Quantum Interactive Proofs and Semidefinite Programming

John Watrous

Institute for Quantum Computing
University of Waterloo

Talk overview

The main goal of the talk is to introduce the **quantum interactive proof system** model and explain how **semidefinite programming** is useful in its analysis.

There are 6 main sections of the talk:

1. Classical interactive proof systems overview.
2. Quantum interactive proof systems: definitions and basic facts.
3. Brief introduction to semidefinite programming.
4. A semidefinite programming formulation of quantum interactive proof systems.
5. QIP = PSPACE.
6. A different semidefinite programming formulation of quantum interactive proof systems (and other interactions).

Please feel free to ask questions at any time!

Background knowledge

I will assume that you are familiar with the basics of quantum information and computation, including:

1. States (as density operators), measurements, and channels.
2. Quantum circuits and related notions (such as BQP).
3. Basic linear algebra and matrix theory.

For the purposes of this talk, a **register** is a collection of qubits to which we assign a name and view as a single object.

Typical names for registers will be X, Y, Z, and W (often with subscripts), and we use the same letter in a scripted font to denote its associated Hilbert space (e.g., $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, and $\mathcal{W}$).

Assume quantum circuits are composed of gates from a finite universal gate set, such as **Toffoli**, **Hadamard**, and (imaginary) **phase** gates. We may include **ancillary** and **erasure** gates if we want to consider circuits that implement general channels.

1. Classical interactive proof systems.

Proofs in theoretical computer science, definition of interactive proof systems and complexity classes, and a few simple examples.

The notion of proofs in theoretical computer science

The general notion of a **proof** has central importance in the theory of computation, both historically and mathematically speaking.

- Gottfried Leibniz (1646–1716) built a calculating machine and developed an early form of symbolic logic. He wrote of extending his machine to reason using this symbolic logic.
- Alonzo Church (1903–1995) and Alan Turing (1912–1954) developed the first formal models of computation. They were both influenced and motivated by work of David Hilbert (1862–1943) and Kurt Gödel (1906–1978) on mathematical logic.
- The theory of NP-completeness, pioneered by Stephen Cook, Leonid Levin, and Richard Karp in the early 1970s, is closely linked to the efficient verification of proofs.

The connections are natural: classical notions of proofs and computations may both be abstracted as manipulations of symbols according to fixed sets of rules.

Interactive proofs and quantum computation

The 1980s saw the introduction of two fascinating ways of extending the traditional notions of proofs and computations.

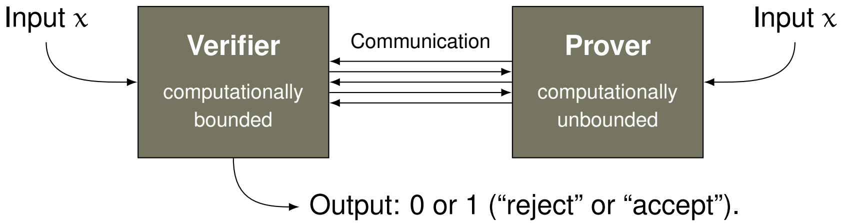
- The **interactive proof system** model was proposed by Shafi Goldwasser, Silvio Micali, and Charles Rackoff, and independently by László Babai in 1985.
- **Quantum computation** was proposed independently by Yuri Manin and Richard Feynman in the early 1980s. David Deutsch defined the quantum Turing machine model of computation in 1985.

The interactive proof system model has had a major impact on complexity theory. The model is also important in theoretical cryptography, and many variations on it have been studied.

Further comments on the history of quantum computation are presumably not needed for this audience. . .

Interactive proof systems

Interactive proof systems have the following structure:



To say that a computational decision problem has an interactive proof system means that there exists a verifier meeting two conditions:

Completeness: For every yes-input, there must exist a prover strategy causing the verifier to accept with high probability.

Soundness: For every no-input, all prover strategies must cause the verifier to reject with high probability.

Informal example: the Pepsi challenge

Consider the following claim:

Coke and Pepsi taste different.

Suppose that you believe this claim, and that indeed you can taste the difference. (Substitute “salt” and “sugar” for “Coke” and “Pepsi” if it helps with the example.)

How would you convince a skeptic?

Non-interactive proof: hopeless.

Interactive proof: allow the skeptic to perform a randomized blind taste test on you... When you correctly identify Coke or Pepsi 100 times in a row, the skeptic should be convinced.

A computational example: graph isomorphism

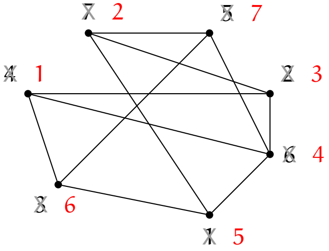
The **graph isomorphism problem** is as follows:

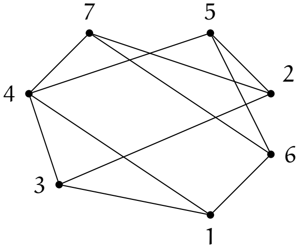
Input: Two simple, undirected graphs G_0 and G_1.

Yes: G_0 and G_1 are isomorphic ($G_0 \cong G_1$).

No: G_0 and G_1 are not isomorphic ($G_0 \ncong G_1$).

This problem is in NP: it is easy to prove that two graphs are isomorphic by simply exhibiting an isomorphism from G_0 to G_1.





Isomorphism: $1 \to 5,\ 2 \to 3,\ 3 \to 6,\ 4 \to 1,\ 5 \to 7,\ 6 \to 4,\ 7 \to 2$.

Another example: graph non-isomorphism

The completeness and soundness conditions are **not symmetric**...

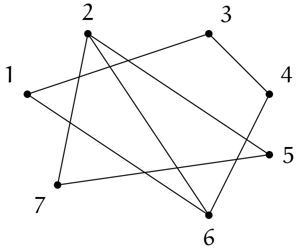
Consider the **graph non-isomorphism** problem:

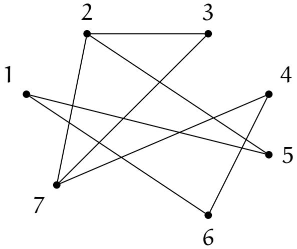
Input: Two simple, undirected graphs G_0 and G_1.

Yes: G_0 and G_1 are not isomorphic ($G_0 \ncong G_1$).

No: G_0 and G_1 are isomorphic ($G_0 \cong G_1$).

Consider proving that these two graphs are non-isomorphic:





It is not known whether or not graph non-isomorphism is in NP.

Blind taste test for graphs

There is a simple (classical) interactive proof system for the graph non-isomorphism problem requiring just one question and response:

1. The verifier randomly chooses a bit $b \in \{0, 1\}$ and a permutation $\sigma \in S_n$, sets $H = \sigma(G_b)$, and sends H to the prover.
2. Implicitly, the prover is being challenged to identify whether $b = 0$ or $b = 1$. If the prover guesses correctly, the verifier accepts (or outputs 1), otherwise he rejects (or outputs 0).

Intuitively speaking, we may think of **isomorphism classes** of graphs as having **different flavors**. The prover, being computationally unbounded, can taste the difference between the graphs (even after a random permutation).

This protocol may be repeated many times (sequentially or in parallel) to decrease the error probability.

Classical complexity classes from interactive proofs

Several variants of classical interactive proof systems have been studied, and many results are known about these models.

Two fundamental complexity classes based on these models:

AM	The class of decision problems having classical interactive proof systems in which a **constant number of messages** are exchanged between the prover and verifier.
IP	The class of decision problems having classical interactive proof systems in which a **polynomial number of messages** are exchanged between the prover and verifier).

It is known that IP = PSPACE.
[Lund, Fortnow, Karloff, & Nisan 1990; Shamir 1990]

Both classes are highly robust with respect to choices of error bounds.

2. Quantum interactive proof systems.

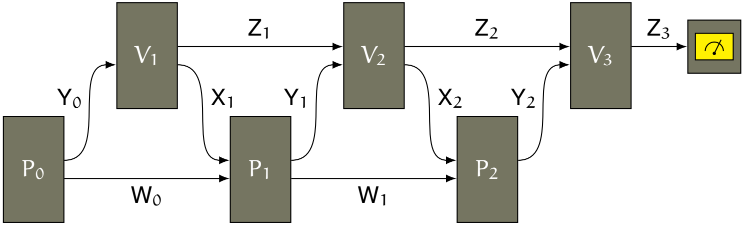
Definition of the model and complexity classes, relationships to other classes.

Quantum interactive proof systems

The **quantum interactive proof system** model works exactly the same as the classical model, except that the prover and verifier may exchange and process quantum information.

General assumptions and notions of completeness and soundness are unchanged…

The model may be formalized using quantum circuits. Here is an illustration of an interaction (on a fixed input string):



The class QIP

We define complexity classes based on quantum interactive proofs as follows:

QIP	The class of decision problems having quantum interactive proof systems (allowing polynomially many messages to be exchanged between the prover and verifier).
QIP(m)	The class of decision problems having quantum interactive proof systems, where at most m messages are exchanged in total.

It holds that

$$\text{QIP}(3) = \text{QIP} = \text{PSPACE};$$

quantum interactive proof systems are no more powerful than classical ones, but offer a reduction in the number of required required.

Diagram of Classes

$$\begin{tikzcd}
 & \mathsf{EXP} \arrow[d, dash] & \\
 & \mathsf{PSPACE} = \mathsf{QIP} \arrow[dl, dash] \arrow[dr, dash] & \\
\mathsf{PP} \arrow[d, dash] & & \mathsf{QIP}(2) \arrow[dll, dash] \arrow[d, dash] \\
\mathsf{QMA} = \mathsf{QIP}(1) \arrow[d, dash] \arrow[drr, dash] & & \mathsf{AM} \arrow[d, dash] \\
\mathsf{BQP} \arrow[d, dash] & & \mathsf{MA} \arrow[dll, dash] \arrow[d, dash] \\
\mathsf{BPP} \arrow[dr, dash] & & \mathsf{NP} \arrow[dl, dash] \\
 & \mathsf{P} &
\end{tikzcd}$$

Upper bounds by classical simulations

To prove **upper bounds** (i.e., limitations) on the power of quantum interactive proof systems, we consider **classical simulations** of quantum interactive proofs.

General goal: Given that A is a decision problem having a quantum interactive proof system (of some particular variety), find a classical algorithm for A obeying one of more resource constraints of interest.

Several results of this type make use of **semidefinite programming**.

Examples:

QIP $\subseteq$ EXP [KITAEV & W. 2000].

QIP = PSPACE [JAIN, JI, UPADHYAY, & W. 2009].

QRG = EXP [GUTOSKI & W. 2006].

QRG(2) = PSPACE [GUTOSKI & WU 2010].

QIP_{log} = BQP [BEIGI, SHOR, & W. 2010].

3. Semidefinite programming.

Definitions, a simple example, weak and strong duality, summary of algorithms.

Some linear algebra notation

Let $\mathrm{L}(\mathcal{X},\mathcal{Y})$ denote the set of all linear mappings (or **operators**) from a vector space $\mathcal{X}$ to a vector space $\mathcal{Y}$, and let $\mathrm{L}(\mathcal{X})$ denote $\mathrm{L}(\mathcal{X},\mathcal{X})$.

The **adjoint** (or conjugate transpose) of an operator A will be denoted A^* (as opposed to $A^\dagger$).

For a given finite-dimensional Hilbert space $\mathcal{X}$, we will let $\mathrm{Herm}(\mathcal{X})$, $\mathrm{Pos}(\mathcal{X})$, and $\mathrm{D}(\mathcal{X})$ denote the sets of **Hermitian operators**, **positive semidefinite operators**, and **density operators** acting on $\mathcal{X}$.

If $A, B \in \mathrm{Herm}(\mathcal{X})$, then the notations $A \leq B$ and $B \geq A$ mean that $B - A \in \mathrm{Pos}(\mathcal{X})$.

When we refer to the **inner product** of two operators $A, B \in \mathrm{L}(\mathcal{X},\mathcal{Y})$, we mean the Hilbert–Schmidt inner product:

$$\langle A, B\rangle = \mathrm{Tr}\big(A^* B\big).$$

Linear maps on spaces of operators

Linear maps of the form

$$\Phi : \mathrm{L}(\mathcal{X}) \rightarrow \mathrm{L}(\mathcal{Y})$$

are important in quantum information theory. (E.g., **quantum channels** are important examples of maps of this form.)

For such a map, we define the **adjoint map** $\Phi^* : \mathrm{L}(\mathcal{Y}) \rightarrow \mathrm{L}(\mathcal{X})$ to be the unique map that satisfies

$$\langle Y, \Phi(X) \rangle = \langle \Phi^*(Y), X \rangle$$

for all $X \in \mathrm{L}(\mathcal{X})$ and $Y \in \mathrm{L}(\mathcal{Y})$.

A map Φ of the form above is said to be **Hermiticity-preserving** if $\Phi(X) \in \mathrm{Herm}(\mathcal{Y})$ for all $X \in \mathrm{Herm}(\mathcal{X})$.

Semidefinite programs

A **semidefinite program** (or **SDP** for short) is a pair of optimization problems, specified by a triple (Φ, A, B), where:

1. $\Phi : \mathrm{L}(\mathcal{X}) \rightarrow \mathrm{L}(\mathcal{Y})$ is a Hermiticity-preserving linear map,
2. $A \in \mathrm{Herm}(\mathcal{X})$, and
3. $B \in \mathrm{Herm}(\mathcal{Y})$.

The pair of optimization problems is as follows:

	Primal problem		Dual problem
maximize:	$\langle A, X \rangle$	minimize:	$\langle B, Y \rangle$
subject to:	$\Phi(X) = B$	subject to:	$\Phi^*(Y) \geq A$
	$X \in \mathrm{Pos}(\mathcal{X})$		$Y \in \mathrm{Herm}(\mathcal{Y})$

(There are other equivalent formulations of these problems, including the so-called **standard form**, that you might be familiar with.)

Optimal values

The **optimal value** of the **primal problem**

$$\begin{aligned} \text{maximize:} \quad & \langle A, X \rangle \\ \text{subject to:} \quad & \Phi(X) = B \\ & X \in \mathrm{Pos}(\mathcal{X}) \end{aligned}$$

is defined as

$$\alpha = \sup\bigl\{ \langle A, X \rangle \,:\, X \in \mathrm{Pos}(\mathcal{X}),\ \Phi(X) = B \bigr\}.$$

Similarly, the **optimal value** of the **dual problem**

$$\begin{aligned} \text{minimize:} \quad & \langle B, Y \rangle \\ \text{subject to:} \quad & \Phi^*(Y) \geq A \\ & Y \in \mathrm{Herm}(\mathcal{Y}) \end{aligned}$$

is defined as

$$\beta = \inf\bigl\{ \langle B, Y \rangle \,:\, Y \in \mathrm{Herm}(\mathcal{Y}),\ \Phi^*(Y) \geq A \bigr\}.$$

Example

As a very simple example, we may take:

$$\mathcal{X} = \mathbb{C}^n \text{ (arbitrary } n) \qquad \text{and} \qquad \mathcal{Y} = \mathbb{C},$$

let $A \in \operatorname{Herm}(\mathcal{X})$ be arbitrary, and let $\Phi = \operatorname{Tr}$ and $B = 1$.

The primal problem looks like this:

$$\boxed{\begin{array}{rl} \text{maximize:} & \langle A, X \rangle \\ \text{subject to:} & \Phi(X) = B \\ & X \in \operatorname{Pos}(\mathcal{X}) \end{array}} \xrightarrow{\text{simplify}} \boxed{\begin{array}{rl} \text{maximize:} & \langle A, \rho \rangle \\ \text{subject to:} & \rho \in \mathrm{D}(\mathcal{X}) \end{array}}$$

The optimal value of the primal problem is

$$\alpha = \lambda_1(A)$$

(the **largest eigenvalue** of A).

Example (continued)

The dual problem looks like this:

$$\boxed{\begin{array}{rl} \text{minimize:} & \langle B, Y\rangle \\ \text{subject to:} & \Phi^*(Y) \geq A \\ & Y \in \mathrm{Herm}(\mathcal{Y}) \end{array}} \xrightarrow{\text{simplify}} \boxed{\begin{array}{rl} \text{minimize:} & \lambda \\ \text{subject to:} & A \leq \lambda \mathbb{1} \\ & \lambda \in \mathbb{R} \end{array}}$$

Here we used the fact that the adjoint of the trace map is given by

$$\mathrm{Tr}^*(\lambda) = \lambda \mathbb{1}$$

for all $\lambda \in \mathbb{C}$. This must be so, because $\langle \lambda, \mathrm{Tr}(X)\rangle = \langle \lambda\mathbb{1}, X\rangle$.

The optimal value of the dual problem is

$$\beta = \lambda_1(A).$$

It is not a surprise that $\alpha = \beta$; this usually happens.

Duality

For a semidefinite program specified by (Φ, A, B), we have defined the optimal primal value α and the optimal dual value β as

$$\begin{aligned}\alpha &= \sup\bigl\{\langle A, X\rangle \,:\, X \in \mathrm{Pos}(\mathcal{X}),\ \Phi(X) = B\bigr\},\\ \beta &= \inf\bigl\{\langle B, Y\rangle \,:\, Y \in \mathrm{Herm}(\mathcal{Y}),\ \Phi^*(Y) \geq A\bigr\}.\end{aligned}$$

Weak duality: it $\underline{\text{always}}$ holds that $\alpha \leq \beta$.

To see this, suppose X is **primal feasible** and Y is **dual feasible**:

$$\begin{gathered} X \in \mathrm{Pos}(\mathcal{X}) \text{ and } \Phi(X) = B,\\ Y \in \mathrm{Herm}(\mathcal{Y}) \text{ and } \Phi^*(Y) \geq A. \end{gathered}$$

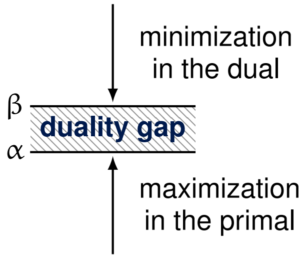
It follows that

$$\langle A, X\rangle \leq \langle \Phi^*(Y), X\rangle = \langle Y, \Phi(X)\rangle = \langle Y, B\rangle = \langle B, Y\rangle.$$

(The inequality follows from $\langle P, Q\rangle \geq 0$ for $P, Q \geq 0$.)

Duality (continued)

The relationship between the primal and dual optimal values is represented by this figure:



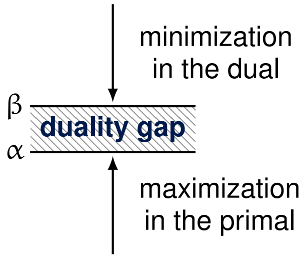
If we ever find a primal feasible X and a dual feasible Y such that

$$\langle A, X \rangle = \langle B, Y \rangle,$$

then we know we have found the optimal values; and moreover we have $\alpha = \beta$ (a condition known as **strong duality**).

Duality (continued)

The relationship between the primal and dual optimal values is represented by this figure:



Also note:

$$Y \text{ dual feasible} \Rightarrow \alpha \leq \langle B, Y \rangle$$
$$X \text{ primal feasible} \Rightarrow \beta \geq \langle A, X \rangle.$$

Every feasible point provides a bound on the complementary problem.

Strong duality

Strong duality refers to the situation in which the optimal primal and dual values are equal: $\alpha = \beta$. It is possible to construct SDPs for which **strong duality fails**. (We could have $\alpha = 0$ and $\beta = 1$, for instance.)

However, for most SDPs that arise naturally, strong duality will hold.

Slater conditions: strong duality holds under either of the following conditions:

1. The primal is **feasible** (there exists $X \in \mathrm{Pos}(\mathcal{X})$ with $\Phi(X) = B$) and the dual is **strictly feasible** (there exists $Y \in \mathrm{Herm}(\mathcal{Y})$ with $\Phi^*(Y) > A$).

   (Moreover, the optimal primal value is achieved in this case.)

2. The primal is **strictly feasible** (there exists $X > 0$ with $\Phi(X) = B$) and the dual is **feasible** (there exists $Y \in \mathrm{Herm}(\mathcal{Y})$ with $\Phi^*(Y) \geq A$).

   (Moreover, the optimal dual value is achieved in this case.)

More general forms of SDPs

It is common that one sees semidefinite programs having multiple variables and both equality and inequality constraints, e.g.,

$$\begin{aligned} \text{maximize:} \quad & \langle A_0, X_0 \rangle + \langle A_1, X_1 \rangle \\ \text{subject to:} \quad & X_0 + X_1 \leq B, \quad X_0, X_1 \in \mathrm{Pos}(\mathcal{X}). \end{aligned}$$

We can convert such SDPs to ones of the form described before by using **block matrices** and **slack variables**:

$$\text{maximize:} \quad \left\langle \begin{pmatrix} A_0 & 0 & 0 \\ 0 & A_1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} X_0 & \cdot & \cdot \\ \cdot & X_1 & \cdot \\ \cdot & \cdot & Y \end{pmatrix} \right\rangle$$

$$\text{subject to:} \quad \Phi \begin{pmatrix} X_0 & \cdot & \cdot \\ \cdot & X_1 & \cdot \\ \cdot & \cdot & Y \end{pmatrix} \overset{\text{def}}{=} X_0 + X_1 + Y = B, \quad \begin{pmatrix} X_0 & \cdot & \cdot \\ \cdot & X_1 & \cdot \\ \cdot & \cdot & Y \end{pmatrix} \geq 0.$$

Algorithms for approximating semidefinite programs

There exist efficient algorithms for approximating optimal solutions of semidefinite programs, provided that they meet certain conditions.

1. **Ellipsoid method:** Not useful in practice, but provides a provably polynomial-time algorithm for a very general class of SDPs (having “well-bounded” feasible sets).
2. **Interior point algorithms:** These algorithms are useful in practice. (The CVX system for MATLAB provides a very nice, easy-to-use implementation.)
3. **Matrix multiplicative weights update method:** this is a “meta-algorithm” that describes certain highly-efficient algorithms for special classes of semidefinite programs.

Not all semidefinite programs can be solved efficiently. Some have exponential-size solutions, and others are related to the notorious sum of square-roots problem.

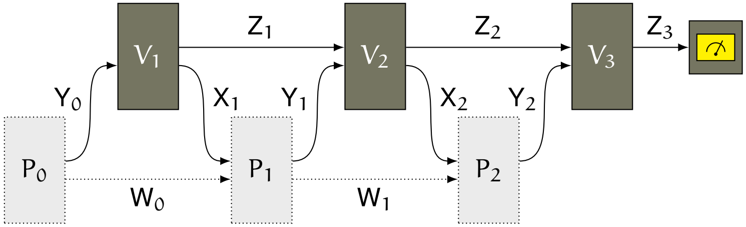
4. An SDP for quantum interactive proofs.

One of two general ways to represent quantum interactive proof systems by semidefinite programs, yields QIP $\subseteq$ EXP and perfect parallel repetition.

Optimization over compatible provers

Let us begin by considering the general problem at hand.

From any quantum interactive proof on a **fixed input string**, we obtain a **quantum verification process**, taking inputs and producing outputs over the course of multiple rounds:

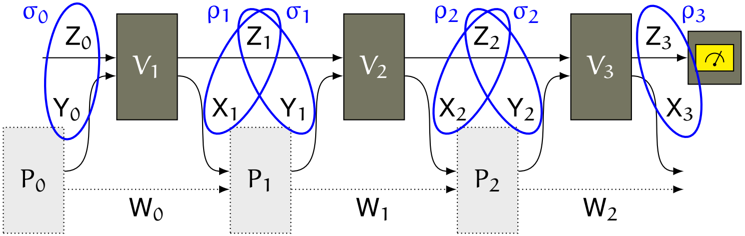


The goal is to optimize over all possible **compatible provers** to determine the **maximum probability** with which a particular measurement outcome will be produced.

States of the registers the verifier touches

To phrase the optimization problem as an SDP, we make the simplifying assumption that the verifier's transformations $V_1, \ldots, V_n$ are **unitary**.

Consider an optimization over the possible **states** of the registers the verifier touches (for each choice of **co-existing** registers).

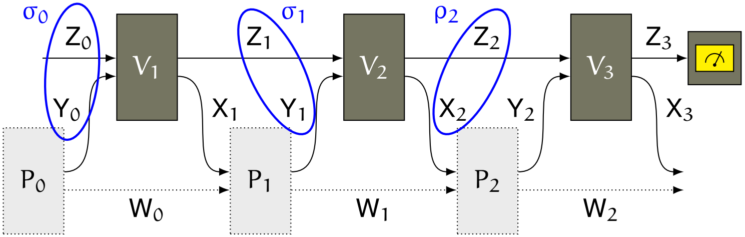


Certain constraints must hold. For example, if $\rho_1 \in \mathrm{D}(\mathcal{X}_1 \otimes \mathcal{Z}_1)$ and $\sigma_1 \in \mathrm{D}(\mathcal{Y}_1 \otimes \mathcal{Z}_1)$ represent states of (X_1, Z_1) and (Y_1, Z_1), then

$$\mathrm{Tr}_{\mathcal{X}_1}(\rho_1) = \mathrm{Tr}_{\mathcal{Y}_1}(\sigma_1).$$

States of the registers the verifier touches

Another type of constraint corresponds to a consistency between the states before and after each verifier transformation.



For each $k = 1, \ldots, n$ (for n the number of verifier operations) we must have

$$V_k \sigma_{k-1} V_k^* = \rho_k.$$

Finally, σ_0 must be a valid starting state of (Y_0, Z_0). (In other words, Z_0 must be initialized to its starting state.)

Quantum prover optimization as an SDP

Now, if we optimize over all states obeying these constraints (where the objective function is the probability of acceptance), we get an SDP:

$$\begin{aligned}
\text{maximize:}\quad & \left\langle V_n^* \Pi V_n, \sigma_{n-1} \right\rangle \\
\text{subject to:}\quad & \operatorname{Tr}_{\mathcal{Y}_0}(\sigma_0) = |0\cdots 0\rangle \langle 0\cdots 0|, \\
& \operatorname{Tr}_{\mathcal{Y}_1}(\sigma_1) = \operatorname{Tr}_{\mathcal{X}_1}\left(V_1 \sigma_0 V_1^*\right), \\
& \quad\vdots \\
& \operatorname{Tr}_{\mathcal{Y}_{n-1}}(\sigma_{n-1}) = \operatorname{Tr}_{\mathcal{X}_{n-1}}\left(V_{n-1} \sigma_{n-2} V_{n-1}^*\right), \\
& \sigma_0 \in \mathrm{D}(\mathcal{Y}_0 \otimes \mathcal{Z}_0) \\
& \quad\vdots \\
& \sigma_{n-1} \in \mathrm{D}(\mathcal{Z}_{n-1} \otimes \mathcal{Y}_{n-1}).
\end{aligned}$$

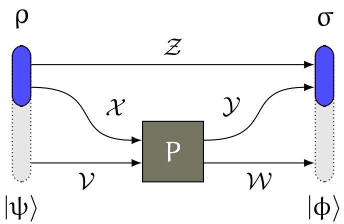
(Π is a measurement operator corresponding to acceptance. Because $\rho_k = V_k \sigma_{k-1} V_k^*$, we don't really need $\rho_1, \ldots, \rho_n$ as variables.)

All feasible points represent valid provers

One important question: does every feasible point represent a valid strategy for the prover?

Remarkably, the answer is **yes**—this fact follows from the **unitary equivalence of purifications** (and this is why we have assumed the verifier applies unitary operations).

Suppose the prover wishes to perform a transformation like this:



If the prover holds a **purification** $|\psi\rangle$ of ρ, then the transformation is possible, provided that $\mathrm{Tr}_{\mathcal{X}}(\rho) = \mathrm{Tr}_{\mathcal{Y}}(\sigma)$.

Consequence: QIP $\subseteq$ EXP

As a consequence of the SDP formulation just described, we obtain QIP $\subseteq$ EXP. In more detail:

- We are given a decision problem $A \in$ QIP, and we wish to prove $A \in$ EXP (i.e., there exists an exponential-time algorithm, for a classical deterministic Turing machine, that decides A).
- On any given input string x, we compute a description of the SDP in deterministic exponential time. (The description of the SDP will be exponentially large in x, and the matrices required to specify it can be obtained from the circuit description of the verifier.)
- Finally, we run a polynomial-time SDP algorithm (such as the ellipsoid algorithm), and accept or reject x depending on the approximate optimal value obtained.

(One must prove that the SDP possesses the properties required for the chosen algorithm to run in polynomial time, but this is not difficult.)

Dual form of the SDP

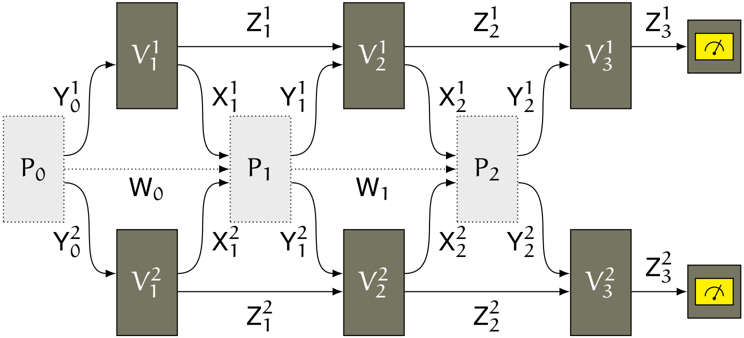
If we compute (and simplify) the dual form of the SDP for optimizing over states of a quantum interactive proof, we obtain this problem:

$$\begin{aligned}
\text{minimize:}\quad & \lambda \\
\text{subject to:}\quad & \lambda\mathbb{1} \geq (\langle 0\cdots 0| \otimes \mathbb{1})V_1^*(Z_1 \otimes \mathbb{1})V_1(|0\cdots 0\rangle \otimes \mathbb{1}), \\
& Z_1 \otimes \mathbb{1} \geq V_2^*(Z_2 \otimes \mathbb{1})V_2, \\
& \qquad\vdots \\
& Z_{n-2} \otimes \mathbb{1} \geq V_{n-1}^*(Z_{n-1} \otimes \mathbb{1})V_{n-1}, \\
& Z_{n-1} \otimes \mathbb{1} \geq V_n^*\Pi V_n, \\
& \lambda \in \mathbb{R} \\
& Z_1 \in \mathrm{Herm}(\mathcal{Z}_1),\ \ldots,\ Z_{n-1} \in \mathrm{Herm}(\mathcal{Z}_{n-1}).
\end{aligned}$$

Strong duality can be verified through the Slater conditions.

Application: analysis of parallel repetition

Consider the problem of **parallel repetition**: we run two (or more) quantum verifiers simultaneously (i.e., in parallel).



Nothing forces a prover to treat the proof systems independently. What is the optimal probability with which both (or all) of the verifiers can be made to accept?

Proving parallel repetition

Let $\omega(V)$ denote the maximum acceptance probability of a verifier V (for a prover interacting with V in isolation).

It is reasonable to conjecture that

$$\omega(V^1 \otimes \cdots \otimes V^m) = \omega(V^1) \cdots \omega(V^m). \tag{1}$$

(An analogous fact holds for classical single-prover interactive proofs, but fails for multi-prover interactive proofs.)

It is evident that

$$\omega(V^1 \otimes \cdots \otimes V^m) \geq \omega(V^1) \cdots \omega(V^m);$$

one of the prover's options is to play optimally against each verifier independently. To prove (1), it therefore suffices to prove

$$\omega(V^1 \otimes \cdots \otimes V^m) \leq \omega(V^1) \cdots \omega(V^m).$$

Proving parallel repetition

The inequality

$$\omega(V^1 \otimes \cdots \otimes V^m) \leq \omega(V^1) \cdots \omega(V^m)$$

can be proved using the dual form of our SDP. (Let us focus on the case $m = 2$ for simplicity.)

Consider the dual form of the SDP for V^1, V^2, and $V^1 \otimes V^2$. Strong duality holds for all three SDPs (and in particular for V^1 and V^2).

Take optimal dual solutions for the first two SDPs:

1. $\lambda^1 = \omega(V^1)$, $Z_1^1, \ldots, Z_{n-1}^1$ optimal dual solution for the V^1 SDP.
2. $\lambda^2 = \omega(V^2)$, $Z_1^2, \ldots, Z_{n-1}^2$ optimal dual solution for the V^2 SDP.

One can prove (using a simple and well-known operator inequality) that

$$\lambda = \lambda^1 \lambda^2, \quad Z_1 = Z_1^1 \otimes Z_1^2, \quad \ldots, \quad Z_{n-1} = Z_{n-1}^1 \otimes Z_{n-1}^2$$

is feasible for the $V^1 \otimes V^2$ SDP, and thus $\omega(V^1 \otimes V^2) \leq \omega(V^1)\omega(V^2)$.

5. QIP = PSPACE.

A high-level overview of the idea behind the proof.

PSPACE and bounded-depth Boolean circuits

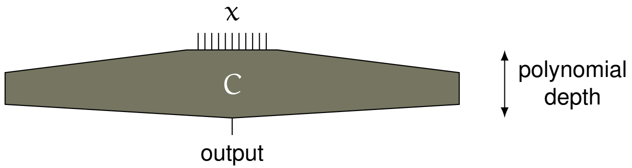
To prove QIP $=$ PSPACE, we just need to prove QIP $\subseteq$ PSPACE. (The other containment is easy: PSPACE $=$ IP $\subseteq$ QIP.)

Working directly with space-bounded algorithms is often not very natural or amenable to our "usual" intuition about algorithms. . .

To get around this, we make use of an alternative characterization of PSPACE in terms of **bounded-depth Boolean circuits**:

$$\text{PSPACE} = \text{NC}(\text{poly}).$$

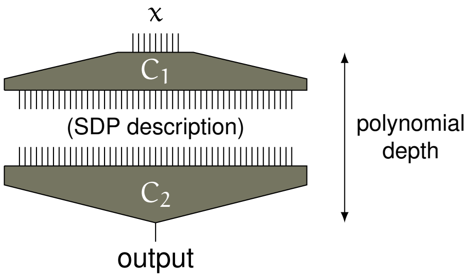
NC(poly) refers to the class of problems solvable by **exponentially large** Boolean circuits with **polynomial depth**.



A natural approach

Suppose that A is a decision problem having a quantum interactive proof system. Our goal is to prove $A \in \text{NC(poly)}$.

We might hope to obtain an NC(poly) circuit for A by solving an SDP along simular lines to before, splitting the computation as so:



In fact it is not difficult to perform C_1 as suggested, using well-known **parallel** algorithms for matrix operations. The difficult part is C_2...

Parallel algorithms for SDPs

To perform the second part of the computation, represented by the circuit C_2 in the picture, in such a way that the overall circuit depth is polynomial in x, we need a **parallel algorithm** for solving SDPs.

Unfortunately, general SDPs cannot be approximated well in parallel, unless something unexpected happens in complexity theory ($\mathrm{NC} = \mathrm{P}$).

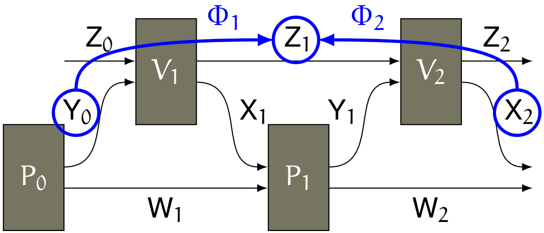
We will therefore need a **special purpose** parallel SDP algorithm for the task at hand. Our first step will be to obtain the simplest SDP formulation we can for the problem at hand.

There are multiple ways that are known to work:

- The original proof used a so-called **public-coin** quantum interactive proof system (requiring just a single coin-flip).
- In this talk we will consider an alternative formulation (due to Xiaodi Wu) based on a **min-max formulation** of the problem.

QIP and maximum output fidelity

$A \in$ QIP, so there is a 3 message quantum interactive proof system for A with perfect completenes and small soundness error probability.



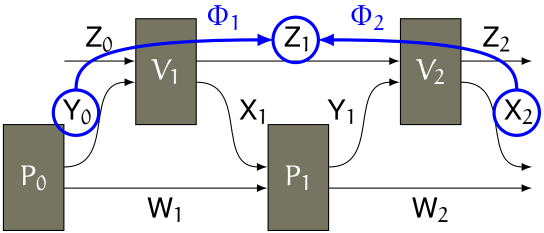
Define channels Φ_1 and Φ_2 as

$$\Phi_1(\rho_1) = \operatorname{Tr}_{\mathcal{X}_1}\left(V_1(|0\cdots0\rangle\langle0\cdots0| \otimes \rho_1)V_1^*\right)$$
$$\Phi_2(\rho_2) = \operatorname{Tr}_{\mathcal{Y}_1}\left(V_2^*(|1\rangle\langle1| \otimes \rho_2)V_2\right)$$

for every $\rho_1 \in \mathrm{D}(\mathcal{Y}_0)$ and $\rho_2 \in \mathrm{D}(\mathcal{X}_2)$.

QIP and maximum output fidelity

$A \in$ QIP, so there is a 3 message quantum interactive proof system for A with perfect completenes and small soundness error probability.



A convenient fact:

$$\omega(V) = \max_{\rho_1,\rho_2} F(\Phi_1(\rho_1), \Phi_2(\rho_2))^2$$

where $F(\sigma_1, \sigma_2) = \left\| \sqrt{\sigma_1}\sqrt{\sigma_2} \right\|_1$ is the **fidelity function**.

QIP as a min-max quantity

We can now use our expression

$$\omega(V) = \max_{\rho_1,\rho_2} \mathrm{F}(\Phi_1(\rho_1), \Phi_2(\rho_2))^2$$

to represent the problem as a min-max approximation.

Define a map Ξ so that

$$\Xi(\rho_1 \otimes \rho_2) = \Phi_1(\rho_1) - \Phi_2(\rho_2)$$

and define

$$\eta = \min_{\sigma} \max_{P} \langle P, \Xi(\sigma) \rangle$$

(where the minimum is over density operators σ and the maximum is over operators with $0 \leq P \leq \mathbb{1}$). Key fact:

$$\begin{aligned} x \text{ is a yes input } &\Rightarrow \ \eta = 0 \\ x \text{ is a no input } &\Rightarrow \ \eta \geq 1/2 \end{aligned}$$

Parallel algorithm for approximating min-max value

The following algorithm approximates the min-max quantity

$$\eta = \min_{\sigma} \max_{P} \big\langle P, \Xi(\sigma) \big\rangle.$$

1. Set $X_1 = \mathbb{1}$ and T sufficiently large (polynomial in x).
2. For each $t = 1, \ldots, T$, let

   $$\sigma_t = \frac{X_t}{\operatorname{Tr}(X_t)},$$

   let Π_t be the projection operator corresponding to the positive eigenspace of $\Xi(\sigma_t)$, and let

   $$X_{t+1} = \exp\big(-\varepsilon\, \Xi^{*}\big(\Pi_1 + \cdots + \Pi_t\big)\big).$$

3. Output

   $$\frac{1}{T} \sum_{t=1}^{T} \big\langle \Pi_t, \Xi(\sigma_t) \big\rangle.$$

Analogy to the probabilistic experts algorithm

The specific way that the algorithm improves its guesses $\sigma_1, \sigma_2, \sigma_3, \ldots$ is analogous to the **probabilistic experts algorithm**. . .

We represent confidence in a collection of n experts with a probability vector $(p_1, \ldots, p_n)$, initially set to $(1/n, \ldots, 1/n)$.

For a sequence of games or investments, we base our own predictions on the **weighted majority prediction** of the experts (in accordance with our confidence vector).

After each prediction, we penalize each expert that was wrong:

$$p_k \to (1 - \varepsilon)p_k$$

and then renormalize $(p_1, \ldots, p_n)$.

The algorithm very quickly approximates the expected performance of the **leading expert**.

Noncommutative updates

The update rule in the min-max algorithm is more complicated than the simple experts update rule. . . this is because of the noncommutative nature of the problem.

Intuitively speaking, the operator $\Xi^*\big(\Pi_1 + \cdots + \Pi_t\big)$ in the update rule

$$X_{t+1} = \exp\big(-\varepsilon\,\Xi^*\big(\Pi_1 + \cdots + \Pi_t\big)\big)$$

represents a **penalty function** of sorts. . . we may think of it as an **observable** whose value is large for bad choices of σ, small for good choices of σ. Setting X_{t+1} in this way and normalizing is similar in spirit to repeatedly penalizing experts and renormalizing the probability vector representing confidence.

Similar to the experts algorithm, the min-max algorithm converges very quickly to an approximation of the min-max value. **Parallelizing every step** (using known methods) gives a parallel algorithm to approximate the min-max value η, due to this very fast convergence.

6. A different SDP formulation of QIP.

An alternative SDP formulation based on Choi operators, generalization to other interactions, applications to quantum complexity and coin-flipping.

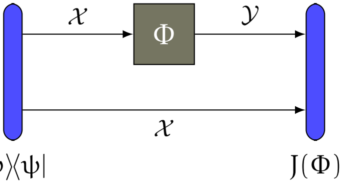
Quantum process optimization as an SDP

A second way to represent prover optimization as an SDP is based on **Choi operators** associated with linear maps of the form $\Phi : \mathrm{L}(\mathcal{X}) \to \mathrm{L}(\mathcal{Y})$.

Given such a map, its Choi operator is defined as

$$J(\Phi) = \sum_{a,b} \Phi(|a\rangle\langle b|) \otimes |a\rangle\langle b| \, .$$

For a quantum channel Φ, one may view this operator as being the unnormalized state obtained by applying Φ to half of the unnormalized maximally entangled state $|\psi\rangle = \sum_a |a\rangle |a\rangle$:



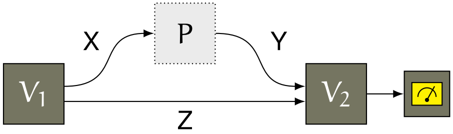
Quantum process optimization as an SDP

It is well-known that a linear map $\Phi : \mathrm{L}(\mathcal{X}) \rightarrow \mathrm{L}(\mathcal{Y})$ is a channel if and only if

(1) $\mathrm{J}(\Phi)$ is positive semidefinite, and

(2) $\mathrm{Tr}_{\mathcal{Y}}(\mathrm{J}(\Phi)) = \mathbb{1}_{\mathcal{X}}$.

Consider a single-round proof system on a fixed input string:

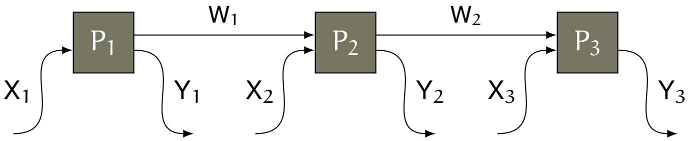


For Q determined in a simple way by V_1, V_2, and the measurement, the optimal acceptance probability is given by this SDP:

$$\begin{aligned} &\text{maximize:} && \langle Q, X \rangle \\ &\text{subject to:} && \mathrm{Tr}_{\mathcal{Y}}(X) = \mathbb{1}_{\mathcal{X}},\ X \in \mathrm{Pos}(\mathcal{Y} \otimes \mathcal{X}). \end{aligned}$$

Generalization to multiple rounds

Now consider a multiple-round prover strategy, like this one:



One may (momentarily) ignore the time-ordering, regarding it as a map of the form $\Phi : \mathrm{L}(\mathcal{X}_1 \otimes \mathcal{X}_2 \otimes \mathcal{X}_3) \rightarrow \mathrm{L}(\mathcal{Y}_1 \otimes \mathcal{Y}_2 \otimes \mathcal{Y}_3)$.

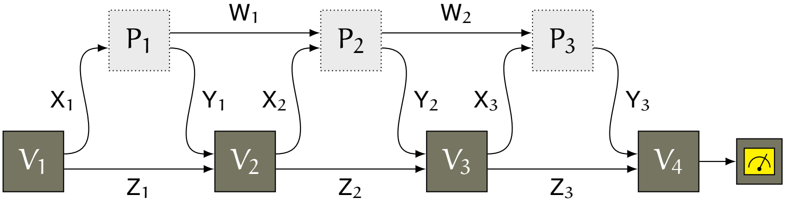
A **perfect characterization** of the possible Choi operators $X_3 = J(\Phi)$ that may be obtained in this way:

$$\begin{aligned} \mathrm{Tr}_{\mathcal{Y}_3}(X_3) &= X_2 \otimes \mathbb{1}_{\mathcal{X}_3}, & X_3 &\in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{Y}_2 \otimes \mathcal{Y}_3 \otimes \mathcal{X}_1 \otimes \mathcal{X}_2 \otimes \mathcal{X}_3), \\ \mathrm{Tr}_{\mathcal{Y}_2}(X_2) &= X_1 \otimes \mathbb{1}_{\mathcal{X}_2}, & X_2 &\in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{Y}_2 \otimes \mathcal{X}_1 \otimes \mathcal{X}_2), \\ \mathrm{Tr}_{\mathcal{Y}_1}(X_1) &= \mathbb{1}_{\mathcal{X}_1}, & X_1 &\in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{X}_1). \end{aligned}$$

[GUTOSKI & W. 2006; CHIRIBELLA, D'ARIANO, & PERINOTTI 2007]

SDPs from the second method

Returning to the sort of optimization problem discussed before. . .



For Q determined from $V_1, \ldots, V_4$ and the measurement, the optimal acceptance probability is represented by this SDP:

maximize: $\langle Q, X_3 \rangle$

subject to:

$$\begin{aligned}
&\mathrm{Tr}_{\mathcal{Y}_3}(X_3) = X_2 \otimes \mathbb{1}_{\mathcal{X}_3}, && X_3 \in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{Y}_2 \otimes \mathcal{Y}_3 \otimes \mathcal{X}_1 \otimes \mathcal{X}_2 \otimes \mathcal{X}_3),\\
&\mathrm{Tr}_{\mathcal{Y}_2}(X_2) = X_1 \otimes \mathbb{1}_{\mathcal{X}_2}, && X_2 \in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{Y}_2 \otimes \mathcal{X}_1 \otimes \mathcal{X}_2),\\
&\mathrm{Tr}_{\mathcal{Y}_1}(X_1) = \mathbb{1}_{\mathcal{X}_1}, && X_1 \in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{X}_1).
\end{aligned}$$

Complexity applications of the second SDP method

Using the second SDP formulation, we obtain an alternative proof that

$$\mathsf{QIP} \subseteq \mathsf{EXP}.$$

The second SDP method may be used to obtain an exponential-time simulation even the verifier interacts with two **competing provers**:

$$\mathsf{QRG} = \mathsf{EXP}.$$

When combined with **quantum tomography**, a quantum simulation is obtained when only a logarithmic number of qubits are exchanged:

$$\mathsf{QIP}_{\mathsf{log}} = \mathsf{BQP}.$$

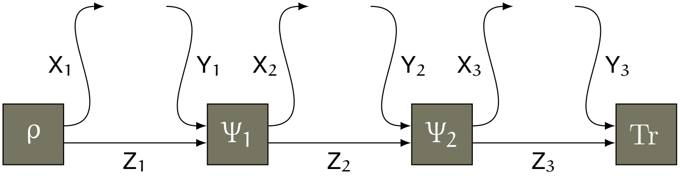
The second SDP formulation is also useful for proving other facts about quantum interactions in different settings.

Dual form of maximum output probability SDP

The dual form of the SDP described a couple of slides ago is remarkably simple:

$$\begin{aligned} \text{minimize:} \quad & \lambda \\ \text{subject to:} \quad & Q \leq \lambda R \end{aligned}$$

for R ranging over all operators $R \in \mathrm{Pos}(\mathcal{Y}_1 \otimes \mathcal{Y}_2 \otimes \mathcal{Y}_3 \otimes \mathcal{X}_1 \otimes \mathcal{X}_2 \otimes \mathcal{X}_3)$ that arise from a similar representation to Q, for processes of this form:



From this fact, we may obtain a very simple proof of Kitaev's bound on strong quantum coin-flipping. . .

Strong quantum coin-flipping

A **strong quantum coin-flipping protocol** with bias ε is an interaction between two (honest) players Alice and Bob, both having output sets $\{0, 1, \text{abort}\}$.

Required properties:

1. The interaction between the honest players Alice and Bob produces the same outcome $b \in \{0, 1\}$ for both players, with probability $1/2$ for each outcome.
2. If one of the players does not follow the protocol but the other does, neither of the outcomes $b \in \{0, 1\}$ is output by the honest player with probability greater than $1/2 + \varepsilon$.

Theorem (Kitaev): All strong quantum coin-flipping protocols have bias at least

$$\frac{1}{\sqrt{2}} - \frac{1}{2} \approx 0.207.$$

Simple proof of Kitaev's coin-flipping bound

Consider any quantum coin-flipping protocol (any number of rounds). Let the honest strategies for Alice and Bob be represented by operators

$$\{A_0,\ A_1,\ A_{\text{abort}}\} \quad \text{and} \quad \{B_0,\ B_1,\ B_{\text{abort}}\}.$$

Let p be the maximum probability a cheating Bob can force outcome 0 for Alice. This implies that there must be a (cheating) strategy R for Alice such that $A_0 \leq pR$.

The strategy R would cause Bob to output 0 with probability

$$\langle R, B_0 \rangle \geq \frac{1}{p} \langle A_0, B_0 \rangle = \frac{1}{2p}.$$

For $p > 0$ we have

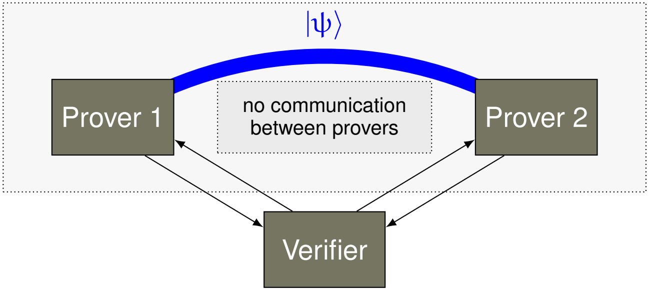
$$\max\left\{p,\ \frac{1}{2p}\right\} \geq \frac{1}{\sqrt{2}};$$

one of the players can force outcome 0 with probability at least $\frac{1}{\sqrt{2}}$. □

Conclusion

Semidefinite programming is an indispensable tool in the study of quantum interactive proof systems (and in quantum information and computation more generally).

The next frontier in the study of quantum interactive proof systems is the multi-prover quantum interactive proof system model.



Combined actions of entanged provers are not likely to be represented succinctly by semidefinite programs. . .