Super-Efficient Rational Proofs

by

Pablo Daniel Azar

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
August 12, 2014

Signature redacted

Certified by .....
Silvio Micali
Professor of Computer Science
Thesis Supervisor

Signature redacted

Accepted by .....
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Theses

Super-Efficient Rational Proofs

by

Pablo Daniel Azar

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

Information asymmetry is a central problem in both computer science and economics. In many fundamental problems, an uninformed principal wants to obtain some knowledge from an untrusted expert. This models several real-world situations, such as a manager's relation with her employees, or the delegation of computational tasks to workers over the internet.

Because the expert is untrusted, the principal needs some guarantee that the provided knowledge is correct. In computer science, this guarantee is usually provided via a *proof*, which the principal can verify. Thus, a dishonest expert will always get caught and penalized. In many economic settings, the guarantee that the knowledge is correct is usually provided via *incentives*. That is, a game is played between expert and principal such that the expert maximizes her utility by being honest.

A rational proof is an interactive proof where the prover, Merlin, is neither honest nor malicious, but rational. That is, Merlin acts in order to maximize his own utility. I previously introduced and studied Rational Proofs when the verifier, Arthur, is a probabilistic polynomial-time machine [3].

In this thesis, I characterize *super-efficient rational proofs*, that is, rational proofs where Arthur runs in logarithmic time. These new rational proofs are very practical. Not only are they much faster than their classical analogues, but they also provide very tangible incentives for the expert to be honest. Arthur only needs a *polynomial-size budget*, yet he can penalize Merlin by a large quantity if he deviates from the truth.

Thesis Supervisor: Silvio Micali
Title: Professor of Computer Science

Acknowledgments

So many people helped me during my time at MIT CSAIL. I am grateful to Chuck, Nina, JoAnne, Be, Holly and Linda for helping me navigate the administrative side of CSAIL and the EECS department, as well as for providing an endless supply of candy to fuel the writing of this thesis.

I am grateful to my advisor Silvio Micali for helping me to develop my problem solving skills, in all areas of life.

I am grateful to Shafi Goldwasser and Andrew Lo, who together with Silvio formed my thesis comittee, and supported me in both good times and bad times.

I am grateful to my family, Hector, Sara, Gie, Lisa, Jose, Deborah and Jasper, without whose encouragement and support this work would not be possible.

And most of all, I am grateful to my wonderful wife, Stephanie Liem Azar, whose endless supply of love, encouragement, jokes you cannot put in a thesis acknowledgement section, passion and laughter were the true fuel for this thesis and beyond.

Contents

1 Introduction **9**

2 Related Work **13**

3 Preliminaries **19**

4 Rational Proof Characterization of TC^0 **25**

5 Rational Proof Characterization of $P^{\|NP}$ **33**

6 Rational Proof Characterization of $P^{\|MA}$ **41**

7 Conclusion **45**

Chapter 1

Introduction

Exchanging knowledge for money is central in a market economy. Two concrete examples of knowledge being sold are *Amazon's Mechanical Turk*, where individuals are paid to solve problems that are impractical to automate, and *cloud computing* where computational resources are rented from far away servers. In order to fully understand the potential of these markets, we need to develop a theory of delegation of computation to rational agents.

A classical way to delegate computation is by using *interactive proofs* [5] [19]. An interactive proof system guarantees that (1) there is a way for a prover (colorfully called Merlin) to convince a skeptical verifier (colorfully called Arthur) that a theorem is true, but (2) there is no way (with overwhelming probability) to convince Arthur that a false theorem is true. Accordingly, Arthur can agree to pay Merlin a fixed amount of money (e.g., a thousand dollars) if Merlin manages to convince him that a given computation is correct, and zero dollars otherwise. Interactive proofs can be much faster than traditional proofs which use no interaction. Unfortunately, they are not always as efficient as we would like them to be. This is so because Arthur needs to *verify* whether Merlin is being honest or dishonest.

Rational proofs [3] are an alternative to interactive proofs where Merlin is neither honest nor arbitrarily malicious. Instead, he is economically motivated and seeks to maximize a monetary reward. Thus, a rational proof system needs to ensure that Merlin maximizes this reward if and only if he honestly reports the correct

answer to Arthur's problem. In such proofs, even though Arthur cannot run the computation himself and cannot even verify whether the provided result is correct, he can surprisingly come up with an easy to evaluate reward function—to be evaluated on Merlin's answer—that is only maximized by Merlin whenever he gives a correct answer.

By considering rational provers, we can significantly improve the efficiency of interactive proofs. Indeed, in this thesis, we present new rational proofs with two strong properties

1. Super-Efficiency. Most of the proofs in this thesis involve a *logarithmic time verifier*. Furthermore, our super-efficient rational proofs are *prover feasible* [24]. That is, all provers (including unbounded ones) are incentivized to give an honest answer and the amount of computation required by an honest prover is no larger than the amount required to solve the problem at hand.

2. Bounded Budget. All the rational proofs we present (1) enable Arthur to reward Merlin with a *polynomial* (in the input length) budget and (2) ensure that Merlin suffers *polynomial* losses from giving a wrong answer. This gives Merlin very strong incentives to be truthful, and is an improvement over previous rational proofs where Merlin would only lose an exponentially small amount if he deviated from the truth.

Budgets are important when Merlin has costly computation. In particular, assume that every computational step in an algorithm costs Merlin \$1. If Merlin can run said algorithm in a polynomial number of steps (say n^c where n is the size of the algorithm's input), his total cost would be $\$n^c$. If Merlin does not get compensated for his costs, then he could presumably make a larger profit by giving a default answer that does not cost him anything to compute, such as 0. By giving this answer, he gets some reward and does not have to pay any computational cost. Thus, a properly structured rational proof with polynomial budget must make the difference in reward between a correct answer and an incorrect answer larger than Merlin's computational cost of computing the right answer.

Super-efficiency is also important, as we want to make the verifier exert as little effort as possible. Even when Merlin is a polynomial-time machine, he can still solve problems that a super-efficient Arthur running in logarithmic time cannot solve. For many problems we study in this thesis, there is no known classical proof where Arthur runs in $O(\log n)$ time. However, even when Arthur can run polynomial time algorithms, it is possible that Merlin holds a "trade secret" that allows him to solve problems than Arthur cannot solve. For example, Merlin may have a quantum computer which allows him to factor numbers. Another example is that Merlin may have found an algorithm for the graph non-isomorphism problem. Instead of revealing his algorithm in full, Merlin can sell his knowledge as a service to Arthur using a rational proof. We remark that for these two problems (factoring and graph non-isomorphism), there exist classical interactive proofs which allow Merlin to convince Arthur that his work is correct. However, these proofs require Arthur to use polynomial time to verify the proofs. As mentioned before, all our rational proofs (including for NP and *co-NP* problems) only require Arthur to use $O(\log n)$ time to compute Merlin's reward.

Our Results In this thesis we study rational proofs where Arthur only uses $O(\log n)$ time and Merlin's reward sensitivity is polynomial in n, where n is the size of the input. This allows us to work in a setting where all possible losses from deviating from the truth are larger than \$1000, and where Arthur's budget is a polynomial function $p(n)$ of the problem size n.[1]

We focus on three complexity classes:

- Uniform TC^0. This class is powerful enough to include ***all multi-variable rational functions of constant degree [20].*** More formally, it is the set of languages decidable by uniform, polynomial-size, constant-depth threshold circuit families.
- $P^{\|NP}$. Finding the largest clique in a graph, finding the maximum number of satisfying assignments in a boolean formula, and finding the maximum of an

[1]Note that, if we scale rewards by a large enough polynomial factor $q(n)$, we can make Merlin's losses from lying as large as $\$1000q(n)$, while still keeping Arthur's budget polynomial in n. For convenience of notation, we will in most cases normalize the budget so that Merlin loses at least \$1000 when he is dishonest.

arbitrary function $f : \{0,1\}^{n^{O(1)}} \to \{1,...,n^{O(1)}\}$ can be reduced to decision problems in $P^{\|NP}$. More formally, this class is the set of languages decidable by a polynomial time machine that can make *parallel* queries to an NP oracle.

Let us emphasize that this class also includes all problems in *co-NP*, which do not admit constant-round classical interactive proofs even when Arthur is allowed to use polynomial time.[2] In contrast, the rational proofs that we give for this class, will always have *a single round of interaction.*

- $P^{\|MA}$. This is the analogue of $P^{\|NP}$ where instead of making queries to an NP oracle, we can make queries to an even "more powerful" oracle.

In particular, we prove that

1. *Uniform* TC^0 coincides with the set of languages L that admit a rational proof with an $O(\log n)$-time Arthur, $O(\log n)$ communication, a constant number of rounds and a polynomial-size budget.
2. $P^{\|NP}$ coincides with the set of languages having a rational proof with an $O(\log n)$-time Arthur, $poly(n)$ communication, one round, and a polynomial-size budget.
3. $P^{\|MA}$ coincides with the set of languages having a rational proof with a poly-time Arthur, $poly(n)$ communication, one round, and a polynomial-size budget.

Remarks When Arthur has logarithmic time, he can only query a few bits of Merlin's messages via random access. This is the same model of interaction as that given in Probabilistically Checkable Proofs (PCPs). Our second result thus says that "*rational PCPs*" can be much more efficient than traditional PCPs. Indeed, in classical PCPs the verifier —although reading very few of the bits transmitted by the prover— runs in time at least linear in n, whereas our verifier runs in time $O(\log n)$.

Our third result gives a limit on the power of polynomial budget rational proofs, and makes explicit the intuition that, in order to give rational proofs for $\#P$ or higher classes one needs an exponential budget (under standard complexity theoretic assumptions).

[2]More precisely, if *co-NP* admitted constant round interactive proofs, the polynomial hierarchy would collapse. This is not believed to be true under standard complexity theoretic assumptions.

Chapter 2

Related Work

Rational Proofs for high complexity classes A previous paper [3] introduced rational proofs and showed that the complexity class $\#P$ admitted one-round rational proofs. We also showed that the languages admitting constant round rational proofs are exactly those in the counting hierarchy CH. In this thesis, I extend these previous results by giving characterizations of (relatively) lower complexity classes such as TC^0, $P^{||NP}$ and $P^{||MA}$. Our previous rational proofs for CH required Merlin to be sensitive to exponentially small losses in reward, and they required Arthur to be a polynomial time machine. In contrast, the new rational proofs that I present in this thesis only require Merlin to be sensitive to polynomially large differences in reward, and (for TC^0 and $P^{||NP}$) only require Arthur to do logarithmic time computations.

Interactive Proofs Interactive Proofs were introduced by Goldwasser, Micali and Rackoff [19] and by Babai and Moran [5], from whom we have borrowed the Arthur-Merlin terminology. In both these papers, Arthur is assumed to be a polynomial time machine. Many ways of weakening Arthur's computational requirements have been considered, including, but not limited to, the work of Dwork and Stockmeyer [13][14], who have studied interactive proofs with verifiers that are probabilistic finite state automata, and the work of Condon and Lipton [11], who study interactive proofs for space bounded verifiers.

A closer point of comparison for our proofs of circuit evaluation are Goldwasser,

Kalai, and Rothblum's notions and techniques for *interactive proofs for muggles.* [27]. Their main result gives interactive proofs for uniform NC circuits of size $S(n)$ and depth $d(n)$, where the verifier acts in time $(n + d(n)) \cdot polylog(S(n))$ and the prover acts in time $poly(S(n))$. In contrast, by switching to a rational model, we can give interactive proofs for more general uniform TC circuits, and reduce the verifier time to $O(d(n)\log(n))$.

For the rational proof characterization of $P^{||NP}$, the relevant point of comparison is the literature on probabilistically checkable proofs (PCPs) [1][16][2], and holographic proofs [4]. Here, in essence, to prove that an n-bit string x belongs to an NP language L, the prover provides the verifier with a $poly(n)$ bit string, which is sampled at a constant number of locations by a verifier who then performs a polynomial time computation.[1] As already mentioned in the introduction, in our case, Arthur not only reads only $O(\log n)$ bits, but also performs only $O(\log n)$ computation.

There are other alternative models of PCPs which have weak verifiers. Batu, Rubinfeld and White [8] and Ergun, Kumar and Rubinfeld [15] study a variant of $PCPs$, where the solutions to the given problems are only *approximate solutions.* In this setting, and for many problems of interest, they give PCPs which can be verified in polylogarithmic time.

In addition, the notion of committing the computation of a circuit, and then allowing the verifier to locally de-commit and check it, is deeply rooted in the cryptographic literature. In particular, CS Proofs [21, 24] present a variant of this approach that allows for just a logarithmic amount of communication, but still requiring a polynomial amount of computation from the verifier. We also remark that, by switching to the rational model, we do not need to make any cryptographic assumptions.

Optimization Problems One of the main results in this thesis is a characterization of $P^{||NP}$ in terms of super-efficient rational proofs. There are other characterizations of $P^{||NP}$, especially in terms of optimization problems. First, we remark that it is well known that $P^{||NP} = P^{NP[O(\log n)]}$, the class of languages decidable in polynomial

[1] It is important that this polynomial only depends on the input size $|x|$, but not on the language L for which the proof is given.

time with $O(\log n)$ queries to an NP oracle. Furthermore, Krentel [23] shows that the corresponding function class $FP^{NP[O(\log n)]}$ is equal to $OptP[O(\log n)] = \{y(\cdot) : y(x) = \text{argmax}_y f(x,y)$ where $f(x,y)$ is computable in polynomial time and can be written with $z(|x|)$ bits$\}$. That is, for any function $g(x) \in FP^{NP[O(\log n)]}$, there exists another function $f(x,y)$ computable in polynomial time and taking polynomially many different values such that $g(x) = \text{argmax}_y f(x,y)$.

One immediate consequence of this characterization of $FP^{NP[O(\log n)]}$ is that we can give the following informal rational proof for it. Given a function g which we want to evaluate on input x, ask Merlin for a value y, allegedly equal to $g(x)$. In order to incentivize Merlin to give the correct answer, give him a reward equal to $f(x,y)$. This is a rational proof with polynomial budget, and can be used to give polynomial budget rational proofs for $P^{\|NP}$. However, ***it is not super-efficient.*** Arthur needs to evaluate the function $f(x,y)$, which can take polynomial time. While the above rational proof is simple, and recasts optimization problems as a tool to obtain trust from incentives, we also give a more sophisticated rational proof for $P^{\|NP}$, one which only requires Arthur to act in logarithmic time, an exponential savings in verifier time.

Game-Theoretic Characterizations of Complexity Classes Rational proofs are not the first characterization of complexity classes in terms of the actions of economic agents. Most of the existing work focuses on the complexity of finding properties of games with two or more players, each of whom has at least polynomial time to act. In contrast, the results in this thesis focus on an honest logarithmic time referee (Arthur) who interacts with a single utility-maximizing agent (Merlin). Furthermore, and to the best of my knowledge, these results are the first to give game-theoretic characterizations of "low" complexity classes such as TC^0. In the following paragraphs, we give a brief (and by no means exhaustive) summary of work in this area.

The idea of a referee interacting with utility-maximizing players has been explored by Feige and Kilian [17]. They define the class $RG(k)$ of languages that can be decided

by a polynomial-time verifier who "pits two experts against each other." That is, one prover is honest, the other one is dishonest, and the referee needs to determine which one's which. This can be interpreted as deciding which player is the winner in a zero-sum game with k rounds.

Kol and Raz [22] combine the refereed games framework with interactive proofs for muggles. That is, they give two-prover proofs for circuits and delegation of computation, where one prover is honest and the other one dishonest. Their results imply that for a uniform NC circuit of depth d and size S, one can give an interactive proof with r rounds where the verifier complexity is $d^{\frac{1}{r}}polylog(S)$.

Feigenbaum, Koller and Shor [18] give a framework to characterize complexity classes based on two-player games with imperfect information and imperfect recall. They show that languages in EXP and $co\text{-}NEXP$ can be characterized as games where players have imperfect information (and one player has imperfect recall). They also apply their framework to characterizations of $PSPACE$ in terms of games.

As mentioned above, all of the work on refereed games and debates involves a referee encouraging two expert to participate in a zero-sum game, whereas the rational proofs model concerns only one expert. A closer point of comparison is Papadimitriou's characterization of PSPACE in terms of games against nature [25]. He introduces the problem of stochastic satisfiability, where a player and "nature" alternate assigning boolean variables in a random formula (with the player seeking to make the formula satisfied), and shows that this problem is $PSPACE$-complete.

One can also give game-theoretic characterizations of complexity classes in terms of properties of non-zero sum games. Schoenebeck and Vadhan [26] give characterizations of $P^{\#P}$, $co\text{-}NP^{\#P}$ and $co\text{-}NP$ in terms of deciding whether a profile of strategies is an exact equilibria in games defined by succinct boolean circuits. The problem of deciding whether a profile of strategies is an approximate Nash equilibrium characterizes classes such as P^{BPP} and $co\text{-}NP^{BPP}$. Other game-theoretic problems on concise games characterize different classes such as $NEXP$ and MA. We refer the reader to their work for more details. Daskalakis, Goldberg and Papadimitriou [12] show that finding an ϵ-Nash equilibrium on graphical games with 4 players is complete for the

complexity class PPAD. Chen and Deng [10] extend this result to graphical games with 2 players.

Chapter 3

Preliminaries

Strings and Circuits Given a string $x \in \{0,1\}^*$, $|x|$ represents the description length of x. A circuit family $\{C_n\}_{n=1}^{\infty}$ is a sequence of boolean circuits such that $C_n : \{0,1\}^n \to \{0,1\}$.

DLOGTIME Uniformity Because Arthur will be computationally limited, he may not be able to handle arbitrary non-uniform circuits. This is why we will require our circuits families to be uniform. Traditionally, a circuit family is said to be DLOGTIME-uniform if the connection language

$$\{(1^n, g, h, i) : h \text{ is the } i^{th} \text{ input gate to } g \text{ in } C_n\} \cup \{(1^n, g, t) : g \text{ is a gate of type } t \text{ in } C_n\}$$

is decidable in logarithmic time [7]. This is a decision version of uniformity. We will use a search variant of this definition, that enables us to make our results tight. (For notational convenience we actually treat output and input wires as gates, so that all wires run from gates to gates. A gate representing the i^{th} input x_i takes no arguments and produces the output x_i.)

Definition 1. *Let $t : \mathbb{N} \to \mathbb{N}$ be a time-constructible function. A circuit family $\{C_n\}_{n=1}^{\infty}$ is $t(n)$-search uniform if there exists a Turing Machine M running in time $t(n)$ such that*

- *On input $(1^n, OUTPUT)$, M outputs a pointer to the output gate of C_n.*

- *On input* $(1^n, g, i)$*, where* g *is a pointer to a gate and* i *is an integer,* M *outputs a pointer to the* i^{th} *input of gate* g*.*

- *On input* $(1^n, g)$*, where* g *is a pointer to a gate,* M *outputs the type of gate* g*. The types of possible gates are* $AND, OR, NOT, THRESHOLD$ *and* $INPUT$*. Furthermore, if* g *is an input gate then* M *outputs which input* x_i *it corresponds to.*

We denote by $Search - Uniform - TC_d$ the class of $DLOGTIME$-Search-Uniform threshold circuits of depth d and by $Search{-}Uniform{-}TC^0$ the class of $DLOGTIME$-Search-Uniform threshold circuits of constant depth.

Rational Proofs for Circuits Given an input x of length n, and a circuit $C : \{0,1\}^n \to \{0,1\}$, we can define a rational proof that $C(x) = y$. Informally, Arthur and Merlin perform several rounds of interaction. After this interaction is complete, Arthur should be able to

1. Quickly compute the value of $C(x)$.

2. Quickly compute a reward $R(x, \mathcal{T})$ for Merlin which depends on the input x, and the transcript $\mathcal{T}$ from Merlin and Arthur's communication.

Merlin should be incentivized to always act truthfully. That is, if the protocol up to round k has followed a truthful execution, then Merlin will maximize his expected reward in the future by following the protocol truthfully at round $k+1$.

More formally, we can define Interactive Protocols and resource-bounded rational interactive proofs as follows.

Definition 2 (Interactive Protocols)**.** *A* k*-round interactive protocol consists of two randomized functions* $P, V : \{0,1\}^* \to \{0,1\}^*$*. Given an input* x *and a round* i*, they define the transcript* $\mathcal{T}_i$*, the prover's view* $\mathcal{P}_i$*, the verifier's view* $\mathcal{V}_i$*, the prover's message* a_i *and the verifier's message* b_i *as follows:*

Set $\mathcal{T}_0, \mathcal{V}_0, \mathcal{P}_0 \equiv \{x\}$*. Then, for* $i = 1$ *to* k*,*

- $\mathcal{P}_i \triangleq (\mathcal{P}_{i-1}, \mathcal{T}_{i-1}, r_i)$, *where* r_i *is a sequence of random coin tosses, and* $a_i \triangleq P(\mathcal{P}_i)$.

- $\mathcal{V}_i \triangleq (\mathcal{V}_{i-1}, \mathcal{T}_{i-1}, s_i, a_i)$, *where* s_i *is a sequence of random coin tosses, and* $b_i \triangleq V(\mathcal{V}_i)$.

- $\mathcal{T}_i \triangleq (\mathcal{T}_{i-1}, a_i, b_i)$.

We use the term honest transcript of (P, V) *on input* x *for any* $\mathcal{T}_k$ *computed using this protocol on input* x.

The protocol is public-coin if, for every round i, *Arthur's message* b_i *at round* i *consists of just his random coins* s_i. *Given fixed coin tosses* $b = (b_1, ..., b_k)$, *and an arbitrary prover* P', *we define the* transcript generated by P' and b *to be*

$$\mathcal{T}(P', b) = (P'(x), b_1, P'(x, b_1), ..., P'(x, P'(x), ..., b_{k-1}))$$

Definition 3 (Rational Interactive Proof). *Let* $t, cc, rc, b : \{0,1\}^* \to \mathbb{N}$ *be non-decreasing functions. A circuit family* $\{C_n\}_{n \in \mathbb{N}}$ *has a* rational interactive proof with time complexity t, communication complexity cc, round complexity rc, and budget b *(for short, a (t,cc,rc,b)-Rational Interactive Proof) if there exists a public coin interactive protocol* (P, V), *a reward function* $R(\cdot, \cdot)$ *and a predicate* $\pi(\cdot, \cdot)$ *such that, for every input* x

- *The verifier is a machine that makes at most* $t(|x|)$ *computational steps in each round.*

- *The number of bits exchanged in each round is at most* $cc(|x|)$.

- *The number of rounds is* $k = rc(|x|)$.

- *The reward* $R(x, \mathcal{T})$ *and the predicate* $\pi(x, \mathcal{T})$ *can be computed in time* $k \cdot t(|x|)$.

- *If* $\mathcal{T}_k$ *is an honest transcript given* P, V *and* x, *then* $\pi(x, \mathcal{T}) = C_n(x)$.

- *Given a prover P', the expected reward of P',*

$$R(x, P') \triangleq \mathbb{E}_b[R(x, \mathcal{T}(P', b))],$$

is an integer in $[0, b(|x|)]$.

- *For any $P' \neq P$, we have $R(x, P) - R(x, P') > 1000$.*

Our first four conditions require that a rational proof has limited time complexity, communication complexity, and
round complexity. The fifth condition ensures that an honest transcript will always help the verifier decide the underlying language L correctly. The sixth condition ensures not only that all expected rewards are integers (which can always be done by multiplying any rational reward by a large enough number) but also that they are bounded by the budget function $b(|x|)$. Unless specified otherwise, the budget that we use in our results is polynomial in $|x|$. Our last condition ensures that honesty is by far the only rational behavior. If Merlin deviates from the honest protocol P and instead acts like a dishonest prover P', then he expects to lose at least \$1000. We remark that this quantity can be an arbitrary polynomial $q(|x|)$, while still keeping a polynomial budget.

Definition 4 (DRMA(t,cc,rc,b)). *Let t, cc, rc, b be as in the definition above. The set of all languages which admit a (t, cc, rc, b)-rational interactive proof with public coins is called $DRMA(t, cc, rc, b)$. For most results in this thesis, the budget $b(n)$ will be a polynomial function of the input size n. Thus, we will often write $DRMA(t, cc, rc, poly(n))$ as $DRMA(t, cc, rc)$ and assume implicitly in this notation that Arthur's budget is polynomial.*

Proper Scoring Rules Let Ω be a finite state space, and $\Delta(\Omega)$ be the set of distributions over Ω. A scoring rule is a function $S : \Delta(\Omega) \times \Omega \to \mathbb{R}$. Such a function can be interpreted as a *reward* that the expert obtains for reporting a probabilistic prediction $\mathcal{P}$ when the realized state of the world turns out to be ω. The scoring rule

S is *strictly proper* if the expert strictly maximizes her reward by announcing the true distribution $\mathcal{D}$. That is, $\mathcal{D} = \text{argmax}_{\mathcal{P}} \mathbb{E}_{\omega \leftarrow \mathcal{D}}[S(\mathcal{P}, \omega)]$. A well known strictly proper scoring rule is *Brier's scoring rule*

$$BSR(\mathcal{P}, \omega) = 2\mathcal{P}(\omega) - \sum_{x \in \Omega} \mathcal{P}(x)^2 + 1.$$

Besides being strictly proper, Brier's scoring rule is always bounded in $[0, 2]$.

Chapter 4

Rational Proof Characterization of TC^0

We first consider rational proofs with a constant number of rounds where Merlin can only send logarithmic length messages and Arthur only has logarithmic computation. We can prove the following theorem

Theorem 1. $DRMA[O(\log n), O(\log n), O(1)] = Search - Uniform - TC^0$

We prove this theorem via two lemmas, each showing a side of the inclusion.

Lemma 1. *For any constant* $d > 0$, $Search{-}Uniform{-}TC_d \subset DRMA[O(\log n), O(\log n), d]$

Proof. First we remark that threshold gates can simulate AND and OR gates, so it suffices to focus on circuits composed entirely of threshold gates and their negations. Let $C_n : \{0,1\}^n \to \{0,1\}$ be a uniform threshold circuit of depth d. Let $C_n(x) = y$. C_n can be thought of as the composition of two circuits $A \circ B$, where A is a threshold circuit of depth 1 and B is a multi-output threshold circuit of depth $d-1$. This allows us to use induction to prove our lemma.

First, we prove the base case: any threshold circuit Th_k of depth 1 has a rational proof with $O(\log n)$ time complexity, $O(\log n)$ communication complexity, one round, and budget $O(n^2)$. It suffices, in order to evaluate $Th_k(x_1, ..., x_n)$, to know how many input bits are equal to 1. Arthur can incentivize Merlin to reveal this count by using a *proper scoring rule.* The rational proof proceeds as follows

1. Merlin announces a probability distribution Y over the domain $\{0,1\}$, with $Prob[Y=1] \in \frac{\mathbb{Z}}{n} \cap [0,1]$.

2. Arthur draws an index $r \leftarrow \{1,...,n\}$ at random, and looks at the input bit x_r. He rewards Merlin using Brier's scoring rule $\mathtt{BSR}(Y, x_r)$.

3. If $n \cdot Prob[Y=1] > k$, then Arthur outputs 1. Otherwise, Arthur outputs 0. [1]

Because Arthur draws x_r at random from $\{x_1,...,x_n\}$ and uses a strictly proper scoring rule to compute Merlin's reward, Merlin is incentivized to report Y such that $Pr[Y=1] = \frac{\#\{i:x_i=1\}}{n}$. The proof obviously proceeds in one round, and Merlin needs $\log n$ bits to communicate the distribution Y. So now all we need to show is that Arthur uses $O(\log n)$ computation and can communicate Merlin's reward using $O(\log n)$ bits. Note that $BSR(Y, x_r) = 2 \cdot Pr[Y = x_r] - (Pr[Y=1]^2 + Pr[Y=0]^2) + 1$, and that both $Pr[Y=0]$ and $Pr[Y=1]$ are $\log n$-bit numbers. Thus, the reward itself is an $O(\log n)$ bit number and can be computed in time $O(\log n \cdot \log\log n)$, which is the time required to square a $\log n$-bit number.

We can improve this running time to $O(\log n)$ by using a ***randomized version*** of Brier's scoring rule. This randomized version is computed by the following procedure

1. Merlin announces a distribution Y over $\{0,1\}$, where $Pr[Y=1] \in \frac{\mathbb{Z}}{n}$.

2. Arthur randomly samples a $y \leftarrow Y$.

3. Arthur randomly samples $x_r \leftarrow \{x_1,...,x_n\}$.

4. Arthur gives Merlin a reward equal to $2Pr[Y=x_r] - Pr[Y=y] + 1$.

Notice that Arthur can sample $y \leftarrow Y$ in time $O(\log n)$ by choosing a number $k \leftarrow \{1,...,n\}$ and setting $y=1$ if $\frac{k}{n} \leq Pr[Y=1]$ and $y=0$ otherwise. He can sample x_r in time $O(\log n)$ (by uniformity of the given circuit), and he can compute the reward in time $O(\log n)$ because he is adding $\log n$-bit numbers. Notice furthermore, that

$$\mathbb{E}_{y \leftarrow Y}[2Pr[Y=x_r] - Pr[Y=y] + 1] =$$

[1] If instead we want to evaluate $\neg Th_k(x_1,...,x_n)$, Arthur outputs 0 when $n \cdot Prob[Y=1] > k$ and 1 otherwise.

$$2Pr[Y = x_r] - (Pr[Y = 0]^2 + Pr[Y = 1]^2) + 1$$

so Merlin's expected reward (where the expectation is taken over Arthur's random choice of y) is exactly equal to $BSR(Y, x_r)$. Thus, Merlin is still incentivized to strictly announce $Pr[Y = 1] = \frac{\#\{i:x_i=1\}}{n}$.

Finally, note that the budget from this rational proof is $O(n^2)$. Merlin's expected reward from announcing Y is

$$Pr[x_r = 1] \cdot \mathbf{BSR}(Y, 1) + Pr[x_r = 0] \cdot \mathbf{BSR}(Y, 0) =$$

$$2Pr[x_r = 1]Pr[Y = 1] + 2Pr[x_r = 0]Pr[Y = 0] -$$

$$-(Pr[Y = 0]^2 + Pr[Y = 1]^2) + 1.$$

Since all the probabilities are integer multiples of $\frac{1}{n}$, Merlin's expected reward is always an integer multiple of $\frac{1}{n^2}$. By multiplying all rewards by $1000n^2$, Arthur can guarantee that if Merlin deviates from the truth, he will always receive a penalty of at least \$1000 in his reward.

This proves the base case of the induction. To prove the inductive step, assume that we have shown that, for circuits of depth $d-1$, there exist rational proofs with $O((d-1)\log n)$ time and communication complexity, $d-1$ rounds and $b(n) = O(n^{2(d-1)})$ budget. By shrinking rewards appropriately, we can assume that the reward for any $d-1$ rational proof with budget $b(n)$ is in $[0, 1]$ and Merlin loses at least $\frac{1}{b(n)}$ by deviating from the honest protocol.

Decompose circuit C_n of depth d into the composition $A \circ B$ where A is a threshold circuit of depth 1 and B is a multi-output threshold circuit of depth $d-1$. Let $B_i(x)$ be the i^{th} output of B. The proof proceeds as follows

1. Merlin announces a distribution Y, allegedly satisfying $Pr[Y = 1] = Pr_i[B_i(x) = 1]$.

2. Arthur chooses one input to A at random using the uniformity of the circuit. Let's say input r. He needs to compute $B_r(x)$, which he does by using a $d-1$-

round rational proof, since B_r has depth $d-1$. Let R and π be the reward and output functions associated with this $d-1$ rational proof, and let $\mathcal{T}$ be the transcript.

3. Arthur gives Merlin an extra reward of $\delta_d \cdot \mathbf{BSR}(Y, B_r(x))$, where δ_d is a scaling factor which only depends on d, and is defined below.

4. If Merlin is truthful, Arthur can use the announced distribution Y to compute $n \cdot Pr[Y=1] = \#\{i : B_i(x) = 1\}$, and using this number Arthur can compute $A(B(x))$.

First, note that the number of rounds in this protocol is d: there are $d-1$ rounds to compute $B_r(x)$, and 1 round to compute $A(B(x))$. Furthermore, Arthur only adds $O(\log n)$ computation steps, the ones he needs to pick the random index r, to compute $\mathbf{BSR}(Y, B_r(x))$,[2] and to compute $A(B(x))$ from the distribution Y. Finally, the communication complexity increases by $O(\log n)$, the amount needed to communicate the distribution Y and the reward $\mathbf{BSR}(Y, B_r(x))$.

Now we need to show that Merlin has the right incentives to tell the truth. Note that, because we are interacting with Merlin multiple times, there is a dynamic incentive compatibility problem. The only reason that Arthur knows the value of $B_r(x)$ is because *Merlin told him this value with a* $d-1$ *round rational proof.* For this information, Merlin receives an expected reward $\mathbb{E}_{\mathcal{T}}[R(x, \mathcal{T})]$ where $\mathcal{T}$ is the transcript of the interaction and the expectation is taken over Arthur's coin tosses. Merlin maximizes this reward by following the protocol honestly and producing a transcript $\mathcal{T}$ compatible with the true value $B_r(x)$. Thus, if Merlin's only reward was $R(x, \mathcal{T})$, he would correctly report $B_r(x)$. However, Merlin's *total reward* is $R(x, \mathcal{T}) + \delta_d \mathbf{BSR}(Y, B_r(x))$. It is possible that Merlin is willing to deviate at some point and produce a *dishonest transcript* of $\mathcal{T}'$ together with a dishonest $B'_r(x)$ because it would increase his reward from $\mathbb{E}_r[\mathbf{BSR}(Y, B_r(x))]$ to some higher $\mathbb{E}_r[\mathbf{BSR}(Y, B'_r(x))]$. To prevent Merlin from doing this, we need to guarantee that $\mathbb{E}_r[\delta_d(\mathbf{BSR}(Y, B'_r(x)) - \mathbf{BSR}(Y, B_r(x)))] < \mathbb{E}_{\mathcal{T}}[R(x, \mathcal{T})] - \mathbb{E}_{\mathcal{T}'}[R(x, \mathcal{T}')]$. That is, any

[2] Again, using the randomized version of Brier's scoring rule.

gains that Merlin obtains from reporting the fake value $B'_r(x)$ are offset by losses that he gets by generating a corresponding dishonest transcript $\mathcal{T}'$.

Recall that the budget function $b(n)$ satisfies $\mathbb{E}_{\mathcal{T},\mathcal{T}'}[R(x,\mathcal{T}) - R(x,\mathcal{T}')] > \frac{1}{b(n)}$, where the expectation is taken with respect to Arthur's random coin tosses in the transcripts, and that by inductive hypothesis $b(n) = O(n^{2(d-1)})$. Recall also that $\mathbf{BSR}(\cdot,\cdot) \in [0,2]$ and thus $\mathbf{BSR}(Y, B'_r(x)) - \mathbf{BSR}(Y, B_r(x)) < 2$. Thus, if we set $\delta_d = \frac{1}{2 \cdot b(n)}$ we will have $\mathbb{E}_r[\delta_d(\mathbf{BSR}(Y, B'_r(x)) - \mathbf{BSR}(Y, B_r(x)))] \leq \frac{1}{b(n)} < \mathbb{E}_{\mathcal{T}}[R(x,\mathcal{T})] - \mathbb{E}_{\mathcal{T}'}[R(x,\mathcal{T}')]$. This implies that Merlin strictly maximizes his reward by reporting both $\mathcal{T}$ and Y honestly.

Finally, we need to show that Arthur's budget is $O(n^{2d})$. Note that Merlin's expected reward is $\mathbb{E}_r[\frac{1}{2b(n)}\mathbf{BSR}(Y, B_r(x))] + \mathbb{E}_{\mathcal{T}}[R(x,\mathcal{T})]$. If Merlin deviates and produces an alternative dishonest transcript $(\mathcal{T}', Y')$, his expected reward is $\mathbb{E}_r[\frac{1}{2b(n)}\mathbf{BSR}(B'_r(x), Y')] + \mathbb{E}_{\mathcal{T}'}[R(x,\mathcal{T}')]$. The difference in rewards is $\mathbb{E}_r[\frac{1}{2b(n)}(\mathbf{BSR}(Y, B_r(x)) - \mathbf{BSR}(B'_r(x), Y'))] + (\mathbb{E}_{\mathcal{T}}[R(x,\mathcal{T})] - \mathbb{E}_{\mathcal{T}'}[R(x,\mathcal{T}')])$. Note that, since $(\mathcal{T}, Y) \neq (\mathcal{T}', Y')$, both terms cannot be zero simultaneously.[3] The first difference satisfies $\mathbb{E}_r[\frac{1}{2b(n)}(\mathbf{BSR}(Y, B_r(x)) - \mathbf{BSR}(B'_r, Y')] \in \frac{1}{2b(n)} \cdot \frac{1}{n^2} \cdot \mathbb{Z}$. If $\mathcal{T} \neq \mathcal{T}'$, the second difference satisfies $\mathbb{E}_{\mathcal{T}}[R(x,\mathcal{T})] - \mathbb{E}_{\mathcal{T}'}[R(x,\mathcal{T}')] > \frac{1}{b(n)}$. Multiplying the reward by $2000b(n) \cdot n^2$, we get that the loss that Merlin gets by producing a dishonest transcript $(\mathcal{T}', Y')$ is always greater than 1000. Since $2000b(n) \cdot n^2 = O(n^{2(d-1)} \cdot n^2) = O(n^{2d})$, we conclude that our rational proof's budget is $O(n^{2d})$.

Thus, we have shown that depth d threshold circuits admit rational proofs with $O(d \cdot \log n)$ communication and time complexity, that these proofs can be completed in d rounds, and that they can be done with an $O(n^{2d})$ budget. *Q.E.D.*

Lemma 2. $DRMA[O(\log n), O(\log n), O(1)] \subset$
$Search - Uniform - TC^0$

Proof. We prove this by induction. We begin with the case $d = 1$. In this case, we want to show that rational proofs with $\log n$ bits of communication and with one round can be simulated by a $DLOGTIME$-search-uniform circuit in TC^0.

[3] Otherwise, the dishonest transcript $(\mathcal{T}', Y')$ would maximize Merlin's reward.

Let L be a language that can be decided with a d-round rational proof that uses $k(n) = O(\log n)$ communication and time complexity per round. The transcript $\mathcal{T}$ of this interaction is $(a_1, b_1, ..., a_d, b_d)$ where a_i is Merlin's message in round i and b_i consists of Arthur's random coins in that round. In round i, Merlin chooses $a_i^*(a_1, b_1, ..., a_{i-1}, b_{i-1})$ as a function of the transcript up to that round in order to maximize

$$V(a_i; a_1, ..., b_{i-1}) =$$

$$\sum_{b_i \in \{0,1\}^{k(n)}} \max_{a_{i+1}} \sum_{b_{i+1}} \ldots \max_{a_d} \sum_{b_d} R(x, (a_1, b_1, ..., a_d, b_d)).$$

A threshold circuit can reconstruct Merlin's choices by doing backwards induction. Given as input a transcript

$a_1, ..., a_{d-1}, b_{d-1}$ of the interaction up to round $d-1$, a uniform threshold circuit of constant depth can compute $\max_{a_d} \sum_{b_d} R(x, (a_1, ..., a_d, b_d))$ and choose the input a_d^* that maximizes this expected reward. Similarly, when given as input the transcript $\mathcal{T}_{i-1}$ up to round $i-1$, a depth $O(d-i)$ circuit can compute $a_i^* = \text{argmax}_{a_i} V(a_i; a_1, ..., b_{i-1})$. Thus, we can reconstruct Merlin's choices using a uniform threshold circuit of depth $O(d)$. Given the input x to the rational proof, the circuit first computes Merlin's action

$$a_1^* = \text{argmax}_{a_1} \sum_{b_1} \max_{a_2} \ldots \sum_{b_d} R(x, (a_1, b_1, ..., a_d, b_d)).$$

Because each message a_i or b_i is restricted to be of length $O(\log n)$, every MAX gate is taking the maximum of $poly(n)$ numbers and every SUM gate is taking the sum of $poly(n)$ numbers. Furthermore, all numbers have $poly(n)$ bits.[4] Thus, this can be computed using a search-uniform threshold circuit of polynomial size and depth d.[5] Merlin then tosses $O(\log n)$ random coins to choose b_1^* and computes $a_2^*(a_1^*, b_1^*)$, which again can be computed with a polynomial size threshold circuit of depth d. Continuing with this procedure, we can reproduce an honest transcript $\mathcal{T} = (a_1^*, b_1^*, a_2^*, ..., a_d^*, b_d^*)$

[4]Each $R(x, \mathcal{T})$ must be computed in time $O(\log n)$ and hence has $O(\log n)$ bits. The addition of $poly(n)$ such numbers will be a $poly(n)$-bit number.

[5]The only operations we use are iterated addition and taking maxima. For both these operations, one can construct threshold circuits where the i^{th} input to an arbitrary gate g can be computed in logarithmic time. See [9] for a description of such circuits.

with a circuit of depth $O(d^2)$. Finally, given $\mathcal{T}$, we can compute the output $\pi(x, \mathcal{T})$ of the rational proof in time $O(\log n)$, and hence in TC^0.

This shows that any language decidable by a d-round rational proof with $O(\log n)$ communication per round is decidable by a search-uniform threshold circuit of depth $O(d^2)$. *Q.E.D.*

Remark Our proof also applies to circuits with non-constant depth $d(n)$, as long as Arthur only performs $O(\log n)$ computation per round, and communication is limited to $O(\log n)$ bits per round. In this case, a $d(n)$-round rational proof with budget $O(n^{2d(n)})$ can be given for any threshold circuit of depth $d(n)$. Furthermore, any $d(n)$ round proof can be simulated by a circuit of depth $O(d^2(n))$.

Chapter 5

Rational Proof Characterization of $P^{||NP}$

In this chapter, we study a rational analogue of Probabilistically Checkable Proofs (PCPs). We show that the set of languages for which there exists a rational proof with $O(\log n)$ verification time, $poly(n)$ communication and 1 round is exactly $P^{||NP}$, the set of languages that can be decided by a polynomial time machine that can make non-adaptive queries to NP.

In addition, while in Theorem 1 we prove our results for search-uniform circuits, in this theorem our proof will actually hold for the decision version of uniformity. Nevertheless, our proof will proceed as if our circuits were search-uniform until the very end, when we explain how to modify it for decision-uniform circuits.

Theorem 2. $DRMA[\log(n), poly(n), 1] = P^{||NP}$.

Proof. We will first show a rational proof for any language in P. Combining this with a rational proof for SAT, we will show how to obtain such proofs for any language in $P^{||NP}$.

Let $\{C_n\}_{n=1}^{\infty}$ be a uniform[1] polynomial size circuit family, where the gates are AND, OR and NOT gates. Given an input $x \in \{0,1\}^n$, a circuit C_n and a gate

[1]The class P can be characterized as the set of languages decidable by $DLOGTIME$-uniform circuits [7],[6]. We give our proof for Search-Uniform circuits (see chapter 3), but it can be adapted to uniform circuits using the definition of [7], as we remark below.

$g \in C_n$, let $v_g(x)$ be the value that gate g outputs when the circuit is evaluated on input x. Input wires are considered gates of the circuit. If a gate g corresponds to the i^{th} input wire, then $v_g(x) = x_i$.

Our rational proof proceeds as follows. Given an input x and a circuit C_n, Merlin sends as his message a vector $\{\alpha_g\}_{g \in C_n}$. Allegedly, $\alpha_g = v_g(x)$, the correct output that gate g would produce when the circuit is evaluated on x. To incentivize Merlin to fill out the circuit correctly, Arthur can choose a gate g at random using $O(\log n)$ coin tosses. If g is an input gate representing input x_i, Arthur can pay Merlin \$1000 if $\alpha_g = x_i$ and \$0 if it is not. If g is an AND gate with input gates i, j, Arthur can pay Merlin \$1000 if $\alpha_g = AND(\alpha_i, \alpha_j)$ and zero otherwise. Analogously, Arthur can compute the rewards for OR and NOT gates.

It is clear that the maximum expected payment that Merlin can obtain is \$1000, and that he receives this payment if and only if he fills out the circuit with the correct values at every step. Given the filled out circuit, Arthur can simply read $\alpha_{g_{out}}$ for the output gate g_{out} which, given that Merlin is incentivized to be truthful, should be the value of $C(x_1, ..., x_n)$.

As a remark, let us point out that this proof requires, for any gate g in a polynomial size circuit, the ability of finding any given input to g in logarithmic time. For example, if g was an AND gate, then we needed to check that $\alpha_g = AND(\alpha_i, \alpha_j)$ where i, j were the input gates to g. If we want to use the ***decision version*** of uniformity, we can alter our proof so Arthur chooses three gates (g, i, j) at random, and proceeds with the proof only if i, j are the inputs to g (which he can check by deciding whether the tuples $(1^n, g, i, 1)$ and $(1^n, g, j, 2)$ are in the connection language of the circuit). If they are not the inputs to g, the proof halts and Merlin gets nothing. Note that this does not affect Merlin's incentives and scales down his expected reward by at most a polynomial factor, so the budget of the proof is still polynomial. Thus, we can modify our rational proof so it applies to the decision version of $DLOGTIME$ uniformity.

This shows that there exist one round rational proofs with $O(\log n)$ complexity, $poly(n)$ communication and one round for languages in P. We now show such proofs

also exist for the language $SAT = \{\phi : \phi \text{ is a satisfiable boolean formula }\}$. Given a formula $\phi : \{0,1\}^n \to \{0,1\}$ with m clauses, Merlin sends Arthur a pair (y, k), where $y \in \{0,1\}^n$ is an assignment of the variables that (allegedly) maximizes the number of satisfied clauses in ϕ and k is (allegedly) the number $\#\phi(y)$ of clauses satisfied by y. Arthur gives Merlin the following rewards to ensure that k and y are reported truthfully:

1. To incentivize Merlin to announce y to satisfy as many clauses as possible, Arthur chooses a clause c of ϕ at random by tossing $O(\log n)$ coins. Denote the clause by $c = x_{i_1} \vee x_{i_2} \vee x_{i_3}$ (to make notation simpler, we assume the inputs are not negated). Arthur can now read $y_{i_1}, y_{i_2}, y_{i_3}$ from Merlin's message y and check whether $c(y) = y_{i_1} \vee y_{i_2} \vee y_{i_3}$ is satisfied. If it is, then Arthur pays Merlin \$1000. Otherwise, Arthur pays Merlin \$0. Call this reward $R_1(\phi, y, c)$.

2. Even knowing ϕ and y, and knowing that y satisfies as many clauses as possible, Arthur still does not know if $\phi(y) = 1$ (implying ϕ is satisfiable) or $\phi(y) = 0$ because evaluating ϕ requires polynomial time. Thus, Arthur needs to trust that Merlin correctly reports k to be $\#\phi(y)$, the number of clauses satisfied by y. To incentivize Merlin to make this report, Arthur samples another clause c' from ϕ uniformly at random, evaluates $c'(y)$ and gives Merlin a reward equal to $\delta \cdot \mathbf{BSR}(K, c'(y))$,[2] where K is a random variable over $\{0,1\}$ constructed so that $Pr[K = 1] = \frac{k}{m}$ and δ is a scaling factor to be chosen later. Note that, since c' is a uniformly random clause, the probability $Pr[c'(y) = 1]$ is equal to $\frac{\#\phi}{m}$. Assume momentarily that y was honestly reported and is now out of Merlin's control, so the only way that Merlin controls this reward is by setting k. Since BSR is a strictly proper scoring rule, he will want to announce $Pr[K = 1] = Pr[c'(y) = 1]$, or equivalently $k = \#\phi(y)$. Call this reward $\delta \cdot R_2(\phi, y, k, c')$.

Note that if the two rewards were given to two separate Merlins, we could guarantee that (1) the first Merlin correctly announces a y maximizing the number of

[2] As in theorem 1, use the randomized Brier scoring rule so the computation can be done in $O(\log n)$ time.

satisfied clauses in ϕ and (2) the second Merlin correctly announces $k = \#\phi(y)$. However, since we only have one expert, we need to scale his rewards so that he will simultaneously announce all of this information truthfully.

When we work with one Merlin, we can guarantee that his incentives are correct by scaling the rewards, as in our proof of theorem 1. Merlin's *total expected reward* is

$$\mathbb{E}_{c \leftarrow \phi}[R_1(\phi, y, c)] + \delta \cdot \mathbb{E}_{c' \leftarrow \phi}[R_2(\phi, y, k, c')]$$

where δ is a scaling factor that we will control. Note that $R_2(\phi, y, k, c') = \mathbf{BSR}(K, c'(y)) < 2$, and that for any $y \neq y'$ we have $\mathbb{E}_{c \leftarrow \phi}[R_1(\phi, y, c) - R_1(\phi, y', c)] > \frac{1}{m}$. If we make $\delta < \frac{1}{2m}$, then Merlin always has a much higher incentive to report a y maximizing the number of clauses in ϕ (and hence get a very large reward), over reporting a "fake" y to manipulate the (much smaller) reward obtained from $\frac{1}{2m}\mathbf{BSR}(K, c'(y))$. This shows that Merlin is incentivized to report a correct y. Given that he reports a correct y, he is also incentivized via the scoring rule to report a correct k. Finally, Arthur can read k and determine whether the formula is satisfiable or not by checking whether $k = m$ or $k < m$. This shows that SAT admits a one-round rational proof. Note that $\mathbf{BSR}(K, c'(y))$ is always an integer multiple of $\frac{1}{m^2}$, and hence the proof has a polynomial size budget. Merlin's smallest loss from lying is $\frac{1}{2m} \cdot \frac{1}{m^2} = \frac{1}{2m^3}$. Note that this rational proof pays at most \$1002. We will use this fact in our construction of rational proofs for $P^{||NP}$.

So far, we have shown that any language in $P \cup \{SAT\}$ admits a rational proof with $O(\log n)$ complexity, $poly(n)$ communication and 1 round. Now we need to show that such proofs also exist for languages in $P^{||NP}$.

Let L be a language in $P^{||NP}$. This means that there exists a polynomial time machine with an access oracle to SAT that, on input x, can make polynomially many non-adaptive queries $\phi_1, ..., \phi_\ell$ to the SAT oracle. Let $SAT(\phi_j)$ be the oracle's answer on query ϕ_j. The machine can then continue its computation on input x and the oracle answers $SAT(\phi_1), ..., SAT(\phi_\ell)$ to decide whether $x \in L$. More formally, there exist two uniform polynomial-size circuit families $\{A_n\}_{n=1}^{\infty}, \{B_n\}_{n=1}^{\infty}$ such that

on input $x \in \{0,1\}^n$

1. $A(x)$ outputs m boolean formulas $\phi_1, ..., \phi_\ell$.

2. $B(x, SAT(\phi_1), ..., SAT(\phi_\ell))$ outputs $L(x)$.

To produce our rational proof, we need to incentivize Merlin to *simultaneously*

1. Evaluate the circuit $A(x) = (\phi_1, ..., \phi_\ell)$ correctly.

2. Give correct answers to $SAT(\phi_1), ..., SAT(\phi_\ell)$.

3. Evaluate the circuit $B(x, SAT(\phi_1), ..., SAT(\phi_\ell))$ correctly.

Any of these three problems can be solved individually. However, because we are asking Merlin to solve the three problems ***simultaneously***, there might be a conflict of incentives for Merlin. The formulas $\phi_1, ..., \phi_\ell$ depend on Merlin's computation of $A(x)$. Arthur only knows what queries to ask the NP oracle because Merlin computed $A(x)$ for him. If Merlin does not perform this computation truthfully, then Arthur might query Merlin on different boolean formulas $\psi_1, ..., \psi_\ell$, and reporting his knowledge about ψ might give Merlin a higher reward than reporting his knowledge about ϕ. Thus, Merlin might be incentivized to lie on the computation of $A(x)$ in order to get a higher reward for his knowledge about boolean formulas.

We solve this problem by scaling Merlin's reward. Let n^c be the size of circuit A_n. This is also a bound on the number ℓ of queries to the SAT oracle. Let $\alpha(\phi_1), ..., \alpha(\phi_\ell)$ be Merlin's answers to the SAT queries. Let n^d be a bound on the size of the circuit $B_{n+n^c}(x_1, ..., x_n, \alpha(\phi_1), ..., \alpha(\phi_{n^c}))$. We note that the size of this circuit does not depend on the answers $\alpha(\phi_1), ..., \alpha(\phi_\ell))$.

Regardless of Merlin's answers $\alpha(\phi_1), ..., \alpha(\phi_n)$ to the SAT queries, Merlin always maximizes his reward by evaluating $B_{n+n^c}(x_1, ..., x_n, \alpha(\phi_1), ..., \alpha(\phi_{n^c}))$ correctly. If he fills out this circuit correctly, he gets \$1000 in expectation, and he gets less money if he gives incorrect answers. Thus, we do not need to scale down Merlin's reward for computing the circuit B in order to incentivize him to truthfully compute $\alpha(\phi_i) = SAT(\phi_i)$.

However, Merlin can still lie on the computation of $A(x) = (\phi_1, ..., \phi_\ell)$ in order to obtain more money for his knowledge about boolean formulas. To prevent this, we note that the maximum amount of money that Merlin can make by answering queries about $\phi_1, ..., \phi_\ell$ is $\ell \cdot \$1002$ because Merlin makes at most \$1002 on each SAT query, and there are ℓ such queries. Since $\ell < n^c$, Merlin makes at most $\$1002 \cdot n^c$ by answering SAT queries.

How do we prevent Merlin from lying on the computation of $A_n(x)$? Since A_n has size n^c, by lying on one gate g of $A_n(x)$ Merlin loses at least $\$\frac{1}{n^c}1000$. Thus, if Arthur scales down the reward for each SAT query by a factor of $\gamma = \frac{1}{2n^{2c}}$, Merlin cannot make more than $\$\gamma 1002n^c < \frac{1}{n^c}\$1000$ by answering SAT queries. Thus, Merlin will never lie on the computation of $A(x)$ (obtaining an expected loss of at least $\frac{1000}{n^c}$) in order to gain only a small reward on his answers to SAT queries.

This shows that any language in $P^{\|NP}$ has one-round rational proofs with logarithmic complexity and polynomial communication. Note that if we scale all the rewards by a large enough polynomial (depending on the circuit size) we can make all rewards integers (that are bounded above by a polynomial in n) and any differences in reward larger than \$1000. Thus, the proof has polynomial budget.

Now we need to show that $\text{DRMA}[\log(n), poly(n), 1] \in P^{\|NP}$. Let L be a language that admits a one-round rational proof with reward function R, predicate function π. For any input x, message a from Merlin and random coin tosses b from Arthur, let $V(a) = \mathbb{E}_b[R(x, a, b)]$.[3] Let $n = |x|$ and let $B = n^c$ be the budget of the rational proof. This means that we can think of the value function $V(a)$ as taking integer values in the set $[0, B]$. Let $i^* = \max_a V(a)$ and let $A^* = \text{argmax}_a V(a)$. Since any $a^* \in A^*$ maximizes Merlin's value $V(a)$, we must have that $\pi(x, a^*)$ correctly tells us whether $x \in L$ or not. Furthermore, for any $a \notin A^*$ we have $V(a^*) - V(a) > 1000$

Let M be the language of pairs $(i, c) \in \{0, ..., B\} \times \{0, 1\}$ such that there exists a message a satisfying $V(a) \geq i$ and $\pi(x, a) = c$. Note that the language M is in NP. Since Arthur only tosses logarithmically many coins, the value $V(a) = \mathbb{E}_b[R(x, a, b)]$

[3]We use the notation $V(a)$ to denote the value of message a. It should not be confused with the verifier in the formal definition of rational proofs, who we just call Arthur in this chapter.

can be computed in polynomial time. Once this value has been computed, we can verify in polynomial time whether $V(a) \geq i$, and whether $\pi(x, a) = c$.

Our goal now is finding the largest value $i^* \in \{0, ..., B\}$ such that there exists a c with $(i^*, c) \in M$. Such i^* will satisfy $i^* = V(a^*)$ for any a^* that maximizes V. For any such a^*, we must have $\pi(x, a^*) = L(x)$. Therefore, exactly one of $(i^*, 0)$ or $(i^*, 1)$ belongs to the language M. This unique bit c allows us to determine whether $x \in L$ or not.

To find i^*, we can make $2 \cdot B$ parallel queries to the NP language M, one for each pair (i, c). Since $(i, c) \in M$ if and only if there exists a such that $V(a) \geq i$ and $\pi(x, a) = c$, i^* will be the largest value of i for which our oracle tells us $(i, c) \in M$. Since the bit c associated with i^* is unique, we conclude that L can be decided in $P^{||NP}$. *Q.E.D.*

Chapter 6

Rational Proof Characterization of $P||MA$

The rational proof for $P^{||NP}$ shows that, even for languages in $co-NP$ —which do not admit classical Merlin Arthur proofs unless the polynomial hierarchy collapses— there exist polynomial budget, logarithmic time, 1 round rational proofs. How limited are we by the restriction that Arthur act in logarithmic time? Is it possible to give a constant round rational proof with polynomial budget and a ***polynomial time verifier*** that for languages much more powerful than those in $P^{||NP}$? We show that this is not the case: the set of languages decidable by one round polynomial budget rational proofs with a polynomial time verifier is exactly $P^{||MA}$. That is, the NP oracle in our previous proof is replaced by an MA oracle.

Theorem 3. $DRMA[poly(n), poly(n), 1] = P^{||MA}$

Proof. First, it is clear that MA admits one-round rational proofs with polynomial (actually constant) budget. Let L be a language in MA and let x be an input. There exists a verifier $V(\cdot,\cdot,\cdot)$ such that, if $x \in L$, there exists a message a from Merlin that $Pr_b[V(x,a,b)=1] \geq \frac{2}{3}$ where the probability is taken over Arthur's random coin tosses b. If $x \notin L$, then for every message a by Merlin we have $Pr_b[V(x,a,b)=0] \geq \frac{2}{3}$. The rational proof for L proceeds in the following way. Merlin reports a certificate a and a bit $c \in \{0,1\}$. If $c=1$ (that is, Merlin reports that $x \in L$, then Arthur tosses

random coins b and gives Merlin a reward equal to $V(x,a,b)$. If $c=0$ (that is, Merlin reports that $x \notin L$, then Arthur gives Merlin a reward equal to $\frac{1}{2}$.

Clearly Merlin maximizes his reward by being honest and reporting $c = L(x)$. When $L(x) = 1$, he gets an expected reward of $Pr[V(x,a,b) = 1] \geq \frac{2}{3}$ for being honest and reporting $c = 1$, and an expected reward of $\frac{1}{2}$ for lying and reporting $c = 0$. When $L(x) = 0$, he gets an expected reward of $\frac{1}{2}$ for being honest and reporting $c = 0$ and an expected reward of $Pr[V(x,a,b) = 1] \leq \frac{1}{3}$ for lying and reporting $c = 1$.

Now we show that $P^{\|MA}$ admits one round rational proofs. In our rational protocol, the verifier uses the prover as an oracle for the MA queries. The verifier can ask all these queries in parallel, which does not affect the prover's incentives. Finally, he uses the prover's answers as oracle answers in his $P^{\|MA}$ computation.

The proof that DRMA$[poly(n), poly(n), 1)] \subset P^{\|MA}$ is similar to our proof above that DRMA$[O(\log n), poly(n), 1]$ is in $P^{\|NP}$. Let L be a language admitting a one-round rational proof with reward function R and predicate π. Let a be Merlin's message and b be Arthur's coin tosses. Recall that we defined the value of Merlin's message as $V(a) = \mathbb{E}_b R(x,a,b)$, where x is the input whose membership in L we are trying to decide. The key is that, since the rational proof has a polynomial budget, the value $V(a)$ is an integer in $\{0,...,B\}$ for some B that is polynomial in $|x|$. Let M be the language of pairs $(i,c) \in \{0,...,B\} \times \{0,1\}$ such that there exists a with $V(a) \geq i$ and $\pi(x,a) = c$. In our proof for showing DRMA$[O(\log n), poly(n), 1] \subset P^{\|NP}$, we made use of a language $M \in NP$. M was in NP because when Arthur tosses logarithmically many coins, the value $V(a)$ can be computed in polynomial time.

What happens if we allow Arthur to use polynomially many coins? In this case, we can't necessarily verify whether a pair $(i,c) \in M$ in polynomial time because computing $V(a) = \mathbb{E}_b[R(x,a,b)]$ requires adding up exponentially many terms. However, the language M is in the class MA. To see this, consider the following classical Merlin-Arthur protocol. On input a pair (i,c), Merlin sends Arthur a message a, allegedly one that makes $V(a) = \mathbb{E}_b[R(x,a,b)] \geq i$. Arthur now generates m uniformly random strings $b^1,...,b^m$, where each $b^i \leftarrow \{0,1\}^{poly(n)}$ and computes the sample aver-

age $S = \frac{1}{m}\sum_{j=1}^{m} R(x,a,b^j)$. Arthur accepts if $S > i - \frac{1}{3}$ and $\pi(x,a) = c$, and rejects otherwise.

We need to prove soundness and completeness for this protocol. We first prove completeness. That is, the case where $(i,c) \in M$ and Merlin is honest. That is, Merlin sends a message a such that $V(a) \geq i$ and $\pi(x,a) = c$. One can show using Chernoff's bound that if m is large enough (but still polynomial), then $Pr[S > V(a) - \frac{1}{3}] \geq \frac{2}{3}$. Now we prove soundness. That is the case where $(i,c) \notin M$. Given Merlin's message a, Arthur can quickly detect whether $\pi(x,a) = c$, so Merlin will get caught if he gives the wrong c. Since $V(a)$ is an integer and $V(a) < i$, we have $V(a) \leq i - 1$. Thus, by flipping a polynomial number of coins Arthur can guarantee that $Pr[S < i - \frac{1}{3}] \geq Pr[S < V(a) - \frac{1}{3}] \geq \frac{2}{3}$. This proves soundness.

By making $2B$ parallel queries of the form "is (i,c) in M?", the $P^{||MA}$ machine can decide whether $x \in L$ or not. *Q.E.D.*

Remark It is easy to generalize the argument in theorem 2 to show that d-round rational proofs with polynomial budget and polynomial communication are contained in the polynomial hierarchy. This suggests that the polynomial hierarchy might admit d-round rational proofs. However, in order to construct good multi-round rational proofs, we need to prevent Merlin from lying in one round in order to answer more (or better) queries in the next rounds. While we have been able to avoid this problem for TC^0 and the counting hierarchy by using random coin tosses to decide future queries, this technique is tied to the nature of these "majority" complexity classes. It is not clear whether such techniques can be extended to give rational proofs for the polynomial hierarchy.

Chapter 7

Conclusion

This thesis introduced a new way of delegating computation, and showed that, as long as we trust our computation providers to act in their best interest, verification is not necessary. Thus, the rational proofs that we propose are much faster and efficient ways of delegating computation than traditional interactive proofs that rely on verifying that the prover is not cheating.

With these tools, I was able to give new characterizations of complexity classes, such as TC^0, $P^{\|NP}$ and $P^{\|MA}$ based on incentives. One ambitious avenue for future work would be to use these rational characterizations in order to better understand the power of these complexity classes.

I believe that there is a large potential role for incentives in computation. The world is not full of malicious adversaries who will go out of their way to sabotage our best intentions. Instead, it is full of rational people who act in their best interest (and even sometimes altruistically). Being aware of this rationality may hold the key to designing faster and more powerful interactive protocols.

Bibliography

[1] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[2] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of np. *Journal of the ACM*, 45(1):70–122, 1998.

[3] Pablo Daniel Azar and Silvio Micali. Rational proofs. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1017–1028. ACM, 2012.

[4] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32. ACM, 1991.

[5] Laszlo 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.

[6] David A. Mix Barrington and Neil Immerman. Time, hardware, and uniformity. In *Proceedings of Ninth Annual IEEE Structure in Complexity Theory Conference*, pages 176–185. IEEE Computer Society, 1994.

[7] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within nc^1. *Journal of Computer Systems Science*, 41(3):274–306, 1990.

[8] Tugkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate pcps for multidimensional bin-packing problems. *Information and Computation*, 196(1):42–56, 2005.

[9] Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. Constant depth reducibility. *SIAM Journal on Computing*, 13(2):423–439, 1984.

[10] Xi Chen and Xiaotie Deng. Settling the complexity of two-player nash equilibrium. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 261–272. IEEE Computer Society, 2006.

[11] Anne Condon and Richard J. Lipton. On the complexity of space bounded interactive proofs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 462–467. IEEE Computer Society, 1989.

[12] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[13] Cynthia Dwork and Larry J. Stockmeyer. Finite state verifiers i: The power of interaction. *Journal of the ACM*, 39(4):800–828, 1992.

[14] Cynthia Dwork and Larry J. Stockmeyer. Finite state verifiers ii: Zero knowledge. *Journal of the ACM*, 39(4):829–858, 1992.

[15] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Information and Computation*, 189(2):135–159, 2004.

[16] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[17] Uriel Feige and Joe Kilian. Making games short. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 506–516. ACM, 1997.

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

[19] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[20] 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.

[21] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 723–732. ACM, 1992.

[22] Gillat Kol and Ran Raz. Competing provers protocols for circuit evaluation. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:122, 2011.

[23] Mark W Krentel. The complexity of optimization problems. *Journal of computer and system sciences*, 36(3):490–509, 1988.

[24] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2001.

[25] Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[26] Grant Schoenebeck and Salil Vadhan. The computational complexity of nash equilibria in concisely represented games. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 270–279. ACM, 2006.

[27] Yael Tauman Kalai Shafi Goldwasser and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th annual ACM Symposium on Theory of computing*, pages 113–122. ACM, 2008.