapter 4.3, we focus on time-efficient $O(\log n)$ time verifiers. Thus, their space and randomness is also $O(\log n)$. We denote the class of languages that have time-efficient RIP protocols, that is, protocols with a $O(\log n)$ time verifier as $\mathsf{RIP}^{\mathsf{t}}$. Multi-prover notation $\mathsf{MRIP}^{\mathsf{t}}$ is analogous.

As a logarithmic verifier cannot even read the entire input, it is difficult to obtain protocols with good utility gap guarantees using these verifier. To achieve better utility gap, in Chapter 4.4, we restrict the verifier's space usage and randomness, instead of its running time and consider verifiers that use $O(\log n)$ space and $O(\log n)$ randomness. We denote the class of languages that have an RIP protocol with space- and randomness-efficient verifiers, that is, verifiers with $O(\log n)$ space and $O(\log n)$ randomness as $\mathsf{RIP}^{\mathsf{s,r}}$.

Protocol costs. A rational interactive proof protocol has three main ingredients: communication cost, number of provers and rounds and utility gap.

In Chapter 4.3, we study the effect of varying the communication complexity of a protocol on its power when we have a logarithmic time verifier. The number of rounds in all the protocols in the chapter is $O(1)$.

We denote the class of languages that have an MRIP protocol with $C(n)$ communication cost, $p(n)$ provers, $k(n)$ rounds and $O(\gamma(n))$ utility gap as $\gamma(\mathsf{n})\text{-}\mathsf{RIP}[\mathsf{C}(\mathsf{n}), \mathsf{p}(\mathsf{n}), \mathsf{k}(\mathsf{n})]$. For single-prover protocols with the same parameters, we keep the same notation with $p(n) = 1$ for consistency, that is, we use $\gamma(\mathsf{n})\text{-}\mathsf{RIP}[\mathsf{C}(\mathsf{n}), 1, \mathsf{k}(\mathsf{n})]$.

When the verifier runs in $O(\log n)$ (as in Chapter 4.3), then the utility gap is trivially at least polynomial in n. Thus, when considering a protocol in $\mathsf{RIP}^{\mathsf{t}}$, we drop $\gamma(n)$ from the notation for simplicity when $\gamma(n) = \text{poly}(n)$. That is, $O(\text{poly}(n))\text{-}\mathsf{RIP}^{\mathsf{t}}[\mathsf{C}(\mathsf{n}), \mathsf{p}(\mathsf{n}), \mathsf{k}(\mathsf{n})] = \mathsf{RIP}^{\mathsf{t}}[\mathsf{C}(\mathsf{n}), \mathsf{p}(\mathsf{n}), \mathsf{k}(\mathsf{n})]$. The multi-prover case is analogous.

4.3 Verification in Logarithmic Time

In this section we consider *time-efficient* verifiers that run in time logarithmic in the input size. We show that for time-efficient verifiers, access to multiple provers is fundamentally linked to the communication cost of the protocol: any single-prover protocol with high communication costs can be reduced to a communication-efficient multi-prover protocol. On the other hand, multiple provers give no extra power for communication-efficient protocols.

Since the utility gap of all the protocols in this section is polynomial in n, we drop it the utility gap from the notations.

Constant communication. We first show that multiple provers do not increase the power of a rational proof system when the communication complexity of the protocol is very small, that is, only $O(1)$ bits. Recall that with a single prover, $\mathsf{RIP}^{\mathsf{t}}[\mathsf{O}(\log \mathsf{n}), \mathsf{1}, \mathsf{O}(\mathsf{1})] = \mathsf{RIP}^{\mathsf{t}}[\mathsf{O}(\mathsf{1}), \mathsf{1}, \mathsf{O}(\mathsf{1})] = \mathsf{UniformTC_0}$ [11, 80].

Theorem 4.1. $\mathsf{MRIP}^{\mathsf{t}}[\mathsf{O}(\mathsf{1}), \mathsf{O}(\mathsf{1}), \mathsf{O}(\mathsf{1})] = \mathsf{UniformTC_0}$.

Proof. The lower bound follows directly from the single prover result [11, 80].

Now, we prove that $\mathsf{MRIP}^{\mathsf{t}}[\mathsf{O}(\mathsf{1}), \mathsf{O}(\mathsf{1}), \mathsf{O}(\mathsf{1})] \subseteq \mathsf{UniformTC_0}$.

Let L be a language with a k-round MRIP protocol $(V, \vec{P})$ where V runs in $O(\log n)$ time, and the transcript of $(V, \vec{P})$ has size $O(1)$, where k is a constant.

Note that the strategy profile s of the provers $\vec{P}$ for a protocol with $O(1)$ communication can be specified in $O(1)$ bits. Thus, there can be at most $O(1)$ possible strategy profiles for the provers to choose from. We first construct a circuit that decides L and then show that the circuit can be simulated by a $\mathsf{UniformTC_0}$ machine.

We construct the gates in independent blocks (i.e. there are no wires between two gates in different blocks). We denote the ith block by G_i for $1 \leq i \leq t$ for some constant t. The purpose of G_i is to "try out" strategy profile s^i. In particular, the output of the block G_i is the expected payment over all possible coin flips of the verifier when the strategy followed by the provers is s^i. All blocks finally output their solution (the expected payment) to a single max gate that finds the maximum over the expected payments.

The structure inside a block G_i is as follows: for each possible randomness r of the verifier we have an input wire to the block gate (note that there are at most polynomially many r). Given r, executing the strategy of the provers in a step by step manner (using the truth table) can be simulated by a depth k circuit using using AND, OR, and NOT gates. Thus, for each r, a constant sized circuit can compute the final payment.

Finally, a SUM gate at the end of G_i can sum over the payments to compute the expected payment.[7] This final expectation is the output of the block G_i.

The final MAX gate over the output all G_i gives the maximum possible expected reward. If the first bit of the corresponding strategy matches the first bit of the MRIP protocol's transcript, then the circuit outputs 1, else 0.

We note that the above circuit structure can be simulated by a constant-depth uniform threshold circuit since SUM gates and MAX gates with polynomial input wires can be implemented using $\mathsf{UniformTC_0}$ circuits [11]. □

Logarithmic and polynomial communication. We characterize the power of MRIP protocols with $O(\log n)$-time verification, when the communication complexity of the protocol is logarithmic and polynomial in n.

Theorem 4.2. $\mathsf{MRIP^t}[\mathsf{poly(n)}, \mathsf{poly(n)}, \mathsf{poly(n)}] = \mathsf{MRIP^t}[\mathsf{O(\log(n))}, \mathsf{O(1)}, \mathsf{O(1)}] = \mathsf{P_{||}}^{\mathsf{NP}}$.

Azar and Micali [11] characterized the class $\mathsf{P_{||}^{NP}}$ in terms of single-prover rational proofs with $O(\log n)$ verification and $O(\text{poly}(n))$ communication. In particular, they proved that $\mathsf{RIP^t}[\mathsf{O(poly(n))}, \mathsf{1}, \mathsf{O(1)}] = \mathsf{P_{||}^{NP}}$.

To prove Theorem 4.2, we first show that using two provers reduces the communication complexity of the RIP protocol for $\mathsf{P_{||}^{NP}}$ exponentially. In fact, we show prove a more general statement—any MRIP protocol (thus any RIP protocol as well) with a logarithmic time verifier and polynomial communication can be simulated using two provers, five rounds and logarithmic communication.

Lemma 4.3. *An MRIP protocol with $p(n)$ provers, $k(n)$ rounds, $T(n)$ verification cost, and $C(n)$ communication cost can be simulated by an MRIP protocol with 2 provers, 5 rounds, $O(T(n) + \log C(n))$ verification cost and $O(T(n) + \log C(n))$ communication cost.*

Proof. Let $(V, \vec{P})$ be the MRIP protocol for a language L with $p(n)$ provers where V's running time is $T(n)$ and $C(n)$ bits of communication are exchanged over $k(n)$ rounds. Without loss of generality, suppose each message is of length $\ell(n)$. Note that $k(n) \leq C(n)$ and $\ell(n) \leq C(n)$. We shift and scale the payment function R of $(V, \vec{P})$ such that $R \in [0, 1]$. The 2-prover 5-round MRIP protocol (V', P_1', P_2') in Figure 10 simulates $(V, \vec{P})$.

The string $\tilde{m}$ in step 3 is the *effective transcript* of the protocol $(V, \vec{P})$. Since V runs in time $T(n)$, for a given randomness r, V can access at most $T(n)$ bits from the $C(n)$-size transcript $\vec{m}$ of the protocol (P, V). Thus, $|\tilde{m}| \leq T(n)$.

[7] We could normalize by dividing by the number of possible r, but this scaling is unnecessary as we only care about relative payments.

For any input string x of length n, the protocol (V', P_1', P_2') works as follows.

1. P_1' sends m_1 to V', where m_1 is the message sent by P_1 in the first round of $(V, \vec{P})$ according to the best strategy profile s of $\vec{P}$. V' outputs c, the first bit of m_1 at the end of the protocol.

2. V' generates the random string r used by V in $(V, \vec{P})$ and sends it to P_1'.

3. P_1' sends a string $\tilde{m}$, which is a concatenation of bits accessed by V in order in the transcript $\vec{m} = (V, \vec{P})(x, r, s)$.

4. V' chooses a round j from $\{1, 2 \ldots, k(n)\}$, a prover index $i \in \{1, \ldots, p(n)\}$ and a bit index k from $\{1, 2, \ldots, \ell(n)\}$ uniformly at random.

5. V' simulates V using $\tilde{m}$ and sends all messages sent by V to P_i up to round $j-1$ to P_2. P_2 sends a bit b to V', where b is the kth bit of the round-j message sent by P_i.

6. V' simulates V to check if V ever accesses the kth bit of P_i's round j message in $\vec{m}$. If V does not, then the protocol ends and $R' = 0$.

7. Finally, V' computes the payment R' as follows.

(a) If b does not match the kth bit of P_i's round-j message in $\tilde{m}$, $R' = -1$.
(b) Else, V' computes the payment R in $(V, \vec{P})$, and $R' = R/(2C(n))$.

Figure 10: Simulating an RIP protocol using 2 provers and reduced communication.

Furthermore, since any index i, j where $1 \leq i \leq \ell(n)$ and $1 \leq j \leq k(n)$ is of size at most $O(\log C(n))$, the communication complexity of the protocol (V', P_1', P_2') is $O(T(n) + \log C(n))$. Similarly, i, j, k can be randomly selected in $O(\log C(n))$ time, leading to total time $O(T(n) + \log C(n))$.

We now prove correctness of the protocol in Figure 10. Note that P_2' commits to a strategy profile s' of the provers $\vec{P}$ in step 5. We consider two cases.

Case 1. For some randomness r, suppose P_1' and P_2' do not agree on the effective transcript $\tilde{m}$. Then without loss of generality, there exist indices i, j, k such that the corresponding bit is part of the effective transcript, and the verification in step 7 fails with $R' = -1$. The probability that V' chooses such indices i, j, k in step 4 is at least $1/C(n)$. Thus, the expected payment of the provers is at most:

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

where the last inequality follows from the fact that $R \leq 1$. If P_1' and P_2' are consistent on $\tilde{m}$ and r (keeping the rest of their strategy the same) they can improve their expected payment since their payment under r would be least 0. Thus, this case does not occur under the best strategy profile of P_1' and P_2'.

Case 2. P_1' and P_2' agree on the effective transcript $\tilde{m}$ for every randomness r, but the answer bit c' under the strategy s' committed by P_2 for $\vec{P}$ is incorrect.

For a given randomness r and indices i, j, and k such that V accesses kth bit of the round-j message of P_i in $(V, \vec{P})$, $R' = R(s', x)/(2C(n))$, where $R(s', x)$ is the payment of protocol $(V, \vec{P})$ under strategy s'. By the correctness and utility gap of $(V, \vec{P})$, we know that the expected payment $u(s', x) + 1/\text{poly}(n) < u(s, x)$, where s is the best strategy of $\vec{P}$. Thus, in this case, P_1' and P_2' lose a polynomial amount if they use strategy s' instead of s. □

Lemma 4.3 demonstrates the importance of two provers over one in rational proofs to save on communication.

Corollary 4.4. *Since* $\mathsf{RIP}^{\mathsf{t}}[\mathsf{O}(\text{poly}(\mathsf{n})), \mathsf{O}(1), \mathsf{O}(1)] = \mathsf{P}_{||}^{\mathsf{NP}}$*, using Lemma 4.3, we have,* $\mathsf{P}_{||}^{\mathsf{NP}} \subseteq \mathsf{MRIP}^{\mathsf{t}}[\mathsf{O}(\text{poly}(\mathsf{n})), \mathsf{O}(\text{poly}(\mathsf{n}), \text{poly}(\mathsf{n})] \subseteq \mathsf{MRIP}^{\mathsf{t}}[\mathsf{O}(\log \mathsf{n}), \mathsf{O}(1), \mathsf{O}(1)]$.

To complete the proof Theorem 4.2, we prove the following upper bound.

Lemma 4.5. $\mathsf{MRIP}^{\mathsf{t}}[\mathsf{O}(\log(\mathsf{n})), \mathsf{O}(1)] \subseteq \mathsf{P}_{||}^{\mathsf{NP}}$.

The proof of Lemma 4.5 is similar to the proof of $\mathsf{MRIP} \subseteq \mathsf{EXP}^{||\mathsf{NP}}$ in Chapter 3.5.

Proof. Fix a language $L \in \mathsf{MRIP}(\log(\mathsf{n}), \log(\mathsf{n}), \mathsf{O}(1))$ and let $(V, \vec{P})$ be an MRIP protocol for L. Since V runs in $O(\log n)$ time, there exists a constant k such that, for any two payments

R and R' computed by V for some input of length n and some randomness are such that $R \neq R' \Rightarrow |R - R'| \geq \frac{1}{n^k}$.

Moreover, since V uses $O(\log n)$ coin flips, 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 $\frac{1}{n^{k'}}$. Therefore, for any input x of length n and any two strategy profiles s and s' of the provers, we have $|u(s,x) - u(s',x)| \geq \frac{1}{n^{k+k'}}$.

Consider the following deterministic oracle Turing machine M: Given any input x of length n, it divides the interval $[0,1]$ into $2n^{k+k'}$ subintervals of length $\frac{1}{2n^{k+k'}}$. For any $i \in \{1, \ldots, 2n^{k+k'}\}$, the ith interval is $[(i-1)/2n^{k+k'}, i/2n^{k+k'}]$. M then makes $4n^{k+k'}$ queries of the form (i,j), where $i \in \{1, ..., 2n^{k+k'}\}$ and $j \in \{0,1\}$. For each query (i,j), if $j = 0$ then the corresponding question is "whether there exists a strategy profile s of the provers such that $u(s,x)$ is in the ith interval"; and if $j = 1$ then the corresponding question is "whether there exists a strategy profile s such that $u(s,x)$ is in the ith interval *and the first bit sent by* P_1 *is* $c = 1$". Note that all queries are nonadaptive. We say that interval i is *non-empty* if the query $(i,0)$ is answered 1, and *empty* otherwise.

Given the answers to all the queries, 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. Given correct oracle answers, we show that M decides L.

For by the definition of MRIP, there exists a strategy profile whose expected payment is non-negative and thus in $[0,1]$. Thus there exists an interval i such that $(i,0)$ is answered 1. Also by definition, the best strategy profile s has the highest expected payment, and thus $u(s,x)$ falls into interval i^*.

Any strategy profile s' with $u(s',x) < u(s,x)$ has $u(s',x)$ not in interval i^*, since the difference between $u(s',x)$ and $u(s,x)$ is larger than the length of the interval. And so all strategy profiles s' with $u(s',x)$ in interval i^* satisfies $u(s',x) = u(s,x)$, that is, they are all the best strategy profiles of the provers. P_1 must send the same first bit c under all such 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 iff $c = 1$.

We now show that the oracle queries can be answered by an NP oracle. Since the communication complexity and verification complexity is $O(\log n)$ a strategy profile has size polynomial in n. Thus, an NP machine can guess a strategy profile $\tilde{s}$, simulate the protocol, and compute the expected payment $u(\tilde{s}, x)$. □

4.4 Verification in Logarithmic Space

The protocols in Chapter 4.3 have a polynomial utility gap. For a constant budget this means that the provers who mislead the verifier to an incorrect answer lose at least a polynomial amount from their expected payment.

As utility gap is analogous to the soundness gap in classical proofs, which is constant (independent of n), it is desirable to have rational protocols with constant utility gap as well.

Constant utility gap is difficult to achieve when the verifier is $O(\log n)$ time and cannot even read the entire input. This is true even for classical proofs with a $O(\log n)$-time verifier where the soundness conditioned is weakened to design "proofs of proximity" [19,20,82,124]. In particular, the soundness guarantees of such proofs depend on how far (usually in terms of hamming distance) the input string x is from the language L. Note that all existing $O(\log n)$-time rational proofs [11, 80, 81] have polynomial utility gap (under a constant budget).

To design protocols with a strong utility gap such as logarithmic or constant, in this section we consider verifier's that use only $O(\log n)$ space and randomness.

Let $\gamma(n)$ be any polynomial-time computable function (given 1^n) that is polynomially bounded. For example, $\gamma(n)$ can be a constant, $\log n$, or $\sqrt{n}$. We prove the characterization in general for a utility gap of $\gamma(n)$.

Theorem 4.6. *The class of languages that have a one-round RIP protocol with polynomial communication, $\gamma(n)$-utility gap and a verifier that uses $O(\log n)$ space and $O(\log n)$ randomness is exactly $\mathsf{P}_{||}^{\mathsf{NP}[\gamma(n)]}$. That is, $\gamma(\mathsf{n})\text{-}\mathsf{RIP}^{\mathsf{r},\mathsf{s}}[\mathrm{poly}(\mathsf{n}),1,1] = \mathsf{P}_{||}^{\mathsf{NP}[\gamma(n)]}$.*

First, we give a space-efficient RIP for the class NP using the log-space interactive proof for the language given by Condon and Ladner [47] as a blackbox.

Lemma 4.7. $\mathsf{NP} \in \gamma(\mathsf{n})\text{-}\mathsf{RIP}^{\mathsf{r},\mathsf{s}}[\mathrm{poly}(\mathsf{n}),1,1]$.

Proof. Let (V, P) denote the 1-round log-space interactive proof for a language $L \in \mathsf{NP}$ given in [47]. The one-round log-space RIP for Ł, (V', P') is given. P' sends message m' which is the concatenation of answer bit c with a bit string m. If $c = 0$, then m can be a null string. If $c = 1$, then m must be the message sent by P in (V, P). If $c = 0$, then $R = 1/2$ and the protocol ends. Otherwise, V' simulates V using m and if V accepts, then $R = 1$, else $R = 0$.

The verifier V' uses the same space as V, that is, $O(\log n)$. The communication of (V', P') is the same as (V, P), that is, polynomial in n.

We now argue correctness and show that the protocol has constant utility gap. Suppose $x \in L$ and P sends a message with answer bit $c' = 0$, then its expected payment is $1/2$. On the other hand, if P sends $c = 1$, its expected payment can be 1 by the completeness guarantee of (V, P). Thus in this case the expected payment loss of P is constant. Now

suppose $x \notin L$ and P sends the answer bit $c' = 1$. From the soundness guarantee of (V, P), V accepts with probability at most $1/3$, and thus the expected payment of P is at most $1/3$. On the other hand, if P sent $c = 0$, its expected payment would have been $1/2$. Thus in this case P loses a constant amount as well. □

For the lower bound, we use a different but equivalent complexity class. Recall that $\mathbf{L}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]}$ be a logarithmic space machine that can make $O(\gamma(n))$ nonadaptive queries to an NP oracle and $\mathbf{L}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]} = \mathsf{P}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]}$ [140]. We prove the following.

Lemma 4.8. $\mathsf{P}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]} = \mathbf{L}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]} \subseteq \gamma(\mathsf{n})\text{-}\mathsf{RIP}^{\mathsf{r},\mathsf{s}}[\mathrm{poly}(\mathsf{n}), 1, 1]$

Proof. Consider any language $L \in \mathbf{L}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]}$. Let M be the logarithmic-space oracle Turing machine with nonadaptive accesses to an NP oracle that decides L. Without loss of generality, suppose M makes exactly $\gamma(n) \geq 1$ nonadaptive oracle queries. The RIP protocol for L uses the RIP protocol for NP, given in Lemma 4.7, to simulate the oracle queries.

For any input x of length n, the protocol (V, P) works as follows. Let $R_n = 0$.

1. P sends a message $c, (c_1, m_1), (c_2, m_2), \ldots, (c_{\gamma(n)}, m_\gamma)$ to V, where c is the answer bit, and c_i is the answer to the ith oracle query q_i generated by M and m_i is the message for q_i based on Lemma 4.7. V outputs c at the end of the protocol.
2. V simulates M on x until M outputs queries $q_1, \ldots, q_{\gamma(n)}$.
3. For each $i \in \{1, \ldots, \gamma(n)\}$, V simulates V' in the RIP protocol for NP in Lemma 4.7. In particular, V uses the message c_i and m_i as the prover's messages in the protocol of Lemma 4.7. Let R_i^* be the payment for the ith round. V updates $R_n \leftarrow R_n + R_i^*$.
4. V continues simulating M till the end using $c_1, \ldots, c_{\gamma(n)}$. If c does not match M's output, then $R = -1$; otherwise $R = R_n/\gamma(n)$.

Figure 11: An RIP protocol for $\mathbf{L}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]}$.

We now prove correctness of the protocol in Figure 11. Note that an honest strategy of P, that is, reporting the correct answer bits $c, c_1, \ldots, c_{\gamma(n)}$, and sending correct proof strings m_i whenever $c_i = 1$, leads to a payment $R \geq 1/2$ (this is because of the payment-structure of the protocol for NP in Lemma 4.7).

Suppose P reports the incorrect answer bit c', then either (a) the output of M in Step 4 does not match c' and $R = -1$; or (b) there exists an oracle query q_i such that c_i is incorrect.

In case (a), the expected payment loss of P is at least $1/2 + 1 = 3/2 > 1/\gamma(n)$, as $\gamma(n) \geq 1$. In case (b), because the protocol in Lemma 4.7 has $O(1)$ utility gap, the provers' expected payment loss in the overall protocol is $1/O(\gamma(n))$. □

To complete the proof of Theorem 4.6 we prove the following upper bound.

Lemma 4.9. $\gamma(\mathsf{n})\text{-}\mathsf{RIP}^{\mathsf{r},\mathsf{s}}[\mathrm{poly}(\mathsf{n}),1,1] \subseteq \mathsf{P}_{||}^{\mathsf{NP}[\gamma(\mathsf{n})]}$

Proof. Given any $L \in \gamma(\mathsf{n})\text{-}\mathsf{RIP}$, let (V,P) be the a 1-round RIP protocol for L with $\gamma(n)$ utility gap, and $\mathrm{poly}(n)$ communication cost, where V uses $O(\log n)$ space and randomness.

Consider a polynomial-time Turing machine M which can make $O(\gamma(n))$ nonadaptive queries to an NP oracle. Given an input x of length n, M divides $[0,1]$ into $2\gamma(n)$ intervals, each of length $1/(2\gamma(n))$. That is, the ith interval is $[i/2\gamma(n),(i+1)/2\gamma(n))$ for each $i \in \{1,\ldots,2\gamma(n)-1\}$. For each such interval, M makes the following queries to its oracle:

1. Does there exist a message m sent by P such that the expected payment $u_{(V,P)}(s;x)$ is in the ith interval?
2. Does there exist a message m sent by P such that the expected payment $u_{(V,P)}(s;x)$ is in the ith interval and the corresponding answer bit $c=1$?

Note that M makes $O(\gamma(n))$ nonadaptive queries and clearly runs in polynomial time.

We now show that it is an NP machine can answer the oracle queries. The key point to note here is that protocol is one round and thus the size of a provers' strategy is polynomial in n. Thus, an NP oracle can guess a strategy and compute a payment $R(s,x,(V,P))$. Since the verifier uses $O(\log n)$ randomness, an NP machine can enumerate over all possible polynomial coin flips and compute the expected payment $u_{(V,P)}(s,x)$. If $u_{(V,P)}(s,x)$ is in the ith interval the oracle returns 1 to query (1) and 0 otherwise. Similarly, if $u_{(V,P)}(s,x)$ is in the ith interval and $c=1$ for this strategy, the oracle responds 1 to query (2) and 0 otherwise. Thus, M's queries can be answered by an NP oracle.

Finally, M finds the highest index i^* such that the oracle returns 1 to query (1) with respect to the i^*. M accepts if the oracle returns 1 to query 2 for the i^*th interval, and rejects otherwise. Note that M clearly runs in polynomial time.

M decides L given correct answers to its oracle queries, because the maximum expected payment $u_{(V,P)}(s^*,x)$ falls in the i^*th interval. As $(V,\vec{P})$ has $\gamma(n)$-utility gap, by construction and definition of utility gap, all strategies with expected payments in the i^*th interval must have the same answer bit c as that in s^*. Thus, $x \in L$ if and only if $c=1$, which occurs if and only if the oracle's answer to query 2 for interval i^* is 1. □

Chapter 5

Relationship Between Classical and Rational Proofs

5.1 Introduction

In Chapter 3 and Chapter 4, we designed several simple and efficient MRIP protocols with strong utility-gap guarantees for important and powerful complexity classes.

In this chapter, we closely compare the guarantees provided by rational interactive proofs to that of classical interactive proofs.

Classical interactive proofs provide *completeness* and *soundness* guarantees. In particular, they guarantee that if $x \in L$, there exists a strategy of the provers such that the verifier accepts with probability at least $2/3$ (completeness), and if $x \notin L$, then no strategy of the provers can make the verifier accept with probability more than $1/3$ (soundness).

On the other hand, the strength of the guarantee provided by rational proofs is captured by the notion of *utility gap*. The high level idea behind utility gap is that provers who are not perfectly rational may not care about small losses in payments and may lazily give the incorrect answer. If a rational protocol has a utility gap of u, then the provers who mislead the verifier to an incorrect answer are guaranteed to lose at least $1/u$.

Utility gap in rational proofs is analogous to the difference between completeness and soundness in rational proofs. In this chapter, we make this connection explicit. In particular, we construct a condition on the expected payments and utility gap of a rational proof which, if satisfied, turns it into a classical proof with completeness and soundness guarantees.

First, we show that any rational proof protocol can be converted to one where the payments are 0 or 1, where 1 represents verifier's "acceptance" of the provers' claim and 0 represents the verifier's "rejection" of the provers' claim. In such a protocol, the probability of acceptance is then exactly equal to the expected payment of the provers. We then use this to prove that if the expected payments of all inputs $x \in L$ are noticeably far away from that of all inputs $x \notin L$, the rational protocol can be converted to a classical interactive protocol.

This result demonstrates the usefulness of studying the *incentives* of the provers in a proof system. The way interactive proofs are presented in general, the provers are said to be *malicious*, however, they are not really malicious in the true sense of the term. Instead, classical provers only have one goal—to maximize the probability of the verifier accepting the claim $x \in L$. If x happens to be in L, all provers are honest, and if x happens not be in

L, all provers are malicious.

Rational proofs, by virtue of being designed with the objective of specifically incentivizing provers to give the "correct" answer, have guarantees that depend on the string x. In particular, if an $x \in L$, then each prover strategy that reports $x \in L$ is honest and all other strategies are dishonest. Similarly, if $x \notin L$, then each prover strategy that reports $x \notin L$ is honest and all other are dishonest.

Thus, rational proofs incentivize the provers to always use an honest strategy and rational proofs with a strong utility gap guarantee ensures that the expected payment of a dishonest strategy is significantly less than an honest one.

5.2 Converting a Rational Proof to a Classical Proof

We show under what conditions does a rational interactive proof reduces to a classical interactive proof. The results in this section are stated in terms of the multi-prover model (that is, MRIP and MIP) which is more general, and thus they also hold for the single prover model (that is, RIP and IP).

To compare the two proof models, we explore their differences. In rational interactive proofs, the provers are allowed to give an answer bit $c = 1$ claiming $x \in L$ or $c = 0$ claiming $x \notin L$.[8] In other words, the question is "is $x \in L$?" and the rational provers can say "yes" or "no" based on their incentives. Furthermore, for a particular input x of size n, if the provers' claim c about x is incorrect, they lose at least a $1/\gamma(n)$, where $\gamma(n)$ is the utility gap.

On the other hand, in classical proofs, the provers are only allowed to prove membership, that is, $x \in L$. Furthermore, given completeness and soundness parameters c and s respectively, where $0 \leq s < c \leq 1$, we have that "for any $x \in L$", there exists a strategy such that V accepts with probability at least c and "for any $x \notin L$", for any strategy V rejects with probability at most s. Thus, given L, the guarantees are independent of x.

In this section, we show under what conditions a rational proof reduces to a classical proof. Intuitively, this happens when the utility gap guarantee of a rational protocol is made to hold for all x and in particular, it is enforced to be the gap between the expected payments for all $x \in L$ and all $x \notin L$.

We first show that without loss of generality we can restrict the expected payments of the provers in a rational protocol to be either 1 or 0, where 1 corresponds to "accept" and 0 to "reject" respectively.

Lemma 5.1. *Any MRIP protocol $(V, \vec{P})$ with payment $R \in [0, 1]$ and utility gap $\gamma(n)$ can be simulated by an MRIP protocol $(V', \vec{P})$ with payment $R' \in \{0, 1\}$ and utility gap $\gamma(n)/2$. In*

[8] Thus it is not surprising that rational proofs are closed under complement.

particular, for any strategy s and any input x,

$$u_{(V,\vec{P})}(x;s) \leq u_{(V',\vec{P})}(x;s) \leq u_{(V,\vec{P})}(x;s) + \gamma(n)/2.$$

V' uses $1 + \lceil \log_2 \gamma(n) \rceil$ more random bits than V.

Proof. We go one by one through the payments made in the protocol $(V, \vec{P})$ and replace them with payments in $\{0, 1\}$. At a high level, V' makes a payment 1 in $(V', \vec{P'})$ with probability $R(x, r, \vec{m})$ where $R(x, r, \vec{m})$ is the payment made by V in $(V, \vec{P})$, and 0 otherwise.

Assume without loss of generality that each random string r is a bit string that corresponds exactly to the result of $|r|$ coin flips. Let $G = 2^{1+\lceil \log_2 \gamma(n) \rceil}$; thus $1/G \leq 1/2\gamma(n)$.

We create a new protocol $(V', \vec{P'})$. First, V' runs the original protocol $(V, \vec{P})$ to obtain a transcript $\vec{m}$ (given the provers' strategy s) and randomness r of V. Let $R(x, r, \vec{m})$ be the payment made by V. Then, V' flips $1 + \lceil \log_2 \gamma(n) \rceil$ extra coins; call this string r'. Momentarily treat r' as an integer. If $r' \leq \lceil G \cdot R(x, r, \vec{m}) \rceil$ then V' pays 1; otherwise V' pays 0. Let this final payment be denoted by $R(x, r \circ r', \vec{m})$.

Note that in the above, because r' is uniformly selected from G distinct values, for any r, the V' pays 1 with probability

$$\mathbf{E}_{r'}[R(x, r \circ r', \vec{m})] = \frac{\lceil G \cdot R(x, r, \vec{m}) \rceil}{G};$$

thus

$$R(x, r, \vec{m}) \leq \mathbf{E}_{r'}[R(x, r \circ r', \vec{m})] \leq R(x, r, \vec{m}) + 1/G \leq R(x, r, \vec{m}) + 1/2\gamma(n).$$

The expected payment in $(V, \vec{P'})$, given input x and transcript $\vec{m}$, is $\sum_r R(x, r, \vec{m}) \Pr(r)$. The expected payment in $(V', \vec{P})$ is (using independence of r' and r)

$$\mathbf{E}_{r,r'}[R(x, r \circ r', \vec{m})] = \sum_r \sum_{r'} R(x, r \circ r', \vec{m}) \Pr(r') \Pr(r) = \sum_r \mathbf{E}_{r'}[R(x, r \circ r', \vec{m})] \Pr(r).$$

Substituting with the above, the expected payment is bounded by

$$\sum_r R(x, r, \vec{m}) \Pr(r) \leq \mathbf{E}_{r,r'}[R(x, r \circ r', \vec{m})] \leq 1/2\gamma(n) + \sum_r R(x, r, \vec{m}) \Pr(r). \qquad \square$$

Any rational protocol with zero-one payments immediately gives us an accept-reject protocol such that for a given x, the probability that the verifier accepts is exactly the expected payment of the original protocol. More formally let $(V, \vec{P})$ be a rational protocol with $R \in \{0, 1\}$ and utility gap $\gamma(n)$. Let $(V', \vec{P'})$ be defined as follows: V' simulates V,

ignores the answer bit c, and if the payment in $(V, \vec{P})$ is $R = 1$ then accept, else reject.

Thus, for a given input string x, the expected payment in $(V, \vec{P})$ is equal to the probability that V' accepts in $(V', \vec{P}')$. That is,

$$u_{(V,\vec{P})}(x; s) = \mathbf{E}_r[R(x, r, (V, \vec{P})(x, r, s))] = \sum_r \Pr(r \mid R(x, r, (V, \vec{P})(x, r, s)) = 1)$$
$$= \sum_r \Pr(r \mid V' \text{accepts } (V', \vec{P}')) = \Pr(V' \text{ accepts } (V', \vec{P}')). \tag{3}$$

Furthermore, $(V', \vec{P}')$ satisfies the following instance-specific properties similar to completeness and soundness in interactive proofs. For any $x \in L$, let s^* denote the optimal strategy of the provers $\vec{P}$, that is, s^* maximizes their expected payment. Then for $\vec{P}'$ following s^*, V' accepts with probability exactly $c(x, n) = u_{(V,\vec{P})}(x; s^*)$. Furthermore, we know from the utility gap condition that for any $x \notin L$, for any strategy s', the probability that V' accepts is at most $u_{(V,\vec{P})}(x; s') < u_{(V,\vec{P})}(x; s^*) - 1/\gamma(n)$, that is, the probability that V' accepts is at most $s(x, n) < c(x, n) - 1/\gamma(n)$. Similar guarantees hold for any $x \notin L$.

However, if we want $(V', \vec{P}')$ to be an interactive proof protocol with completeness and soundness guarantees that hold for all $x \in L$ and for all $x \notin L$ respectively, we need to impose restrictions on the expected payments of the rational proof protocol.

Theorem 5.2. *Let $(V, \vec{P})$ be an MRIP protocol for a language L such that*

$$\min_{x \in L} u_{(V,\vec{P})}(x; s^*) > \max_{x \notin L} u_{(V,\vec{P})}(x; s^*) + \frac{1}{\gamma(n)} \tag{4}$$

where x is any input of length n, s^* is the strategy of the provers that maximizes their expected payment in $(V, \vec{P})$ and $\gamma(n)$ is any function such that $\gamma(n) > 1$ and $\gamma = O(\text{poly}(n))$. Then, $(V, \vec{P})$ can be simulated by a MIP protocol for L.

We prove this theorem in two parts. First, we show prove the following lemma which proves Theorem 5.2 with weak completeness and soundness guarantees.

Lemma 5.3. *Let $(V, \vec{P})$ be an MRIP protocol for a language L that satisfies the condition 4 in Theorem 5.2. Then, $(V, \vec{P})$ can be simulated by MIP protocol with completeness and soundness $c(n)$ and $s(n)$ respectively such that $c(n) > s(n) + 1/2\gamma(n)$ and $c(n), s(n) \geq 0$.*

Proof. Using Lemma 5.1, without loss of generality, let the payment of $(V, \vec{P})$ be $R \in \{0, 1\}$. Since the expected reward under each strategy is changed by at most $1/2\gamma(n)$, the condition of Theorem 5.2 is still satisfied with $\gamma(n) \leftarrow 2\gamma(n)$. In the MIP protocol $(V', \vec{P})$, V' simulates V, ignores the answer bit c, and if the payment in $(V, \vec{P})$ is 1, then accepts, else rejects. The expected payment in $(V, \vec{P})$ is equal to the probability that V' accepts; see Equation 3.

Define $c(n) = \min_{x \in L} u_{(V,\vec{P})}(x, r, s^*)$ and $s(n) = \max_{x \notin L} u_{(V,\vec{P})}(x, r, s^*)$. Then, by definition, we have $c(n), s(n) \geq 0$ and $c(n) > s(n) + 1/2\gamma(n)$.

We now show that $c(n)$ and $s(n)$ are the completeness and soundness of the MIP $(V', \vec{P'})$ respectively. For any $x \in L$, there exists $\vec{P'}$ where $\vec{P'}$ uses a strategy s^*, such that the probability V' accepts is exactly $u_{(V,\vec{P})}(x, s^*) \geq c(n)$. For any $x \notin L$, for all $\vec{P'}$ using any strategy s, the probability V' accepts is exactly $u_{(V,\vec{P})}(x, s) \leq u_{(V,\vec{P})}(x, s^*) \leq s(n)$. Thus, $(V', \vec{P'})$ is an MIP with completeness and soundness $c(n)$ and $s(n)$ respectively. □

We amplify the "gap" of an MIP by repeating the protocol sufficiently many times and then using Chernoff bounds. The techniques are mostly standard, although the parameters must be set carefully to deal with the case $s(n) = 0$.

Lemma 5.4. *Given an MIP protocol for a language L, with completeness $c(n) > 0$ and soundness $s(n) \geq 0$ such that $c(n) > s(n) + 1/\gamma'(n)$ for some $\gamma'(n) > 1$ and $\gamma' = O(\text{poly}(n))$, can be converted to an MIP protocol for L with completeness at least $1 - 1/\text{poly}(n)$ and soundness at most $1/\text{poly}(n)$.*

Proof. We repeat the MIP protocol $\rho(n) = 96(\log n)\gamma'(n)^2/c(n)$ times and accept if more than $\tau(n) = \rho(n)c(n)(1 - 1/4\gamma'(n))$ of the instances end in accept.

Let the random indicator variable X_i be 1 if the verifier in the ith repetition accepts, otherwise $X_i = 0$. Let $X = \sum_{i=1}^{\rho(n)} X_i$ be the total number of accepts.

Consider an $x \in L$. Then if provers use their best strategy of the original MIP protocol in each iteration, we have

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{i=1}^{\rho(n)} X_i\right] = \sum_{i=1}^{\rho(n)} \mathbf{E}\left[X_i\right] \geq \sum_{i=1}^{\rho(n)} c(n) = c(n)\rho(n).$$

Using Chernoff bounds, we obtain[9]

$$\begin{aligned}\Pr(X < \tau(n)) &= \Pr\left(X < \left(1 - \frac{1}{4\gamma'(n)}\right) c(n)\rho(n)\right) \\ &\leq e^{-\frac{c(n)\rho(n)}{32\gamma'(n)^2}} < 1/n.\end{aligned}$$

Now consider an $x \notin L$. For any strategy of the provers, by the soundness guarantee at most $s(n)$ of the original protocol (and using linearity of expectation as above) we have $\mathbf{E}(X) \leq s(n)\rho(n) < \rho(n)(c(n) - 1/2\gamma'(n))$. Note $c(n) - 1/2\gamma'(n) > c(n)/2$ and $\tau(n) >$

[9]This uses a slight extension of Chernoff bounds that uses a bound on the expectation rather than the expectation itself; see Exercise 4.7 in [110] for example.

$\rho(n)(c(n) - 1/2\gamma'(n))(1 + 1/4\gamma'(n))$. Then we can use the following Chernoff bound

$$\begin{aligned}\Pr(X > \tau(n)) &\leq \Pr\left(X > \left(1 + \frac{1}{4\gamma'(n)}\right)\rho(n)(c(n) - 1/2\gamma'(n))\right) \\ &\leq e^{-\rho(n)(c(n)-1/2\gamma'(n))/48\gamma'(n)^2} \leq e^{-\rho(n)c(n)/96\gamma'(n)^2} \leq 1/n.\end{aligned}$$

Note that the same analysis extends to any $1/\text{poly}(n)$ instead of $1/n$ when $\rho(n)$ is increased by a constant. □

Remark 5.5. *The repetition of the MIP protocol to amplify its completeness and soundness guarantee used in Lemma 5.4 is not efficient as it blows up the number of rounds. There exist more efficient techniques to amplify IP guarantees by parallel repetition that can be used instead; for example, see [16, 62, 74, 122].*

Chapter 6

Rational Proofs with Non-Cooperative Provers

6.1 Introduction

The motivation behind the study of efficient interactive proofs and rational interactive proofs is largely to design efficient computation-outsourcing protocols. However, in all existing interactive-proof systems with multiple provers, the provers' interests either perfectly align as in multi-prover interactive proofs and cooperative rational proofs (see Chapter 3, Chapter 4 and Chapter 5), or directly conflict as in refereed games (see Chapter 2). Neither of these extremes truly capture the strategic nature of service providers in outsourcing applications. How to design and analyze non-cooperative interactive proofs is an important open problem.

In this chapter, we introduce the model of rational proofs with *non-cooperative* provers [44], where each prover acts selfishly to maximize its utility in the resulting game.

This incentive model is different from all interactive-proof systems with multiple provers. In classical multi-prover interactive proofs [17], the provers are totally cooperative with each other and their utility is equal to the probability that the verifier accepts the statement $x \in L$. On the other hand, in refereed games [40, 63, 65, 67], the provers compete with one another in a zero-sum game. The utility of each prover in a refereed game is the probability that the verifier accepts their claim $x \in L$ or $x \notin L$ respectively.

Several computation-outsourcing protocols that make use of either MIP or refereed games have been designed [26,35–37,97]. However, the service providers in outsourcing applications, may neither be completely cooperative nor completely conflicting.

In this chapter, we use a mechanism design approach to define the new interactive-proof model, *non-cooperative rational interactive proofs (ncRIP)*, in which each prover acts to maximize its own expected utility given other provers' strategies. Unlike refereed games, no prover is required to be honest.

In the non-cooperative setting, the utility of non-cooperative provers may no longer directly correspond to the verifier's acceptance probability (as in MIP and refereed games). To assign meaningful incentives, we draw on our cooperative model of rational proofs in Chapter 3, in which the provers' utility is a monetary payment given at the end of the protocol. The provers are cooperative and rational—that is, they work together to maximize their total expected payment received from the verifier. Since the provers cooperate, rational

proofs are not too different from MIP—the total payment is analogous to the verifier's acceptance probability (see Chapter 5 for a detailed discussion on this).

We borrow the notion of payments as utilities in our new model with the difference that instead of a total payment, each prover receives an individual expected payment. While this leads to a well-defined utility structure, there are still significant challenges in designing the overall model of a non-cooperative proof system and in understanding its full power.

In particular, we need to (a) select a meaningful equilibrium concept for the specific game structure of interactive proofs, (b) design protocols where the verifier learns the correct answer under such an equilibrium, and finally (c) bound the exact power of such equilibria-based proofs. In this work, we address these challenges.

Finding the right solution concept. We describe the main obstacles to finding an appropriate solution concept. First, the solution concept should not be susceptible to empty threats. This rules out Nash and maximum Nash as they cannot handle empty threats.

It is worth pointing out that a possible approach to defining non-cooperative rational proofs is to simulate an MRIP protocol using non-cooperative provers, and split the final payment evenly among them and analyze the resulting game using maximum Nash. While such an approach may not suffer from empty threats, it forces non-cooperative provers to act cooperatively and fails to lead us to a more meaningful and general non-cooperative model.

Second, the solution concept should have an elegant way to handle provers' beliefs at unreachable information sets. In particular, we want to avoid artificial consistency requirements on the beliefs of the players at these information sets, like those of sequential equilibrium (SE).[10] Finally, the solution concept must handle equilibrium selection.

Strong sequential equilibrium. Taking the above constraints into account, we define a new solution concept to help analyze ncRIP: *strong sequential equilibrium* (SSE). SSE is a refinement of SE which, unlike SE, may not always exist. Thus, it is up to the mechanism designer to ensure that their protocol does have an SSE. We believe that SSE is of independent interest as a solution concept for designing extensive-form mechanisms (e.g. [57, 72, 135]). In Chapter 6.3, we prove important properties of SSE that are used in this dissertation and may prove useful in future studies.

To resolve the problem of equilibrium selection, we define a maximum variant of SSE and apply it recursively to a larger class of "subgames."[11] Informally, in a rational proof with

[10] To quote Kreps, one of the inventors of SE, "rather a lot of bodies are buried in this definition" [100,117].

[11] Similar to subgame-perfect equilibrium, the recursive aspect is to ensure that provers play their best response at *each* subform, an extension of subgame for imperfect-information games.

non-cooperative provers, whenever the strategy profile used by the provers is a *recursive-maximum SSE* (rmSSE), it leads the verifier to the correct answer. The ncRIP framework is formally defined in Chapter 6.2.

Utility gap for non-cooperative provers. Similar to the model of cooperative rational proofs, we need a notion of utility gap for the non-cooperative model as well so that we can analyze the guarantees provided by the protocols.

This notion is straightforward to define for cooperative MRIP protocols—they have a utility gap of u if the *total* expected payment decreases by $1/u$ whenever the provers report the wrong answer. In the non-cooperative model, however, it is not a priori clear how to define such a payment loss or and which prover to impose it on. A payment loss imposed solely on the total payment may not prevent some provers from deviating, and a loss solely imposed a prover's final payment may not prevent it from deviating within subgames.

In this chapter, we define a meaningful notion of utility gap for ncRIP that captures how the provers reason in the resulting extensive-form game, and is naturally incorporated into the framework of recursive-maximum SSEs.

6.2 Model and Preliminaries

In this section we define the model for ncRIP formally.

Notation. The interactive model of non-cooperative rational proofs is similar to interactive proofs described in Chapter 2, except for the following differences.

Similar to MRIP, in ncRIP, a *round* of interaction consists of either messages sent in parallel by all or some provers to the verifier or messages sent by the verifier to all or some provers, 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 $c = 0$). The length of each message $\ell(n)$, and the number of rounds $k(n)$ are both polynomial in n. Let r be the random string used by V. Given r, let $\vec{m}$ be the final transcript.

At the end of the communication, based on x, r, and $\vec{m}$, the verifier computes an *answer bit* $c \in \{0, 1\}$ for the membership of x in L, and a payment vector $\vec{R} = (R_1, R_2, \ldots, R_{t(n)})$, where R_i is the payment given to P_i and $R_i \in [-1, 1]$, and the total $\sum_{i=1}^{p(n)} R_i \in [-1, 1]$ as well.[12] The protocol and the payment function $\vec{R}$ are public knowledge.

[12]Negative payments are used to reflect punishment. They can be shifted and scaled to lie in $[0, 1]$. Similarly, we may allow V's total budget to be a larger constant for simplicity as it can be scaled down.

Each prover P_i's strategy at round j maps the transcript seen at the beginning of round j to the message it sends in that round. Let $s_i = (s_{i1}, \ldots, s_{ik(n)})$ be the *strategy* of prover P_i, and $s = (s_1, \ldots, s_{t(n)})$ be the strategy profile of the provers. Given input x, and strategy profile s, let $\mu_k(x, s, (V, \vec{P}))$ denote the expected payment of prover P_k in the protocol $(V, \vec{P})$ based on r, x and s; if $(V, \vec{P})$ is clear from context, this is shortened to $\mu_k(x, s)$.

At a high level, the provers choose their individual strategy to maximize their own payment; we formalize the solution concept in Chapter 6.3. The protocol and solution concept should be such that when the provers reach an equilibrium, V learns the correct answer c.

Extensive-form games formed by IP protocols. An interactive proof protocol results in an *extensive-form game with imperfect information*. The game tree is naturally induced by the possible coin flips and messages of the verifiers, as well as the possible messages of the provers. For a detailed exposition on extensive-form games, we refer the readers to the textbook by Osborne and Rubinstein [117].

In a protocol $(V, \vec{P})$ with input x, the set of provers $\vec{P} = (P_1, \ldots, P_{t(n)})$ are the *players* and the verifier V's random coin flips are treated as the moves of *Nature*. The *history* $h = (a^1, a^2, \ldots, a^K)$ of a decision node is the sequence of actions taken by Nature and the players along the path from the root to the decision node. The set of valid histories (including ϕ, the empty history corresponding to the root) is denoted by H.

A history h is *terminal* if it belongs to a leaf in the game tree, and *non-terminal* otherwise. Let $Z(h)$ denote the player whose turn it is to act following a non-terminal history h—note that even though in an ncRIP protocol more than one prover may send a message to the verifier in a round, without loss of generality we can increase the number of rounds such that only a single prover acts in each round. Let $A(h)$ denote the set of actions available to the acting player at a non-terminal history h: that is, $A(h) = \{a : (h, a) \in H\}$. If $Z(h)$ is Nature, then $A(h)$ is the set of possible coin flips and messages of the verifier following h; otherwise $A(h)$ is the set of possible messages that $Z(h)$ may send to the verifier. For each terminal history h, the *utility* of a player i following h, $u_i(h)$, is the payment R_i computed by the verifier given x and h.

Since the verifier's coins are private and a prover does not see the messages exchanged between the verifier and the other provers, an IP protocol represents an extensive-form game of imperfect information.

An *information set* I_i of a player P_i is a subset of all possible histories h with $Z(h) = P_i$, and represents all the information that the player knows when acting in one of the decision nodes in I_i. That is, when a decision node in I_i is reached, P_i knows that I_i has been reached but does not know exactly which node it is at. Naturally, $A(h) = A(h')$ for all $h, h' \in I_i$

—that is, the set of actions available to player i at every decision node in a particular information set is the same. Let $A(I_i)$ denote the set of available actions at an information set I_i. The set of all information sets of P_i forms a partition of the set $\{h \in H : Z(h) = P_i\}$, and let $\mathcal{I}_i$ to denote this partition, referred to as the *information partition* of P_i. In terms of the protocol, $\mathcal{I}_i$ is in a one-to-one correspondence with the set of possible message sequences $(m_{i1}, \ldots, m_{ij})$ seen by P_i, where $j \in \{1, \ldots, p(n)\}$ and P_i is acting in round j.

A *pure strategy* s_i of a player P_i in an extensive-form game is a function that assigns an action in $A(I_i)$ to each information set $I_i \in \mathcal{I}_i$. A *behavioral strategy* β_i of P_i is a collection $(\beta_i(I_i))_{I_i \in \mathcal{I}_i}$ of independent probability measures, where $\beta_i(I_i)$ is a probability measure over the action set $A(I_i)$. A behavioral strategy β_i is *completely mixed* if each $\beta_i(I_i)$ assigns a positive probability to every action in $A(I_i)$.

In our protocols we only consider deterministic provers and thus analyze only pure strategies. However, our solution concept applies to behavioral strategies as well.

A player i's utility under a strategy profile s, $u_i(s)$, is its expected utility over the distribution of histories induced by s and the verifier's randomness.

The provers are computationally unbounded and never "forget" anything and thus the game has *perfect recall*. That is, for any two histories h and h' in the same information set I_i of a player P_i, h and h' pass the same sequence of information sets to player P_i. Furthermore, for any information set in this sequence, player P_i took the same action in h and h'. This holds in any ncRIP protocol since all histories of prover P_i in the same information set I_i at round j correspond to the sequence of messages $(m_{i1}, \ldots, m_{ij})$ seen by P_i up to round j.

6.3 Strong Sequential Equilibrium and its Properties

We formally define our refinement of sequential equilibrium—*strong sequential equilibrium*.

Recall that a sequential equilibrium considers a strategy profile s together with a *belief system* μ, where μ specifies, for each information set I, the probability assigned to each history in I. SE imposes a *consistency condition* on s and μ: s and μ are *consistent* if there exists a sequence $(s^t, \mu^t)_{t=1}^{\infty}$ that converges to (s, μ), such that each s^t is a profile of completely mixed behavioral strategies and each μ^t is the belief system derived from s^t using Bayes' rule. The pair (s, μ) is an SE if s and μ are consistent and conditioned on any information set I_i being reached by a player i, player i's strategy is a best response to the others' given i's beliefs at I_i (specified by μ). This condition is called *sequential rationality*.

We introduce a refinement of SE that avoids the somewhat artificial consistency condition of SE, which involves computing limits of sequences to argue about players' beliefs at *unreachable information sets*. An information set is unreachable if it is reached with

probability 0 under a equilibrium strategy s.

A strong sequential equilibrium is identical to SE for reachable information sets. At any unreachable information set I, SSE requires that the acting player's strategy be a best response to the others, for *any beliefs* it may hold at I.

Definition 6.1. *(Strong Sequential Equilibrium) A strategy profile s is a* strong sequential equilibrium *(SSE) if for every player i and information set I_i of i:*

- **If I is reachable under s:** *conditioned on I_i being reached, player i's strategy s_i is a best response to s_{-i}, given i's beliefs at I_i being derived from s using Bayes' rule.*
- **If I is not reachable under s:** *conditioned on I_i being reached, player i's strategy s_i is a best response to s_{-i}, given any beliefs of i at I_i.*

Strong sequential equilibrium strengthens SE so that a player's beliefs at unreachable information sets are *irrelevant* in justifying its equilibrium strategy. Unlike SE, not every extensive-form game with perfect recall has an SSE. However, in the context of mechanism design, SSE allows for the design of stronger mechanisms. This is reminiscent of dominant-strategy equilibrium, and the difference between analyzing general games and designing specific mechanisms. Not every game has a dominant-strategy equilibrium, and it cannot be used to analyze general games. However, if the designer can design a mechanism with a dominant-strategy equilibrium, it obtains a strong guarantee on the behavior of the players.

Although we introduce SSE to analyze ncRIP protocols, we believe it is of independent interest as a solution concept for mechanisms based on extensive-form games. Next, we prove several important properties of strong sequential equilibrium.

Strong Sequential Equilibrium Admits a Sequential Equilibrium. We show that, given a strategy profile s that is a strong sequential equilibrium (thus does have a belief system), we can construct a belief system μ such that the pair (s, μ) is a sequential equilibrium.

Lemma 6.2. *For any strategy profile s that is a strong sequential equilibrium, there exists a belief system μ such that (s, μ) is a sequential equilibrium.*

Proof. To prove that s admits a sequential equilibrium, we first construct a belief system μ and show that, there exists a sequence of pairs $(s^\varepsilon, \mu^\varepsilon)_{\varepsilon \to 0}$ which converges to (s, μ), as ε goes to 0, where each s^ε is a profile of completely mixed behavioral strategies and each u^ε is the belief system derived from s^ε using Bayes' rule.

Recall that a strategy profile s defines a probability distribution over the actions available to a player at an information set where it acts. That is, for each information set I_i of a player i, $s_i(I_i)$ is a probability distribution over $A(I_i)$, the set of actions available to player i at

I_i. In particular, if $A(I_i) = (a_1, \ldots, a_k)$, then $s_i(I_i) = (p_i(a_1), \ldots, p_i(a_k))$ where $p_i(a_\ell)$ is the probability that player i chooses action a_ℓ at I_i.

Let $A^+(I_i)$ and $A^0(I_i)$ be the set of actions at information set I_i which player i chooses with positive probability and zero probability respectively; that is,

$$A^+(I_i) = \{a_\ell \in A(I_i) \mid p_i(a_\ell) > 0\} \text{ and } A^0(I_i) = A(I_i) \setminus A^+(I_i).$$

For any $\varepsilon \in (0, 1)$, we define s_i^ε for player i at information set I_i as follows: if $A^0(I_i) = \emptyset$ then $s_i^\varepsilon(I_i) = s_i(I_i)$; otherwise,

$$s_i^\varepsilon(I_i)(a_\ell) = \begin{cases} (1-\varepsilon) \cdot p_i(a_\ell) & \text{for each } a_\ell \in A^+(I_i); \\ \frac{\varepsilon}{|A^0(I_i)|} & \text{for each } a_\ell \in A^0(I_i). \end{cases}$$

By construction, $s_i^\varepsilon(I_i)$ is a valid probability distribution over I_i and is completely mixed, that is, assigns a positive probability to every action in I_i. Indeed, because $\sum_{\ell=1}^k p_i(a_\ell) = \sum_{a_\ell \in A^+(I_i)} p_i(a_\ell) = 1$, when $A^0(I_i) \neq \emptyset$ we have $\sum_{a_\ell \in A(I_i)} s_i^\varepsilon(I_i)(a_\ell) = \sum_{a_\ell \in A^+(I_i)} (1 - \varepsilon) p_i(a_\ell) + \varepsilon = 1$. It is easy to see that s_i^ε converges to s_i when $\varepsilon \to 0$.

Given the strategy profile s^ε, to define μ_i^ε, the belief system of a player i, consider an arbitrary information set I_i where player i acts. The probability that a particular history $h = (a^1, \ldots, a^K) \in I_i$ occurs can be derived from s^ε as follows. For any history $h' = (a^1, \ldots, a^w)$ with $0 \leq w \leq K-1$, recall that $Z(h')$ is the player acting following history h'. For any action $a \in A(h')$, let $s_{Z(h')}^\varepsilon(h')(a)$ denote the probability assigned by $s_{Z(h')}^\varepsilon$ to action a at history h' (i.e., at the information set containing h'). We have

$$\Pr\{h \text{ occurs under } s^\varepsilon\} = \prod_{w=0}^{K-1} s_{Z(a^1,\ldots,a^w)}^\varepsilon(a^1, \ldots, a^w)(a^{w+1}) = c_h \varepsilon^{e_h} (1-\varepsilon)^{f_h},$$

where c_h, e_h and f_h are positive constants depending on s and h, but not on ε. In particular, letting S^0 be the set of actions a^{w+1} in h that are assigned zero probability by $s_{Z(h')}$ at history $h' = (a^1, \ldots, a^w)$, we have $e_h = |S^0|$. f_h is the number of actions a^{w+1} in h such that a^{w+1} is not in S^0 but $s_{Z(h')}$ is not completely mixed at h' either. Finally,

$$c_h = \prod_{\substack{0 \leq w \leq K-1 \\ a^{w+1} \notin S^0}} s_{Z(a^1,\ldots,a^w)}(a^1, \ldots, a^w)(a^{w+1}) \cdot \prod_{\substack{0 \leq w \leq K-1 \\ a^{w+1} \in S^0}} \frac{1}{|A^0(a^1, \ldots, a^w)|},$$

where the second term is defined to be 1 if $S^0 = \emptyset$. Note that $\Pr\{h \text{ occurs under } s^\varepsilon\} > 0$ for every $h \in I_i$.

The probability that the information set I_i is reached under s^ε is

$$\mathcal{P}(I_i) \triangleq \sum_{h \in I_i} \Pr\{h \text{ occurs under } s^\varepsilon\} = \sum_{h \in I_i} c_h \varepsilon^{e_h} (1-\varepsilon)^{f_h} > 0.$$

Then $\mathcal{P}(I_i)$ can be written as a polynomial in ε, that is, $\mathcal{P}(I_i) = b_0 + b_1\varepsilon + b_2\varepsilon^2 + \ldots + b_r\varepsilon^r$, where the coefficients $b_0, \ldots, b_r$ may be zero, positive or negative. Following Bayes' rule, for any history $h \in I_i$,

$$\mu_i^\varepsilon(I_i)(h) = \frac{c_h \varepsilon^{e_h}(1-\varepsilon)^{f_h}}{\mathcal{P}(I_i)} = \frac{c_h \varepsilon^{e_h}(1-\varepsilon)^{f_h}}{b_0 + b_1\varepsilon + b_2\varepsilon^2 + \ldots + b_r\varepsilon^r} > 0.$$

To define the belief system μ, let d be the minimum degree of ε in $\mathcal{P}(I_i)$ such that $b_d \neq 0$. As the minimum degree of ε in each term $c_h\varepsilon^{e_h}(1-\varepsilon)^{f_h}$ is e_h with coefficient $c_h > 0$, we have $d = \min_{h \in I_i} e_h$ and $b_d = \sum_{h \in I_i, e_h = d} c_h > 0$. For any $h \in I_i$, we define $\mu_i(I_i)(h) = c_h/b_d (> 0)$ if $e_h = d$, and $\mu_i(I_i)(h) = 0$ if $e_h > d$. It is easy to see that $\mu_i(I_i)$ is a probability distribution over I_i. Moreover, $\lim_{\varepsilon \to 0} \mu_i^\varepsilon(I_i)(h) = c_h/b_d$ when $e_h = d$, and $\lim_{\varepsilon \to 0} \mu_i^\varepsilon(I_i)(h) = 0$ when $e_h > d$. Thus, $\lim_{\varepsilon \to 0} \mu_i^\varepsilon(I_i)(h) = \mu_i(I_i)(h)$ for any player i, information set I_i of i and history $h \in I_i$, and μ^ε converges to μ as $\varepsilon \to 0$. Since s^ε converges to s, s and μ are consistent.

To prove the condition of sequential rationality, we show that at a reachable information set, the belief specified by μ is derived from s using Bayes' rule. Consider an arbitrary player i and an information set I_i of i that is reachable by s. By definition, there exists $h \in I_i$ such that $e_h = 0$, thus $d = 0$ for $\mathcal{P}(I_i)$ and $b_0 = \sum_{h \in I_i, e_h = 0} c_h$. Therefore $\mu_i(I_i)$ is the probability distribution derived from s using Bayes' rule. Sequential rationality of s (with respect to μ) then follows from the definition of SSE. Thus (s, μ) is a sequential equilibrium. □

Alternative Definition of Strong Sequential Equilibrium. The notion of strong sequential equilibrium, as stated in Definition 6.1, requires that at any unreachable information set, regardless of the belief the acting player holds at that set, its action should be a best response to that belief and the other players' strategies.

We now give an equivalent definition of strong sequential equilibrium, which says that a player's strategy at an unreachable information set should be optimal following *every history* in that information set. This definition is more convenient when proving that a strategy profile is a strong sequential equilibrium.

Definition 6.3 (Strong Sequential Equilibrium: Alternate Definition)**.** *A strategy profile s is a* strong sequential equilibrium *if for every player i and information set I_i of i, we have:*

- **If I is reachable under s:** *conditioned on I_i being reached, player i's strategy s_i is a*

best response to s_{-i}, given i's beliefs at I_i being derived from s using Bayes' rule.

- **If I is unreachable under s:** *for every history $h \in I_i$, conditioned on I_i being reached, player i's strategy s_i is a best response to s_{-i}, given player i's belief that h occurs with probability 1.*

We now prove the equivalence of the two definitions of SSE in the following lemma. Without loss of generality, let s be a profile of pure strategies.

Lemma 6.4. *For any strategy profile s, any player i and information set I_i of i that is not reached with positive probability under s, conditioned on I_i being reached, s_i is a best response to s_{-i} with respect to all possible beliefs that player i may hold at I_i if and only if for every history $h \in I_i$, s_i is a best response to s_{-i} given i's belief that h occurs with probability 1.*

Proof. The "only if" part is immediate, because for any history $h \in I_i$, at h with probability 1 (and any other history with probability 0) is a specific belief that i may hold at I_i.

Suppose that s_i is a best response to s_{-i} conditioned on every history $h \in I_i$, that is, conditioned on reaching h with probability 1. To show that s_i is a best response to s_{-i} conditioned on all possible beliefs player i may hold at information set I_i, fix an arbitrary belief $\mu_i(I_i)$ over I_i and a strategy s_i'. Let $I_i = \{h_1, h_2, \ldots, h_m\}$ and $\mu_i(I_i) = (\mu_i(I_i)(h_1), \mu_i(I_i)(h_2), \ldots, \mu_i(I_i)(h_m))$, where $\mu_i(I_i)(h_k)$ is the probability with which player i believes that history h_k occurs conditioned on I_i being reached. Then, player i's expected utilities under s_i and s_i' respectively, conditioned on I_i, $\mu_i(I_i)$ and s_{-i}, are

$$u_i(s_i, s_{-i}|\mu_i(I_i)) = \sum_{k=1}^{m} \mu_i(I_i)(h_k) \cdot u_i(s_i, s_{-i}|h_k) \text{ and}$$
$$u_i(s_i', s_{-i}|\mu_i(I_i)) = \sum_{k=1}^{m} \mu_i(I_i)(h_k) \cdot u_i(s_i', s_{-i}|h_k),$$

where $u_i(s_i, s_{-i}|h_k)$ is player i's utility under (s_i, s_{-i}), conditioned on history h_k being reached at I_i. Since s_i is a best response to s_{-i} at every $h_k \in I_i$, we have $u_i(s_i, s_{-i}|h_k) \geq u_i(s_i', s_{-i}|h_k)$ $\forall k \in \{1, \ldots, m\}$. Thus $u_i(s_i, s_{-i}|\mu_i(I_i)) \geq u_i(s_i', s_{-i}|\mu_i(I_i))$. □

Corollary 6.5. *Definition 6.1 and Definition 6.3 are equivalent.*

One-Shot Deviation for Strong Sequential Equilibrium. Informally, a solution concept satisfies the one-shot deviation principle says that if whenever a strategy is at equilibrium then no player can change its action at a single information set (without changing the rest of its and other players' strategy) and improve its utility.

In the context of sequential equilibrium, it is well known that given a consistent belief system μ, (s, μ) is a sequential equilibrium if and only if the *one-shot deviation principle holds*, that is, no player i has an information set I_i at which a change in $s_i(I_i)$—holding the remaining of s_i fixed—increases its expected utility conditioned on reaching I_i [88, 117].

Since strong sequential equilibrium does not require artificial notion of beliefs for unreachable information sets, we define a stronger notion of one-shot deviation at those information sets— for every decision node (i.e., history) in an unreachable information set of player i, there does not exist a one-shot deviation at that node which improves player i's utility conditioned on that node being reached. At reachable information sets, both the definition and proof of the one-shot deviation condition for SSE are exactly the same as in SE [88].

Lemma 6.6 (One-Shot Deviation for Strong Sequential Equilibrium)**.** *For any strategy profile s, s is a strong sequential equilibrium if and only if it satisfies the following* one-shot deviation principle*: For every player i and every information set I_i of i,*

- **If I_i is reachable under s:** *there does not exist a change in $s_i(I_i)$ (holding the rest of s_i fixed) that increases player i's expected utility conditioned on reaching I_i, given i's belief at I_i derived using Bayes' rule.*
- **If I_i is unreachable under s:** *for every history $h \in I_i$, there does not exist a change in $s_i(I_i)$ (holding the rest of s_i and s_{-i} fixed) that increases player i's expected utility conditioned on reaching h.*

Proof. The "only if" part follows immediately from Definition 6.3 and the fact that a one-shot deviation results in a different strategy for the deviating player. We now prove that if s satisfies the one-shot deviation principle then it is a strong sequential equilibrium.

Reachable information sets. First, similar to the proof of Lemma 6.2, we can construct a belief system μ such that s and μ are consistent. Indeed, the construction of μ only depends on the actions taken by s and does not depend on the utilities induced by s at all. Since s satisfies the one-shot deviation principle at every reachable information set and at every history in each unreachable information set, it is not hard to see that s satisfies the one-shot deviation principle with respect to μ. Thus (s, μ) is a sequential equilibrium. Furthermore, for any player i and information set I_i of i that is reachable by s, s_i is a best response to s_{-i} conditioned on $\mu_i(I_i)$ (which is derived from s using Bayes' rule at I_i).

Unreachable information sets. Next, we use backward induction to show that, for any player i, information set I_i of i that is unreachable by s, and history $h \in I_i$, s_i is a best response to s_{-i} conditioned on reaching h. To begin with, if h is of height 1 then this

immediately holds: indeed, the strategy induced by s_i following h is exactly the action $s_i(I_i)$, thus the one-shot deviation principle implies that s_i is a best response to s_{-i} at h.

Consider an arbitrary player i, information set I_i of i unreachable by s, and a history $h \in I_i$ of height larger than 1. By induction, assume that for any information set I'_i of i unreachable by s, and history $h' \in I'_i$ of height smaller than that of h, s_i is a best response to s_{-i} at h'. For the sake of contradiction, suppose player i can deviate to strategy s'_i and increase its utility conditioned on reaching h, that is, $u_i(s'_i, s_{-i}|h) > u_i(s_i, s_{-i}|h)$.

If $s'_i(I_i) = s_i(I_i)$, consider the first history h' following h where player i acts and s'_i differs from s_i. As h is unreachable by s, h' is unreachable by s as well. However, the height of h' is smaller than that of h and $u_i(s'_i, s_{-i}|h') = u_i(s'_i, s_{-i}|h) > u_i(s_i, s_{-i}|h) = u_i(s_i, s_{-i}|h')$, contradicting the inductive hypothesis. Thus we have $s'_i(I_i) \neq s_i(I_i)$.

If s'_i is the same as s_i at all the histories following $(h, s'_i(I_i))$ where player i acts, then the one-shot deviation principle is violated. Thus, there must exist a history following $(h, s'_i(I_i))$, where player i acts and s'_i differ from s_i. Letting h' be the first such history, we have that the height of h' is smaller than that of h. Since h' is unreachable by s, by the inductive hypothesis we have that s_i is a best response to s_{-i} at h'. Thus $u_i(s_i, s_{-i}|h') \geq u_i(s'_i, s_{-i}|h')$. As $u_i(s'_i, s_{-i}|h') = u_i(s'_i, s_{-i}|h) > u_i(s_i, s_{-i}|h)$, we have $u_i(s_i, s_{-i}|h') > u_i(s_i, s_{-i}|h)$.

Let strategy s''_i be such that, it follows s_i till history h, then follows action $s'_i(I_i)$, then follows s'_i (and s_i as well, because they are the same after $(h, s'_i(I_i))$ and before h') till history h', and then follows s_i for the rest. Note that s''_i can be obtained from s_i by a one-shot deviation from $s_i(I_i)$ to $s'_i(I_i)$. However,

$$u_i(s''_i, s_{-i}|h) = u_i(s''_i, s_{-i}|h') = u_i(s_i, s_{-i}|h') > u_i(s_i, s_{-i}|h),$$

contradicting the one-shot deviation principle and s_i is a best response to s_{-i} conditioned on reaching h. Thus, by Definition 6.3, s is an SSE and Lemma 6.6 holds. □

Verifying Strong Sequential Equilibrium. Given an extensive-form game with arbitrary number of players, it is possible to decide whether a pair (s, μ) is a sequential equilibrium in time polynomial in the size of the game tree [70].

However, if only a strategy profile s is given, then it is NP-hard to decide whether s is part of an SE (that is, whether there exists a belief system μ such that (s, μ) is an SE) [86]. As strong sequential equilibrium does not rely on belief systems, we prove the following.

Lemma 6.7. *Given an extensive-form game and a strategy profile s of the players, deciding whether s is an SSE of the game can be done in time polynomial in the size of the game tree.*

Proof. In time polynomial in the size of the game tree we can traverse it, mark each in-

formation set whether it is reachable by s or not, and compute, for each player i and each reachable information set I_i of i, the belief $\mu_i(I_i)$ derived from s using Bayes' rule. Next, we apply the one-shot deviation principle (Lemma 6.6).

We start from the bottom level of the tree and proceed up. For every player i and every information set I_i of i, if I_i is unreachable under s, then we go through each $h \in I_i$ and each $a \in A(I_i)$, and check if changing $s_i(I_i)$ to a improves i's utility conditioned on reaching h. If so then s is not an SSE. If I_i is reachable under s, then we go through every $a \in A(I_i)$, and check if changing $s_i(I_i)$ to a improves i's expected utility conditioned on I_i and $\mu_i(I_i)$. If so then again s is not an SSE. If all the checks above pass, then s is an SSE.

Since we go through each decision node of the game tree at most once, and since it takes polynomial time to compute expected utilities of the players under s, deciding whether s is an SSE takes polynomial time in the size of the tree. □

6.4 Recursive-Maximum SSE

We define *maximum SSE*. A strategy profile s is a maximum SSE if it is an SSE and for any player i and SSE s', $u_i(s) \geq u_i(s')$. However, maximum SSE alone is not enough to resolve equilibrium-selection problems in extensive-form games, as the maximality is only imposed at the root. Instead, we impose the maximality condition at *every subgame*. This is analogous to the backward induction in subgame-perfect equilibrium. As the correctness of ncRIP protocols only hold at max SSE, the provers must play their max SSE strategy when restricted to subforms as well.

A subgame is a subtree that can be singled out from the game tree and treated as a separate well-defined game. As an extensive-form game with imperfect information may have very few proper subgames, we use the extended notion of subgames, *subforms*, defined by Kreps and Wilson[13] [101] to ensure that the solution concept "has enough bite."

Subforms. For any information set I, let H_I be the forest *rooted at* I, that is, $H_I = \cup_{h' \in I}\{h \mid (h', h) \in H\}$, where H is the set of all valid histories in the game. For a history $h \in H$, let $I(h)$ be the unique information set containing h. Let F_I be the set of all *information sets following* I, that is, $F_I = \{I(h', h) \mid h' \in I, (h', h) \in H\}$. As h can be the empty history ϕ, we have $I \in F_I$.

Definition 6.8. *(Subform [101]) For any information set I, H_I is a* subform rooted at I *if for every $I' \in F_I$ and every $\hat{h} \in F_I$, $\exists h' \in I$ and $\exists h \in H_I$ such that $\hat{h} = (h', h)$.*

[13] We would like to thank an anonymous reviewer of [44] for pointing us towards this citation.

Roughly speaking, a subform H_I completely contains all the information sets following I, so there is no information asymmetry between the players acting within H_I. A subform H_I and a probability distribution μ_I on I together form a well-defined game, where the players' expected utilities under s are based on μ_I and the Nature moves in H_I.

A subform H_I is *reachable under* s if I is reachable under s. Given a strategy profile s, let s_I be the strategy profile induced by s in the subform H_I. The *height* of a subform H_I is the length of the longest path in the game following the information set I.

Recursive-Maximum SSE. To deal with equilibrium selection, we enforce the solution concept to hold recursively on every subform, as in subgame-perfect equilibrium.

Definition 6.9 (Recursive-Maximum Strong Sequential Equilibrium)**.** *A strategy profile s is a recursive-maximum strong sequential equilibrium (or rmSSE) if s is an SSE and*

- *for every subform H_I of height 1:*
  - **if I is reachable under s:** *s_I is a maximum SSE on H_I with respect to μ_I (deduced from s using Bayes' rule),*
  - **if I is unreachable under s:** *s_I is a maximum SSE on H_I with respect to any distribution μ_I,*
- *for every subform H_I subgame of height > 1:*
  - **if I is reachable under s:** *s_I is maximum among all SSEs on H_I that are rmSSEs in all subforms following I (with respect to μ_I deduced from s using Bayes' rule),*
  - **if I is unreachable under s:** *s_I is maximum among all SSEs on H_I that are rmSSEs in all subforms following I (with respect to any distribution μ_I).*

We are ready to define rational proofs with non-cooperative provers.

Definition 6.10. *For any language L, an interactive protocol $(V, \vec{P})$ is a* non-cooperative rational interactive proof *(ncRIP) protocol for L if, for any $x \in \{0,1\}^*$, there exists a strategy profile s of the provers that is a recursive-maximum SSE in the resulting extensive-form game, and under any recursive-maximum SSE, the answer bit c output by V is correct (i.e., $c = 1$ iff $x \in L$) with probability 1, where the probability is over V's randomness.*

6.5 Utility Gap in ncRIP Protocols

The notion of rmSSE provides a strong guarantee that rational non-cooperative provers will act as prescribed by the protocol and lead the verifier to the correct answer.

However, this is only true in the classic game-theoretic sense where the players are perfectly rational and "sensitive" to arbitrarily small utility losses. In reality, some provers may

not care about small payment losses. Such provers could still deviate to lead the verifier to the wrong answer. To make ncRIP protocols robust against such "insensitive" provers, we define utility gap for ncRIP protocols.

Informally, a utility gap of u means that if a strategy profile s leads the verifier to the wrong answer, there must exist a subform such that some provers must lose at least a $1/u$ amount from their payments (compared to if they played their best strategy in that subform). Thus, these provers will not deviate to s, *as long as* they care about $1/u$ payment losses.

Recall that the notion of utility gap is analogous to *soundness gap*—the difference between completeness and soundness—in interactive proofs. If an IP protocol has a large soundness gap, then the probability that malicious provers can make the verifier accept when $x \notin L$ is low. Similarly, if an ncRIP protocol has a large utility gap, then even provers who are not perfectly rational and are insensitive to small losses will not deviate to the wrong answer. We proved this connection formally for the cooperative model in Chapter 5.

Definition 6.11. *Let $(V, \vec{P})$ be an ncRIP protocol for a language L and s^* be a recursive-maximum SSE in the resulting game. The protocol $(V, \vec{P})$ has an $\gamma(n)$-*utility gap *or $\gamma(n)$-gap, if for any strategy profile s' under which the answer bit c' is wrong, there exists a subform H_I reachable under s', and a prover P_j acting in H_I who has deviated from s^* such that*

$$u_j(x, (s'_{-I}, s^*_I), (V, \vec{P})) - u_j(x, (s'_{-I}, s'_I), (V, \vec{P})) > 1/\gamma(n),$$

where s'_{-I} denotes the strategy profile s' outside subform H_I, that is, $s'_{-I} = s' \setminus s'_I$.

The class of languages that have an ncRIP protocol with *constant*, *polynomial* and *exponential* utility gap, are denoted by $\mathsf{O(1)\text{-}ncRIP}$, $\mathsf{poly(n)\text{-}ncRIP}$, and ncRIP respectively.[14] Note that these terms refer to $\gamma(n)$, so exponential gives the weakest gap guarantees.

[14]These classes are formally defined by taking the union over languages with $\gamma(n)$ utility gap, for every $\gamma(n)$ that is constant, polynomial and exponential in n respectively.

Chapter 7

Tight Characterizations of ncRIP Classes

7.1 Introduction

In Chapter 6, we defined the model of non-cooperative rational proofs (ncRIP) using the solution concept of recursive-maximum strong sequential equilibrium (rmSSE).

In this chapter, we construct ncRIP protocols with constant, polynomial, and exponential utility gaps for powerful (and tight) complexity classes, demonstrating the strength of our solution concept. Our protocols are simple and intuitive (in fact requiring few changes from their cooperative counterparts), and are thus easy to explain and implement. However, proving their correctness requires significant effort to analyze the provers' incentives and to show that the protocol meets the strong solution-concept and utility-gap requirements.

A natural question to ask is whether we are "overfitting" the solution concept so as to give the verifier unrealistic or unlimited power in leveraging the provers' rationality. We show that this is not the case by proving tight upper-bounds for all three ncRIP classes.

Proving tight upper bounds on the classes of ncRIP protocols is quite technically challenging. For example, while an NEXP oracle can guess a strategy profile and verify if it is an SSE, it cannot itself verify recursive-maximum SSEs. Furthermore, the polynomial randomness of the verifier can induce an exponential-sized game tree. The key lemma that helps us overcome these challenges is the *pruning lemma* (Lemma 7.11). At a high level, it shows how we can prune the nature moves of the verifier in the resulting game tree, while preserving the recursive-maximum SSE and utility-gap guarantees.

Our results are stated in Figure 12. Recall that $\mathsf{O(1)\text{-}ncRIP}$, $\mathsf{poly(n)\text{-}ncRIP}$ and ncRIP denote ncRIP classes with constant, polynomial and exponential utility gaps respectively. The notations are analogous for MRIP. Recall that $\mathsf{P}^{\mathsf{NEXP}[\mathsf{O(1)}]}$ is the class of languages decided by a polynomial-time Turing machine that makes $O(1)$ queries to an NEXP oracle, and $\mathsf{EXP}^{\mathsf{poly\text{-}NEXP}}$ is the class decided by an exponential-time Turing machine with polynomial-length queries to an NEXP oracle. (This class is not known to be equivalent to $\mathsf{EXP}^{\mathsf{NP}}$).

Figure 13 gives a pictorial hierarchy of the power of MIP, MRIP, and ncRIP in terms of different utility gaps. These exact characterizations confirm that our solution concept properly reflects the game-theoretic nature of rational provers in interactive proofs. The

Theorem 7.1. $\mathsf{O(1)\text{-ncRIP}} = \mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]}$	**Corollary 7.4.** $\mathsf{O(1)\text{-ncRIP}} = \mathsf{O(1)\text{-MRIP}}$
Theorem 7.2. $\mathsf{poly(n)\text{-ncRIP}} = \mathsf{P}^{\mathsf{NEXP}}$	**Corollary 7.5.** $\mathsf{poly(n)\text{-ncRIP}} \supseteq \mathsf{poly(n)\text{-MRIP}}$
Theorem 7.3. $\mathsf{ncRIP} = \mathsf{EXP}^{\mathsf{poly\text{-}NEXP}}$	**Corollary 7.6.** $\mathsf{ncRIP} = \mathsf{MRIP}$

Figure 12: Summary of the tight characterizations of ncRIP classes.

techniques we use for the upper bounds are quite different from those used for the lower bounds, and require a deep understanding of the solution concept and extensive-form games.

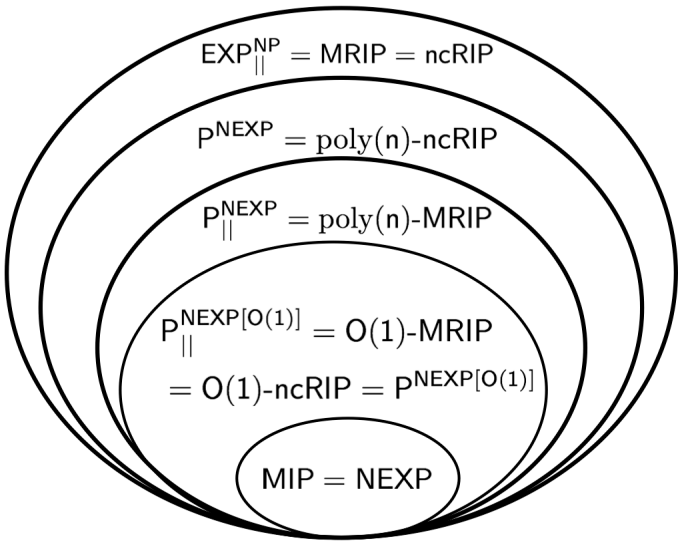

Figure 13: The computation power hierarchy of classical, rational cooperative and rational non-cooperative interactive proof systems, assuming $\mathsf{P}^{\mathsf{NEXP}}_{||} \neq \mathsf{P}^{\mathsf{NEXP}}$.

Non-Cooperative vs. Cooperative and Competitive Provers. Interestingly, in the case of constant and exponential utility gap, ncRIP and MRIP classes coincides. This can be explained by the power of adaptive versus nonadaptive queries in oracle Turing machines.

Indeed, our results reveal the main difference between non-cooperative and cooperative provers: the former can be used to handle adaptive oracle queries, the latter cannot. Intuitively, this makes sense—cooperative provers may collude across adaptive queries, answering some incorrectly to gain on future queries. On the other hand, ncRIP protocols allow us to dissociate the provers handling oracle queries from the others. Thus, whenever adaptive queries to the oracle reduce to nonadaptive queries, the two classes coincide.

Furthermore, non-cooperative provers are more powerful than competing provers—the power of refereed games with imperfect information and perfect recall is equal to EXP [63].

MRIP vs. ncRIP protocols. The ncRIP protocols presented in this chapter are largely the same as the corresponding MRIP protocols in Chapter 3—this is somewhat unsurprising as both are tailored tightly to a circuit representation of the classes in Figure 12.

However, the techniques used in the analysis are substantially different. The proofs in this chapter require careful analysis of the information sets, and subtleties of identifying when they form proper "subgames". On the other hand, ncRIP protocols and proofs are in some places much more intuitive—for example, they let us avoid the difficult and somewhat artificial payment scaling between rounds, often required in RIP and MRIP [10, 43, 80].

7.2 Lower Bounds on ncRIP Classes

In this section, we design a $O(1)$-utility gap ncRIP protocol for the class NEXP and use it to give a $O(\gamma(n))$-utility gap ncRIP protocol for the class $\mathsf{P}^{\mathsf{NEXP}[\gamma(\mathsf{n})]}$. Setting $\gamma(n)$ to a constant or polynomial gives us $\mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]} \subseteq \mathsf{O}(1)\text{-ncRIP}$ and $\mathsf{P}^{\mathsf{NEXP}} \subseteq \mathsf{poly(n)}\text{-ncRIP}$ respectively.

Then, we show how to simulate any cooperative multi-prover rational interactive proof (MRIP) using an ncRIP protocol with exponential utility gap. Since $\mathsf{EXP}^{\mathsf{NP}}_{||} \subseteq \mathsf{MRIP}$, and $\mathsf{EXP}^{\mathsf{NP}}_{||} = \mathsf{EXP}^{\mathsf{poly-NEXP}}$ [43], this proves that $\mathsf{EXP}^{\mathsf{poly-NEXP}} \subseteq \mathsf{ncRIP}$.

Constant-utility gap ncRIP protocol for NEXP. The ncRIP protocol for any language in NEXP is in Figure 14.[15] While the protocol is simple and uses the 2-prover 1-round MIP for NEXP [64] as a blackbox, in the analysis we have to open up the black-box. In particular, if P_1 sends $c = 0$ in round 1, all the information sets of P_1 and P_2 in round 3 become unreachable. To show that a strong sequential equilibrium exists, we need to show that the provers have a best response at these unreachable sets, which is argued based on the messages exchanged in the MIP procotol.

This protocol is a good example to highlight the differences between ncRIP and MRIP protocols—the ncRIP protocol appears almost identical to the MRIP protocol for NEXP in Chapter 3.3. However, the analyses are significantly different—the correctness of the MRIP protocol follows almost immediately from that of the blackbox MIP protocol, while in the case of the ncRIP protocol, we have to show that it meets all the conditions of rmSSE.

Lemma 7.7. *Any language $L \in \mathsf{NEXP}$ has an ncRIP protocol that uses two provers, three rounds and has a utility gap of $6/5$.*

[15] It is also possible to give a scoring-rule based ncRIP protocol for NEXP, as in Chapter 3.3. However, such a protocol has an exponential utility gap and is subsumed by our simulation of the MRIP in Figure 16.

For any input x and language $L \in \mathsf{NEXP}$, the protocol (V, P_1, P_2) for L is:

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 payments are $R_1 = R_2 = 1/2$.
3. Otherwise, V runs the 2-prover 1-round MIP protocol for NEXP [64] with P_1 and P_2. If the MIP protocol accepts then $R_1 = 1$, $R_2 = 1$; else, $R_1 = -1$, $R_2 = -1$.

Figure 14: A $O(1)$-utility gap ncRIP protocol for NEXP.

Proof. The ncRIP protocol for any language $L \in \mathsf{NEXP}$ is given in Figure 14.

We show that there exists a strategy profile $s = (s_1, s_2)$ of provers P_1 and P_2 respectively that is a recursive-maximum SSE of the game tree corresponding to the protocol (V, P_1, P_2) and under any recursive-maximum SSE, the answer bit $c = 1$ if and only if $x \in L$.

In the protocol, if $c = 0$, no player acts. If $c = 1$, the verifier executes the 1-round blackbox MIP protocol with P_1 and P_2. To exhibit a strategy that is a best response for P_1 and P_2 on their information sets at step 3, we look at the messages the verifier sends to each prover in the classic MIP protocol. In the MIP protocol, the verifier sends P_1 a tuple of message pairs $\vec{m}_1 = ((q_1, x_1), \ldots, (q_m, x_m))$ where m is a polynomial in n and V sends P_2 a tuple of random messages $\vec{m}_2 = (y_1, \ldots, y_m)$. P_1 sends back a polynomial $P(t)$ and P_2 sends back the value of the polynomial $P(t)$ for t satisfying $q_j + tx_j = y_j$. The verifier rejects if their answers are inconsistent.

To analyze the SSE strategy, without loss of generality, suppose P_1 moves last in the MIP protocol. Any information set I_1 of P_1 at step 3 is characterized by the message $\vec{m}_1$ it receives. The decision nodes in I_1 correspond to each possible message $\vec{m}_2$ to P_2.

Given P_2's strategy, if any information set I_1 of P_1 is reached under s then P_1's best response at I_1 is to maximize the acceptance-probability of the MIP protocol given its beliefs on I_1. Similarly, given P_2's strategy, if any information set I_1 of P_1 is unreachable under s then, P_1's best response at I_1 for every decision node in I_1 is the following: given $\vec{m}_1 = ((q_1, x_1), \ldots, (q_m, x_m))$, respond with a polynomial $P(t)$ such that $P(t)$'s value at all t coincides with P_2's reply on all y_j where $q_j + tx_j = y_j$.

Given P_1's strategy of committing to a polynomial $P(t)$ that matches P_2 on all values of t, P_2' best response at any information set I_2 (reachable or unreachable under s) at step 3 at every decision node in I_2 is to answer the tuple of queries $(y_1, \ldots, y_m)$ so as to maximize the acceptance probability of the MIP protocol. The verifier's move at step 3 starts a subform. Conditioned on step 3 being reached, any maximum SSE at this subform corresponds to a strategy profile s that is an SSE, which when restricted to this subform, maximizes the

acceptance probability of the MIP protocol. Under any such recursive-maximum SSE, we show that P_1's best response at step 1 is to send the correct answer bit.

Suppose $x \in L$. If P_1 sends $c = 0$, then $R_1 = 1/2$ with probability 1. On the other hand, if P_1 sends $c = 1$, by the soundness condition of MIP, the acceptance probability is 1, leading to $R_1 = 1$. Thus for $x \in L$, s is a recursive-maximum SSE iff P_1 sends $c = 1$.

Suppose $x \notin L$. If P_1 reports $c = 0$, then $R_1 = 1/2$ with probability 1. On the other hand if P_1 reports $c = 1$, then by the soundness condition of the MIP protocol, the maximum acceptance probability is $1/3$ leading to $R_1 = 1$. The protocol rejects with probability at least $2/3$ leading to $R_1 = -1$. Thus, P_1's expected payment for misreporting the answer bit is at most $R_1 = -1/3$. Thus for $x \notin L$, s is a recursive-maximum SSE iff P_1 sends $c = 0$.

Thus, under s which is a recursive-maximum SSE, $c = 1$ if and only if $x \in L$. Furthermore, the payment incurred by the provers when the answer bit sent in the first round is incorrect is at least $5/6$ for both provers and thus the protocol has constant utility gap. □

$O(\gamma(n))$-utility gap ncRIP protocol for $\mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$. Next, we give an ncRIP protocol with $O(\gamma(n))$-utility gap for the class $\mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$, where $\gamma(n)$ is a function of n which (1) only takes positive integral values, (2) is upper-bounded by a polynomial in n, and (3) is polynomial-time computable. For example, $\gamma(n)$ can be $\log n$, $\sqrt{n}$, etc.

The ncRIP protocol for any $L \in \mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$ is in Figure 15. It is fairly intuitive: V simulates the polynomial-time machine and uses the ncRIP protocol for NEXP for the queries.

The analysis of the protocol illustrates the robustness of the solution concept. In particular, the NEXP queries start the non-trivial subforms in the game, and which of them are reachable under any strategy profile s is determined solely by P_1' strategy. To avoid suboptimal equilibria and the problem of empty threats, recursive-maximality must hold at both reachable and unreachable subforms. Otherwise, P_1 cannot unilaterally deviate out of a bad strategy where it is lying on an NEXP gate to a strategy giving the correct answers (and thus making a previously unreachable NEXP query reachable), if P_2 and P_3 are giving wrong answers at those NEXP queries.

The analysis of the protocol in Figure 15 shows that even though V can only check the NEXP queries that P_1 wants V to see, the protocol is designed to ensure that deviating provers in some reachable subform suffer an $1/O(\gamma(n))$ loss in their overall expected payment.

Lemma 7.8. *Any language $L \in \mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$ has an ncRIP protocol that uses three provers, five rounds and has a utility gap of $6/(5\gamma(n))$.*

Proof. Consider any language $L \in \mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$. Let M be a polynomial-time Turing machine deciding L, with access to an oracle O for an NEXP language.

For any input x of length n, the protocol $(V, \vec{P})$ works as follows.

1. P_1 sends $(c, c_1, \ldots, c_{\gamma(n)}) \in \{0,1\}^{\gamma(n)+1}$ to V. V outputs c at the end of the protocol.
2. V simulates M on x using the bits $c_1, \ldots, c_{\gamma(n)}$ as answers to NEXP queries $\phi_1, \ldots, \phi_{\gamma(n)}$ generated by M respectively. If M accepts and $c = 0$ or M rejects and $c = 1$, then the protocol ends and $R_1 = -1, R_2 = R_3 = 0$.
3. V picks a random index i' from $\{1, \ldots, \gamma(n)\}$ and sends $(i', \phi_{i'})$ to P_2 and P_3.
4. V runs the 2-prover 3-round $O(1)$-gap ncRIP protocol for NEXP (Figure 14) with P_2 and P_3 on ϕ_i. P_2 and P_3 get payments R_2 and R_3 based on the protocol. Let $c^*_{i'}$ be the answer bit in the NEXP protocol. If $c^*_{i'} \neq c_{i'}$, then $R_1 = 0$; otherwise $R_1 = 1$.

Figure 15: An $O(\gamma(n))$-utility gap ncRIP protocol for $\mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$.

The ncRIP protocol for L is given in Figure 15.

Let s_1, s_2, s_3 be the strategy used by P_1, P_2 and P_3 respectively in the protocol in Figure 15, and $s = (s_1, s_2, s_3)$. First, note that regardless of s_2 and s_3, P_1's best response at step 1 is to send the bits $c, c_1, \ldots, c_\gamma(n)$ such that the verification in step 2 goes through. In particular, if s_1 is such that the output of M on input x, using $c_1, \ldots, c_{\gamma(n)}$ as answers to NEXP queries $\phi_1, \ldots, \phi_{\gamma(n)}$ is consistent with c, then P_1 gets $R_1 \geq 0$. Meanwhile, if the verification in step 2 fails then $R = -1$. Thus, under any SSE s, the answer bits $c_1, \ldots, c_{\gamma(n)}$ sent by P_1 must be consistent with the computation of M on x and the final the answer bit c, regardless of s_2 and s_3.

We now argue using backward induction. Each random index i' chosen by V in step 3 together with $\phi_{i'}$ starts a subform. In particular, since P_2 and P_3 both know $(i', \phi_{i'})$, all their information sets starting from step 4 are completely disjoint from information sets reached under a different index and NEXP query. By Lemma 7.7, there exists a recursive-maximum SSE s on each such subform simulating an NEXP query, and under any recursive-maximum SSE, s_2 and s_3 are such that $c^*_{i'}$ is the correct answer to the NEXP query.

Moving up the tree, the next subform is induced by V's nature move at step 3 assigning a probability to each subsequent subform. Since under any recursive-maximum SSE, the expected payments of P_2 and P_3 (conditioned on reaching these subforms) are maximized, the overall expected payments under V's nature move at step 3 is also maximized.

We move up a further level in the tree to the root. We show that P_1's best response at step 1 is to send the correct answer bits, given that under any recursive-maximum SSE s:

- P_2 and P_3 answer each NEXP query $\phi_{i'}$ determined by s_1 and index i' correctly, and

- the verification in step 2 goes through (i.e. P does not set $R_1 = -1$) under s_1.

Suppose s_1 is such that there exists an NEXP query where P_1 lies. Let k be the first NEXP query index such that c_k is not the correct answer to query ϕ_k, where $1 \leq k \leq \gamma(n)$. In particular, the instance ϕ_k is evaluated correctly (by running M on x using the correct answers to previous queries: $c_1, \ldots, c_{k-1}$ but the answer c_k is not evaluated correctly based on ϕ_k. Then with probability $1/\gamma(n)$, V picks k in step 3 and crosschecks the c_k with $c^*_{i'}$, in which case the verification fails and $R_1 = 0$. Thus, P_1's expected payment is at most $1 - 1/\gamma(n)$. If P_1 answers all NEXP queries correctly, since the verification in step 2 goes through, P_1 gets $R_1 = 1$ with probability 1. Thus, $c, c_1, \ldots, c_{\gamma(n)}$ are correct under any recursive-maximum SSE s, and $c = 0$ if and only if $x \in L$.

Now, we show that protocol $(V, \vec{P})$ has $O(\gamma(n))$ utility gap. Let s^* be a recursive-maximum SSE of the game resulting from $(V, \vec{P})$. Suppose s' is such that the answer bit c' under s' is incorrect. We go "bottom-up" in the game tree and exhibit a subform H_I (reachable under s') such that some prover acting in that subform loses $O(1/\gamma(n))$ compared to the strategy where s^*_I is played on H_I, keeping the rest of the strategy fixed.

First, consider all the NEXP queries at step 4 that start subforms. Suppose there exists a query ϕ_k committed under s'_1, for $1 \leq k \leq \gamma(n)$, such that c_k* is the wrong answer to ϕ_k. By Lemma 7.7, both P_2 and P_3 lose 5/6 from their expected payment (conditioned on reaching this subform) compared to the recursive-maximum SSE strategy profile $s^*_{\phi_k}$ which reports the correct answer to ϕ_k. Since V chooses ϕ_k with probability $1/\gamma(n)$, P_2 and P_3 can gain $O(1/\gamma(n))$ in their overall expected payment by deviating to strategy profile s_{ϕ_k}, at the subform corresponding to (k, ϕ_k) keeping $s'_{-\phi_k}$ fixed. Specifically,

$$\begin{aligned}&\mu_i\left(x, r, (s'_{-\phi_k}, s^*_{\phi_k}), (V, \vec{P})\right) - \mu_i\left(x, r, (s'_{-\phi_k}, s'_{\phi_k}), (V, \vec{P})\right)\\ &> \frac{1}{\gamma(n)}\left(\frac{5}{6}\right), \quad \text{for} \quad i \in \{2, 3\}.\end{aligned}$$

Finally, suppose P_2 and P_3 answer all NEXP queries (reachable under s') correctly. Then, P_1 loses at least $1/\gamma(n)$ at the subform at the root—the entire game. Since the answer bit c' under s' is incorrect, either step 2 fails or P_1 lies on some NEXP query. In the first case, P_1 gets -1 with probability 1 compared to an expected payment of 1 under s^*. In the second case, P_1 gets caught in step 4 with probability $1/\gamma(n)$, and gets an expected payment of at most $1 - 1/\gamma(n)$, losing at least $1/\gamma(n)$ compared to s^*. □

The ncRIP protocol in Figure 15 is a good example to demonstrate the problem of empty threats that can occur under Nash and maximum Nash equilibrium.

In this protocol, P_2 and P_3 obtain a higher expected payment if the NEXP instance they are queried is not satisfiable. Thus, they can blackmail P_1—if it does not give answers that lead them to a satisfiable instance, they will lie about the answer to the NEXP instance in step 4, which would result in P_1 being punished. This is an empty threat (as it is not a rational move for P_2 and P_3), however it leads to an unnatural Nash equilibrium. Similarly, there does exist a maximum Nash equilibrium for this protocol.

Simulating any MRIP protocol using an ncRIP Protocol. Any MRIP protocol $(V, \vec{P})$ with $p(n)$ provers and $k(n)$ rounds can be simulated by a 2-prover 3-round ncRIP protocol (V', P_1', P_2') with exponential utility gap.

Essentially, V' gives all the randomness of V to P_1' and asks for the entire transcript and uses P_2' to commit to a single prover's message, and cross-checks their answers. However, we don't want P_1' who has access to all the randomness to dictate what information sets of P_2' are reachable. Because the ncRIP protocol need only have an exponential utility gap, V' asks one prover a totally random question (independent of P_1'), and with exponentially small probability this random message is exactly the message V' intended to check. This protocol shows why exponential gap guarantees do not lead to meaningful protocols—a verifier that asks random questions can still extract honest behavior from rational provers through the exponentially small changes in expected payments.

Lemma 7.9. *Any MRIP protocol can be simulated by an ncRIP protocol that uses two provers, three rounds and has exponential utility gap.*

Proof. Let $(V, \vec{P})$ be an MRIP protocol with $p(n)$ provers and $k(n)$ rounds for a language L. Without loss of generality, each message in the protocol is of length $\ell(n)$ for any input of length n, where $\ell(n)$ is a polynomial in n. We shift and rescale the payment function of V, so that the payment is always in $[0, 1]$, and the expected payment is strictly greater than 0 under the provers' best strategy profile.

We simulate $(V, \vec{P'})$ using an ncRIP protocol $(V', (P_1', P_2'))$, given in Figure 16.

Let s_1' and s_2' denote the strategy of the provers P_1' and P_2' respectively and $s' = (s_1', s_2')$. Since P_2' is queried only once and about a single message in step 3, any strategy s_2' of P_2' de facto commits to a strategy profile for the provers in $(V, \vec{P})$.

We analyze the game tree of the protocol $(V', \vec{P'})$ bottom-up.

The last move is by P_1' sending the entire transcript $\vec{m}$ at step 5. Any information set I_1' of P_1' is characterized by the randomness r received by P_1' in step 4 and all information sets are reachable under any s'. The decision nodes in I_2' correspond to different strings $\tilde{m}_{ij}$ that P_2' could have been asked in step 2. Given s_2', the best response of P_1' at any information set I_1',

Given input x of length n, and an MRIP protocol $(V, \vec{P})$, the ncRIP protocol $(V', \vec{P}')$ is:

1. P'_1 sends the round 1 messages $m_{11}, \ldots, m_{p(n)1}$ of $(V, \vec{P})$ to V'. V' outputs c, the first bit of m_{11}, at the end of the protocol.
2. V' selects a random prover index $i \in \{1, \ldots, p(n)\}$ and a random round $j \in \{1, \ldots, k(n)\}$. Then, V' generates a random string $\tilde{m}_{ij}$ of length $(j-1)\ell(n)$.
3. V' sends $(i, j, \tilde{m}_{ij})$ to P'_2. P'_2 simulates P_i on round j and sends back the message m'_{ij}.
4. V' generates all the randomness r used by V and sends it to P'_1.
5. P'_1 uses r to simulate the protocol $(V, \vec{P})$, and sends the resulting transcript $\vec{m}$ to V'.
6. If $\tilde{m}_{ij} \neq (m_{i1}, \ldots, m_{i(j-1)})$, where m_{ij} denotes prover P_i's message in round j according to $\vec{m}$ sent by P_1', then the protocol ends and $R'_1 = R'_2 = 0$.
7. Otherwise, if $m_{ij} \neq m'_{ij}$, then $R'_1 = R'_2 = -1$.
8. Else, V' computes the payment R in $(V, \vec{P})$ using x, r and $\vec{m}$, and sets $R'_1 = 0$, $R'_2 = R$.

Figure 16: Simulating any MRIP using an ncRIP protocol with exponential utility gap.

for any beliefs at I'_1, is to match the transcript committed by P'_2 and make the verification in step 7 go through. Suppose there exists a prover index i and round j such that the message m_{ij} in $\vec{m}$ that is inconsistent with the corresponding message m'_{ij} committed under s'_2. With probability $\frac{1}{2^{(j-1)\ell(n)}}$, the random string $\tilde{m}_{ij}$ generated by V' in Step 2 is equal to $(m_{i1}, \ldots, m_{i(j-1)})$, otherwise the protocol ends with $R'_1 = 0$. With probability at least $\frac{1}{p(n)k(n)}$, V' chooses (i, j) in step 2, and queries P'_2 for m'_{ij} and $R'_1 = -1$. If (i, j) is not chosen then $R'_1 = 0$. Thus, P'_1 expected payment at I'_1 is at most

$$\sum_{i \leq p(n), 1 \leq j \leq k(n)} \frac{1}{2^{(j-1)\ell(n)}} \cdot \frac{1}{p(n)k(n)} \cdot \left(\mathbb{I}_{m_{ij} \neq m'_{ij}} \cdot (-1) + \mathbb{I}_{m_{ij} = m'_{ij}} \cdot 0 \right) < 0.$$

On the other hand, matching s'_2 on all messages gets P'_1 an expected payment of 0 at I'_1.

Given that P'_1 best response is to make the verifier in step 7 go through for every randomness r, we analyze P'_2 move at step 3. Any information set I'_2 of P'_2 is characterized by the random string $\tilde{m}_{ij}$ received by P'_2 in step 2 and all information sets are reachable under any s'. The decision nodes in I'_1 correspond to different random strings r that P'_1 could have been asked in step 2. The best response of P'_2 at any information set I'_1, for any beliefs at I'_1, is to commit to the correct strategy profile s of the provers $\vec{P}$. Suppose P'_2 commits to a strategy profile s' such that the answer bit under s' is wrong. With probability $\frac{1}{2^{(j-1)\ell(n)}}$,

the random string $\tilde{m}_{ij}$ generated by V' in Step 2 matches $(m_{i1}, \ldots, m_{i(j-1)})$, otherwise the protocol ends with $R'_2 = 0$. If it matches, then P'_2 expected payment is determined by the expected payment that $\tilde{s}$ gets in $(V, \vec{P})$ given x and randomness r, which is strictly less than the expected payment under the strategy profile s which commits to the correct answer bit (by correctness of the original MRIP protocol). That is,

$$\sum_{1 \le j \le k(n)} \frac{1}{k(n)} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot u_{(V,\vec{P})}(x, \tilde{s}) < \sum_{1 \le j \le k(n)} \frac{1}{k(n)} \cdot \frac{1}{2^{(j-1)\ell(n)}} \cdot u_{(V,\vec{P})}(x, s).$$

Thus, given that s'_1 matches s'_2 for every randomness r, the best response by P'_2 is to commit to a strategy profile $s'_2 = s$ that maximizes the total expected payment of the original protocol $(V, \vec{P})$ and thus has the correct answer bit.

There are no non-trivial subforms in the game. Any maximum SSE is a recursive-maximum SSE, under which both P'_1 and P'_2 maximize their expected payments—P'_1 matches P'_2 on all messages and P'_2 commits to the correct strategy profile s. □

7.3 Upper Bounds on ncRIP Classes

In this section, we prove matching upper bounds on the classes of ncRIP protocols with constant, polynomial and exponential utility gaps. We focus on the upper bound for $\mathsf{O(1)}$-ncRIP and $\mathsf{poly(n)}$-ncRIP, in which a polynomial-time Turing machine needs to simulate the protocol with a constant and polynomial number of queries to an NEXP oracle respectively.

To simulate an ncRIP protocol, we need to find a strategy profile "close enough" to a recursive-maximum SSE so that the answer bit is still correct, that is, it is sufficient to find a strategy profile that satisfies the utility gap guarantee. We formalize this restatement of Definition 3.6 as the following observation.

Observation 7.10. *Given input x of length n and an ncRIP protocol $(V, \vec{P})$ with a utility gap of $\gamma(n)$, let s be a strategy profile such that for all subforms H_I (reachable under s), and for all provers P_j acting in H_I, we have*

$$u_j(x, (V, \vec{P}), (s_{-I}, s^*_I)) - u_j(x, (V, \vec{P}), (s_{-I}, s_I)) < \frac{1}{\gamma(n)},$$

where s^* is a recursive-maximum SSE. Then, the answer bit c under s must be correct.

There are several challenges involved in finding a strategy satisfying Observation 7.10.

First, the size of the game tree of any ncRIP protocol—small gap notwithstanding—can be exponential in n. Even if the polynomial-time Turing machine considers a single strategy

profile s at a time, since V can flip polynomial coins, the part of the tree "in play"—the number of decision nodes reached with nonzero probability under s—can still be exponential.

The second and related problem is that while the NEXP oracle can guess and verify an SSE, it cannot help the Turing machine directly with maximum SSEs. In particular, the polynomial-time machine must go bottom-up in the game tree and find an SSE that is recursively maximal on all its reachable subgames (which can again be exponential in n).

Finally, the polynomial-time machine needs to search through the exponentially large strategy-profile space in an efficient way to find one which leads to the correct answer.

We now prove a fundamental lemma about ncRIP protocols with utility gap that lets us get around the first two challenges mentioned above.

Pruning Nature moves in ncRIP protocols. A verifier's coin flips in an ncRIP protocol represent *Nature moves* in the resulting game. A Nature move that imposes nonzero probabilities over exponentially many outcomes can cause the game tree under play to be exponential in size. We prune the Nature moves so that a polynomial-time Turing machine simulating an $\gamma(n)$-utility gap protocol can traverse the game tree reached under any given s.

Lemma 7.11 (Pruning Lemma). *Let $L \in \gamma(\mathsf{n})$-ncRIP and let $(V, \vec{P})$ be an ncRIP protocol for L with $\gamma(n)$ utility gap and $p(n)$ provers. Given an input x and a strategy s, the protocol $(V, \vec{P})$ can be transformed in exponential time to a new protocol, say $(V', \vec{P})$, where*

- *the probability distributions imposed by the nature moves of V' have $O(\gamma(n))$ support,*
- *if s is a rmSSE of $(V, \vec{P})$, s induces a rmSSE in $(V', \vec{P})$,*
- *$|u_j(x, s, (V, \vec{P})) - u_j(x, s, (V', \vec{P}))| < 1/(4\gamma(n))$ for all $j \in \{1, \ldots, p(n)\}$, and*
- *if the answer bit under s is wrong, then there exists a subform H_I in the game $(V', \vec{P})$ (reachable under s) and a prover P_j acting at H_I, such that P_j loses a $1/(2\gamma(n))$ amount in its expected payment compared to a strategy profile where s_I (induced by s on H_I) is replaced by s_I^* (the recursive-maximum SSE on H_I), keeping the strategy profile outside H_I, s_{-I}, fixed.*

We prove Lemma 7.11 in several parts. First, we show how to transform any nature move of V that imposes a nonzero probability distribution on exponentially many outcomes into a probability distribution with $O(\gamma(n))$ support, given an input x and a strategy s.

Let $(V, \vec{P})$ use $p(n)$ provers and let the running time of V be n^k for some constant k. There can be at most 2^{n^k} different payments that V can generate for a particular prover given the input x. Given x and s, fix a prover index $j \in \{1, p(n)\}$. Let $R_1, R_2, \ldots, R_m$ be the

payments generated by V on s for P_j. Let V's randomness assign probability distribution $\mu = (p_1, p_2, \ldots, p_m)$ to $R_1, R_2, \ldots, R_m$ respectively. Then, the expected payment of P_j under s, $u_j(x, s, (V, \vec{P})) = \sum_{i=1}^{m} p_i R_i$.

Recall that $u_j(x, s, (V, \vec{P})) \in [-1, 1]$ for all $1 \leq j \leq p(n)$. For each prover P_j, divide the interval $[-1, 1]$ into $4\gamma(n)$ intervals, each of length $1/(2\gamma(n))$. In other words, prover P_j's ith interval is $[i/2\gamma(n), (i+1)/2\gamma(n))$, for each $i \in \{-2\gamma(n), \ldots, 2\gamma(n) - 1\}$.[16]

We round the possible payments for P_j to a representative of the interval they belong. Specifically, we map each payment R_i to r_j as described in Equation 5. There are potentially

$$r_j = \begin{cases} \frac{4\ell+1}{4\gamma(n)} & \text{if } R_i \in \left[\frac{\ell}{2\gamma(n)}, \frac{2\ell+1}{4\gamma(n)}\right) \\ \\ \frac{4\ell+3}{4\gamma(n)} & \text{if } R_i \in \left[\frac{2\ell+1}{4\gamma(n)}, \frac{\ell+1}{2\gamma(n)}\right) \end{cases} \quad (5) \qquad p'_i = \begin{cases} \sum_{k \in T_j} p_k & \text{if } i = f(S(i)) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

exponentially many different payments R_i, and only polynomially many different payments r_j, so several R_i must map to the same r_j. Let $T_j = \{i : R_i \text{ maps to } r_j\}$. Let $\mathcal{T} = \cup_j \{T_j\}$. Thus the total number of distinct r_j's is $8\gamma(n)$, $|\mathcal{T}| = O(\gamma(n))$. Let $S : \{1, \ldots, m\} \to \mathcal{T}$, such that $S(i) = T_j$ iff $i \in T_j$.

For each $T_j \in \mathcal{T}$, let $f(T_j)$ denote a unique index in the set T_j. Without loss of generality, let $f(T_j)$ be the lowest index in T_j.

We define a new probability distribution $\mu' = (p'_1, \ldots, p'_h)$ over the payments $R_1, \ldots, R_h$ respectively given by Equation 6. In particular, for every $T_j \in \mathcal{T}$, assign $R_{f(T_j)}$ probability $\sum_{k \in T_j} p_k$ and for every other index $\ell \in T_j$, $\ell \neq f(T_j)$, assign R_ℓ probability 0.

Given x, define V' as a polynomial-time verifier that simulates all deterministic computation of V and imposes a probability distribution μ' with $O(\gamma(n))$ support for any probability distribution μ imposed by V. For other inputs, V' simulates V without any modification.

Note that given x, a strategy profile s and the protocol $(V, \vec{P})$, transforming the distribution μ to μ' takes time linear in the size of the game tree, and thus exponential in n. (Thus an NEXP oracle, given x, can guess a particular s and perform the transformation.)

Next, we show that if a strategy s is a recursive-maximum SSE of $(V, \vec{P})$, then s restricted to the pruned game tree of $(V', \vec{P})$ imposes a recursive-maximum SSE on $(V', \vec{P})$ as well.

Claim 7.12. *Any recursive-maximum SSE s in the game tree of protocol $(V, \vec{P})$ induces a recursive-maximum SSE in the game tree of protocol $(V', \vec{P})$.*

Proof. By contradiction, suppose s is not an SSE of $(V', \vec{P})$. Then there exists an information set $I = \{h_1, \ldots, h_m\}$, such that, conditioned on reaching I, the prover acting at I can improve

[16] To include 1 as a payment, interval $2\gamma(n) - 1$ should be closed on both sides; we ignore this for simplicity.

its expected payment by deviating (given its belief u'_I at I if I is reachable under s and for any belief it may hold at I if I is unreachable under s).

We split into two cases: I is either reachable or unreachable under s.

By construction, if I is reachable under s in $(V', \vec{P})$, then I must also be reachable under s in $(V, \vec{P})$. Let $\mu'_I = (p'_1, \ldots, p'_m)$, where p'_i is the probability assigned to h_i and the support of μ'_I is $O(\gamma(n))$. Let $R_1, \ldots, R_m$ be the payments that the player acting on I gets under s conditioned on reaching $h_1, \ldots, h_m$ respectively. Similarly, let $R'_1, \ldots, R'_m$ be the payments conditioned on reaching $h_1, \ldots, h_m$ respectively under the strategy to which the player at I deviates from s. Then, $\sum_{i=1}^{m} p'_i R'_i > \sum_{i=1}^{m} p'_i R_i$. Let $\mu_I = (p_1, \ldots, p_m)$ be the beliefs on I under s in $(V, \vec{P})$. We use the relationship between the distributions μ'_I and μ_I, to show that such a deviation in $(V', \vec{P})$ would imply a deviation in $(V, \vec{P})$. In particular, mapping μ'_I back to μ_I, using Equation 6 we get:

$$\sum_{i=1}^{m} \left(\mathbb{I}_{i=f(S(i))} \cdot \sum_{k \in S(i)} p_k \right) R'_i > \sum_{i=1}^{m} \left(\mathbb{I}_{i=f(S(i))} \cdot \sum_{k \in S(i)} p_k \right) R_i$$

$$\sum_{i=1}^{m} \left(\mathbb{I}_{i=f(S(i))} \cdot \sum_{k \in S(i)} p_k \right) \cdot \min_{k \in S(i)} R'_k > \sum_{i=1}^{m} \left(\mathbb{I}_{i=f(S(i))} \cdot \sum_{k \in S(i)} p_k \right) \cdot \max_{k \in S(i)} R_k \tag{7}$$

$$\sum_{i=1}^{m} \left(\mathbb{I}_{i=f(S(i))} \cdot \sum_{k \in S(i)} p_k R'_k \right) > \sum_{i=1}^{m} \left(\mathbb{I}_{i=f(S(i))} \cdot \sum_{k \in S(i)} p_k R_k \right)$$

$$\sum_{i=1}^{m} p_i R'_i > \sum_{i=1}^{m} p_i R_i \tag{8}$$

Inequality 7 holds because $R'_{f(S(i))} > R_{f(S(i))}$, and so the two payments lie in different intervals in the mapping (Equation 5). Thus the minimum payment in the interval of $R'_{f(S(i))}$ will be greater than the maximum payment in the interval of $R_{f(S(i))}$. Finally, Inequality 8 contradicts the fact that s is an SSE in $(V, \vec{P})$.

Consider an information set I unreachable under s in $(V', \vec{P})$, then I must be unreachable under s in $(V, \vec{P})$. If the action of prover acting at I is not its best response in $(V', \vec{P})$ for some history $h \in I$ then, it contradicts the fact that s is an SSE of $(V, \vec{P})$.

Suppose s is not a recursive-maximum SSE of $(V', \vec{P})$. Then there exists a subgame H_I of height k such that s is recursive-maximum on all subgames following H_I of height $< k$ but not maximum at H_I (among SSE's that are recursively-maximum at all subforms following H_I). Let s^* be recursive-maximum on H_I, then the expected payment of at least one prover P_j is better under s^*, while everyone else does just as well (given the beliefs at I derived using Bayes' rule if I is reachable under s or given any beliefs if I is unreachable under s). Writing the expression of expected payment of P_j conditioned on reaching H_I and "unfolding" the

probability distribution back to the original game, we get a contradiction that s is not its recursive-maximum SSE, as s^* gives P_j a better expected payment at H_I while doing just as well for other provers. The proof is similar as before; we omit the details. □

We now show that for a given s, the expected payments of the provers under $(V, \vec{P})$ and under $(V', \vec{P})$ are not too far off. In particular, we prove the following claim.

Claim 7.13. *For all $j \in \{1, \ldots, p(n)\}$, $|u_j(x, s, (V, \vec{P})) - u_j(x, s, (V', \vec{P}))| < 1/(4\gamma(n))$.*

Proof. Given input x and strategy profile s, fix a prover P_j. Let V generate payments $R_1, R_2, \ldots, R_m$ under s for P_j, and assign the probability distribution $\mu = (p_1, p_2, \ldots, p_m)$ on $R_1, R_2, \ldots, R_m$ respectively. Using Equations (5) and (6) we compare P_j's expected payment:

$$|u_j(s, x, (V, \vec{P})) - u_j(s, x, (V', \vec{P}))| = \left| \sum_{i=1}^{m} p_i R_i - \sum_{T_j \in \mathcal{T}} \left(\sum_{k \in T_j} p_k \right) R_{f(T_j)} \right|$$

$$= \sum_{T_j \in \mathcal{T}} \sum_{k \in T_j} p_k \Big(|R_{f(T_j)} - R_i| \Big) < \sum_{T_j \in \mathcal{T}} \sum_{k \in T_j} p_i \left(\frac{1}{4\gamma(n)} \right) = \left(\sum_{i=1}^{m} p_i \right) \frac{1}{4\gamma(n)} = \frac{1}{4\gamma(n)} \qquad \square$$

To complete the proof of Lemma 7.11, we show that $(V', \vec{P})$ preserves utility gap.

Claim 7.14. *Given input x, if the answer bit under s is wrong, then there exists a subform H_I reachable under s in $(V', \vec{P})$ and P_j acting at H_I, such that P_j's expected payment under s is $1/2\gamma(n)$ less than that under (s_{-I}, s_I^*), where s_I^* is a recursive-maximum SSE on H_I.*

Proof. Consider a strategy profile s^* that is a recursive-maximum SSE of $(V, \vec{P})$. Since s gives the wrong answer bit, from the $\gamma(n)$-utility gap guarantee of $(V, \vec{P})$ and Definition 3.6, there exists a subform H_I reachable under s, such that a prover P_j acting in H_I loses $1/\gamma(n)$ in its expected payment under s compared to the strategy profile (s_{-I}, s_I^*). That is,

$$u_j(x, (s_{-I}, s_I^*), (V, \vec{P})) - u_j(x, (s_{-I}, s_I), (V, \vec{P})) > \frac{1}{\gamma(n)}. \tag{9}$$

Using Claim 7.12, s^* also induces a recursive-maximum SSE of $(V', \vec{P})$. And since H_I is reachable under s in $(V, \vec{P})$, it is reachable under s in $(V', \vec{P})$ as well. We show that:

$$u_j(x, (s_{-I}, s_I^*), (V', \vec{P})) - u_j(x, (s_{-I}, s_I), (V', \vec{P})) > \frac{1}{2\gamma(n)}. \tag{10}$$

Using Claim 7.13, P_j's expected payments in the two protocols under s and s^* follow:

$$|u_j(x,(s_{-I},s_I^*),(V,\vec{P})) - u_j(x,(s_{-I},s_I^*),(V',\vec{P}))| < \frac{1}{4\gamma(n)} \tag{11}$$

$$|u_j(x,(s_{-I},s_I),(V,\vec{P})) - u_j(x,(s_{-I},s_I),(V',\vec{P}))| < \frac{1}{4\gamma(n)} \tag{12}$$

There are four cases depending on the sign of the left hand side of Inequalities (11) and (12). We show Claim 7.14 holds for one case and omit others, which are similar.

Suppose the left hand side of both inequalities is positive, that is, $u_j(x,(s_{-I},s_I^*),(V,\vec{P})) > u_j(x,(s_{-I},s_I^*),(V',\vec{P}))$, and $u_j(x,(s_{-I},s_I),(V,\vec{P})) > u_j(x,(s_{-I},s_I),(V',\vec{P}))$. Then,

$$\begin{aligned}
&u_j(x,(s_{-I},s_I^*),(V',\vec{P})) - u_j(x,(s_{-I},s_I),(V',\vec{P})) \\
&> \left(u_j(x,(s_{-I},s_I^*),(V,\vec{P})) - \frac{1}{4\gamma(n)}\right) - u_j(s',x,(V',\vec{P})) \\
&> \left(u_j(x,(s_{-I},s_I),(V,\vec{P})) + \frac{1}{\gamma(n)}\right) - \frac{1}{4\gamma(n)} - u_j(x,(s_{-I},s_I),(V',\vec{P})) > \frac{3}{4\gamma(n)}
\end{aligned}$$ □

Searching through strategy-profile space efficiently. Using Lemma 7.11, given an input x and ncRIP protocol $(V,\vec{P})$ with utility gap $\gamma(n)$ that is constant or polynomial, a polynomial-time oracle Turing machine can use its NEXP oracle to guess a strategy s, prune the Nature moves of V, and report the resulting $O(\gamma(n))$-support distribution bit-by-bit. Thus, it can simulate the new distribution to figure out decision nodes reachable under s.

The next question then is, how should the polynomial-time Turing machine navigate the potential strategy-profile space in polynomial-time to find the strategy profile that satisfies Observation 7.10 (and thus gives the correct answer bit)? To do this, we invoke a recurring idea: divide each prover's expected payment interval $[-1,1]$, evenly into $8\gamma(n)$ *subintervals* of length $1/(4\gamma(n))$, and consider *subinterval profiles* (a tuple of subintervals, one for each prover) to cut down on the search space.

Claim 7.15. *Given an input x and an ncRIP protocol $(V,\vec{P})$ with $\gamma(n)$-utility gap, consider a subinterval profile $(L_1,\ldots,L_{p(n)})$, where each $L_i = [k/(4\gamma),(k+1)/(4\gamma+1))$ denotes a subinterval for prover P_i's expected payment in $[-1,1]$, for some $k \in \{-2\gamma(n),\ldots,2\gamma(n)-1\}$. If an SSE s has an expected payment profile $\tilde{u}(x,s)$ such that $u_i(x,s) \in L_i$ for all $1 \le i \le p(n)$, and s does not satisfy Observation 7.10, then there exists a prover index j such that $u_j(x,s^*) \notin L_j$, where s^* is a recursive-maximum SSE.*

Proof. Since s does not satisfy Observation 7.10, there exists a reachable subform H_I and

prover P_j acting on H_I such that (without loss of generality, let $u_j(s,x) \in L_k$):

$$u_j(x,(s_{-I},s_I^*),(V,\vec{P})) - u_j(x,(s_{-I},s_I),(V,\vec{P})) > \frac{1}{\gamma(n)}$$
$$u_j(x,s^*,(V,\vec{P})) > \frac{1}{\gamma(n)} + \frac{k}{4\gamma(n)} \implies u_j(x,s^*,(V,\vec{P})) \notin L_k \qquad \square$$

Using Claim 7.15, if the polynomial-time machine is able to test *any* SSE s with $\tilde{u}(x,s)$ in a subinterval profile, for all subinterval profiles, it is guaranteed to find one that satisfies Observation 7.10. This is because a recursive-maximum SSE of an ncRIP protocol is guaranteed to exist and its expected payment profile must belong to some subinterval profile.

However, as there are $O(\gamma(n)$ subintervals for each of the $p(n)$ provers, there are $O(\gamma(n)^{p(n)})$ total subinterval profiles and a polynomial-time machine cannot test SSEs for each of them.

To reduce the search space further, we show that it is sufficient to consider subintervals of the *total expected-payment* interval and test an SSE s for each of them. To test if an SSE s satisfies Observation 7.10, we go bottom-up in the game tree reachable under s to find the maximum SSE for all subforms (always exists in an ncRIP protocol).

Recall that a maximum SSE is an SSE where for any player i and SSE s', $u_i(s) \geq u_i(s')$. We find such an SSE by querying about total expected payments only using the next lemma.

Lemma 7.16. *If a maximum SSE exists then a strategy profile s is a maximum SSE if and only if s is an SSE and s maximizes the sum of utilities of all players among all SSEs.*

Using Lemma 7.16, we can divide $[-1,1]$ into $\gamma(n)/4$-sized subintervals, and query whether a strategy profile has a total expected payment in a given interval.[17]

We are now ready to prove the upper bounds on the power of our ncRIP classes.

Constant utility gap. Using Lemma 7.11 and Lemma 7.16, simulating a constant-gap protocol using a $\mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]}$ machine M is easy. In particular, there are at most $O(1)$ subforms that are reachable under any strategy profile s, and the total expected payment of the provers conditioned on reaching these subforms will be in one of the $O(1)$ subintervals. Thus, there are $O(1)$ combinations of total expected payments on all subforms (including the whole game). M queries its NEXP oracle whether there exists an SSE that achieves that combination of total expected payments on those subforms, for all combinations. Then, M finds the maximum among all of the combinations that got a "yes".

[17] We maintain a constant total budget for V; the payments in our protocols can be scaled so that total is in $[-1,1]$.

Lemma 7.17. $\mathsf{O(1)\text{-ncRIP}} \subseteq \mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]}$.

Proof. Given any $L \in \gamma(\mathsf{n})\text{-ncRIP}$, let $(V, \vec{P})$ be the MRIP protocol with $\gamma(n)$ utility gap for L, where $\gamma(n)$ is a constant.

Given an input x of length n, consider the following deterministic polynomial-time oracle Turing machine M with access to an oracle O for an NEXP language. Similar to the proof of Lemma 7.18, M divides $[-1, 1]$ into $8\gamma(n)$ intervals, each of length $1/4\gamma(n)$. In other words, the ith interval is $[i/4\gamma(n), (i+1)/4\gamma(n))$ for each $i \in \{-4\gamma(n), \ldots, 4\gamma(n) - 1\}$.

Using Lemma 7.11, under a given input x and strategy profile s, there are at most $8\gamma(n)$ subforms are reached under any s in the modified game. Total expected payment of provers acting within any subform (conditioned on reaching the subform) must lie in any one of the $8\gamma(n)$ intervals in $[-1, 1]$. Thus overall, there are $O(\gamma(n)^{\gamma(n)})$ combinations of total expected payments over subforms, which is still $O(1)$. Let $(u, u_{I_1}, \ldots, u_{I_k})$ be a tuple of total expected payments, where $k = 8\gamma(n)$, the maximum number of subforms reachable under any s, and u represents the total expected payment of the whole game, whereas u_{I_j} represents total expected payment of the provers acting in subform I_j (conditioned on reaching I_j).

For each combination $(u, u_{I_1}, \ldots, u_{I_k})$, M queries O: *does there exists a strategy profile that is an SSE and the total expected payments over reachable subforms under s and $O(\gamma(n))$ support Nature moves imposed by Lemma 7.11 is $(u, u_{I_1}, \ldots, u_{I_k})$ (conditioned on reaching the subforms)?* Among the queries to which the oracle's answer is "yes", M finds the combination that achieves maximum expected payment for all subforms. Such a combination is guaranteed to exist because $(V, \vec{P})$ is an ncRIP protocol, and a recursive-maximum SSE of the game exists and maximizes expected payment on all subforms. □

The polynomial-time oracle Turing machine in Lemma 7.17 can issue all its queries *non-adaptively* That is, $\gamma(\mathsf{n})\text{-ncRIP} \subseteq \mathsf{P}_{||}^{\mathsf{NEXP}[\mathsf{O}(1)]}$. Furthermore, we know that $\mathsf{O(1)\text{-ncRIP}} \subseteq \mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]}$. Indeed, the two classes are equal: $\mathsf{P}_{||}^{\mathsf{NEXP}[\mathsf{O}(1)]} = \mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]}$.

Since we prove that $\mathsf{O(1)\text{-MRIP}} = \mathsf{P}_{||}^{\mathsf{NEXP}[\mathsf{O}(1)]}$ in Chapter 3, this shows that cooperative provers are as powerful as non-cooperative provers under constant utility-gap guarantees, and we obtain Corollary 7.4.

Polynomial utility gap. To simulate a polynomial-utility gap ncRIP protocol $(V, \vec{P})$, using a $\mathsf{P}^{\mathsf{NEXP}}$ machine M, we put to use all the structure we have established in this section. We note that the simple strategy of querying all possible payment combinations as in Lemma 7.17 does not work (there are total $O(\gamma(n)^{\gamma(n)})$ combinations).

Lemma 7.18. $\mathsf{poly(n)\text{-ncRIP}} \subseteq \mathsf{P}^{\mathsf{NEXP}}$.

Proof. Given any $L \in \mathsf{poly(n)\text{-}ncRIP}$, let $(V, \vec{P})$ be the ncRIP protocol with $\gamma(n)$ utility gap for L, where $\gamma(n) = n^k$ for some constant k.

Given an input x of length n, consider the following deterministic polynomial-time oracle Turing machine M with access to an oracle O for an NEXP language. M divides $[-1, 1]$ into $8\gamma(n)$ intervals, each of length $1/4\gamma(n)$. In other words, the ith interval is $[i/4\gamma(n), (i+1)/4\gamma(n))$ for each $i \in \{-4\gamma(n), \ldots, 4\gamma(n) - 1\}$.

For each interval $[i/4\gamma(n), (i+1)/4\gamma(n))$, M makes the following queries to O: *does there exist a strategy profile s that is an SSE and the sum of expected payments of all provers $u(x, s)$ is in the ith interval?* Let L denote the set of intervals for which the answer is "yes".

For each interval $[\ell/4\gamma(n), (\ell+1)/4\gamma(n)) \in L$, M queries O: *does there exist a strategy profile s that is an SSE and the sum of expected payments of all provers $u(x, s)$ is in the first half of the ℓth interval?* If the answer is "yes", M recurses on the first half, else M recursives on the second half of the interval. In polynomial time and queries, M can find the exact total expected payment $u(x, s, (V, \vec{P}))$ in the interval that is generated by an SSE. M asks further queries to figure out the exact payment profile under such an SSE. For $k \in \{1, \ldots, p(n)\}$, where $p(n)$ is the total number of provers in $(V, \vec{P})$, and for each $j \in \{1, \ldots, n^{k'}\}$, where $n^{k'}$ is the running time of V (k' is a constant), M asks the following queries adaptively: *under an SSE where $\sum_{i=1}^{p(n)} \mu_i(x, s) = u(x, s)$, what is the jth bit in the expected payment $\mu_k(x, s)$ of prover P_k, given and the first $j-1$ bits of $\mu_k(x, s)$ and $\mu_1(x, s), \ldots, \mu_{k-1}(x, s)$.* In $O(n^{k'}p(n))$ queries, M can figure out the exact payment profile $\tilde{u}(x, s) = (\mu_1, (x, s) \ldots, \mu_k(x, s))$ under an SSE s, such that the total expected payment is in the ℓth interval.

M now verifies whether the SSE corresponding to the payment profile $\tilde{u}(x, s)$ satisfies the condition of Observation 7.10. M proceeds in two phases: first, M goes "top-down" figuring out what part of the game tree is being played under s on input x, using the oracle to simulate the provers and the verifier. Then, it goes "bottom-up" in the tree under play to check whether all subforms are "$(1/\gamma(n))$-close" to the recursive-maximum at that subform.

Top-down phase. Let $k(n)$ be the total number of rounds in $(V, \vec{P})$. Note that $k(n)$ is polynomial in n. Let m_{ij} denote the message sent by prover P_i at round j. Then, for each round j and each prover i where $1 \leq j \leq k(n)$ and $1 \leq k \leq p(n)$, M first asks the oracle to give the "pruned" $O(\gamma(n))$ support distribution imposed by the Nature move of V at round j bit by bit as follows: *"under an SSE where the expected payment profile is $\tilde{u}(x, s)$, what is the rth bit of the distribution imposed by V' using V and Lemma 7.11?"* This requires a polynomial number of bits (and therefore queries) because the distribution is polynomial sized. The pruned distribution preserves the recursive-maximum SSE and changes the utility gap by only a factor 2 (this factor does not affect the proof as our intervals are scaled down

to handle it). Given this distribution, M simulates V on the support of the distribution to figure out the messages that V sends to the provers in round j. In particular, M does not have access to random bits, so instead it simulates *every* action of V in the support. To simulate the provers at round j, M similarly queries O bit by bit: *"under an SSE where the expected payment profile is $\tilde{u}(x,s)$, what is the rth bit of the message sent by P_k"*. Thus, after simulating the moves of V and P under s, M has sketched out the $O(\gamma(n))$ size part of the game tree being played under s corresponding to $\tilde{u}(x,s)$.

Bottom-up phase. Given the $O(\gamma(n))$ nodes reached under s, M can mark out the subforms reachable under s (corresponding to $\tilde{u}(x,s)$). Starting from last level and going up, for each subform H_I reachable under s, M uses the oracle to figure out which payment interval the expected payments of the maximum SSE on H_I lie in (given the expected maximum SSE payments on the reachable subforms verified so far), until it finds a subform that violates the condition of Observation 7.10.

In particular, for each subform H_I of height k, let $\tilde{u}(x,s,I')$ denote the tuple of total expected payments under s on all subforms $H_{I'}$ of height $< k$ following I (conditioned on reaching I) verified so far. M divides the interval $[-1,1]$ into $8\gamma(n)$ intervals of size $\gamma(n)/4$ as before and for each interval queries the oracle O: *does there exist a strategy s_I on subform H_I that is an SSE and the total expected payments of provers $u(x,s,I)$ is in the xth interval, and gets a total expected payments on subforms $H_{I'}$ of height $< k$ following I equal to $\tilde{u}(x,s,I')$.*[18]

Then, M finds the maximum interval $[i/4\gamma(n),(i+1)/4\gamma(n))$ among the intervals for which the oracle says yes. By Lemma 7.16, the maximum SSE, s_I^* at H_I also lies in the ith interval. Using the probability p_I assigned by H_I (M knows the distribution imposed by all "pruned" Nature moves), M checks whether the total expected payment of maximum SSE s_I^* is in the same interval as the sum of expected payments of provers in Z_I under s. If it is not, then s fails the test and M continues to the next interval in L. Otherwise, M continues to the next reachable subform.

If s passes the test for all subforms, then by Observation 7.10, the answer bit under s is correct. M's final query to O is: *"under an SSE where the expected payment profile is $\tilde{u}(x,s)$, what is the answer bit c?* If $c=1$, then M accepts x, otherwise M rejects x.

M is guaranteed to find a payment profile $\tilde{u}(x,s)$ (and thus a strategy profile s) that passes the test. Since $(V,\vec{P})$ is an ncRIP protocol for L, there exists a recusive-maximum SSE s^* in some interval in L. By Observation 7.10, if a strategy profile s' fails the test,

[18] M does not *need* to send the total expected payments of the subforms at lower levels (that have already been verified). Instead, M can just send the total expected payment $u(x,s)$ at the root and ask O to guess s as well. While an NEXP oracle cannot find max SSEs, it can guess two strategy profiles and compare their expected payments. This observation is crucial in extending this proof to exponential utility gap.

the recursive-maximum SSE can not get a total expected payment in the same interval as s'. Thus, we can rule out intervals by checking any SSE with total expected payment in that interval. Since a recursive-maximum SSE s^* exists, M must eventually find an interval, where the corresponding SSE passes the test.

To complete the proof, we prove that M runs in polynomial time, and the oracle queries can be answered by a NEXP machine.

M is polynomial time because each top-down and bottom-up phase is executed $O(\gamma(n))$ times and both take polynomial time. In the top-down phase, M simulates the protocol on strategy s with Nature moves of $O(\gamma(n))$ support. In the bottom-up phase, M finds maximum SSEs at $O(\gamma(n))$ reachable subforms and $O(\gamma(n))$ interval queries for each subform.

Finally, the oracle queries can be answered by an NEXP machine because it can guess a strategy s of the provers (which is at most exponential in size), compute expected payments under s, and and by Lemma 6.7 verify whether s is an SSE in exponential time. □

Exponential utility gap. Finally, we prove a tight upper bound on the class of ncRIP protocols with exponential utility gap. The proof follows immediately from that of Lemma 7.18. In fact, it is simpler as the exponential-time Turing machine is powerful enough to (a) simulate V's Nature moves directly, and (b) test all possible payment profiles.

Lemma 7.19. $\mathsf{ncRIP} \subseteq \mathsf{EXP}^{\mathsf{poly-NEXP}}$.

Since $\mathsf{EXP}^{\mathsf{poly-NEXP}} \subseteq \mathsf{EXP}_{||}^{\mathsf{poly-NEXP}} = \mathsf{EXP}_{||}^{\mathsf{NP}}$, and $\mathsf{EXP}_{||}^{\mathsf{NP}} \subseteq \mathsf{MRIP}$ [43], Lemma 7.19 shows that $\mathsf{ncRIP} \subseteq \mathsf{MRIP}$ and using Lemma 7.9, we get that in general the two classes coincide. In other words, non-cooperative rational proofs are as powerful as cooperative multi-prover rational proofs under exponential utility gap and we obtain Corollary 7.6.

7.4 Optimal Number of Provers and Rounds

Theorem 7.2 and Lemma 7.8 together show that the full power of ncRIP protocols with polynomial gap can be captured by 3 provers and 5 rounds. In this section, we prove that this is tight, that is, 3 provers and 5 rounds are optimal for polynomial-gap ncRIP protocols.

We first start with a structural lemmas about recursive-maximum SSEs and extensive form games with less than 3 provers and less than 5 rounds.

Lemma 7.20. *Consider an extensive-form game with 2 players and k rounds. Then any maximum SSE of the game is also a recursive maximum SSE.*

Proof. Let s be the maximum SSE of the extensive-form game with players P_1 and P_2 such that s is not a recursive-maximum SSE. Then, there exists an information set I and a

subform F_I rooted at I such that s_I is not a maximum SSE (with respect to μ_I deduced by Bayes' rule if I is reachable or with respect to any distribution μ_I if I is unreachable).

Thus, there exists another strategy s' such that s'_I is an SSE and without loss of generality P_1 gets a strictly better payment under s' in F_I (without hurting P_2's payment in F_I). That is, $u_1(x, s'_I) > u_1(x, s_I)$ and $u_2(x, s'_I) \geq u_2(x, s_I)$. Consider the strategy profile $s^* = (s_{-I}, s'_I)$, that is, s on the entire game tree except subform F_I, and s' on F_I. Then s^* is an SSE of the entire game. This is because both s and s' are an SSE. Furthermore, $u_1(x, s^*) > u_1(x, s)$ and $u_2(x, s^*) \geq u_2(x, s)$. This is a contradiction as s is the maximum SSE. □

For the next lemma, we recall that we count the provers' and verifier's messages as separate rounds. Furthermore, the first prover sends the message in the first round by default. Thus three rounds are optimal in a non-trivial ncRIP protocol.

Lemma 7.21. *Consider an ncRIP protocol with p provers and* 3 *rounds. Then any maximum SSE of the resulting game is also a recursive-maximum SSE.*

Proof. In the extensive-form game resulting from a 3-round ncRIP protocol $(V, \vec{P})$, the only subforms start in the last and first round (where latter is the entire game). This is because the verifier's messages in round 2 only imposes a probability distribution over the round-3 information sets of the provers, and any strategy that maximizes their expected payment conditioned on reaching the last round, also maximizes it under the verifier's distribution. All the subforms starting in round 3 have height 1, and thus consist of a single information set I. Let s be a maximum SSE of the entire game. If s is not a recursive-maximum SSE then there must exist a round-3 information set I such that, conditioned on reaching this information set, s_I is not the maximum SSE.

That is, there exists another SSE s' such that the single prover $P(I)$ acting in I gets a better payment under s'_I compared to s_I. Consider the strategy $s^* = (s_I, s'_I)$, that is, replace the strategy on information set I by s' keeping the rest the same. Then, $u_j(x, s^*) \geq u_j(x, s)$ for all $j \in \vec{P} \setminus P(I)$. The expected payment of $P(I)$ is strictly better under s^*. This contradicts the assumption that s is the recursive maximum SSE. □

Finally, using Lemma 7.20 and Lemma 7.21, we show that any ncRIP protocol with polynomial utility gap that has less than 3 provers, or has less than 5 rounds can be simulated by a polynomial-time Turing machine with nonadaptive access to an NEXP oracle, that is, the power of the class reduces to that of cooperative rational provers.

Theorem 7.22. *Let* $\mathsf{poly(n)\text{-}ncRIP(p, k)}$ *denote the class of languages decided by an ncRIP protocol with polynomial gap using p provers and k rounds, then we have the following:* $\mathsf{poly(n)\text{-}ncRIP(2, k)} \subseteq \mathsf{P}_{||}^{\mathsf{NEXP}}$ *and* $\mathsf{poly(n)\text{-}ncRIP(p, 3)} \subseteq \mathsf{P}_{||}^{\mathsf{NEXP}}$.

Proof. Given any $L \in \mathsf{poly(n)\text{-}ncRIP}$, let $(V, \vec{P})$ be the ncRIP protocol with p provers and k rounds and $\gamma(n)$ utility gap for L, where $\gamma(n) = n^k$ for some constant k. Assume that $p = 2$ or $k = 3$, then by Lemma 7.20 and Lemma 7.21, we only need to a maximum SSE.

Given an input x of length n, consider the deterministic polynomial-time oracle machine M with access to a NEXP oracle. M divides $[-1, 1]$ into $8\gamma(n)$ intervals, each of length $1/4\gamma(n)$. For each interval $[i/4\gamma(n), (i+1)/4\gamma(n))$, M makes the following oracle queries:

1. Does there exist a strategy profile s that is an SSE and the sum of expected payments of all provers $u(x, s)$ is in the ith interval?
2. Does there exist a strategy profile s that is an SSE and the sum of expected payments of all provers $u(x, s)$ is in the ith interval and corresponding answer bit $c = 1$?

Note that M makes $O(\gamma(n))$ nonadaptive queries, each of polynomial size. Furthermore, the queries by M can be answered by an NEXP oracle as in the proof of Lemma 7.18. Finally, M finds the highest index i^* such that interval i^* is "non-empty": that is, the oracle has answered 1 for query 1 for this interval. M accepts if the oracle's answer to query 2 for this interval is 1, and rejects otherwise. The correctness follows from the fact that the protocol $(V, \vec{P})$ has a utility gap of γ and from Lemma 7.16. □

Corollary 7.23. *We need at least* 3 *provers and* 5 *rounds to design an ncRIP protocol for* $\mathsf{P}^{\mathsf{NEXP}}$*, assuming* $\mathsf{P}^{\mathsf{NEXP}} \neq \mathsf{P}_{||}^{\mathsf{NEXP}}$*. Thus, the prover and round complexity of our ncRIP protocol for* $\mathsf{P}^{\mathsf{NEXP}}$ *in Figure 15 is optimal.*

Lemma 7.20 and Lemma 7.21 simplify the solution concept and thus the construction and analysis of ncRIP protocols that do not require more than 2 provers and more than 5 rounds (perhaps for deciding problems in smaller complexity classes). This is because verifying and satisfying a maximum SSE condition is easier than a recursive-maximum SSE condition.

Chapter 8

Scaled-Down Non-Cooperative Rational Proofs

8.1 Introduction

In Chapter 7, we characterized the power of the general class of non-cooperative protocols with constant, polynomial and negligible utility gap. In this chapter, we mirror the work on scaled-down cooperative rational proofs in Chapter 4 to the non-cooperative prover setting.

In particular, we design *highly efficient* non-cooperative rational proofs that have *strong* utility-gap guarantees. Specifically, we design ncRIP protocols for important complexity classes that require only $O(\log n)$ verification and have $O(\log n)$, even $O(1)$ utility gap.

We show that any language in the class NP admits a non-cooperative rational proof with $O(\log n)$ verification and constant utility gap. This result is surprising, since no classical interactive proof for NP is known where the verifier is $O(\log n)$ time while retaining strong soundness guarantees. The work on probabilistically checkable proofs [6, 8, 9, 83, 131] for languages in NP, focuses on improving the query complexity and randomness of the verifier, but the running time of the verifier is still polynomial in n. Making PCP verifier's more efficient, has led to the work on probabilistically checkable "proofs of proximity" [19, 20, 82, 124], where the verifier runs in $O(\log n)$ time, however the soundness conditions are weakened to ensure that the verifier rejects all x "far away" from L (in terms of Hamming distance).

Even with cooperative rational proofs, we were only able to design a protocol for NP with constant utility gap that is efficient in terms of space and randomness, that is, $O(\log n)$ space and randomness, but still polynomial in terms of the verifier's running time.

The main hurdle to making the verification time of these classical and rational protocols sublinear in n, is that if the verifier cannot read the entire input, it is difficult to discover if the provers are lying on a few bits of the proofs. Even in the probabilistic proofs of proximity for NP if an input $x \notin L$ is too close in terms of hamming distance to another input $x \in L$, a sublinear verifier cannot hope to reject it with noticeable probability.

The reason non-cooperative provers are able to overcome this obstacle is because you can use them against each other to find even a single incorrect bit in the proof. In particular, if the first prover gives the verifier a polynomial-size proof that is incorrect at only a couple of bits, the way a log-time verifier catches the mistakes without having to read the entire proof is to incentivize a second prover (who is not cooperative with the first) to report the

discrepancies in the proof. Specifically, if the second prover agrees with the first it gets a payment of 1/2, however if it is able to spot a mistake, which then the $O(\log n)$-time verifier can verify is indeed a mistake by a local spot check, it gets 1 instead. This idea of exploiting the provers' non-cooperativeness directly leads to extremely efficient and powerful protocols.

8.1.1 Overview of Results and Contributions

Next, we summarize the results of this chapter.

Log-time, constant gap protocol for any language in P and NP. We start by designing highly efficient protocols for any language in P and NP. The verification-time of these protocols is only $O(\log n)$, and the utility gap is constant. Another interesting and surprising aspect of these protocols is that the verifier is deterministic. In contrast, both in classical and cooperative rational interactive proofs, the randomness of the verifier is essential to providing the correctness and soundness guarantees of the protocol.

Our protocol for NP uses the protocol for P as a subroutine for the reduction of any $L \in \mathsf{NP}$ to $\mathsf{3\text{-}SAT}$. To facilitate this reduction, we prove a general composition lemma (Lemma 8.3) about ncRIP protocols, which may be of independent interest.

A lower bound on the power of ncRIP protocols with $O(\log n)$-time verifier. We prove that the class of languages that admit a $O(\log n)$-time ncRIP protocol with constant provers and rounds, polynomial communication, and $O(\gamma(n))$ utility gap is at least as large as the class of languages decided by a polynomial-time Turing machine with $O(\gamma(n))$ adaptive queries to an NP oracle, denoted as $\mathsf{P}^{\mathsf{NP}[\gamma(n)]}$. It is well-known that $\mathsf{P}^{\mathsf{NP}}_{||} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$. Thus, using non-cooperative provers, we improve both the running time and the utility gap of the rational protocol for $\mathsf{P}^{\mathsf{NP}}_{||}$ in Chapter 4.4, from polynomial to logarithmic.

An upper bound on the power of ncRIP protocols with $O(\log n)$-time verifier. Finally, we show that any $O(\gamma)$-utility gap ncRIP protocol with constant provers and rounds and $O(\log n)$ verification and communication cost can be simulated by a polynomial-time machine with $O(\gamma(n)$ adaptive queries to an NP oracle.

There is a gap between our upper and lower bound—the protocol for $\mathsf{P}^{\mathsf{NP}[\gamma(n)]}$ requires polynomial communication, while the upper bound only holds for ncRIP protocols with logarithmic communication. In case of cooperative rational proofs, polynomial communication does not add any power over logarithmic when the verifier runs in $O(\log n)$ time; see Chapter 4.3. We leave the problem of resolving this gap for ncRIP as future work.

8.2 Model and Preliminaries

We follow our model in Chapter 4.2 and previous literature on PCPs and $O(\log n)$-time verifiers, see for example [11, 82, 124], and allow the verifier to read only limited bits of the transcript. That is, the $O(\log n)$-time verifier can choose to read a constant many bits of a polynomial-size message sent by the prover.

We extend the model to further allow the verifier to pass on polynomial-size messages sent by one provers to another prover without reading it. This assumption is equivalent to saying that the verifier may let a prover observe some messages sent by the other provers.

The non-cooperative rational proof model, including the solution concept of recursive-maximum strong sequential equilibrium (rmSSE) and the definition of utility gap, used in this chapter is defined in Chapter 6. The only difference is that in this chapter, we focus on the computational cost of the verifier and the communication cost of the protocol. In particular, we restrict the verifier to use $O(\log n)$ time (thus, the space and randomness used is also $O(\log n)$). The communication of the ncRIP protocols is polynomial in Chapter 8.3 and Chapter 8.4 and logarithmic in Chapter 8.5.

To prove the upper bound, we use a scaled-down version of the pruning lemma (Lemma 7.11 in Chapter 7.3). We restate the pruning lemma for the case where the verifier's running time is logarithmic in the input size. The bulk of the proof remains the same, except that it takes polynomial-time instead of exponential to do the pruning.

Lemma 8.1 (**Scaled-Down Pruning Lemma**). *Let $(V, \vec{P})$ be an ncRIP protocol with $\gamma(n)$ utility gap, where V runs in $O(\log n)$ time. Given an input x of length n and a strategy s, $(V, \vec{P})$ can be transformed in polynomial time to a new ncRIP protocol $(V', \vec{P})$, such that*

- *the probability distributions imposed by the nature moves of V' have $O(\gamma(n))$ support,*
- *if s is a rmSSE of $(V, \vec{P})$, s induces a rmSSE in $(V', \vec{P})$,*
- *$|u_j(x, s, (V, \vec{P})) - u_j(x, s, (V', \vec{P}))| < 1/(4\gamma(n))$ for each prover P_J, and*
- *if the answer bit under s is wrong, then there exists a subform H_I in the game of $(V', \vec{P})$ that is reachable under s and a prover P_j acting at H_I, such that P_j loses a $1/(2\gamma(n))$ amount in its expected payment compared to a strategy profile where s_I (induced by s on H_I) is replaced by s_I^* (the recursive-maximum SSE on H_I), keeping the strategy profile outside H_I, s_{-I}, fixed.*

Notation. Similar to Chapter 4, we denote the class of languages decided by an ncRIP protocol with a $O\log n)$-time verifier as $\mathsf{ncRIP}^{\mathsf{t}}$. Furthermore, we denote the class of languages

decided by an ncRIP protocol with a $O(\log n)$-time verifier and has $O(\gamma(n))$ utility gap, $C(n)$ communication cost, $p(n)$ provers, and $k(n)$ rounds, as $\gamma(\mathsf{n})\text{-}\mathsf{ncRIP}^{\mathsf{t}}[\mathsf{C}(\mathsf{n}), \mathsf{p}(\mathsf{n}), \mathsf{k}(\mathsf{n})]$.

8.3 Warm up: $O(\log n)$-time ncRIP Protocols for P and NP

We start by giving a highly-efficient ncRIP protocol with constant utility gap for any languages decidable in polynomial time. At a high level, the protocol works as follows—the first prover provides the transcript of the evaluation of all gates in the DLOGTIME uniform circuit for the polynomial-time language. The second prover is then asked if the transcript is correct and incentivized to point out an incorrectly evaluated gate.

Then, we give a composition lemma that says that a verifier can execute two ncRIP protocols one after the other, without modification, to obtain an ncRIP protocol for the union of the languages decided by both as long as the provers in the two protocols are different and their payments are independent.

Using the composition lemma and the protocol for P, we finally give a highly-efficient ncRIP protocol for any language in the class NP.

We note that the protocols in the section use polynomial communication, and only constant number of provers and rounds, and have constant utility gap.

Lemma 8.2. *Any language $L \in \mathsf{P}$ has an ncRIP protocol that uses 2 provers, 3 rounds, $O(\text{poly}(n))$ communication, $O(\log n)$ verification time, and has $O(1)$ utility gap.*

Proof. Let $L \in \mathsf{P}$, then following Lemma 2.6, there exists a DLOGTIME circuit family $\{C_n\}_{n=1}^{\infty}$ that computes L. Let $g = n^k$ 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 ncRIP protocol $(V, \vec{P})$ for L is given in Figure 17.

To see correctness, we show that under a rmSSE, the verifier learns the correct answer bit $c = v_g(x)$. Since the protocol has only two provers, using Corollary 7.23, it is sufficient to prove that under a max SSE, the verifier learns the correct answer bit.

Let P_1 and P_2 use strategies s_1 and s_2, and let $s = (s_1, s_2)$. We show that if s is a maximum strong sequential equilibrium, then $c = 1$ iff $x \in L$.

We argue by backward induction. The last move is by P_2 at step 3. Since P_2 has full information about the history leading up to this step, thus its information set is a singleton consisting of the tuple T of values. If T corresponds to a valid evaluation of all gates of the circuit on input x, then P_2's best response is to say yes, and get a payment of $1/2$. Otherwise, it would be unable to present a gate that evaluates incorrectly and get a payment of 0. On

Given any string x of length n, $(V, \vec{P})$ proceeds as follows.

1. P_1 sends the transcript T of values of all the gates of C_n on x, that is, $T = (v_1(x), \ldots, v_g(x))$ to V. V outputs $c = v_g(x)$ at the end of the protocol.
2. V sends T to P_2 (without reading it) and asks if T sent by P_1 is correct.
3. If P_2 says yes, then $R_1 = 1$, $R_2 = 1/2$ and the protocol ends. If P_2 says no, then it must also provide the gate index j of the gate that is is not been evaluated correctly in T.
4. V uses DC uniformity to find the input gates j_1, j_2 of gate j and the type of gate j. Using T, V checks if v_{j_1}, v_{j_2} and v_j satisfy the Boolean logic of the gate according to its type. If they do, $R_1 = 1$, $R_2 = 0$. Otherwise, $R_1 = 0$ and $R_2 = 1$.

Figure 17: A constant-utility gap, $O(\log n)$-time ncRIP protocol for P.

the other hand, if T corresponds to an invalid evaluation of gates of the circuit on x, that is, there exists a gate index j with input gates indices j_1 and j_2 such that v_{j_1}, v_{j_2} and v_j together do not satisfy the Boolean logic of gate j then P_2's best response is to send V the index j and get a payment of 1. This is because if instead it lies and says T is valid then it only gets a payment of 1/2.

Given that P_2's best strategy is to report an invalid T and agree with a valid T, P_1's best response at step 1 is to commit to a valid transcript T and get a payment of 1. This is because sending an invalid transcript leads to a payment of 0 for P_1, given P_2's best strategy.

Thus, any strategy profile (s_1, s_2) where s_1 commits to a valid transcript T and s_2 agrees is an SSE. Since all such strategies receive an expected payment $R_1 = 1$ and $R_2 = 1/2$, they are also maximum SSEs. Under any maximum SSE, the verifier learns the correct answer bit $c = v_g(x)$, (since the transcript T is correct and includes the output gate).

Finally, note that V runs in $O(\log n)$ time as it only computes the input gates, type of a single gate in step 4 and computes the value of that gate. The communication complexity is the size of the transcript $O(n^k)$. The utility gap is constant because if P_2 lies in step 3 (says yes on an incorrect T or no on a correct T), it loses a constant amount and given P_2 follows its best strategy, P_1 loses a constant amount if it misreports the answer bit. □

The high-level idea of the next protocol is the same as that of Lemma 8.2. In particular, if P_1 claims the input x is a language is a language $L \in \mathsf{NP}$, it is asked to provide a certificate to prove it. Then P_2 is incentivized to call out invalid certificates.

First, we show a composition theorem about the extensive form game resulting from two ncRIP protocols with different provers executed one after the other by the same verifier.

Lemma 8.3. *Let $(V, \vec{P})$ be an ncRIP protocol for a language L and $(V, \vec{P}')$ be an ncRIP protocol for a language L' such that $\vec{P}$ and $\vec{P}'$ do not have a single prover in common. Then the protocol $(V, \vec{P}, \vec{P}')$ where V executes $(V, \vec{P})$ first with $\vec{P}$ including computing their final payment, followed by $(V, \vec{P}')$ with $\vec{P}'$ is an ncRIP protocol for the language $L \cup L'$.*

Proof. To be consistent with the model, add an additional prover P_0 that sends the answer bit $c^* = c \vee c'$, where c is the answer bit sent in $(V, \vec{P})$ and c' is the answer bit sent in $(V, \vec{P}')$. V outputs c^* at the end of the protocol. P_0 gets a payment $R = 1$ if after executing both protocols, V verifies that c^* is in fact $c \vee c'$, and 0 otherwise.

Consider the extensive-form game G resulting from the composed protocol $(V, P_0, \vec{P}, \vec{P}')$. Let s be a rmSSE of $(V, \vec{P})$ and s be a rmSSE of $(V, \vec{P}')$. We know that such s and s' exist from the definition of them being ncRIP protocols. Consider the strategy profile $s^* = s \circ s'$ for the game tree G, where s^* imposes strategy s on the nodes corresponding to the moves of $\vec{P}$ and s^* imposes s' on the moves of $\vec{P}'$. Then s^* is a rmSSE of the entire game tree since the utilities of $\vec{P}$ and $\vec{P}'$ remain the same under s^* compared to s and s' respectively. This is because the payments of $\vec{P}$ and $\vec{P}'$ are completely independent. Thus, under s^*, we have that $c = 1$ iff $x \in L$ and $c' = 1$ iff $x \in L'$ and thus $c^* = 1$ iff $x \in L \cup L'$. □

Lemma 8.4. *Any language $L \in \mathsf{NP}$ has an ncRIP protocol that uses $O(1)$ provers, $O(1)$ rounds, $O(\text{poly}(n))$ communication, $O(\log n)$ verification time, and has $O(1)$ utility gap.*

Proof. Given an input x and a language $L \in \mathsf{NP}$, there exists a polynomial-time computable function f that converts x to a 3-SAT instance ϕ_x such that $x \in L$ if and only if $\phi_x \in$ 3-SAT. Since f is computable in polynomial-time, there exists a DLOGTIME uniform circuit family $\{C_n\}_{n=1}^{\infty}$ that given x computes $f(x) = \phi_x$. We use provers P_1 and P_2 to send a transcript T committing to the correct evaluation of the circuit to compute ϕ_x, similar to the protocol described in Figure 17. In particular, P_1 sends T and P_2 is asked if T is valid or not. From the correctness of protocol Figure 17, we know that under a maximum SSE, P_1 must commit to a valid T and P_2 must agree. Thus, V can obtain the correct 3-SAT instance ϕ_x.

Next, V uses two different provers—P_3 and P_4 to run the ncRIP protocol to determine is $\phi_x \in$ 3-SAT described in Figure 18. By Lemma 8.3 and the correctness of the protocol in Figure 17, it is sufficient to prove correctness of the protocol in Figure 18 for the overall language $L \in \mathsf{P} \cup \mathsf{NP} = \mathsf{NP}$. To see correctness of the protocol for 3-SAT in Figure 18, it is sufficient to prove that under a max SSE, the verifier learns the correct answer bit c.

Let P_3 and P_4 use strategies s_3 and s_4, and let $s = (s_3, s_4)$. We argue by backward induction. The last move is by P_4 at step 4. Since P_4 has full information about the history leading up to this step, its information set is a singleton step consisting of the assignment A. If A is such that all clauses of ϕ_x are satisfied, then P_4's best response is to say yes, and

Given an 3-SAT instance ϕ_x of length n, (V, P_3, P_4) proceeds as follows.

1. P_3 sends an answer bit c and string A. V outputs c at the end of the protocol.
2. If $c = 0$, V ignores A, sets $R_3 = 1/2$, $R_4 = 0$ and the protocol ends.
3. If $c = 1$, A must correspond to the satisfying assignment of ϕ_x. V sends A to P_4 (without reading it) and asks if A is valid.
4. If P_4 says yes, $R_3 = 1$, $R_4 = 1/2$ and the protocol ends. If P_4 says no, then it must provide a clause index j such that the jth clause is not satisfied under A.
5. V uses the circuit for f to compute the clause description $(x_a \vee x_b \vee x_c)$ of the jth clause and compute its value using $A(x_a)$, $A(x_b)$ and $A(x_c)$. If the clause is satisfied, $R_3 = 1$, $R_4 = 0$. Otherwise, $R_3 = 0$, $R_4 = 1$.

Figure 18: A constant-utility gap $O(\log n)$-time ncRIP protocol for 3-SAT.

get a payment of 1/2. Otherwise, it would be unable to present an unsatisfied clause and get a payment of 0. On the other hand, if A is such that there exists a clause j in ϕ_x that is not satisfied, then P_4's best response is to send V the index j and get a payment of 1 (it can only get 1/2 by saying yes in this case).

Given that s_4 is such that P_4 always reports an invalid assignment (that is, an assignment under which ϕ_x is not satisfied), P_1's best response at step 3 is to send a satisfying assignment A and receive 1, otherwise it receives 0. Given that P_1 must send a satisfying assignment in step 3, its best response in step 1 is to commit to the correct answer bit. That is, if $x \in L$, and P_1 sends $c = 0$, P_1 only receives 1/2, while sending $c = 1$ on the other hand can get P_1 a payment of 1. Furthermore, if $x \notin L$ then P_1 should send $c = 0$ and get a payment of 1/2, otherwise its invalid assignment would be reported leading to a payment of 0.

Thus, any strategy profile (s_3, s_4) where s_1 reports $c = 0$ if $x \in L$ and $c = 1$ if $x \notin L$ and s_2 always reports an invalid assignment is an SSE. Since all such strategies receive a payment $R_1 = 1$ and $R_2 = 1/2$ if $x \in L$, and $R_1 = 1/2$ and $R_2 = 0$ if $x \notin L$, which are the maximum possible respectively, they are also maximum SSEs. Thus, under any maximum SSE, the verifier learns the correct answer bit c.

The protocol has constant utility gap because of the following "bottom-up" reasoning. If P_4 lies in step 4 that is, if T is invalid P_4 says yes, or if T is valid and P_4 says no, then in both cases P_4 loses a constant amount from its payment. Given that P_4 follows the max SSE strategy, if P_3 reports the incorrect answer bit c, it loses a constant amount of its payment.

Finally, we note that the running time of the verifier V is $O(\log n)$ because V only checks the satisfiability of a single clause of length $O(\log n)$ bits. The protocol uses five provers—

two provers for the polynomial time reduction to 3-SAT, two provers for the 3-SAT protocol and an additional prover as in proof of Lemma 8.3 to give the overall answer bit. □

8.4 A Lower Bound on $O(\log n)$-time ncRIP Protocols

In this section, we show how to combine the protocols for P and NP we designed in the previous section to obtain an ncRIP protocol with a $O(\log n)$-time verifier and $O(\gamma(n))$ utility gap for any language decided by $\mathsf{P}^{\mathsf{NP}[\gamma(\mathsf{n})]}$. Recall that $\gamma(n)$ is any function that takes positive integral values, is computable in polynomial time (given 1^n) and is polynomially bounded.

In our protocol, we repeatedly make use of the fact that the verifier is allowed to send one provers's message to another without reading it. This is essentially analogous to a game of perfect information where each prover is completely aware of the history leading up to its turn. Thus, the polynomial-size messages sent by the provers count towards the communication complexity of the protocol and not the verifier's running time for verification.

Lemma 8.5. *Any language $L \in \mathsf{P}^{\mathsf{NP}[\gamma(\mathsf{n})]}$ has an ncRIP protocol that uses $O(1)$ provers and rounds,* $\mathrm{poly}(n)$ *communication, $O(\log n)$-time verifier and has $O(1)$ utility gap. That is,*

$$\mathsf{P}^{\mathsf{NP}[\gamma(\mathsf{n})]} \subseteq \gamma(\mathsf{n})\text{-}\mathsf{ncRIP}^{\mathsf{t}}[\mathrm{poly}(\mathsf{n}), \mathsf{O}(1), \mathsf{O}(1)].$$

Proof. For any language $L \in \mathsf{P}^{\mathsf{NP}[\gamma(\mathsf{n})]}$, let M be the polynomial-time oracle Turing machine deciding L and O be the oracle used by M. Without loss of generality, we assume $O = \mathsf{3\text{-}SAT}$. And without loss of generality, we assume that M makes $\gamma(n) \geq 1$ queries on any input x of length n. We use γ to denote $\gamma(n)$ when n is clear from context.

Let $(\phi_1, \phi_2, \ldots, \phi_\gamma)$ denote the sequence of oracle queries in the order they are made by M. The ncRIP protocol that simulates M and the oracle queries is given in Figure 19.

For a given input x of length n, let the oracle O's answer to query ϕ_i be c_i, that is, $c_i = 1$ if and only if $\phi_i \in \mathsf{3\text{-}SAT}$ for all $1 \leq i \leq \gamma$. Then, the polynomial-time computation of machine M on input x, given $c_1, \ldots, c_\gamma$ can be represented by a DLOGTIME uniform circuit C_n of size $O(n^k)$. We use the protocol in Figure 17 to simulate this circuit and obtain the transcript T of it. V uses two more provers to obtain the description of the description of ϕ_i according to T. Note that we allow the verifier to pass the messages sent by one prover to others. Once V obtains the instance ϕ_i, it uses the final two provers to execute the 3-SAT protocol in Figure 18. The complete protocol is described in Figure 19.

We argue correctness using backwards induction. Step 7 starts the first proper subform. By correctness of the protocol in Figure 18, we know that under a recursive-maximum SSE c_i^* is the correct answer to ϕ_i.

For any input x of length n, the ncRIP protocol $(V, \vec{P})$ works as follows.

1. P_1 sends $(c, c_1, \ldots, c_\gamma) \in \{0,1\}^{\gamma+1}$ to V. V outputs c at the end of the protocol.
2. V sends $(c_1, \ldots, c_\gamma)$ to P_2 and P_3 and simulates the protocol in Figure 17 with them to obtain the transcript of values of all gates of circuit C_n corresponding to M, given x and oracle answers $(c_1, \ldots, c_\gamma)$. Let c^* be the answer bit of the protocol, if $c^* \neq c$, then $R_1 = 0$ and the protocol ends.
3. V picks a random index i from $\{1, \ldots, \gamma\}$ sends (i, T) to P_4 and P_5. V asks P_4 for the ith 3-SAT query according to T.
4. P_4 sends ϕ_i'. V sends ϕ_i' to P_5 and asks if ϕ_i matches the ith 3-SAT query ϕ_i in T.
5. If P_5 says yes, $R_4 = 1$, $R_5 = 1/2$ and protocol continues. If P_5 says no, it must point the gate index g_j such that the output of gate g_j is the jth bit of ϕ_i which does not match the jth bit of ϕ_i'.
6. V uses uniformity to check if g_j is the gate that outputs the jth bit of ϕ_i. If it is not, then $R_4 = 0, R_5 = 0$ and the protocol ends. Else if, the jth bit of ϕ_i' does not match the jth bit of ϕ_i, $R_4 = 0$, $R_5 = 1$ and the protocol ends. Else, $R_4 = 1$, $R_5 = 0$.
7. V sends ϕ_i to P_6 and P_7 and executes the protocol for 3-SAT in Figure 18. Let c_i^* be the answer bit. If $c_i^* \neq c_i$, $R_1 = 0$; otherwise, $R_1 = 1$.

Figure 19: An $O(\gamma(n))$-utility gap ncRIP protocol for $\mathsf{P}^{\mathsf{NEXP}[\gamma(n)]}$.

The second-last move is by P_5 in step 5. Regardless of the strategy of others, under a recursive-maximum SSE, given a transcript T and query ϕ_i', P_5's must say yes if they are consistent and get 1/2 (instead of 0), or say no if they are inconsistent and get 1 (instead of 1/2). Given that P_5 reports any inconsistency between T and ϕ_i', in step 4 P_4 should send ϕ_i' that is consistent with T and get a payment of 1, instead of lying and getting 0.

One level up, at step 2, we know from the correctness of the protocol in Figure 17, that given $c_1, \ldots, c_\gamma$, under a recursive-maximum SSE, P_2 must commit to the correct transcript T and P_3 must agree with P_2.

Finally, given that P_2 and P_3 commit to the correct transcript T, and P_6 and P_7 report the correct answer to the 3-SAT query in step 7, under a recursive-maximum SSE, P_1 must commit to the correct answer bits $c, c_1, \ldots, c_\gamma$ in step 1. Suppose there exists a j such that c_j is incorrect, where $1 \leq j \leq \gamma$, then since ϕ_j is evaluated correctly based on x and $c_1, \ldots, c_{j-1}$ and V picks j in step 3 with probability $1/\gamma$, we have that c_j^* will not match c_j in step 7 and thus P_1 gets 0, making its overall expected payment at most $1 - 1/\gamma$ in this case. On the other hand, if all answer bits are reported correctly, its expected payment is 1 under a recursive-maximum SSE. Thus, its best strategy is to report each c_j correctly. Finally, given $(c_1, \ldots, c_\gamma)$ are correct, if P misreports the answer bit c, it will not match c^* in step 2 leading to a payment of 0. Thus, under a recursive-maximum SSE, $c = 1$ if and only if $x \in L$.

We now show that the utility gap of the protocol is $O(\gamma)$. First, consider all the 3-SAT queries at step 7 that start subforms. Suppose there exists a query ϕ_k, for $1 \leq k \leq \gamma$, such that c_k* is the wrong answer to ϕ_k. Since the protocol in Figure 18 has $O(1)$ utility gap, either P_6 or P_7 loses a constant amount from their payment (conditioned on reaching this subform). V chooses ϕ_k with probability $1/\gamma$, P_2 and P_3 lose $1/O(\gamma)$ from their expected payment in this case. Given P_6 and P_7 answer all 3-SAT queries correctly, it is easy to see that P_4 and P_5 lose a constant amount from the expected payment if they deviate from the recursive-maximum SSE in steps 4 and 5. Moving up, since the protocol in Figure 17 has constant utility gap, P_2 and P_3 lose a constant amount if they deviate from their best strategy. Finally, given everyone else plays their best strategy, we argued above that P_1 loses $1/O(\gamma)$ on deviating from the recursive-maximum SSE.

Since the verification is $O(\log n)$ time for the protocols in Figure 17 and Figure 18, and V only checks $O(1)$ gates and probes $O(1)$ string indices in step 5 and uses $O(\log n)$ random bits in step 3, the overall verification time is $O(\log n)$ time. Finally, the protocol has polynomial communication as γ, T, ϕ_i are polynomial in n. □

8.5 A Upper Bound on $O(\log n)$-time ncRIP Protocols

In this section, we prove a weak upper bound on the power of $O(\log n)$-time ncRIP protocols with communication complexity of $O(\log n)$. In particular, we prove the following.

Lemma 8.6. *If a language L has an ncRIP protocol that uses $O(1)$ provers and rounds, $O(\log n)$ communication, $O(\log n)$-time verifier and has $O(1)$ utility gap, then $L \in \mathsf{P}^{\mathsf{NP}[\gamma(n)]}$. That is,*

$$\gamma(\mathsf{n})\text{-}\mathsf{ncRIP}^{\mathsf{t}}[\mathsf{O}(\log \mathsf{n}), \mathsf{O}(1), \mathsf{O}(1) \subseteq \mathsf{P}^{\mathsf{NP}[\gamma(n)]}.$$

Thus, there is a communication-complexity gap between between the lower bound in Lemma 8.5 which requires polynomial communication and this upper bound which holds for logarithmic communication. We leave the problem of resolving this gap (either by improving the communication cost of the ncRIP protocol in Figure 19 or by tightening the upper bound to hold for polynomial communication protocols) for future work.

Proof of Lemma 8.6. Given an input x of length n, let L be language that has an ncRIP protocol $(V, \vec{P})$ with $O(1)$ provers and rounds, $O(\log n)$ communication and verification and $O(\gamma(n))$ utility gap. Consider the following deterministic polynomial-time oracle Turing machine M with access to an oracle O for an NP language. Similar to all utility-gap based upper bounds, M divides $[-1, 1]$ into $8\gamma(n)$ intervals, each of length $1/4\gamma(n)$. In other words, the ith interval is $[i/4\gamma(n), (i+1)/4\gamma(n))$ for each $i \in \{-4\gamma(n), \ldots, 4\gamma(n) - 1\}$.

We note that since the verifier's running time and communication complexity is $O(\log n)$, and the number of provers and rounds is $O(1)$, the size of the complete game tree of the protocol $(V, \vec{P})$ is polynomial in n. Thus, a strategy profile s of the provers can be specified in polynomial bits and an NP oracle can guess such a s in non-deterministic polynomial time.

We divide the proof into two cases depending on the value of $\gamma(n)$.

Case 1. $\gamma(n) = O(1)$.

The proof for this case is similar to the proof of $\mathsf{O(1)\text{-}ncRIP} \subseteq \mathsf{P}^{\mathsf{NEXP}[\mathsf{O}(1)]}$ (Lemma 7.17 in Chapter 7.3). In particular, using Lemma 8.1, under a given input x and strategy profile s, there are at most $O(\gamma(n))$ subforms that are reached under any s in the modified game. Total expected payment of provers acting within any subform (conditioned on reaching the subform) must lie in any one of the $8\gamma(n)$ intervals in $[-1, 1]$. Thus overall, there are $O(\gamma(n)^{\gamma(n)})$ combinations of total expected payments over subforms, which is $O(1)$ for $\gamma(n) = O(1)$. Let $(u, u_{I_1}, \ldots, u_{I_k})$ be a tuple of total expected payments, where $k = 8\gamma(n)$, the maximum number of subforms reachable under any s, and u represents the total expected

payment of the whole game, whereas u_{I_j} represents total expected payment of the provers acting in subform I_j (conditioned on reaching I_j).

For each combination $(u, u_{I_1}, \ldots, u_{I_k})$, M queries O: *does there exists a strategy profile s that is an SSE and the total expected payments over reachable subforms under s and $O(\gamma(n))$ support Nature moves imposed by Lemma 8.1 is $(u, u_{I_1}, \ldots, u_{I_k})$ (conditioned on reaching the subforms)?* Among the queries to which the oracle's answer is "yes", M finds the combination that achieves maximum expected payment for all subforms and accepts if the answer bit under such a strategy is 1, otherwise rejects. Such a combination is guaranteed to exist because $(V, \vec{P})$ is an ncRIP protocol, and thus a rmSSE of the game exists.

Case 2. $\Omega(\log n) \leq \gamma(n) \leq O(\text{poly}(n))$.

For each of the $O(\gamma(n))$ subintervals of the interval $[-1, 1]$, M first queries the oracle to find out if there exists an SSE with total expected payment $u(x, s)$ in the subinterval. For all the subintervals that correspond to an SSE, M does a recursive search to find an exact total expected payment $u(x, s)$ that is generated by an SSE. In particular, M queries the NP oracle: *Does there exist an SSE with total expected payment in the first half of the ith interval?* If the oracle says yes, M then recurses on the first half if the answer is *yes*, else recurses on the second half. Since the verifier runs in $O(\log n)$ time, any expected payment generated is at least $1/n^k$, for some constant k. Thus, in $O(\log n)$ time and $O(\log n) = O(\gamma(n))$ oracle queries, M can find an exact $u(x, s)$ for an SSE s in the subinterval using its *adaptive* queries.

Using Lemma 8.1, under a given input x and strategy profile s, there are at most $O(\gamma(n))$ subforms that are reached s in the modified game. Without loss of generality, let the subform reached (in order) under any s be indexed $1, \ldots, \ell\gamma(n)$ for some constant ℓ. Then for each $j \in \{1, \ldots, \ell\gamma(n)\}$, M asks the NP oracle: *does there exist an SSE with total expected payment $u(x, s)$ and another SSE s' such that for the jth subform rooted at information set I there exists a prover P_k acting in the subform such that: $u_j(x, (s_{-I}, s'_I)) - u_j(x, s) > 1/\gamma(n)$.* We note that an NP oracle can guess two strategy profile s, s', as both of them are polynomial in size, check if they are an SSE, execute them and compare their expected payments. If for any index j, the oracle says yes, we say that this interval and this s have "failed" the test of Observation 7.10. Using Claim 7.15, we know that the rmSSE cannot be in this subinterval.

Since $(V, \vec{P})$ is an ncRIP protocol, a recursive-maximum SSE exists and thus there exists a subinterval and an SSE s with $u(x, s)$ in that subinterval that passes the above test for all subforms. The answer bit under such s must be correct. For that subinterval, M asks a final question: *"under an SSE where the expected payment profile is $\tilde{u}(x, s)$, what is the answer bit c?* If $c = 1$, M accepts, else M rejects. □

Chapter 9

Adaptive Bloom Filters to Minimize Remote Accesses

9.1 Introduction

So far in this dissertation, we focused on designing efficient protocols for verifiable computation outsourcing. We used payments to leverage the rationality of service provers in a marketplace, where the service providers can be cooperative or non-cooperative.

In this chapter, we focus on a different scenario that also arises in settings involving a weak client—*storage outsourcing*. In particular, most memory-constrained devices store their large input data on external servers or clouds. Outsourcing the data to external servers then leads to expensive *remote accesses*. We focus on minimizing such remote accesses in the context of the following specific algorithmic problem—*how do we efficiently test membership and perform updates on a set that is stored remotely?*

In this chapter, we present adaptive and more efficient variants of a *Bloom filter*, a data structure that is widely used to speed up membership queries to a remote set. The results described in this chapter are based on [21].

A Bloom filter [24,29]—or, more generally, an *approximate membership query* data structure (AMQ[19])—maintains a compact, probabilistic representation of a set $\mathcal{S}$ from a universe $\mathcal{U}$. AMQs support queries and inserts, and some AMQs also support deletes. A positive query, i.e., for an $x \in \mathcal{S}$, is guaranteed to return "present." A negative query, i.e., for $x \notin \mathcal{S}$, returns "absent" with probability at least $1-\varepsilon$, where ε is a tunable *false-positive probability*. If the query returns "present," but $x \notin \mathcal{S}$, then x is a *false positive* of the AMQ.

Because Bloom filters and other AMQs have a nonzero false-positive probability, they are able to consume much less space than regular dictionary data structures. Specifically, a Bloom filter can maintain a set $\mathcal{S} \subseteq \mathcal{U}$, where $|\mathcal{S}| = n$ and $|\mathcal{U}| = u$, with a false-positive probability ε using $\Theta(n \log(1/\varepsilon))$ bits [38], which is asymptotically optimal. In contrast, an error-free representation of $\mathcal{S}$ takes $\Omega(n \log u)$ bits.

One of the main uses of AMQs is to speed up dictionaries [29,41,46,54,55,58,59,61,103, 108,111,132,143,144,147]. Often, there is not enough local storage (e.g., RAM) to store the dictionary's internal state, $\mathbf{D}$. Thus, $\mathbf{D}$ must be maintained remotely (e.g., on-disk or across

[19] We refer to a data structure as a "Bloom filter" only if has the same basic structure as the standard Bloom filter. We use "AMQ" to refer to general data structures that give similar approximate-membership-query guarantees, possibly using different techniques.

a network), and accesses to $\mathbf{D}$ are expensive. By maintaining a local AMQ for the set $\mathcal{S}$ of keys occurring in $\mathbf{D}$, the dictionary can avoid accessing $\mathbf{D}$ on most negative queries: if the AMQ says that a key is not in $\mathcal{S}$, then no query to $\mathbf{D}$ is necessary.

Thus, the primary performance metric of an AMQ is how well it enables a dictionary to avoid these expensive accesses to $\mathbf{D}$. The fewer false positives an AMQ returns on a sequence of queries, the more effective it is as a filter.

AMQ guarantees. Most AMQs offer weak guarantees on the number of false positives incurred on a sequence of queries. The false-positive probability of ε holds only for a single query. It does not extend to multiple queries, because queries can be correlated. An adversary can drive an AMQ's false-positive rate towards 1 by simply repeating false-positives.

Even when the adversary is *oblivious*, i.e., selects n queries independent of past queries, most AMQs have weak guarantees. With probability ε, a random query is a false positive, and repeating it n times results in a false-positive rate of 1. Thus, even against an oblivious adversary, most AMQs have $O(\varepsilon n)$ false positives in expectation but not with high probability. This distinction has implications: Mitzenmacher et al. [109] show that on network traces, existing AMQs leave performance on the table by not adapting to past false positives.

Adaptive AMQs. We define an *adaptive AMQ* to be an AMQ that returns "present" with probability at most ε for every negative query, *regardless of answers to previous queries.* For a dictionary using an adaptive AMQ, *any* sequence of n negative queries will result in $O(\varepsilon n)$ false positives, with high probability. This gives a strong bound on the number of (expensive) negative accesses that the dictionary will need to make to D. This is true even if the queries are selected by an adaptive adversary.

Several attempts have been made to move towards adaptivity (and beyond oblivious adversaries). Naor and Yogev [113] considered an adaptive adversary that tries to increase the false-positive rate by discovering collisions in the AMQ's hash functions, but they explicitly forbade the adversary from repeating queries. Chazelle et al. [42] introduced bloomier filters, which can be updated to specify a white list, which are elements in $\mathcal{U} - \mathcal{S}$ on which the AMQ may not answer present. However, bloomier filters are space efficient only when the white list is specified in advance, which makes them unsuitable for adaptivity. Mitzenmacher et al. [109] proposed an elegant variant of the cuckoo filter that stores part of the AMQ locally and part of it remotely in order to achieve adaptivity. They empirically show that their data structure helps maintain a low false-positive rate against temporally-correlated queries.

However, no existing AMQ is provably adaptive.

Feedback, local AMQs, and remote representations. When an AMQ is used to speed up a dictionary, the dictionary always detects which queries are false positives of the AMQ. Thus, the dictionary can provide this feedback to the AMQ. This feedback is free because it does not require any additional accesses to $\mathbf{D}$ beyond what was used to answer the query.

In this chapter we show that, even with this feedback, it is impossible to construct an adaptive AMQ that uses less than $\Omega(\min(n \log\log u, n \log n))$ bits of space; see Theorem 9.3. That is, even if an AMQ is told which are the true and false positives, a very large amount of space is necessary to achieve adaptivity.

This lower bound on the space required by an adaptive AMQ would appear to kill the whole idea of adaptive AMQs, since one of the key ideas of an AMQ is to be small enough to fit in local storage. Remarkably, efficient adaptivity is still achievable.

The way around this impasse is to partition an AMQ's state into a small local state $\mathbf{L}$ and a larger remote state $\mathbf{R}$. The AMQ can still have good performance provided that it accesses the remote state infrequently.

By using this idea, we show how to make an adaptive AMQ that consumes no more local space than the best nonadaptive AMQ (and much less than a Bloom filter). We call this data structure a *broom filter* (because it cleans up its previous mistakes). The broom filter accesses $\mathbf{R}$ only when the dictionary accesses $\mathbf{D}$. Thus, the AMQ's accesses to $\mathbf{R}$ are free; they do not asymptotically affect the number of remote accesses required to perform a query.

Partitioning appears to be essential to creating a space-efficient adaptive AMQ. For example, the adaptive cuckoo filter [109] also partitions its state into local and remote components, but lacks the strong theoretical performance guarantees of the broom filter.

It turns out that the local component of the broom filter is itself a nonadaptive AMQ plus $O(n)$ bits for adaptivity. The purpose of $\mathbf{R}$ is merely to provide a little more information to help $\mathbf{L}$ adapt.

Thus, we have a dual view of adaptivity that helps us interpret the upper and lower bounds. The local representation $\mathbf{L}$ is an AMQ in its own right. The remote representation $\mathbf{R}$ is an "oracle" that gives extra feedback to $\mathbf{L}$ whenever there is a false positive. Because $\mathbf{R}$ is simply an oracle, all the heavy lifting is in the design of $\mathbf{L}$. In the broom filter, $\mathbf{R}$ enables $\mathbf{L}$ to identify an element $y \in \mathcal{S}$ that triggered the false positive. As described above, accessing $\mathbf{R}$ is free because it is paid for by accesses to $\mathbf{D}$.

Putting these results together, we pinpoint how much information is needed for an adaptive AMQ to update its local information. The lower bound shows that simply learning if the query is a false positive is not sufficient. But if this local information is augmented with asymptotically free dictionary lookups, then adaptivity is achievable.

A note on optimality. The broom filter dominates existing AMQs in all regards. Its local state by itself is an optimal conventional AMQ: it uses optimal space up to lower-order terms, and supports queries and updates in worst-case constant time, with high probability. Thus the remote state is only for adaptivity. For comparison, a Bloom filter has a lookup time of $O(\log \frac{1}{\varepsilon})$, the space is suboptimal, and the filter does not support deletes. More recent AMQs [22, 60, 118, 119] also fail to match the broom filter on one or more of these criteria, even leaving aside adaptivity. Thus, we show that adaptivity has no cost.

9.2 Related Work

Bloom filters [24] are a ubiquitous data structure for approximate membership queries and have inspired the study of many other AMQs; see surveys [29, 132]. Here, we only mention a few well-known AMQs in and several results closely related to adaptivity.

Bloom Filters [24]. The standard Bloom filter, representing a set $\mathcal{S} \subseteq \mathcal{U}$, is composed of m bits, and uses k independent hash functions $h_1, h_2, \ldots, h_k$, where $h_i : \mathcal{U} \to \{1, \ldots, m\}$. To insert $x \in \mathcal{S}$, the bits $h_i(x)$ are set to 1 for $1 \leq 1 \leq k$. A query for x checks if the bits $h_i(x)$ are set to 1 for $1 \leq i \leq k$—if they are, it returns "present", and else it returns "absent". A Bloom filter does not support deletes. For a false-positive probability of $\varepsilon > 0$, it uses $m = (\log e) n \log_2(1/\varepsilon)$ bits and has an expected lookup time of $O(\log(1/\varepsilon))$.

Single-Hash-Function AMQs. Carter et al. [38] introduced the idea of an AMQ that uses a single hash function that maps each element in the universe to one of $\lceil n/\varepsilon \rceil$ elements, and then uses a compressed exact-membership tester. Pagh et al. [118] show how to build the first near-optimal single hash-function AMQ by applying a universal hash function $h : U \to \{0, 1, \ldots, \lceil n/\varepsilon \rceil\}$ to $\mathcal{S}$, storing the resulting values using $(1 + o(1))n \log \frac{1}{\varepsilon} + O(n)$ bits. Their construction achieves $O(1)$ amortized insert/delete bounds and has been deamortized, in the case of inserts, by the backyard hashing approach of Arbitman et al. [5].

The quotient filter (QF) [22,119] is a practical variant of Pagh et al.'s single-hash function AMQ. The QF serves as the basis for our adaptive AMQ and is described in Chapter 9.5.

The cuckoo filter [60] is also a practical variant of Pagh et al.'s single-hash function AMQ. However, the implementation is based on cuckoo hashing rather than linear probing. This difference leads to performance advantages for some parameter settings [60, 119].

It would be interesting to develop a provably adaptive variant of the cuckoo filter. Its structure makes it difficult to use our approach of maintaining adaptivity bits. On the other hand, analyzing the Markov chain resulting from the heuristic approach to cuckoo-filter adaptivity in [109] has its own challenges.

Bloomier filters. Chazelle et al. [42]'s Bloomier filters generalize Bloom filters to avoid a predetermined list of undesirable false positives. Given a set S of size n and a whitelist W of size w, a Bloomier filter stores a function f that returns "present" if the query is in the S, "absent" if the query is not in $S \cup W$, and "is a false positive" if the query is in W. Bloomier filters use $O((n + w) \log 1/\varepsilon)$ bits. The set $S \cup W$ cannot be updated without blowing up the space used and thus their data structure is limited to a static whitelist.

Adversarial-resilient Bloom filters. Naor and Yogev [113] study Bloom filters in the context of a repeat-free adaptive adversary, which queries elements until it can find a never-before-queried element that has a false-positive probability greater than ε. (They do not consider the false-positive probability of repeated queries.) They show how to protect an AMQ from a repeat-free adaptive adversaries using cryptographically secure hash functions so that new queries are indistinguishable from uniformly selected queries [113]. Hiding data-structure hash functions has been studied beyond AMQs (e.g., see [25, 71, 87, 107, 112]).

Adaptive cuckoo filter. Recently, Mitzenmacher et al. [109] introduced the *adaptive cuckoo filter* which removes false positives after they have been queried. They do so by consulting the remote representation of the set stored in a hash table. Their data structure takes more space than a regular cuckoo filter but their experiments show that it performs better on real packet traces and when queries have a temporal correlation.

Other AMQs and false-positive optimizations. Retouched Bloom filters [56] and generalized Bloom filter [102] reduce the number of false positives by introducing false negatives. Variants of Bloom filter that choose the number of hash functions assigned to a key based on its frequency in some predetermined query distribution have also been studied [30, 146].

Data Structures and Adaptive Adversaries. Nonadaptive data structures lead to significant problems in security-critical applications, such as algorithmic complexity attacks against hash tables [33, 51], peacock and cuckoo hash tables [18], (unbalanced) binary search trees [51], and quicksort [94]. In these attacks, the adversary adaptively constructs a sequence of inputs and queries that causes the data structure to exhibit its worst-case performance.

9.3 Preliminaries

As in the standard notion of an AMQ, our AMQ is a dynamic set data structure and supports insertions, deletions, and membership queries, and may return false positives on queries.

However, we generalize the standard notion of an AMQ in two ways. First, our AMQ can update its internal representation based on feedback from the system. We also divide the AMQ's internal state into local (i.e., in-memory) and remote (i.e., on-disk) components. The AMQ answers queries based solely on the local state, but can use the remote state to help it correct errors based on feedback.

We begin by defining the operations that our AMQ supports. These operations specify precisely when it can access its local vs. remote state, when it gets to update its state, and how it receives feedback.

Definition 9.1. *An approximate membership query (AMQ) data structure consists of the following deterministic functions. Here ρ denotes the AMQ's private infinite random string, $\mathbf{L}$ and $\mathbf{R}$ denote its private local and remote state respectively, n denotes the maximum set size and ε denotes the false-positive probability.*

- $\textsc{Init}(n, \varepsilon, \rho) \longrightarrow (\mathbf{L}, \mathbf{R})$ *takes n, ε and ρ and returns an initial state $(\mathbf{L}, \mathbf{R})$.*
- $\textsc{Lookup}(\mathbf{L}, x, \rho) \longrightarrow (\mathbf{L}', b)$ *takes a local state $\mathbf{L}$, element $x \in \mathcal{U}$ and ρ, and returns a new local state $\mathbf{L}'$ and bit b indicating whether x is "present" or "absent". Note that* Lookup *does not get access to $\mathbf{R}$.*
- $\textsc{Insert}((\mathbf{L}, \mathbf{R}), x, \rho) \longrightarrow (\mathbf{L}', \mathbf{R}')$ *takes state $(\mathbf{L}, \mathbf{R})$, element $x \in \mathcal{U}$ to be inserted and ρ and returns the new state $(\mathbf{L}', \mathbf{R}')$.* Delete *is defined analogously.*
- $\textsc{Adapt}((\mathbf{L}, \mathbf{R}), x, \rho) \longrightarrow (\mathbf{L}', \mathbf{R}')$ *takes state $(\mathbf{L}, \mathbf{R})$, false-positive element[20] x and ρ, and returns the new state $(\mathbf{L}', \mathbf{R}')$.*

Note that this definition generalizes existing AMQs [22, 24, 42, 60, 118, 119]. For example, we can view Bloom filters, cuckoo filters, etc., as AMQs that never make any modifications or accesses to $\mathbf{R}$, and that have an Adapt function that returns $(\mathbf{L}, \mathbf{R})$ unmodified. An AMQ is *local* if it never accesses $\mathbf{R}$. An AMQ is *oblivious* if its Adapt is the identity function.

Histories and adaptivity. A *history* is an invocation of Init followed by a sequence of Inserts, Deletes, Adapts, and Lookups. Histories serve two purposes. First, a history $\mathcal{H}$ encodes the set $\mathcal{S}_{\mathcal{H}}$ of true positives, i.e., the elements that have been inserted into the AMQ (but not deleted). Second, the calls to Adapt in the history encode the information that an adversary can learn from queries—which queries resulted in false positives. Thus, in an adaptive AMQ when we analyze the probability that each query is a false positive independent of the results of past queries, we mean independent of the AMQ's history.

For notational convenience, for a given n and ε, we view a history $\mathcal{H}$ as a function $\mathcal{H}(\rho)$ that returns the final state of the AMQ that results from calling $\textsc{Init}(n, \varepsilon, \rho)$ and then

[20] We precisely define false-positive below.

performing the sequence of INSERTs, DELETEs, LOOKUPs, and ADAPTs specified in $\mathcal{H}$. Let $\mathbf{L}_{\mathcal{H}(\rho)}$ and $\mathbf{R}_{\mathcal{H}(\rho)}$ be the local and remote components, respectively, of AMQ state $\mathcal{H}(\rho)$.

The AMQs we consider only make guarantees on their correctness and false-positive rate for histories that use the AMQ properly. For example, if a history builds the AMQ with maximum size n, then the history should never have more than n items inserted at one time. A history should also never delete an item that is not in the set. Throughout the rest of this chapter, we consider only histories that obey these rules.

A history should call ADAPT immediately after each LOOKUP that returns a false positive, and at no other time. However, whether a call to LOOKUP yields a false positive depends on ρ. Thus a history $\mathcal{H}$ might obey this rule for some ρs but not for others. A history $\mathcal{H}$ and a string ρ are *consistent* if, during the execution of $\mathcal{H}(\rho)$, each call to ADAPT immediately follows a call to LOOKUP that returned a false positive.

We can now define false positives, negatives, and adaptivity.

Given consistent history $\mathcal{H}$ and random string ρ, x is a *false positive* of $\mathbf{L}_{\mathcal{H}(\rho)}$ with respect to $\mathcal{H}$ if $x \notin S_{\mathcal{H}}$ but LOOKUP$(\mathbf{L}_{\mathcal{H}(\rho)}, x, \rho)$ returns "present". We define *false negative* similarly. We consider only AMQs that have no false negatives whenever $\mathcal{H}$ and ρ are consistent.

We define the standard *static (or single-query) false-positive rate* of an AMQ as follows. An AMQ supports static false-positive rate ε if, for all lookup-free histories $\mathcal{H}$ (i.e., histories that do not contain any calls to LOOKUP) that call INIT with false-positive-rate parameter ε and all elements $x \in \mathcal{U} \setminus \mathcal{S}_{\mathcal{H}}$,

$$\Pr_{\rho}[\text{LOOKUP}(\mathbf{L}_{\mathcal{H}(\rho)}, x, \rho) \text{ returns "present"}] \leq \varepsilon.$$

We now extend the notion of false-positive rate to handle adaptive queries.

Definition 9.2. *An AMQ supports* sustained false-positive rate ε *if, for every history* $\mathcal{H}$ *that calls* INIT *with false-positive-rate parameter* ε *and all elements* $x \in \mathcal{U} \setminus \mathcal{S}_{\mathcal{H}}$,

$$\Pr_{\rho}[\text{LOOKUP}(\mathbf{L}_{\mathcal{H}(\rho)}, x) \textit{ returns "present"} | \rho \textit{ is consistent with } \mathcal{H}] \leq \varepsilon.$$

The probability in the definition of sustained false-positive rate is taken only over ρ that are consistent with the false positives specified in $\mathcal{H}$. In other words, an AMQ has a sustained false-positive rate ε if, no matter the results of past queries, the next query always has a false-positive probability of at most ε.

An AMQ is *adaptive* if it guarantees a sustained false-positive rate of ε for some $\varepsilon < 1$.

Cost model. We measure AMQ performance in terms of RAM model operations to examine and update $\mathbf{L}$ and in terms of updates and queries to $\mathbf{R}$. We measure these three quantities (RAM operations, remote updates, and remote queries) separately.

We follow the standard practice of analyzing AMQ performance in terms of the AMQ's maximum capacity, n.[21] We assume the RAM model with a word size $w = \Omega(\log u)$. For simplicity of presentation, we assume that $u = \text{poly}(n)$ but our results generalize.

Hash functions. In this work, we assume that the adversary cannot find a never-queried-before element that is a false positive of the AMQ with probability greater than ε based on the result of previous queries. Ideal hash functions have this property for arbitrary adversaries. If the adversary is polynomially bounded, one-way functions are sufficient to prevent them from generating new false positives [113].

Notation. An event occurs *with high probability* (w.h.p.) if it occurs with probability at least $1 - 1/n^c$ for a tunable constant c.

9.4 A Lower Bound on Local AMQs

In this section, we show that a local AMQ cannot maintain adaptivity along with space efficiency. More formally, if the ADAPT function depends only on $\mathbf{L}$, x and ρ (and not on a remote representation $\mathbf{R}$), then the AMQ must use $\Omega(\min\{n \log n, n \log \log u\})$ bits, much larger than our desired space when $\mathcal{U}$ is large. Thus, an AMQ cannot be adaptive if it only learns whether each query x is a false positive or a true positive.

In particular, we prove the following lower bound on the space required by a local AMQ to maintain adaptivity.

Theorem 9.3. *Any local adaptive AMQ storing a set of size n from a universe of size $u > n^4$ requires $m = \Omega(\min\{n \log n, n \log \log u\})$ bits of space with high probability to maintain any constant sustained false-positive rate $\varepsilon < 1$.*

Interestingly, a similar lower bound was studied in the context of Bloomier filters [42]. The Bloomier filter is an AMQ designed to solve the problem of storing n items for which it must return "present", along with a whitelist of $\Theta(n)$ items for which it must return "absent". Other queries must have a static false-positive rate of ε. Chazelle et al. [42] give a lower bound on any data structure that updates this whitelist dynamically, showing that such a data structure must use $\Omega(n \log \log(u/n))$ space. Their lower bound implies that if the

[21] This originates from the Bloom filter literature, since Bloom filters cannot be resized.

adversary gives an AMQ a dynamic white list of false positives that it needs to *permanently* fix, then it must use too much space. We generalize this bound to all adaptive local AMQs.

9.4.1 Notation and Adversary Model

We begin by formalizing our notation and defining the adversary used in the lower bound.

We fix n and ε and drop them from most notation. We use $\textsc{Build}(\mathcal{S}, \rho)$ to denote the state that results from calling $\textsc{Init}(n, \varepsilon, \rho)$ followed by $\textsc{Insert}(x, \rho)$ for each $x \in \mathcal{S}$ (in lexicographic order).

The adversary does not have access to the AMQ's internal randomness ρ, or any internal state $\mathbf{L}$ of the AMQ. The adversary can only issue a query x to the AMQ and only learns the AMQ's output—"present" or "absent"—to query x.

The adversary's goal is to adaptively generate a sequence of $O(n)$ queries and force the AMQ to either use too much space or to fail to satisfy a sustained false-positive rate of ε.

Let $\varepsilon_0 = \max\{1/n^{1/4}, (\log^2 \log u)/\log u\}$. Thus, our lower bound can be rephrased as $m = \Omega(n \log 1/\varepsilon_0)$. Note that $\varepsilon_0 \leq \varepsilon$; otherwise the classic AMQ lower bound of $m \geq n \log 1/\varepsilon$ [38, 104] is sufficient to prove Theorem 9.3. We can think of ε_0 as a maximum bound on how often the AMQ encounters elements that are hard to fix.

Attack description. First, the adversary chooses a set $\mathcal{S}$ of size n uniformly at random from $\mathcal{U}$. After choosing the initial set, the adversary's attack proceeds in rounds. The adversary selects a set Q of size n uniformly at random from $\mathcal{U} - \mathcal{S}$. Starting from Q, in each round, it queries the elements that were false positives in the previous round. To simplify analysis, we assume that the adversary orders its queries in lexicographic order. Let FP_i be the set of queries that are false positives in round $i \geq 1$. The attack is described below:

1. In the first round, the adversary queries each element of Q.
2. In round $i > 1$, if $|\mathrm{FP}_{i-1}| > 0$, the adversary queries each element in FP_{i-1}; otherwise the attack ends.

Classifying false positives. The crux of our proof is that some false positives are difficult to "fix"—in particular, these are the queries where an AMQ is unable to distinguish whether or not $x \in \mathcal{S}$ by looking at its local state $\mathbf{L}$.[22] We call $y \in \mathcal{U} \setminus \mathcal{S}$ an *absolute false positive* of a state $\mathbf{L}$ and randomness ρ if there exists a set $\mathcal{S}'$ of size n and a sequence of queries $(x_1, \ldots, x_t)$ such that $y \in \mathcal{S}'$ and $\mathbf{L}$ is the state of the AMQ when queries $x_1, \ldots, x_t$ are performed (some of which may result in $\textsc{Adapt}$s if they are false positives) on $\textsc{Build}(\mathcal{S}', \rho)$.

[22] This is as opposed to easy-to-fix queries where, e.g., the AMQ answers "present" randomly just to confuse an adversary. For all previous AMQs we are aware of, all false positives are absolute false positives.

We use $\text{AFP}(\mathbf{L}, \mathcal{S}, \rho)$ to denote the set of absolute false positives of state $\mathbf{L}$, randomness ρ, and true-positive set $\mathcal{S}$. We call $(\mathcal{S}', (x_1, \ldots, x_t))$ a *witness* to y. We call $y \in \mathcal{U} \setminus \mathcal{S}$ an *original absolute false positive* if and only if $y \in \text{AFP}(\textsc{Build}(\mathcal{S}, \rho), \mathcal{S}, \rho)$. We call the set of original absolute false positives $\text{OFP}(\mathcal{S}, \rho) = \text{AFP}(\textsc{Build}(\mathcal{S}, \rho), \mathcal{S}, \rho)$.

As the AMQ handles queries, it will need to fix some previous false positives. To fix a false positive, the AMQ must change its local state so that it can safely answer "absent" to it. For a state $\mathbf{L}$, we define the set of elements that are no longer false positives by the set $\text{FIX}(\mathbf{L}, \mathcal{S}, \rho) = \text{OFP}(\mathcal{S}, \rho) \setminus \text{AFP}(\mathbf{L}, \mathcal{S}, \rho)$. Note that all fixed false positives are original absolute false positives.

An AMQ without false negatives cannot fix an original absolute false positive y unless it learns that $y \notin \mathcal{S}$. This idea is formalized in the next two observations.

Observation 9.4. *For any randomness ρ, set $\mathcal{S}$, and state $\mathbf{L}$ of the AMQ, if a query $x \in \text{AFP}(\mathbf{L}, \mathcal{S}, \rho)$, then* Lookup $(\mathbf{L}, x, \rho)$ *must return "present".*

Observation 9.5. *Let $\mathbf{L}_1$ be a state of the AMQ before a query x and $\mathbf{L}_2$ be the updated state after x (i.e., after invoking* Lookup *and possibly* Adapt*). Let y be an absolute false positive of $\mathbf{L}_1$ with witness S_y. Then if y is not an absolute false positive of $\mathbf{L}_2$, then $x \in S_y$.*

9.4.2 Analysis

We start with an informal overview of the lower-bound proof.

The following observation shows that there is a 1-to-1 mapping from "fixed" sets of original absolute false positives to AMQ states. Thus, we can lower bound the number of AMQ states (and hence the space needed to represent a state) by lower-bounding the number of sets of original absolute false positives the adversary can force the AMQ to fix.

Observation 9.6. *Given randomness ρ and set $\mathcal{S}$ of size n, consider two fixed false positive sets $\text{FIX}(\mathbf{L}_1, \mathcal{S}, \rho)$ and $\text{FIX}(\mathbf{L}_2, \mathcal{S}, \rho)$. Then if $\text{FIX}(\mathbf{L}_1, \mathcal{S}, \rho) \neq \text{FIX}(\mathbf{L}_2, \mathcal{S}, \rho)$, then $\mathbf{L}_1 \neq \mathbf{L}_2$.*

We begin with a known result showing that if the AMQ cannot use too much space, it must start with a large number of original absolute false positives (Claim 9.7) for almost all $\mathcal{S}$. Given that the AMQ starts with a large number of original absolute false positives, a decent fraction of the adversary's randomly chosen queries Q are original absolute false positives of the AMQ (Lemma 9.8).

Next, we show that through its adaptive queries, the adversary is able to force the AMQ to fix almost all of these discovered original absolute false positives, for most sets Q (Lemma 9.9 and Lemma 9.10).

The crux of the proof relies on Lemma 9.11 which says that the AMQ cannot fix too many *extra* original absolute false positives during the attack—thus, it needs a large number of distinct "fixed" sets to cover all the different sets of original absolute false positives that the adversary forces the AMQ to fix. This is where we use that the AMQ is local, and only receives a limited amount of feedback on each false positive—it cannot fix more false positives without risking some false negatives.

Finally, the proof of Theorem 9.3 pulls these pieces together to derive the lower bound on the space required by a local adaptive AMQ.

Discovering original absolute false positives through random queries. First, we note that for a given randomness ρ, the AMQ must start with a large number of original absolute false positives for most sets $\mathcal{S}$.

While for some special sets $\mathcal{S}$ given in advance, an AMQ may be able to store $\mathcal{S}$ very accurately (with few false positives), this is not true for a random set $\mathcal{S}$ chosen by the adversary. In particular, we note the following claim, also presented by Naor et al. [113].

Claim 9.7 ([113, Claim 5.3])**.** *Given any randomness ρ of AMQ using space $m \leq n \log 1/\varepsilon_0 + 4n$ bits, for any set $\mathcal{S}$ of size n chosen uniformly at random from $\mathcal{U}$, we have:* $\Pr_{\mathcal{S}}\left[|\mathrm{OFP}(\mathcal{S}, \rho)| \leq u\varepsilon_0\right] \leq 2^{-n}$.

For the remainder of this section, we fix a set $\mathcal{S}^* \subseteq \mathcal{U}$ of size n such that $|\mathrm{OFP}(\mathcal{S}^*, \rho)| > u\varepsilon_0$.[23] Let $\mathcal{Q}$ be the set of all possible query sets Q the adversary can choose, that is, $\mathcal{Q} = \{Q \subseteq \mathcal{U} \setminus \mathcal{S}^* \mid |Q| = n\}$. (We do not include S^* in the notation of $\mathcal{Q}$ for simplicity.)

Lemma 9.8. *For a fixed randomness ρ of an AMQ of size $m \leq n \log 1/\varepsilon_0 + 4n$ and fixed set $\mathcal{S}^*$ such that $|\mathrm{OFP}(\mathcal{S}^*, \rho)| > u\varepsilon_0$, we have* $\Pr_{Q \in \mathcal{Q}}\left[|Q \cap \mathrm{OFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0)\right] \geq 1 - 1/\mathrm{poly}(n)$.

Proof. The probability that a given query $x \in Q$ is an original absolute false positive is $\Omega(\varepsilon_0)$ and $\mathbf{E}_{Q \in \mathcal{Q}}[|Q \cap \mathrm{OFP}(\mathcal{S}^*, \rho)|] = \sum_{x \in Q} \Pr_{x \in \mathcal{U} \setminus \mathcal{S}}[x \in \mathrm{OFP}(\mathcal{S}^*, \rho)] = \Omega(n\varepsilon_0)$.

Since $\varepsilon_0 = \Omega(n^{1/4})$, using Chernoff bounds, with high probability over the choice of Q, we have that $|Q \cap \mathrm{OFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0)$. □

Forcing the AMQ to fix large number of original absolute false positives to maintain adaptivity. From the definition of sustained false-positive rate, the AMQ must fix an ε fraction of false positives in expectation in each round. If the expected number of false positives that the AMQ has to fix in each round is high, classic concentration bounds

[23] With probability $1/2^n$, the adversary gets unlucky and chooses a set $\mathcal{S}^*$ that does not satisfy this property, in which case it fails. This is okay, because we only need to show existence of a set $\mathcal{S}^*$ where the AMQ uses large space w.h.p. over ρ.

imply that the AMQ must fix close to this expected number with high probability in each round. This implies that there must be a round where the AMQ fixes a large number of original absolute false positives. The next lemma formalizes this intuition.

For a given Q, let $\text{FORCED}(Q, \mathcal{S}^*, \rho)$ be the largest number of query elements (out of Q) that the AMQ has fixed simultaneously in any state. In particular,[24] $\text{FORCED}(Q, \mathcal{S}^*, \rho) = \text{argmax}_{1 \le t' \le t} \ \text{FIX}(\mathbf{L}_{t'}, \mathcal{S}^*, \rho)$, where $\mathbf{L}_i$ is the state of the AMQ after each query $x_i \in Q$ during an attack consisting of t total queries.

Lemma 9.9. *Consider an AMQ of size $m \le n \log 1/\varepsilon_0 + 4n$. With high probability over ρ, for any set Q satisfying $Q \cap \text{OFP}(\mathcal{S}^*, \rho) = \Omega(n\varepsilon_0)$, there exists a round $T(Q, \rho)$ and a state $\mathbf{L}_{T(Q,\rho)}$ at the beginning of round $T(Q, \rho)$ such that $|\text{FIX}(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0 / \log_\varepsilon \varepsilon_0)$. In other words,*

$$\Pr_\rho \left[|\text{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0 / \log_\varepsilon \varepsilon_0) \;\middle|\; |Q \cap \text{OFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0) \right] \ge 1 - 1/\text{poly}(n).$$

Round $T(Q, \rho)$ is reached in at most $O(n)$ total queries.

Proof. We fix Q and set $T = T(Q, \rho)$.

Recall that FP_T denotes the set of queries that are false positives in round T, and let $T_f = \log_\varepsilon \varepsilon_0$. Since the AMQ has a sustained false-positive rate of ε, we have $|\text{FP}_1| = O(n\varepsilon)$. As $\varepsilon \ge \varepsilon_0 \ge 1/n^{1/4}$, by Chernoff bounds, we have $|\text{FP}_{T+1}| \le \varepsilon |\text{FP}_T|(1 + 1/\log n)$ with high probability for all $1 \le T \le T_f$.

Suppose there does not exist a round $T < T_f$ such that the lemma holds, i.e., in each round $T < T_f$, $|\text{FIX}(\mathbf{L}_T, \mathcal{S}^*, \rho)| \le n\varepsilon_0 / 2 \log_\varepsilon \varepsilon_0$, where $\mathbf{L}_T$ is the state of the AMQ at the beginning of round T. In round T_f, the AMQ is asked $|\text{FP}_{T_f - 1}| \le (\varepsilon(1 + 1/\log n))^{T_f - 2} n = O(n\varepsilon_0)$ queries. From our assumption, $|\text{OFP}(\mathcal{S}^*, \rho) \cap \text{FP}_{T_f - 1}| \ge n\varepsilon_0 (1 - 1/2 \log_\varepsilon \varepsilon_0)^{T_f - 1} = \Omega(n\varepsilon_0)$.

For a sustained false-positive rate of ε, it must hold that $|\text{FP}_{T_f}| = O(n\varepsilon\varepsilon_0)$ with high probability. Thus, in round T_f the AMQ must answer "absent" to $\Omega(n(1 - \varepsilon)\varepsilon_0) = \Omega(n\varepsilon_0)$ original absolute false positives from the set $|(\text{OFP}(\mathcal{S}^*, \rho) \cap \text{FP}_{T_f - 1}) \setminus \text{FP}_{T_f}|$. Let $\mathcal{A}_{T_f}$ be the set of original absolute false positives queries that the AMQ says "absent" to in round T_f.

Let $\mathbf{L}_{T_f, x}$ denote the state of the AMQ in round T_f just before query x is made. Then by Observation 9.4, $x \in \text{FIX}(\mathbf{L}_{T_f, x}, \mathcal{S}^*, \rho)$ for any $x \in \mathcal{A}_{T_f}$. We now show that all $x \in \mathcal{A}_{T_f}$ must simultaneously be in the set of fixed false positives of the state $\mathbf{L}_{T_f}$ at the beginning of round T_f. Note that $x \in \text{OFP}(\mathcal{S}^*, \rho) \cap \text{FP}_{T_{f-1}}$ and all queries between query x in round T_{f-1} and query x in round T_f are distinct from x and were chosen independently from x in round

[24] The argmax is taken over the size of the sets.

1. As there can be at most n queries in between query x in consecutive rounds, using Observation 9.5, the probability that there exists a state $\mathbf{L}_i$ between $\mathbf{L}_{T_{f-1},q}$ and $\mathbf{L}_{T_f,q}$ such that $x \notin \mathrm{FIX}(\mathbf{L}_i, \mathcal{S}^*, \rho)$ is at most $n^2/u < 1/n^2$. Thus, with high probability, $x \in \mathrm{FIX}(\mathbf{L}_{T_f}, \mathcal{S}^*, \rho)$ for any given $x \in \mathcal{A}_{T_f}$. That is, $\mathcal{A}_{T_f} \subseteq \mathrm{FIX}(\mathbf{L}_{T_f}, \mathcal{S}^*, \rho)$, and thus, $|\mathrm{FIX}(\mathbf{L}_{T_f}, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0)$.

Furthermore, round T is reached in $n + \sum_{i=1}^{T} \mathrm{FP}_i \le n(1-\varepsilon^T)/(1-\varepsilon) = O(n)$ queries. □

For simplicity, let $\varepsilon_0' = \varepsilon_0 / \log_\varepsilon \varepsilon_0$. The next lemma shows that (for most ρ), most query sets Q satisfy Lemma 9.9 with high probability.

Lemma 9.10. *With high probability over ρ, for all but a $1/\mathrm{poly}(n)$ fraction of Q there exists a round $T(Q,\rho)$ such that an AMQ of size $m \le n \log 1/\varepsilon_0 + 4n$ is forced to fix $\Omega(n\varepsilon_0')$ original absolute false positive queries. In other words, if $|\mathrm{OFP}(\mathcal{S}^*, \rho)| \ge u\varepsilon_0$,*

$$\Pr_\rho \left[\Pr_{Q \in \mathcal{Q}} \left[|\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0') \right] \ge 1 - \frac{1}{\mathrm{poly}(n)} \right] \ge 1 - 1/\mathrm{poly}(n).$$

Proof.

$$\begin{aligned}
&\Pr_\rho \left[\Pr_{Q \in \mathcal{Q}} \left[|\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega\left(n\varepsilon_0'\right) \right] \ge 1 - \frac{1}{\mathrm{poly}(n)} \right] \\
&\ge \Pr_\rho \left[\Pr_{Q \in \mathcal{Q}} \left[|\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega\left(n\varepsilon_0'\right) \;\middle|\; |Q \cap \mathrm{AFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0) \right] \cdot \right. \\
&\qquad \left. \cdot \Pr_{Q \in \mathcal{Q}} \left[|Q \cap \mathrm{AFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0) \right] \ge 1 - \frac{1}{\mathrm{poly}(n)} \right] \\
&\ge \Pr_\rho \left[\Pr_{Q \in \mathcal{Q}} \left[|\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega\left(n\varepsilon_0'\right) \;\middle|\; |Q \cap \mathrm{AFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0) \right] \ge 1 - \frac{1}{\mathrm{poly}(n)} \right] \\
&\ge \left(1 - \frac{1}{\mathrm{poly}(n)} \right)
\end{aligned}$$

The second step is from Lemma 9.8, and the final step is from Lemma 9.9. □

Local AMQs cannot fix too many original absolute false positives. Next, we show that for a randomly-chosen y, knowing $y \notin \mathcal{S}^*$ does not give any information to the AMQ about which set exact $\mathcal{S}^*$ it is storing.

In particular, a local AMQ may try to rule out false positives that may be correlated to a query y. For example, an AMQ may (without asymptotic loss of space) partition the universe into pairs of elements such that if it learns one item in a pair is a false positive, it is guaranteed that the other is as well. With such a strategy, the AMQ can succinctly fix two false positives per query instead of one. Could a more intricate strategy allow the AMQ to fix more false positives—even enough to be an obstacle to our lower bound?

We rule out this possibility in the following lemma. On a random query, the AMQ is very unlikely to rule out a given false positive. We use this to make a with-high-probability statement about the number of fixed elements using Markov's inequality in our final proof.

Lemma 9.11. *Given randomness ρ and set $\mathcal{S}^*$, consider any sequence of queries $(x_1, \ldots, x_t)$ taken from a uniformly-sampled random set $Q \subset \mathcal{U}$ of size n, and let $\mathbf{L}'$ be the state of the AMQ after these queries are executed. For an element $y \in \mathcal{U}$, the probability over Q that y is a fixed false positive after these queries is n^2/u. That is,* $\Pr_{Q \in \mathcal{Q}}[y \in \mathrm{FIX}(\mathbf{L}', \mathcal{S}^*, \rho)] = n^2/u$.

Proof. If $y \in \mathrm{FIX}(\mathbf{L}', \mathcal{S}^*)$, then by Observation 9.5, for any witness set S' of y, $S' \cap Q$ must be nonempty. Fix a single witness set S'. For a given $s' \in S'$ and $x' \in Q$, $\Pr(s' = x') = 1/u$. Taking the union bound over all n^2 such pairs (s', x') we achieve the lemma. □

Final lower bound. We are now ready to prove Theorem 9.3.

Proof of Theorem 9.3. Assume by contradiction that $m \leq n \log 1/\varepsilon_0 + 4n$ (recall $\varepsilon_0 = \max\{1/n^{1/4}, (\log^2 \log u)/\log u\}$). Applying Observation 9.6 to the state $\mathbf{L}_{T(Q,\rho)}$ from Lemma 9.9 we obtain that $2^m \geq |\{\mathrm{FIX}(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho) \mid Q \subset \mathcal{U}, |Q| = n\}|$. We lower bound how many distinct fixed sets the AMQ needs to store for a given ρ.

By definition, $\mathrm{FORCED}(Q, \mathcal{S}^*, \rho) \subseteq \mathrm{FIX}(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho)$. However, the set of fixed elements cannot be that much bigger than the set of fixed queries. Consider an arbitrary $x \in \mathcal{U}$. By Lemma 9.11, x is a fixed false positive with probability at most n^2/u. Thus, $\mathbf{E}_{Q \in \mathcal{Q}}[|\mathrm{FIX}(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho), \mathcal{S}^*, \rho)|] = n^2$. By Markov's inequality, $\Pr_{Q \in \mathcal{Q}}\left[|\mathrm{FIX}(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho)| = O(n^2)\right] = \Omega(1)$. Then there exists a set[25]

$$\mathcal{Q}^* = \left\{Q \subseteq \mathcal{U} \;\middle|\; |Q| = n, \mathrm{FIX}(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho) = O(n^2)\right\}$$

such that $|\mathcal{Q}^*| = \Omega(|\mathcal{Q}|)$. Thus,

$$\left|\left\{\mathrm{FIX}\left(\mathbf{L}_{T(Q,\rho)}, \mathcal{S}^*, \rho\right) \;\middle|\; Q \in \mathcal{Q}\right\}\right| \geq \left|\left\{\mathrm{FORCED}(Q, \mathcal{S}^*, \rho) \;\middle|\; Q \in \mathcal{Q}^*, |\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0')\right\}\right| \Big/ \binom{O(n^2)}{\Omega(n\varepsilon_0')}.$$

Now, we want to count the number of distinct $\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)$. Let $\mathcal{Z} = \{Z \subseteq \mathrm{OFP}(\mathcal{S}^*, \rho) \mid |Z| = \Theta(n\varepsilon_0')\}$. We show that with high probability a randomly chosen set $Z \in \mathcal{Z}$ belongs to the set of $\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)$ for some Q (because we choose Q uniformly

[25] Again, $\mathcal{Q}^*$ is a function of ρ and $\mathcal{S}$; we do not carry this through the notation for simplicity.

at random, this probabilistic argument lower bounds the number of such Z immediately). Recall that $|\mathrm{OFP}(\mathcal{S}^*, \rho)| > u\varepsilon_0$.

$$\Pr_{Z\in\mathcal{Z}}[\exists Q \in \mathcal{Q}^* \text{ with } Z \subseteq \mathrm{FORCED}(Q, \mathcal{S}^*, \rho)] \geq \Pr_{\substack{Z\in\mathcal{Z}\\ Q\in\mathcal{Q}*, Q\supset Z}}[Z \subseteq \mathrm{FORCED}(Q, \mathcal{S}^*, \rho)\}] \quad (13)$$

$$\geq \Pr_{\substack{Z\in\mathcal{Z}\\ Q\in\mathcal{Q}*, Q\supset Z}}\left[Z \subseteq \mathrm{FORCED}(Q, \mathcal{S}^*, \rho) \;\middle|\; |\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0')\right]$$
$$\cdot \Pr_{Q\in\mathcal{Q}*}\left[|\mathrm{FORCED}(Q, \mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0') \;\middle|\; |Q \cap \mathrm{OFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0)\right]$$
$$\cdot \Pr_{Q\in\mathcal{Q}*}[|Q \cap \mathrm{OFP}(\mathcal{S}^*, \rho)| = \Omega(n\varepsilon_0)] \quad (14)$$

$$\geq \left(1 \Big/ \binom{n}{\Omega(n\varepsilon_0')}\right)\left(1 - \frac{1}{\mathrm{poly}(n)}\right)\left(1 - \frac{1}{\mathrm{poly}(n)}\right). \quad (15)$$

The first term in step (15) above uses the fact that if the AMQ was forced to fix $\Omega(n\varepsilon_0')$ queries out of query set Q, then the probability that a randomly chosen Z corresponds to this forced query set is at most $1/$(total number of possible forced subsets of Q). There can be at most $\binom{n}{\Omega(n\varepsilon_0')}$ such subsets. The second and third term in step (15) follow from Lemma 9.10 and Lemma 9.8 respectively—because $|\mathcal{Q}^*| = \Omega(|\mathcal{Q}|)$, a simple probabilistic argument shows that the probability over $Q \in \mathcal{Q}^*$ rather than $Q \in \mathcal{Q}$ retains the high probability bounds. Thus, we can lower bound the total number of distinct forced sets as

$$\left|\left\{\mathrm{FORCED}(Q, \mathcal{S}^*, \rho) \mid Q \in \mathcal{Q}^*\right\}\right| \geq \binom{u\varepsilon_0}{\Omega(n\varepsilon_0')}\left(1 - \frac{1}{\mathrm{poly}(n)}\right) \Big/ \binom{n}{\Omega(n\varepsilon_0')}.$$

Putting it all together,

$$2^m \geq \binom{u\varepsilon_0}{\Omega(n\varepsilon_0')}\left(1 - \frac{1}{\mathrm{poly}(n)}\right) \Big/ \binom{n}{\Omega(n\varepsilon_0')}\binom{O(n^2)}{\Omega(n\varepsilon_0')}.$$

Taking logs and simplifying (recall $(x/y)^y \leq \binom{x}{y} \leq (xe/y)^y$),

$$m = \Omega\left(n\varepsilon_0'\left(\log\frac{u\log(1/\varepsilon_0)}{n} - \left(\log\frac{n}{\varepsilon_0'} + \log\frac{1}{\varepsilon_0'}\right)\right)\right).$$

We have $\log(u\log(1/\varepsilon_0)/n) - \log(n/\varepsilon_0') = \Omega(\log u)$ because $u \gg n^2/\varepsilon_0'$. Because ε is a constant, $\varepsilon_0' = \Omega(\varepsilon_0/\log(1/\varepsilon_0))$. Thus, $m = \Omega\big(\frac{n\varepsilon_0 \log u}{\log(1/\varepsilon_0)}\big)$.

Using the definition of ε_0, we have two cases.

1. If $1/n^{1/4} \geq (\log^2 \log u)/\log u$, then $m = \Omega(n \log n \log u/\log\log n)$.
2. If $1/n^{1/4} < (\log^2 \log u)/\log u$, then $m = \Omega(n \log\log u)$.

In the first case, we get a contradiction to our assumption that $m \leq n \log 1/\varepsilon_0 + 4n$. In the second, we get that if $m \leq n \log\log u + 4n$, we obtain a bound of $\Omega(n \log\log u)$. □

Matching upper bound. We construct a local AMQ that matches the above space bound.

Lemma 9.12. *There exists a local adaptive AMQ that can handle $O(n)$ adaptive queries using $O(\min\{n \log n, n \log\log u\})$ bits of space with high probability.*

Proof. If $\log n = O(\log\log u)$, we build a standard bloom filter with $\varepsilon_0 = 1/n^c$ for $c > 1$. The adversary never queries an original absolute false positive with high probability. (If it does, the AMQ can store it without affecting the w.h.p. space bound.)

Otherwise, we have $\log\log u = o(\log n)$, and therefore $n/\log u = n^{\Omega(1)}$. We build a standard bloom filter with error rate $\log\log u/\log u$; this requires $O(n \log\log u)$ space.

After $O(n)$ queries we expect $O(n \log\log u/\log u)$ false positives; from the assumption on the size of u this is $n^{\Omega(1)}$ so Chernoff bounds imply $O(n \log\log u/\log u)$ false positives with high probability. We store all false positives in a whitelist. Each requires $O(\log u)$ bits. Thus, the space used is $O(n \log\log u)$. □

9.5 Broom Filter: An Adaptive Bloom filter

We design an adaptive AMQ that is near-optimal on all metrics:

Theorem 9.13. *There exists an adaptive AMQ—the broom filter—that, for any sustained false-positive rate ε and maximum capacity n, attains the following performance:*

Constant local work:	$O(1)$ *worst-case operations for inserts, deletes, and lookups w.h.p.*
Near optimal local space:	$(1 + o(1))n \log_2 \frac{1}{\varepsilon} + O(n)$ *local space w.h.p.*
$O(1)$ *remote accesses:*	$O(1)$ *accesses to the remote representation* $\mathbf{R}$ *for each false positive and each delete. For each insert, there are* $O(1)$ *accesses to* $\mathbf{R}$ *with probability at most* ε.

The broom filter is a single-hash-function AMQ, which means that the AMQ stores fingerprints for each element in $\mathcal{S}$. In Chapter 9.5, we say what fingerprints we store, and how they establish the sustained false-positive rate of broom filters. In Chapter 9.5.1, we show how to maintain the fingerprints space-efficiently and in $O(1)$ work per operation.

We begin our proof of Theorem 9.13 by describing how to modify a single-hash-function AMQ to achieve adaptivity. In particular, known single-function AMQs (i.e., [22, 60, 118]) work by storing a fingerprint for each element in $\mathcal{S}$. In this section, we say what fingerprints

we store and how they can be used to guarantee a sustained false-positive rate of ε; in Chapter 9.5.1 we describe how to maintain the fingerprints efficiently.

9.5.1 Extending Fingerprints Using Adaptivity Bits

The broom filter has a hash function $h : \mathcal{U} \to \{0, \ldots n^c\}$ for some constant $c \geq 4$. Storing an entire hash takes $c \log n$ bits, however, we can only afford approximately $\log_2(1/\varepsilon)$ bits per element. Instead, for set $\mathcal{S} = \{y_1, y_2, \ldots, y_n\}$, the broom filter stores a set of *fingerprints* $\mathcal{P} = \{p(y_1), p(y_2), \ldots, p(y_n)\}$, where each $p(y_i)$ is a *prefix* of $h(y_i)$, denoted $p(y_i) \sqsubseteq h(y_i)$.

Queries. A query for x returns "present" iff there exists a $y \in \mathcal{P}$ such that $p(y) \sqsubseteq h(x)$.

The first $\log n + \log(1/\varepsilon)$ bits of a fingerprint comprise the *baseline fingerprint*, which is subdivided as in a quotient filter [22, 119]. In particular, the first $q = \log n$ bits comprise the *quotient*, and the next $r = \log(1/\varepsilon)$ bits the *remainder*. The remaining bits (if any) comprise the *adaptivity bits*.

Using the parts of the fingerprint. The baseline fingerprint is long enough to guarantee that the false-positive rate is at most ε. We add adaptivity bits to fix false positives, in order to achieve a sustained false-positive rate of ε. Adaptivity bits are also added during insertions. We maintain the following invariant:

Invariant 9.14. *No fingerprint is a prefix of another.*

By this invariant, a query for x can match at most one such $p(y) \in \mathcal{P}$. We show that we can fix a false positive by adding adaptivity bits to the single $p(y)$, for which $p(y) \sqsubseteq h(x)$. Thus, adding adaptivity bits during insertions reduces the number of adaptivity bits added during false positives, which allows us to achieve $O(1)$ work and remote accesses.

There is also a somewhat subtler reason why adaptivity bits are added during insertions—in order to defeat deletion-based timing attacks, which we discuss shortly.

Maintaining the fingerprints. Here we describe what the broom filter does on a call to Adapt. In this section we drop $(\mathbf{L}, \mathbf{R})$ and ρ from the function notation for simplicity.

We define a subroutine of Adapt which we call Extend$(x, \mathcal{P})$. This function is used to maintain Invariant 9.14 and to fix false positives.

Observe that on a query x there exists at most one y for which $p(y) \sqsubseteq h(x)$, by Invariant 9.14. If such a y exists, the Extend$(x, \mathcal{P})$ operation modifies the local representation by appending adaptivity bits to $p(y)$ until $p(y) \not\sqsubseteq h(x)$. (Otherwise, Extend$(x, \mathcal{P})$ does nothing.) Thus, Extend performs remote accesses to RevLookup$_{\mathcal{P}}$, where RevLookup$_{\mathcal{P}}(x)$

returns the (unique) $y \in \mathcal{S}$ such that $p(y) \sqsubseteq h(x)$. RevLookup$_{\mathcal{P}}$ is a part of $\mathbf{R}$, and can be implemented using a dictionary.

We can define Adapt(x) as follows:

- **Queries.** If a query x is a false positive, we call Extend$(x, \mathcal{P})$, after which x is no longer a false positive.
- **Insertions.** When inserting an element x into $\mathcal{S}$, we first check if Invariant 9.14 is violated, that is, if there exists a $y \in \mathcal{S}$ such that $p(y) \sqsubseteq h(x)$.[26] If so, we call Extend$(x, \mathcal{P})$, after which $p(y) \not\sqsubseteq h(x)$. Then we add the shortest prefix of $h(x)$ needed to maintain Invariant 9.14.
- **Deletions.** Deletions do not make calls to Adapt. We defer the details of the deletion operation until after we discuss how to reclaim bits introduced by Adapt. For now we note the naïve approach of deleting an element's fingerprint is insufficient to guarantee a sustained false-positive rate.

Reclaiming bits. Each call to Adapt adds bits, and so we need a mechanism to remove bits. An amortized way to reclaim bits is to rebuild the broom filter with a new hash function every $\Theta(n)$ calls to Adapt.

This change from old to new hash function can be deamortized without losing a factor of 2 on the space. We keep two hash functions, h_a and h_b; any element y greater than frontier z is hashed according to h_a, otherwise, it is hashed according to h_b. At the beginning of a *phase*, frontier $z = -\infty$ and all elements are hashed according to h_a. Each time we call Adapt, we delete the smallest constant $c > 1$ elements in $\mathcal{S}$ greater than z and reinsert them according to h_b. (Finding these elements requires access to $\mathbf{R}$; again this can be efficiently implemented using standard data structures.) We then set z to be the value of the largest reinserted element. When z reaches the maximum element in $\mathcal{S}$, we begin a new phase by setting $h_a = h_b$, picking a new h_b, and resetting $z = -\infty$. We use this frontier method for deamortization so that we know which hash function to use for queries: lookups on $x \leq z$ use h_b and those on $x > z$ use h_a.

Observation 9.15. *A hash function times out after $O(n)$ calls to* Adapt.

Because every call to Adapt introduces an expected constant number of adaptivity bits, we obtain the following lemma.

Lemma 9.16. *In any phase,* Adapt *introduces $O(n)$ adaptivity bits into the broom filter with high probability.*

[26] This step and the following assume x does not already belong to $\mathcal{S}$. If it does, we don't need to do anything during insertions.

Proof. By Observation 9.15, for some constant c_1, there are $c_1 n$ false positives before the entire AMQ gets rehashed. Constant c_1 is determined by the number of elements that get rehashed per false positive, and so can be tuned.

Each time there is a false positive, there is a collision with exactly one element by Invariant 9.14. Conditioned on the fact that there is a collision, the probability that it can be resolved by extending fingerprints by i bits is 2^{-i}. Whenever an element is rehashed, its adaptivity bits get thrown out. Thus, by Chernoff bounds, the number of adaptivity bits in the data structure at any time is $O(c_1 n)$ w.h.p. □

If we did not have deletions, then Observation 9.15 and Lemma 9.16 would be enough to prove a bound on total size of all fingerprints—because adaptivity bits are removed as their hash function times out. We will see that supporting deletions involves introducing adaptivity bits via a second mechanism. We will show that this second mechanism also introduces a total of $O(n)$ adaptivity bits per phase.

Deletions and adaptivity bits. It is tempting to support deletions simply by removing fingerprints from $\mathcal{P}$, but this does not work. To see why, observe that false positives are eliminated by adding adaptivity bits. Removing fingerprints destroys history and reintroduces false positives. This opens up the data structure to timing attacks by the adversary.

We describe one such timing attack to motivate our solution. The adversary finds a false positive x and the element $y \in \mathcal{S}$ that collides with x. (It finds y by deleting and reinserting random elements until x is once again a false positive.) The attack then consists of repeatedly looking up x, deleting y, then inserting y. This results in a false positive on every lookup until x or y's hash function changes.

Thus, the broom filter needs to remember the history for deleted elements, since they might be reinserted. Only once y's hash function has changed can y's history be forgotten. A profligate approach is to keep the fingerprints of deleted elements as "ghosts" until the hash function changes. Then, if the element is reinserted, the adaptivity bits are already there. Unfortunately, remembering deleted elements can blow up the space by a constant factor, which we cannot afford.

Instead, we remember the adaptivity bits and quotient from each deleted element's fingerprint—but we forget the remainder. Only once the hash function has changed do we forget everything. This can be accomplished by including deleted elements in the deamortized rehashing strategy of reclaiming bits. (with deletions, we increase the requirement on adaptivity bits reclaimed at once to $c > 2$).

Now when a new element x gets inserted, we check whether there exists a ghost that matches $h(x)$. If so, then we give x at least the adaptivity bits of the ghost, even if this is

more than needed to satisfy Invariant 9.14. This scheme guarantees the following:

Property 9.17. *If x is a false positive because it collides with y, then it cannot collide with y again until x or y's hash function times out (even if y is deleted and reinserted).*

Sustained false-positive rate. We now establish the sustained false-positive rate of broom filters. We begin by introducing notation:

Definition 9.18. *Hashes $h(x)$ and $h(y)$ have a* soft collision *when they have the same quotient. They have a* hard collision *when they have the same quotient and remainder. Hash $h(x)$ and fingerprint $p(y)$ have a* full collision *if $p(y) \sqsubseteq h(x)$.*

Lemma 9.19. *The probability that any query has a hard collision with any of n fingerprints is at most ε.*

Proof. The probability that any query collides with a single fingerprint is $2^{-(\log n + \log(1/\varepsilon))} = \varepsilon/n$. Applying the union bound, we obtain the lemma. □

Lemma 9.20. *The sustained false-positive rate of a broom filter is ε.*

Proof. We need to establish that any query on any x has $\Pr[\exists y \in \mathcal{S} \mid x \text{ has a full collision with } y] \leq \varepsilon$, regardless of the previous history. Any previous query that is a negative or a true positive has no effect on the data structure. Furthermore, deletions do not increase the chance of any full collision, so we need only consider false positives and insertions, both of which induce rehashing.

We say that $x \in \mathcal{U}$ and $y \in \mathcal{S}$ are *related at time t* if (1) there exists $t' < t$ such that x was queried at time t' and y was in $\mathcal{S}$ at t', and (2) and between times t' and t, the hash functions for x and y have not changed. Suppose x is queried at time t. Then, by Property 9.17, if x and y are related at time t, then $\Pr[x \text{ is a false positive at time } t] = 0$. If x and y are not related at time t, then $\Pr[x \text{ has a full collision with } y] \leq \Pr[h(x) \text{ has a hard collision with } h(y)]$. Finally, by Lemma 9.19, we get $\Pr[x \text{ is a false positive at time } t] \leq \varepsilon$. □

Space bounds for adaptivity bits. We first prove that at any time there are $O(n)$ adaptivity bits. Then we bootstrap this claim to show a stronger property: there are $\Theta(\log n)$ fingerprints associated with $\Theta(\log n)$ contiguous quotients, and these fingerprints have a total of $O(\log n)$ adaptivity bits w.h.p. (thus they can be stored in $O(1)$ machine words).

For the analysis, we partition adaptivity bits into two classes: *extend bits*, which are added by calls to EXTEND, and *copy bits*, which are added on insertion due to partial matches with formerly deleted items. As some bits may be both extend and copy bits, we

partition adaptivity bits by defining all the adaptivity bits in a fingerprint to be of the same type as the last bit, breaking ties in favor of extend. If an item is deleted and then reinserted, its bits are of the same type as when it first got them. (So if an item that gets extend bits is deleted and reinserted with the same adaptivity bits, then it still has extend bits.)

Lemma 9.21. *At any time, the broom filter has $O(n)$ adaptivity bits with high probability.*

Proof. Lemma 9.16 bounds the number of extend bits. We still need to bound the number of copy bits. We do so using a straightforward application of Chernoff bounds.

The number of quotients that have at least k extend bits is $O(n/k)$. This is because the total number of extend bits is $O(n)$. Therefore, the probability that $h(x)$ accumulates k extend bits is $O(1/(k2^k)$. (This is the probability that $h(x)$ matches a quotient with k extend bits times the probability that those extend bits match.)

Thus, the expected number of copy bits from length-k strings is $O(n/2^k)$, for $1 \leq k \leq \Theta(\log n)$. By Chernoff, these bounds also hold w.h.p. for $k \leq (\log n)/\log\log n$; for $k > (\log n)/\log\log n$ Chernoff bounds give that there are $O(\log n)$ bits from length-k strings w.h.p. Thus, the total number of adaptivity bits is $O(n)$, w.h.p. □

Lemma 9.22. *There are $\Theta(\log n)$ fingerprints associated within a range of $\Theta(\log n)$ contiguous quotients, and these fingerprints have a total of $O(\log n)$ extend bits w.h.p.*

Proof. As long as there are $O(n)$ adaptivity bits and $\Theta(n)$ elements in the broom filter, then no matter how the adaptivity bits are distributed: the first time that x is queried or inserted with hash function h, the probability that ADAPT is called is $\Theta(\varepsilon)$. Thus, by Chernoff bounds, before the phase ends, there are $O(n/\varepsilon)$ distinct elements not in $\mathcal{S}$ that are ever queried and $O(n/\varepsilon)$ distinct elements that are ever inserted into $\mathcal{S}$.

We can now calculate an upper bound on the number of adaptivity bits at any time t. Recall that at the very beginning of phase ℓ, there is a unique hash function h_ℓ that is in use, because $h_{\ell-1}$ has expired, and $h_{\ell+1}$ has not been used yet. Any extend adaptivity bits that are in the broom filter at time t in phase ℓ were generated as a result of collisions generated by $h_{\ell-1}$, h_ℓ, or $h_{\ell+1}$.

Now consider all elements that were ever inserted or queried any time during phase $\ell-1$, ℓ, or $\ell+1$ with $h_{\ell-1}$, h_ℓ, or $h_{\ell+1}$. If we took all these elements, and inserted them one at a time into $\mathcal{S}$, calling ADAPT to resolve any collisions, this scheme would generate all the extend adaptivity bits that are present at time t (and more).

It thus suffices to show that even with this overestimate, the fingerprints associated within a range of $\Theta(\log n)$ contiguous quotients have a total of $O(\log n)$ extend bits w.h.p. Call the $\Theta(\log n)$ quotients under consideration the *group*. Define 0/1-random variable $X_i = 1$ iff

element x_i lands in the group and induces a call to extend. Thus, $\Pr[X_i = 1] \leq O(\varepsilon \log n/n)$. There are $O(n/\varepsilon)$ elements inserted, deleted, and/or queried in these rounds. Thus, by Chernoff bounds, the number of elements that land in this quotient group is $O(\log n/\varepsilon)$, and at most $O(\log n)$ of them get adaptivity bits with high probability.

Next we calculate the number of bits it takes to resolve the collisions. There are $O(\log n)$ elements that land in this group. We can model this as a balls and bins game, where elements land in the same bin if they share the same quotient and remainder. Let random variable K_i represent the number of elements in the ith nonempty bin. Then the expected number of bits that get added until all collisions are resolved is $2\sum_{i=1}^{O(\log n)} \log(K_i)$. By the convexity, $\sum_{i=1}^{O(\log n)} \log(K_i) = O(\log n)$, regardless of the distribution of the elements into buckets.

To achieve concentration bounds, we upper bound this process by a different process. Each time we add a bit, there is a probability of at least $1/2$ that it matches with at most half of the remaining strings. Thus, the number of adaptivity bits is stochastically dominated by the number of coin flips we need until we get $\Theta(\log n)$ heads, which is $\Theta(\log n)$ w.h.p. □

Lemma 9.23. *There are $\Theta(\log n)$ fingerprints associated within a range of $\Theta(\log n)$ contiguous quotients, and these fingerprints have a total of $O(\log n)$ adaptivity bits w.h.p.*

Proof. We established the bound on extend bits in Lemma 9.22; now we focus on copy bits.

Consider any time t when there are n elements in the broom filter, and consider any group of $\Theta(\log n)$ contiguous quotients. By Chernoff bounds, $\Theta(\log n)$ of these n elements have hashes that have a soft collision with one of these quotients w.h.p. By Lemma 9.21, there are a total of $O(\log n)$ extend bits in this range. We now show that there are also a total of $O(\log n)$ copy bits.

Our scheme for handling adaptivity bits of deleted elements can be described in terms of balls and bins—there are $\Theta(\log n)$ bins, one for each quotient. Each *string* of adaptivity bits belongs in a bin. Some bins can have multiple strings (but by standard balls-and-bins arguments, the fullest bin has $O(\log n/\log\log n)$ strings of adaptivity bits). When a new element x is inserted, it lands in the bin determined by $h(x)$. Then $p(x)$ inherits the adaptivity bits in the bin iff $h(x)$ matches those adaptivity bits. (Thus any given string of adaptivity bits started out as extend bits, even if it got copied many times as copy bits.)

We now bound the number of adaptivity bits by considering a variation that adds more bits than our scheme. For each element inserted into a bin, we keep appending copy bits as long as there is a match with some string of adaptivity bits in the bin. Once there is a mismatch with every string, we stop. Thus, while our scheme adds copy bits only on *complete* matches, we allow *prefix* matches while still retaining good bounds.

We again overestimate the bounds by assuming that the adaptivity bits are adversarially

(rather than randomly) divided into bit strings and that the bit strings are adversarially distributed among the bins.

Let random variable K_i denote the number of adaptivity bit strings in the bin where the ith element lands. Then, $\Pr[K_i \geq X] < O(1/X)$. This follows from Markov's inequality and Lemma 9.22. Since w.h.p., the total number of adaptivity bits is at most $O(\log n)$, the expected number of bits in a bin, and therefore the expected number of strings, is $O(1)$.

By the previous claim,

$$\Pr[K_i \geq X] \leq \Pr[\text{we flip a coin and get at least } \log(X) - O(1) \text{ tails before any head}].$$

Therefore, the probability that $\sum_{i=1}^{c \log n} \log(K_i) = d \log n$ is at most the probability that we flip a coin $d \log n$ times and get at most $c \log n$ heads. For a suitable choice of constants c and d, this is polynomially small. Thus, $\sum_{i=1}^{\Theta(\log n)} \log(K_i) = O(\log n)$.

Next we bound the total number of adaptivity bits that the elements inherit. Element x_i lands in a bin with K_i adaptivity bit strings. Each time a bit is added, with probability at least $1/2$, the number of adaptivity strings that still match with $h(x_i)$ decreases by half. Specifically, suppose that k adaptivity strings still match x_i. With probability at least $1/2$, after the next bit reveal, at most $\lfloor k/2 \rfloor$ still match. So after an expected $\leq 2 \log(K_i)$ bits, no adaptivity bit strings still match x_i. Once again this game is modeled as flipping a coin until until we get $\Theta(\log n)$ heads, and by Chernoff, only $\Theta(\log n)$ are needed w.h.p. □

Implementing Fingerprints and their Updates

In Chapter 9.5, we showed how to use fingerprints to achieve a sustained false-positive rate of ε (Lemma 9.20). In this section we give space- and time-efficient implementations for the fingerprint operations that are specified in Chapter 9.5. We explain how we store and manipulate adaptivity bits, quotients, and remainders.

We describe two variants of our data structure, because there are two ways to manage remainders, depending on whether $\log(1/\varepsilon) \leq 2 \log \log n$, the *small-remainder case* (Chapter 9.5.2), or $\log(1/\varepsilon) > 2 \log \log n$, the *large-remainder case* (Chapter 9.5.3).

Bit Manipulation within machine words. Next we show how to implement a variety of primitives on machine words in $O(1)$ time using word-level parallelism. The upshot is that from now on, we may assume that the asymptotic complexity for any operation on the broom filter is simply the number of machine words that are touched during the operation.

This section explains that we can store and maintain small strings (fingerprints or metadata) compactly within words, while retaining $O(1)$ lookup (e.g., prefix match), insert, and

delete. Based on this section, we can just focus on where (i.e., in which word) data is stored, and we use lemma Lemma 9.24 as a black box and assume that we can do all manipulations within machine words in $O(1)$ time. (For simplicity, we assume that machine words have $\Theta(\log n)$ bits, since words cannot be shorter and it does not hurt if they are longer.)

Lemma 9.24. *Consider the following input.*

- *A "query" bit string q that fits within a $O(1)$ $O(\log n)$-bit machine words.*
- *"Target" strings $s_1, s_2, \ldots, s_\ell$ concatenated together so that $s = s_1 \circ s_2 \circ \cdots \circ s_\ell$ fits within $O(1)$ $O(\log n)$-bit machine words. The strings need not be sorted and need not have the same length.*
- *Metadata bits, e.g., a bit mask that indicates the starting bit location for each s_i.*

Then the following operations take $O(1)$ time. (Query results are returned as bit maps.)

1. Prefix match query. *Given a range $[j, k]$, find all s_i such that $i \in [j, k]$ and s_i is a prefix of q.*
2. Prefix-length query. *More generally, given a range $[j, k]$, for each s_i such that $i \in [j, k]$, indicate the length of the prefix match between q and s_i.*
3. Insert. *Given an input string $\hat{s}$ and target rank i, insert $\hat{s}$ into s as the new string of rank i.*
4. Delete. *Given a target rank i, delete s_i from s.*
5. Splice, concatenate, and other string operations. *For example, given i, concatenate s_i with s_{i+1}. Given i and length x, remove up to x bits from the beginning of s_i.*
6. Parallel inserts, deletes, splices, concatenations, and queries. *Multiple insert, delete, query, splice, or concatenate operations can be supported in parallel, e.g., remove x bits from the beginning of all strings, and indicate which strings still have bits.*

Proof. These operations can be supported using standard compact and succinct data-structures techniques, such as rank, select, mask, shift, and sublinear-sized lookup tables.

Insertion, deletion, concatenation, etc., are handled using selects, bit-shifts, masks, etc.

We first explain how to find a string s_i that is a prefix match of q for the special case that q has length at most $\log n/8$. We partition s into chunks such that each chunk has size $\log n/8$. Now some s_j are entirely contained within one chunk and some straddle a chunk boundary. As only $O(1)$ strings can straddle a boundary, we can search each of them serially.

In contrast, there may be many strings that are entirely contained within a chunk, and these we need to search in parallel. We can use lookup tables for these parallel searches. Since there are at most $2^{\log n/8} = n^{1/8}$ input choices for query string q and at most $n^{1/4}$ input choices for the concatenated target strings (metadata bits and s), a lookup table with precomputed responses to all possible queries still takes $o(n)$ bits.

The remaining operations are supported similarly using rank, select, and lookup tables. For example, the more general case of querying larger q is also supported similarly by dividing q into chunks, comparing the chunks iteratively, and doing further parallel manipulation on s, also using lookup tables. In particular, since we compare q in chunks, we have to remove all strings s_i that already do not have a prefix in common with q as well as those shorter strings where no prefix is left. □

Encoding adaptivity and deletion bits. We store adaptivity bits separately from the rest of the fingerprint. By Lemma 9.23, all of the adaptivity bits in any range of $\Theta(\log n)$ quotients fit in a constant number of words. Thus, all of the searches and updates to (both copy and extend) adaptivity bits can be achieved in $O(1)$ time by Lemma 9.24.

Encoding quotients. Quotients and remainders are stored succinctly in a scheme similar to quotient filters [22, 119]; we call this high-level scheme *quotienting*.

Quotienting stores the baseline fingerprints succinctly in an array of $\Theta(n)$ slots, each consisting of r bits. Given a fingerprint with quotient a and remainder b, we would like to store b in position a of the array. This allows us to reconstruct the fingerprint based on b's location. So long as the number of slots is not much more than the number of stored quotients, this is an efficient representation. (In particular, we will have a sublinear number of extra slots in our data structure.)

The challenge is that multiple fingerprints may have the same quotient a, and thus contend for the same location. Linear probing is the standard technique for resolving collisions: slide an element forward in the array until it finds an empty slot. Linear probing does not immediately work, however, since the quotient is supposed to be reconstructed based on the location of a remainder. The idea of a quotient filter is to maintain a small number of metadata bits per array slot (e.g., between 2 and 3) along with linear probing. The metadata bits encode the target slot for a remainder even when it is shifted to a different slot.

As-is, the quotient filter does not achieve constant time operations, independent of ε. This is because when the remainder length $r = \log(1/\varepsilon) = \omega(1)$, and the fingerprint is stored in a set of $\Omega(\log n)$ contiguous slots, there can be $\omega(1)$ locations (words) where the target fingerprint could be. (This limitation holds even when the quotient filter is half empty, in which case it is not even space efficient enough for Theorem 9.13.)

Nonetheless, the quotient filter is a good starting point for the broom filter as it allows us to maintain a *multiset* of baseline fingerprints subject to insertions, deletions, and queries. In particular, some queries will have a hard collision with multiple elements.[27] We need to

[27]Given this subtlety, we did not manage to get the single-hash function bloom filters of Pagh et al. [118]

compare the adaptivity bits of the query to the adaptivity bits of *each* colliding element. The quotienting approach guarantees that these adaptivity bits are contiguous, allowing us to perform multiple comparisons at once using word-level parallelism. Specifically, Lemma 9.22 ensures that the adaptivity bits for $O(\log n)$ quotients fit into $O(1)$ machine words.

9.5.2 Broom Filter Design for the Small-Remainder Case

In this section we present a data structure for the case that $r = O(\log \log n)$. This data structure is simple, yet for this range of r its performance dominates all other known AMQs.

High level setup. Our data structure consists of two levels: a primary level and a secondary level. Each level is essentially a quotient filter; however, we slightly change the insert and delete operations for the primary level in order to ensure constant-time accesses.

As in a quotient filter, the primary level consists of $n(1+\alpha)$ slots, where each slot has a remainder of size $r = \log(1/\varepsilon) = O(\log \log n)$. Parameter α denotes the subconstant extra space we leave in our data structure; thus the primary level is a quotient filter, with space parameterized by α (and with slightly modified inserts, queries, and deletes). We require $\alpha \geq \sqrt{(9r \log \log n)/\log n}$; a simpler choice like $\alpha = 1/\log \log n$ works fine.

The secondary level consists of another quotient filter with $\Theta(n/\log n)$ slots. This data structure uses a different hash function h_2. Thus, an element x has two fingerprints $p_1(x)$ and $p_2(x)$. The internals of the two levels are maintained entirely independently: Invariant 9.14 is maintained separately for each level, and adaptivity bits do not carry over from the primary level to the secondary level.

How to perform inserts, queries and deletes. To insert $y \in \mathcal{S}$, we first try to store the fingerprint $p_1(y)$ in the primary level. This uses the technique described in Chapter 9.5.1: we want to store the remainder in the slot determined by the quotient. If the slot is empty, we store the remainder of $p_1(y)$ in that slot. Otherwise, we begin using linear probing to look for an empty slot, updating the metadata bits accordingly; see [22, 119].

However, unlike in previous quotienting-based data structures, we stop our probing for an empty slot early: in the primary level, the data structure only continues the linear probing over $O((\log n)/r)$ slots (and thus $O(1)$ words). If all of these slots are full, the item gets stored in the secondary level. In Lemma 9.26 we show that it finds an empty slot in $O(1)$ words in the secondary level w.h.p.

or the backyard hashing construction of Arbitman et al. [5] to interface with adaptivity bits. Instead we used techniques (e.g., not cuckoo-ing) that permit the same element to be explicitly duplicated multiple times.

We always try to insert into the primary level first. That is, even if x is deleted from the secondary level while reclaiming bits, we still attempt to insert x into the primary level first.

Queries are similar to inserts—to query for y, we calculate $p_1(y)$ and search for it in the primary level for at most $O((\log n)/r)$ slots; if this fails we calculate $p_2(y)$ and search for it in the secondary level.

Lemma 9.25. *The number of elements that get inserted into the secondary level is $O(n/\log^2 n)$ with high probability.*

Proof. Partition the primary level into *primary bins* of $(1+\alpha)(\log n)/r$ consecutive slots. An element is inserted into the secondary level only if it is inserted into a sequence of $\Omega((\log n)/r)$ full slots; for this to happen either the primary bin containing the element is full or the bin adjacent to it is full. We bound the number of full primary bins.

In expectation, each bin is (initially) hashed to by $(\log n)/r$ elements. Thus, by Chernoff bounds, the probability that a given primary bin is hashed to by at least $(1+\alpha)(\log n)/r$ elements is at most $\exp(-(\alpha^2 \log n)/(3r)) \leq 1/\log^3 n$.

Thus, in expectation, $n/\log^3 n$ primary bins are full. Since these events are negatively correlated, we can use Chernoff bounds, and state that $O(n/\log^3 n)$ primary bins are full with high probability.

Each primary bin is hashed to by $O(\log n)$ elements in expectation (even fewer, in fact). Again by Chernoff, each primary bin is hashed to by $O(\log n)$ elements with high probability.

Putting it together, even if all $O(\log n)$ elements hashed into any of the $O(n/\log^3 n)$ overflowing primary bins (or either adjacent bin) are inserted into the secondary level. □

Lemma 9.26. *With high probability, all items in the secondary level are stored at most $O(\log n/r)$ slots away from their intended slot.*

Proof. Partition the secondary level into *secondary bins* of $\Theta(\log n/r)$ consecutive slots. Thus, there are $\Theta(nr/\log^2 n)$ secondary bins. The lemma can only be violated if one of these bins is full.

By Lemma 9.25, we are inserting $O(n/\log^2 n)$ elements into these bins. By classical balls and bins analysis, because there are more bins than balls, the secondary bin with the most balls has $O((\log n)/\log\log n) = O((\log n)/r)$ elements with high probability. Thus, no secondary bin ever fills up with high probability. □

Performance. The $O(1)$ lookup time follows by definition in the primary level, and by Lemma 9.26 in the second level. The total space of the primary level is $O((1+\alpha)n\log(1/\varepsilon))+$

$O(n)$, and the total space of the second level is $O((n \log(1/\varepsilon))/\log n)$. We guarantee adaptivity using the ADAPT function defined in Chapter 9.5, which makes $O(1)$ remote memory accesses per insert and false positive query.

9.5.3 Broom Filter Design for the Large-Remainder Case

We now present a data structure for the the large-remainder case, $\log(1/\varepsilon) > 2 \log\log n$. Here is how this case differs from the previous one. Large remainders are harder to store efficiently since only a small number can fit in a machine word. E.g., we are no longer guaranteed to be able to store the remainders from all hard collisions in $O(1)$ words w.h.p.

However, large remainders also have advantages. We are very likely to be able to search using only a small portion of the remainder—a portion small enough that many can be packed into $O(1)$ words. In particular, we can "peel off" the first $2 \log\log n$ bits of the remainder, filter out collisions just based on those bits, and we are left with few remaining potential collisions. We call these *partial collisions*.

So we have an initial check for uniqueness, then a remaining check for the rest of the fingerprint. This allows us to adapt the small-remainder case to handle larger remainders without a slowdown in time.

Data structure description. As before, our data structure consists of two parts. We refer to them as the *primary level* and the *backyard*. This notation emphasizes the structural difference between the two levels, and emphasizes the relationship with backyard hashing [5]. Unlike the small-remainder case, we use only a single hash function.

The primary level consists of two sets of slots: *signature slots* of size $2 \log\log n$, and *remainder slots* of size $r - 2 \log\log n$. As in Chapter 9.5.2, the number of remainder slots is $(1 + \alpha)n$ and the number of signature slots is $(1 + \alpha)n$, where $\alpha \geq \sqrt{18 \log^2 \log n / \log n}$. Since the appropriate slot is found while traversing the signature slots, we only need to store metadata bits for the signature slots, not for the remainder slots. The signature slots are stored contiguously, thus $O(\log n / \log\log n)$ slots can be probed in $O(1)$ time.

Each item is stored in same the remainder slot as in the normal quotient filter. The signature slots mirror the remainder slots; however, only the first $2 \log\log n$ bits of the remainder are stored, the rest are stored in the corresponding remainder slot.

The front yard. To insert an element y, we first we try to insert $p(y)$ in the front yard. We first find the signature slot corresponding to the quotient of $p(y)$. We then search through at most $O(\log n / \log\log n)$ signatures to find a partial collision (a matching signature) or

an empty slot. We use metadata bits as usual—the metadata bits guarantee that we only search through signatures that have a soft collision with $p(y)$.

If we do find a partial collision—a signature that matches the first $2 \log \log n$ bits of the remainder of $p(y)$—we insert $p(y)$ into the backyard. If there is no empty slot, we insert $p(y)$ into the backyard. If we find an empty slot but do not find a partial collision, we insert $p(y)$ into the empty slot; this means that we insert the signature into the empty signature slot, and insert the full remainder of $p(y)$ into the corresponding remainder slot. We update the metadata bits of the signature slots as in [22, 119].

Querying for an element x proceeds similarly. In the front yard, we find the signature slot corresponding to the quotient of $p(x)$. We search through $O(\log n / \log \log n)$ slots, looking for a matching signature. If we find a matching signature, we look in the corresponding remainder slot to see if we have a hard collision; we return "present" if so. If we do not find a matching signature, or if the corresponding remainder slot does not have a hard collision, we search for $p(x)$ in the back yard.

The back yard. The back yard is just a compact hash table that can store $O(n/\log n)$ elements with $O(1)$ worst-case insert and delete time.[28] One example of such a data structure is by Demaine et al. [53], though there exist other options, such as the backyard hashing scheme of Arbitman et al. [5]. When we store an element y in the back yard, we store its *entire* hash $h(y)$. Thus, w.h.p. there are no collisions in the back yard. Since the back yard has a capacity for $\Theta(n/\log n)$ elements, and each hash has size $\Theta(\log n)$, the back yard takes up $\Theta(n)$ bits, which is a lower-order term.

Lemma 9.27. *With high probability, $O(n/\log^2 n)$ elements are stored in the back yard.*

Proof. An element is stored in the backyard only if 1. it is in a sequence of $\Omega(\log n / \log \log n)$ full slots, or 2. it has a partial collision with some stored element.

The number of elements that are in a sequence of full slots is $O(n/\log^2 n)$ with high probability; this follows immediately from Lemma 9.25 with $r = 2 \log \log n$.

A query element x has a partial collision with an element y if they have the first $\log n + 2 \log \log n$ bits of their fingerprint in common. Thus, x and y collide with probability $1/(n \log^2 n)$; thus x has a partial collision with $1/\log^2 n$ stored elements in expectation. The lemma follows immediately from Chernoff bounds. □

[28] Such a scheme would suffice for the secondary level of the small-remainder case as well. However, the data structure we described for the small remainder case is simpler and likely faster in practice. That scheme does not appear to work for the large-remainder case without significant modifications because the cost analysis breaks down on the second level. Investigating the practical tradeoffs of how to store the secondary level would be interesting future work.

Performance. The back yard requires $O(n)$ total space, since each hash is of length $O(\log n)$. The front yard requires $(1 + \alpha)nr$ space for all primary slots, plus $O(n)$ extra space for the adaptivity bits.

Inserts, deletes, and queries require $O(1)$ time. The search for partial collisions involves $O(\log n / \log \log n)$ signature slots, which fit in $O(1)$ words; these can be searched in constant time using Lemma 9.24. We look at a single remainder slot, which takes $O(1)$ time. If needed, any back yard operation requires $O(1)$ time as well.

9.6 How to Turn any AMQ into an Adaptive AMQ

In this section, we describe how to take an AMQ and modify it to achieve a sustained false-positive rate of ε. Our construction is not quite black box, it's a "grey box"—we make small structural changes to the AMQ, but leave most of its workings intact. This construction does not require accesses to remote representation $\mathbf{R}$ of $\mathcal{S}$ during insertions, just deletions.

Our construction arranges multiple copies of the AMQ in a hierarchical layout. The high level idea of the data structure is that if a query x results in a false positive at some level, then we move the elements in $\mathcal{S}$ that collide with x to the copy of the AMQ at the next level. In the case of Bloom filters and quotient filters, the cost for this data structure over the original AMQ is an extra $1 + O(\varepsilon)$ factor in space usage.

Adaptive Bloom filter. We demonstrate our technique using a Bloom Filter. The data structure consists of d levels (we set d later). Each level consists of a Bloom filter augmented with an extra bit for each cell, denoting whether the cell is *alive* or *dead*. (This extra bit and the resulting changes to LOOKUP are the "grey box" part of the construction.) We call the Bloom filters $B_0, \ldots, B_d$. Bloom filter B_i has k_i randomly chosen hash functions $h_{i,0}, h_{i,1}, \ldots, h_{i,k_i-1}$. Each hash function is selected independently of all other hash functions at all other levels.

Bloom filter B_0 has error rate $\varepsilon_0 = \varepsilon$ and $k_0 = \log(1/\varepsilon)$ hash functions. We specify the parameters of subsequent levels below. At initialization, B_0 contains all n elements of $\mathcal{S}$. All other Bloom filters $B_1 \ldots B_d$ are empty, and all cells in all levels are alive.

We now define the lookup operation in Figure 20, which includes a subroutine to fix x if x is a false positive. LOOKUP(x) begins with a call to LOOKUP$(x, 0)$.

Allocating space for each B_i based on ε_i. We begin by specifying the false-positive probability ε_i at each level. Let $\varepsilon_0 = \varepsilon$, and for $i > 0$, let $\varepsilon_i = \varepsilon_{i-1}^2$. Let n_i be the number of elements that are ever stored in B_i (that is, n_i includes elements that are marked dead and

LOOKUP(x, i):

- If $\exists\ j$ such that $B_i[h_{i,j}(x)] = 0$, return "absent".
- If $B_i[h_{i,j}(x)] = 1\ \forall j$, and no dead cells, then if x is a false positive in B_i, ADAPT$(x,\ i)$, and return "present".
- Else, LOOKUP$(x, i+1)$.

ADAPT$(x,\ i)$:

- Pick a random j from $\{0, \ldots, k_i - 1\}$.
- Mark $h_{i,j}(x)$ as dead in B_i.
- For all $y \in \mathcal{S}$ with $h_{i,j}(y) = h_{i,j}(x)$:
  - Insert y in B_{i+1}, that is, INSERT$(y, i+1)$.

Figure 20: Algorithms for LOOKUP and ADAPT in an adaptive Bloom filter.

inserted into B_{i+1}). Thus, $n_0 = n$. An element is moved to the next level if a false positive collides with it, so $\mathbf{E}[n_i] = \varepsilon_{i-1} n_{i-1}$. We ensure that n_i, the number of elements actually in B_i, is not much bigger than its expectation; if this is violated, we rebuild. Specifically, we guarantee (by Chernoff bounds) that $n_i \leq \mathbf{E}[n_i](1 + 1/\log n)$ with high probability.

We can now compute the space requirement m_i of each level as follows:

$$m_i = n_{i-1}(\varepsilon_{i-1} \log e)(1 + 1/\log n) \log(1/\varepsilon_i) \leq n(\varepsilon_{i-1} \log e)(1 + 1/\log n) \log(1/\varepsilon_{i-1}).$$

Finally, let $d = \log\log\log n - \log\log 1/\varepsilon + O(1)$ be the smallest integer such that $n_d \leq n/(3 \log n)$ with high probability.

When the number of elements n_d is sufficiently small we change technique. There are a variety of solutions we could plug in. For example, we could store the hashes in an open-chaining power-of-two-choices hash table. Since the remainders are so large, we can afford the overhead of a pointer-based data structure and still only incur $O(n)$ bits of space overhead. This would give $O(\log\log n)$-time lookups and updates at the last level, which does not change the ultimate cost of operations. Theorem 9.28 show that this hierarchical construction is adaptive and efficient.

Theorem 9.28. *For any ε and any sequence of up to n queries, there exists a hierarchical construction that takes a (nonadaptive) Bloom filter and generates an adaptive AMQ—the adaptive Bloom filter—that maintains a sustained false-positive rate of ε with high probability. The adaptive Bloom filter guarantees:*

- $O(1)$ *cost in expectation and* $O(\log\log n)$ *w.h.p. for inserts,*
- $O(\log(1/\varepsilon))$ *cost in expectation and* $O(\log\log n)$ *w.h.p. for lookups,*
- $(1 + O(\varepsilon))(\log e)\ n \log(1/\varepsilon)$ *space w.h.p.,*
- $O(1)$ *accesses to the remote representation $\mathcal{R}$ of $\mathcal{S}$ for each (true of false) positive and*

each delete.

Proof. We first show that the total number of levels $d = O(\log\log n)$. After $O(\log\log n)$ levels, we have $\varepsilon_i \le 1/(3\log n(1 + 1/\log n))$, so $n_i \le n/(3\log n)$ with high probability, at which point we switch to an alternative scheme by construction.

Now we bound the total space. Since $\varepsilon_i = \varepsilon_{i-1}^2$, we have $\varepsilon_i \log(1/\varepsilon_i) \le \varepsilon_{i-1}\log(1/\varepsilon_{i-1})$ if $\varepsilon_{i-1} \le 1/2$. In particular, if $\varepsilon \le 1/4$, then $\varepsilon_i \log(1/\varepsilon_i) \le (1/2)\varepsilon_{i-1}\log(1/\varepsilon_{i-1})$. Recall that by our definition of d, $1/\log n = O(\varepsilon_i)$ for all i. Then

$$\begin{aligned}\sum_{i=0}^{d} m_i &\le (\log e)(n\log 1/\varepsilon)\left(1 + \varepsilon(1 + 1/\log n) + (1 + 1/\log n)\sum_{i=2}^{d}\varepsilon_{i-1}\log 1/\varepsilon_{i-1}\right)\\ &= (\log e)(n\log 1/\varepsilon)(1 + O(\varepsilon)).\end{aligned}$$

We also store extra metadata bits. If we store these bits explicitly, we incur an additional space cost of $(\log e)(n\log(1/\varepsilon))$ in B_0 alone, effectively doubling the space. By the above analysis, this is only a problem for B_0—the metadata bits in the recursive Bloom filters do not affect our asymptotics. However, we only set n_i bits in B_{i-1}; in particular, we only set $O(\varepsilon n)$ bits in B_0. Since we only set few bits, we can compress the metadata bits for B_0 while retaining fast query time using standard techniques. For example, we can divide B_0 into $O(\varepsilon n)$ ranges of size $(1/\varepsilon)\log(1/\varepsilon)$, storing the dead bits within each range. This takes $O(\varepsilon n \log(1/\varepsilon))$ total bits. (Since each range has $O(\log(1/\varepsilon))$ expected dead bits, this only adds $O(\log(1/\varepsilon))$ to our expected query time.)

Each query at level i takes $\log(1/\varepsilon_i)$ time. Since a query at level i reaches level $i + 1$ with probability at most ε_i, the total expected cost over all levels is (ignoring constants for simplicity) $\log(1/\varepsilon) + \sum_{i=1}^{d}\varepsilon_i\log(1/\varepsilon_i) \le 2\log(1/\varepsilon)$. The worst-case cost of a query is for one that reaches level d. Since $\log(1/\varepsilon_i) \ge 2\log(1/\varepsilon_i)$, the cost of this query is mostly incurred at level d itself, giving a cost of $O(\log(1/\varepsilon_d)) = O(\log\log n)$.

The sustained false-positive rate follows from the independence of the hash functions in each B_i, that is, if x is a false positive in B_i, one of its cells is marked dead; the probability that x is a false positive in B_{i+1} is $\le \varepsilon$. □

Extending to other AMQs. The above strategy can be used to construct an adaptive AMQ from any nonadaptive AMQ in which we can fix a false positive by marking (as dead) $O(1)$ elements of $\mathcal{S}$ in expectation and then inserting them to the next level.

This strategy works particularly well for single-hash nonadaptive AMQs that represent $\mathcal{S}$ by storing $h(\mathcal{S})$ losslessly. Construct a hierarchy of filters with decreasing false-positive rates as described above. Whenever the adversary discovers a false positive x that collides with

some set of elements Y stored in the ith level, mark $h(x)$ as dead and insert the elements of Y into the $(i+1)$st level. This requires $O(1)$ access to $\mathcal{S}$ in order to invert the hash function used at level i. In general, the new AMQ will have $\log\log n$ levels, and take the same space as the underlying nonadaptive AMQ plus lower order terms. It will support insertions and queries with the same cost as the underlying AMQ in expectation and a factor of at most $O(\log\log n)$ slower than the underlying AMQ with high probability.

However, if the underlying nonadaptive AMQ supports $O(1)$-time insertions and lookups, we can improve the performance of this construction to $O(\log^* n)$-time insertions and lookups. In particular, we set $\varepsilon_0 = \varepsilon$, as before. At level 1, we set $\varepsilon_1 = 2^{-\frac{\log(1/\varepsilon)}{\sqrt{\varepsilon}}} = \varepsilon^{\frac{1}{\sqrt{\varepsilon}}}$. For $i \geq 2$, we set $\varepsilon_i = \varepsilon^{1/(\varepsilon_{i-1})}$. Thus, $d = O(\log^* n)$ with high probability. As in the adaptive Bloom filter, we set the size of the ith filter by bounding $\mathbf{E}[n_i] \leq \varepsilon_{n_{i-1}}\mathbf{E}[n_{i-1}](1 + 1/\log n)$. That is, we set $m_i \leq n\varepsilon_{i-1}(1 + 1/\log n)(\log 1/\varepsilon_i + O(1))$. Thus the total space used is

$$\sum_i m_i = (n\log(1/\varepsilon) + O(n))(1 + 1/\log n)(1 + \sqrt{\varepsilon} + \varepsilon^{1/\sqrt{\varepsilon}} + \ldots) = n\log(1/\varepsilon) + O(n).$$

The following theorem summarizes the performance of this construction.

Theorem 9.29. *Given ε and a sequence of up to n queries, there exists a construction that turns a $O(1)$-expected time and $n\log(1/\varepsilon) + O(n)$ space nonadaptive AMQ, into a adaptive AMQ with a sustained false-positive rate of ε with high probability. This AMQ guarantees:*

- *$O(1)$ cost in expectation and $O(\log^* n)$ w.h.p. for inserts,*
- *$O(1)$ cost in expectation and $O(\log^* n)$ w.h.p. for lookups,*
- *$n\log(1/\varepsilon) + O(n)$ space w.h.p.,*
- *$O(1)$ accesses to the remote representation $\mathbf{R}$ of $\mathcal{S}$ for each false positive and each deletion.*

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] Eric Allender. The permanent requires large uniform threshold circuits. *Chicago Journal of Theoretical Computer Science*, 7:19, 1999.

[4] Eric Allender and Ulrich Hertrampf. On the power of uniform families of constant depth threshold circuits. In *International Symposium on Mathematical Foundations of Computer Science*, pages 158–164, 1990.

[5] Yuriy Arbitman, Moni Naor, and Gil Segev. Backyard cuckoo hashing: Constant worst-case operations with a succinct representation. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 787–796, 2010.

[6] Sanjeev Arora. How np got a new definition: a survey of probabilistically checkable proofs. *arXiv preprint cs/0304038*, 2003.

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

[8] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555, 1998.

[9] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM (JACM)*, 45(1):70–122, 1998.

[10] Pablo Daniel Azar and Silvio Micali. Rational proofs. In *Proceedings of the 44th Annual Symposium on Theory of Computing (STOC)*, pages 1017–1028, 2012.

[11] Pablo Daniel Azar and Silvio Micali. Super-efficient rational proofs. In *Proceedings of the 14th Annual Conference on Electronic Commerce (EC)*, pages 29–30, 2013.

[12] László Babai. Trading group theory for randomness. In *Proceedings of the 17th Annual Symposium on Theory of Computing (STOC)*, pages 421–429, 1985.

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

[14] 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.

[15] Richard Beigel. Bounded queries to SAT and the Boolean hierarchy. *Theoretical Computer Science*, 84(2):199–223, 1991.

[16] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 3(4):319–354, 1993.

[17] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 113–131, 1988.

[18] Udi Ben-Porat, Anat Bremler-Barr, Hanoch Levy, and Bernhard Plattner. On the vulnerability of hardware hash tables to sophisticated attacks. In *Proceedings of the 11th International Conference on Research in Networking*, volume 1, pages 135–148, 2012.

[19] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Short pcps verifiable in polylogarithmic time. In *Proceedings of the 20th Annual Conference on Computational Complexity (CCC)*, pages 120–134. IEEE, 2005.

[20] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. Robust pcps of proximity, shorter pcps, and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, 2006.

[21] Michael A Bender, Martin Farach-Colton, Mayank Goswami, Rob Johnson, Samuel McCauley, and Shikha Singh. Bloom filters, adaptivity, and the dictionary problem. *arXiv preprint arXiv:1711.01616*, 2017.

[22] Michael A Bender, Martin Farach-Colton, Rob Johnson, Russell Kraner, Bradley C Kuszmaul, Dzejla Medjedovic, Pablo Montes, Pradeep Shetty, Richard P Spillane, and Erez Zadok. Don't thrash: how to cache your hash on flash. *Proceedings of the VLDB Endowment*, 5(11):1627–1637, 2012.

[23] Nir Bitansky and Alessandro Chiesa. Succinct arguments from multi-prover interactive proofs and their efficiency benefits. In *Advances in Cryptology–CRYPTO 2012*, pages 255–272. 2012.

[24] Burton H Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[25] Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Proceedings of the Annual International Cryptology Conference*, pages 278–291, 1993.

[26] Andrew J Blumberg, Justin Thaler, Victor Vu, and Michael Walfish. Verifiable computation using multiple provers. *IACR Cryptology ePrint Archive*, 2014:846, 2014.

[27] Benjamin Braun, Ariel J Feldman, Zuocheng Ren, Srinath Setty, Andrew J Blumberg, and Michael Walfish. Verifying computations with state. In *Proceedings of the 24th Symposium on Operating Systems Principles*, pages 341–357, 2013.

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

[29] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[30] J Bruck, Jie Gao, and Anxiao Jiang. Weighted bloom filter. In *Proceedings of the International Symposium on Information Theory*, 2006.

[31] Harry Buhrman, Jim Kadin, and Thomas Thierauf. On functions computable with nonadaptive queries to np. In *Proceedings of the 9th Annual Structure in Complexity Theory Conference*, pages 43–52. IEEE, 1994.

[32] Harry Buhrman and Wim Van Dam. Quantum bounded query complexity. In *Proceedings of the 14th Annual Conference on Computational Complexity (CCC)*, pages 149–156, 1999.

[33] X. Cai, Y. Gui, and R. Johnson. Exploiting unix file-system races via algorithmic complexity attacks. In *Proceedings of the 30th Symposium on Security and Privacy*, 2009.

[34] Matteo Campanelli and Rosario Gennaro. Sequentially composable rational proofs. In *International Conference on Decision and Game Theory for Security*, pages 270–288, 2015.

[35] Ran Canetti, Ben Riva, and Guy N Rothblum. Practical delegation of computation using multiple servers. In *Proceedings of the 18th Conference on Computer and Communications Security*, pages 445–454, 2011.

[36] Ran Canetti, Ben Riva, and Guy N Rothblum. Two 1-round protocols for delegation of computation. In *International Conference on Information Theoretic Security*, pages 37–61, 2012.

[37] Ran Canetti, Ben Riva, and Guy N Rothblum. Refereed delegation of computation. *Information and Computation*, 226:16–36, 2013.

[38] Larry Carter, Robert Floyd, John Gill, George Markowsky, and Mark Wegman. Exact and approximate membership testers. In *Proceedings of the 10th Annual Symposium on Theory of Computing (STOC)*, pages 59–65, 1978.

[39] 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 (CCC)*, pages 217–243, 2015.

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

[41] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *Transactions on Computer Systems*, 26(2):4, 2008.

[42] Bernard Chazelle, Joe Kilian, Ronitt Rubinfeld, and Ayellet Tal. The bloomier filter: an efficient data structure for static support lookup tables. In *Proceedings of the 15th Annual Symposium on Discrete Algorithms (SODA)*, pages 30–39, 2004.

[43] Jing Chen, Samuel McCauley, and Shikha Singh. Rational proofs with multiple provers. In *Proceedings of the 7th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 237–248, 2016.

[44] Jing Chen, Samuel McCauley, and Shikha Singh. Rational proofs with non-cooperative provers. *arXiv preprint arXiv:1708.00521*, 2017.

[45] Zhi-Zhong Chen and Seinosuke Toda. The complexity of selecting maximal solutions. *Information and Computation*, 119(2):231–239, 1995.

[46] Saar Cohen and Yossi Matias. Spectral bloom filters. In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 241–252, 2003.

[47] Anne Condon and Richard Ladner. Interactive proof systems with polynomially bounded strategies. *Journal of Computer and System Sciences*, 50(3):506–518, 1995.

[48] John Conlisk. Why bounded rationality? *Journal of Economic Literature*, 34(2):669–700, 1996.

[49] 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 (ITCS)*, pages 90–112, 2012.

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

[51] Scott A. Crosby and Dan S. Wallach. Denial of service via algorithmic complexity attacks. In *Proceedings of the 12th Conference on USENIX Security Symposium*, volume 12, 2003.

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

[53] Erik Demaine, Friedhelm der Heide, Rasmus Pagh, and Mihai Pătraşcu. On dynamic dictionaries using little space. In *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN)*, pages 349–361, 2006.

[54] Fan Deng and Davood Rafiei. Approximately detecting duplicates for streaming data using stable bloom filters. In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 25–36, 2006.

[55] Sarang Dharmapurikar, Praveen Krishnamurthy, Todd Sproull, and John Lockwood. Deep packet inspection using parallel bloom filters. In *Proceedings of the 11th Symposium on High Performance Interconnects*, pages 44–51, 2003.

[56] Benoit Donnet, Bruno Baynat, and Timur Friedman. Improving retouched bloom filter for trading off selected false positives against false negatives. *Computer Networks*, 54(18):3373–3387, 2010.

[57] John Duggan. An extensive form solution to the adverse selection problem in principal/multi-agent environments. *Review of Economic Design*, 3(2):167–191, 1998.

[58] David Eppstein and Michael T Goodrich. Straggler identification in round-trip data streams via newton's identities and invertible bloom filters. *Transactions on Knowledge and Data Engineering*, 23(2):297–306, 2011.

[59] David Eppstein, Michael T Goodrich, Michael Mitzenmacher, and Manuel R Torres. 2-3 cuckoo filters for faster triangle listing and set intersection. In *Proceedings of the 36th SIGMOD Symposium on Principles of Database Systems (PODS)*, pages 247–260. ACM, 2017.

[60] Bin Fan, Dave G Andersen, Michael Kaminsky, and Michael D Mitzenmacher. Cuckoo filter: Practically better than Bloom. In *Proceedings of the 10th International on Conference on Emerging Networking Experiments and Technologies*, pages 75–88, 2014.

[61] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z Broder. Summary cache: a scalable wide-area web cache sharing protocol. *Transactions on Networking*, 8(3):281–293, 2000.

[62] Uri Feige and Joe Kilian. Two prover protocols: low error at affordable rates. In *Proceedings of the 26th Annual Symposium on Theory of Computing (STOC)*, pages 172–183, 1994.

[63] Uriel Feige and Joe Kilian. Making games short. In *Proceedings of the 29th Annual Symposium On Theory of Computing (STOC)*, pages 506–516, 1997.

[64] Uriel Feige and László Lovász. Two-prover one-round proof systems: their power and their problems. In *Proceedings of the 24th Annual Symposium on Theory of Computing (STOC)*, pages 733–744, 1992.

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

[66] Uriel Feige, Adi Shamir, and Moshe Tennenholtz. The noisy oracle problem. In *Proceedings of the 10th Annual Conference on Advances in Cryptology (CRYPTO)*, pages 284–296, 1990.

[67] Joan Feigenbaum, Daphne Koller, and Peter Shor. A game-theoretic classification of interactive complexity classes. In *Proceedings of 10th Annual Structure in Complexity Theory Conference*, pages 227–237, 1995.

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

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

[70] Nicola Gatti and Fabio Panozzo. New results on the verification of nash refinements for extensive-form games. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 2, pages 813–820, 2012.

[71] Thomas Gerbet, Amrit Kumar, and Cédric Lauradoux. The power of evil choices in bloom filters. In *Proceedings of the 45th Annual International Conference on Dependable Systems and Networks*, pages 101–112, 2015.

[72] Jacob Glazer and Motty Perry. Virtual implementation in backwards induction. *Games and Economic Behavior*, 15(1):27–32, 1996.

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

[74] Oded Goldreich, RL Graham, and B Korte. Modern cryptography, probabilistic proofs and pseudorandomness., 1998.

[75] 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.

[76] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1), 1989.

[77] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual Symposium on Theory of Computing (STOC)*, pages 113–122, 2008.

[78] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[79] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual Symposium on Theory of Computing (STOC)*, pages 59–68, 1986.

[80] Siyao Guo, Pavel Hubáček, Alon Rosen, and Margarita Vald. Rational arguments: single round delegation with sublinear verification. In *Proceedings of the 5th Annual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 523–540, 2014.

[81] Siyao Guo, Pavel Hubáček, Alon Rosen, and Margarita Vald. Rational sumchecks. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 319–351, 2016.

[82] Tom Gur and Ron D Rothblum. Non-interactive proofs of proximity. In *Proceedings of the 6th Annnual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 133–142, 2015.

[83] Johan Haastad and Subhash Khot. Query efficient pcps with perfect completeness. *Theory of Computing*, 1(1):119–148, 2005.

[84] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993.

[85] Joseph Y Halpern. Beyond nash equilibrium: Solution concepts for the 21st century. In *Proceedings of the 27th Symposium on Principles of Distributed Computing (PODC)*, pages 1–10, 2008.

[86] Kristoffer Arnsfelt Hansen, Peter Bro Miltersen, and Troels Bjerre Sørensen. The computational complexity of trembling hand perfection and other equilibrium refinements. In *Proceedings of the International Symposium on Algorithmic Game Theory (SAGT)*, pages 198–209, 2010.

[87] Moritz Hardt and David P Woodruff. How robust are linear sketches to adaptive inputs? In *Proceedings of the 45th Annual Symposium on Theory of Computing (STOC)*, pages 121–130, 2013.

[88] Ebbe Hendon, Hans Jørgen Jacobsen, and Birgitte Sloth. The one-shot-deviation principle for sequential rationality. *Games and Economic Behavior*, 12(2):274–282, 1996.

[89] William Hesse, Eric Allender, and David A Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002.

[90] Paul Hunter, Patricia Bouyer, Nicolas Markey, Joël Ouaknine, and James Worrell. Computing rational radical sums in uniform tc0. In *Proceedings of the Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2010.

[91] Keita Inasawa and Kenji Yasunaga. Rational proofs against rational verifiers. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 100(11):2392–2397, 2017.

[92] Birgit Jenner and Jacobo Torán. Computing functions with parallel queries to np. In *Proceedings of the 8th Annual Structure in Complexity Theory Conference*, pages 280–291. IEEE, 1993.

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

[94] Suraiya Khan and Issa Traore. A prevention model for algorithmic complexity attacks. In *Proceedings of the 2nd International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 160–173, 2005.

[95] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732, 1992.

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

[97] Gillat Kol and Ran Raz. Competing provers protocols for circuit evaluation. In *Proceedings of the 4th Annual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 473–484, 2013.

[98] 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.

[99] Mark W Krentel. The complexity of optimization problems. *Journal of computer and system sciences*, 36(3):490–509, 1988.

[100] David M Kreps. *A course in microeconomic theory*, volume 41. JSTOR, 1990.

[101] David M Kreps and Robert Wilson. Sequential equilibria. *Econometrica: Journal of the Econometric Society*, pages 863–894, 1982.

[102] Rafael P Laufer, Pedro B Velloso, Daniel De O Cunha, Igor M Moraes, Marco DD Bicudo, and Otto Carlos MB Duarte. A new IP traceback system against distributed denial-of-service attacks. In *Proceedings of the 12th International Conference on Telecommunications*, 2005.

[103] Kang Li and Zhenyu Zhong. Fast statistical spam filter by approximate classifications. In *Proceedings of the SIGMETRICS Performance Evaluation Review*, volume 34, pages 347–358, 2006.

[104] Shachar Lovett and Ely Porat. A lower bound for dynamic approximate membership data structures. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*, pages 797–804, 2010.

[105] 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.

[106] Rudolf Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–136, 1979.

[107] Ilya Mironov, Moni Naor, and Gil Segev. Sketching in adversarial environments. *SIAM Journal on Computing*, 40(6):1845–1870, 2011.

[108] Michael Mitzenmacher. Compressed bloom filters. *Transactions on Networking*, 10(5):604–612, 2002.

[109] Michael Mitzenmacher, Salvatore Pontarelli, and Pedro Reviriego. Adaptive cuckoo filters. In *Proceedings of the 20th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 36–47, 2018.

[110] Michael Mitzenmacher and Eli Upfal. *Probability and Computing - Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[111] James K. Mullin. Optimal semijoins for distributed database systems. *Transactions on Software Engineering*, 16(5):558–560, 1990.

[112] Moni Naor and Eylon Yogev. Sliding bloom filters. In *Proceedings of the International Symposium on Algorithms and Computation*, pages 513–523, 2013.

[113] Moni Naor and Eylon Yogev. Bloom filters in adversarial environments. In *Proceedings of the Annual Cryptology Conference*, pages 565–584. Springer, 2015.

[114] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35(1-2):166–196, 2001.

[115] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[116] Pekka Orponen. Complexity classes of alternating machines with oracles. In *Automata, Languages and Programming*, pages 573–584. Springer, 1983.

[117] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[118] Anna Pagh, Rasmus Pagh, and S Srinivasa Rao. An optimal bloom filter replacement. In *Proceedings of the 16th Annual Symposium on Discrete Algorithms (SODA)*, pages 823–829, 2005.

[119] Prashant Pandey, Michael A. Bender, Rob Johnson, and Rob Patro. A general-purpose counting filter: Making every bit count. In *Proceedings of the SIGMOD International Conference on Management of Data*, pages 775–787, 2017.

[120] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *Proceedings of the Symposium on Security and Privacy*, pages 238–252, 2013.

[121] Gary L Peterson and John H Reif. Multiple-person alternation. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 348–363, 1979.

[122] Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

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

[124] Guy N Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the 45th Annual Symposium on Theory of Computing (STOC)*, pages 793–802, 2013.

[125] Ariel Rubinstein. *Modeling bounded rationality*. MIT press, 1998.

[126] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In *Proceedings of the 8th European Conference on Computer Systems*, pages 71–84, 2013.

[127] Srinath TV Setty, Richard McPherson, Andrew J Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, number 9, page 17, 2012.

[128] Srinath TV Setty, Victor Vu, Nikhil Panpalia, Benjamin Braun, Andrew J Blumberg, and Michael Walfish. Taking proof-based verified computation a few steps closer to practicality. In *Proceedings of the USENIX Security Symposium*, pages 253–268, 2012.

[129] Adi Shamir. IP = PSPACE. *Journal of the ACM (JACM)*, 39(4):869–877, 1992.

[130] Janos Simon. On some central problems in computational complexity. Technical report, Cornell University, 1975.

[131] Madhu Sudan. Probabilistically checkable proofs. *Communications of the ACM*, 52(3):76–84, 2009.

[132] Sasu Tarkoma, Christian Esteve Rothenberg, Eemil Lagerspetz, et al. Theory and practice of bloom filters for distributed systems. *Communications Surveys and Tutorials*, 14(1):131–155, 2012.

[133] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Advances in Cryptology–CRYPTO 2013*, pages 71–89. 2013.

[134] Justin Thaler, Mike Roberts, Michael Mitzenmacher, and Hanspeter Pfister. Verifiable computation with massively parallel interactive proofs. In *Proceedings of the 4th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2012.

[135] Hannu Vartiainen. Subgame perfect implementation of voting rules via randomized mechanisms. *Social Choice and Welfare*, 29(3):353–367, 2007.

[136] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human Factors in Computing System*, pages 319–326, 2004.

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

[138] Victor Vu, Srinath Setty, Andrew J Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *Proceedings of the Symposium on Security and Privacy*, pages 223–237, 2013.

[139] Klaus W Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986.

[140] Klaus W Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[141] Michael Walfish and Andrew J Blumberg. Verifying computations without reexecuting them. *Communications of the ACM*, 58(2):74–84, 2015.

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

[143] Jeff Yan and Pook Leong Cho. Enhancing collaborative spam detection with bloom filters. In *Proceedings of the 22nd Annual Conference on Computer Security Applications*, volume 6, pages 414–428, 2006.

[144] Linfeng Zhang and Yong Guan. Detecting click fraud in pay-per-click streams of online advertising networks. In *Proceedings of the 28th International Conference on Distributed Computing Systems*, pages 77–84, 2008.

[145] Yihua Zhang and Marina Blanton. Efficient secure and verifiable outsourcing of matrix multiplications. In *Proceeding of the International Conference on Information Security*, pages 158–178, 2014.

[146] Ming Zhong, Pin Lu, Kai Shen, and Joel Seiferas. Optimizing data popularity conscious bloom filters. In *Proceedings of the 27th Symposium on Principles of Distributed Computing (PODC)*, pages 355–364. ACM, 2008.

[147] Yifeng Zhu, Hong Jiang, and Jun Wang. Hierarchical bloom filter arrays: a novel, scalable metadata management system for large cluster-based storage. In *Proceedings of the International Conference on Cluster Computing*, pages 165–174, 2004.