Programming Telepathy: Implementing Quantum Non-locality Games

Anya Tafliovich, Eric C.R. Hehner

[1]University of Toronto, Toronto ON M5S 3G4, Canada

{anya,hehner}@cs.toronto.edu

Abstract. *Quantum pseudo-telepathy is an intriguing phenomenon which results from the application of quantum information theory to communication complexity. To demonstrate this phenomenon researchers in the field of quantum communication complexity devised a number of quantum non-locality games. The setting of these games is as follows: the players are separated so that no communication between them is possible and are given a certain computational task. When the players have access to a quantum resource called entanglement, they can accomplish the task: something that is impossible in a classical setting. To an observer who is unfamiliar with the laws of quantum mechanics it seems that the players employ some sort of telepathy; that is, they somehow exchange information without sharing a communication channel.*
This paper provides a formal framework for specifying, implementing, and analyzing quantum non-locality games.

1. Introduction

The work develops a formal framework for specifying, implementing, and analyzing quantum pseudo-telepathy: an intriguing phenomenon which manifests itself when quantum information theory is applied to communication complexity. To demonstrate this phenomenon researchers in the field of quantum communication complexity devised a number of quantum non-locality games. The setting of these games is as follows: the players are separated so that no communication between them is possible and are given a certain computational task. When the players have access to a quantum resource called entanglement, they can accomplish the task: something that is impossible in a classical setting. To an observer who is unfamiliar with the laws of quantum mechanics it seems that the players employ some sort of telepathy; that is, they somehow exchange information without sharing a communication channel.

Quantum pseudo-telepathy, and quantum non-locality in general, are perhaps the most non-classical and the least understood aspects of quantum information processing. Every effort is made to gain information about the power of these phenomena. Quantum non-locality games in particular have been extensively used to prove separations between quantum and classical communication complexity. The need for a good framework for formal analysis of quantum non-locality is evident.

We look at quantum non-locality in the context of formal methods of program development, or programming methodology. This is the field of computer science concerned with applications of mathematics and logic to software engineering tasks. In particular, the formal methods provide tools to formally express specifications, prove correctness of implementations, and reason about various properties of specifications (e.g. implementability) and implementations (e.g. time and space complexity).

In this work the analysis on quantum non-locality is based on quantum predicative programming ([Tafliovich and Hehner 2006, Tafliovich 2004]), a recent generalization of the well-established predicative programming ([Hehner 1993, Hehner 2004]). It supports the style of program development in which each programming step is proved correct as it is made. We inherit the advantages of the theory, such as its generality, simple treatment of recursive programs, and time and space complexity. The theory of quantum programming provides tools to write both classical and quantum specifications, develop quantum programs that implement these specifications, and reason about their comparative time and space complexity all in the same framework.

Presenting new non-locality paradigms or new pseudo-telepathy games is not the subject of this work. Our goal is developing a formal framework that encompasses all aspects of quantum computation and information. Formal analysis of quantum algorithms, including their time complexity, is presented in [Tafliovich and Hehner 2006]. This paper focuses on formal analysis of non-locality paradigms; we choose known pseudo-telepathy games as illustrative examples of our formalism.

The rest of this work is organized as follows. Section 2 is a brief introduction to quantum predicative programming. The contribution of this work is Section 3 which introduces a formal framework for specifying, implementing, and analyzing quantum pseudo-telepathy and presents several examples of implementing and analyzing non-locality games. Section 4 states conclusions and outlines directions for future research. A brief introduction to quantum computing is included in the Appendix.

1.1. Our contribution and related work

This work attempts to bring together two areas of active research: the study of quantum non-locality and applications of formal methods to quantum information and computation. Currently, the two worlds rarely meet.

Quantum non-locality has been studied extensively first by physicists and lately by researchers in the fields of quantum information and quantum communication complexity. Since the work of Bell in 1964 ([Bell 1964]), researchers have been trying to provide an intuitive explanation of the genuinely non-classical behaviour produced by quantum mechanics. Today, quantum pseudo-telepathy games are considered one of the best and easiest to understand examples of these non-classical phenomena (e.g. [Galliard et al. 2003, Brassard et al. 2004, Brassard et al. 2005, Brassard et al. 2006]).

Formal approaches to quantum programming include the language qGCL [Sanders and Zuliani 2000, Zuliani 2004, Zuliani 2005], process algebraic approaches developed in [Adao and Mateus 2007, Lalire and Jorrand 2004, Jorrand and Lalire 2004], tools developed in the field of category theory by [Abramsky 2004, Abramsky and Coecke 2004, Abramsky and Duncan 2006, Coecke 2004, Selinger 2004], functional languages of [Arrighi and Dowek 2004, Arrighi and Dowek 2005, Altenkirch and Grattage 2005, Valiron 2004, van Tonder 2004], as well as work of [D'Hondt and Panangaden 2004, D'Hondt and Panangaden 2005], [Danos et al. 2005], and [Gay and Nagarajan 2005]. A detailed discussion of the work related to quantum predicative programming is presented in [Tafliovich and Hehner 2006]. Some researchers address the subject of formalizing quantum non-locality more directly than others (e.g. [Zuliani 2004]). To the best of our knowledge, formal approaches to reasoning about quantum pseudo-telepathy games have not been considered.

2. Quantum Predicative Programming

This section introduces the programming theory of our choice — quantum predicative programming. We briefly introduce parts of the theory necessary for understanding Section 3 of this work. For a course in predicative programming the reader is referred to [Hehner 1993]. An introduction to probabilistic predicative programming can be found in [Hehner 2004]. Quantum predicative programming is developed in [Tafliovich and Hehner 2006, Tafliovich 2004].

2.0.1. Predicative programming

In predicative programming a specification is a boolean expression. The variables in a specification represent the quantities of interest, such as prestate (inputs), poststate (outputs), and computation time and space. We use primed variables to describe outputs and unprimed variables to describe inputs. For example, specification $x' = x + 1$ in one integer variable x states that the final value of x is its initial value plus 1. A computation *satisfies* a specification if, given a prestate, it produces a poststate, such that the pair makes the specification true. A specification is *implementable* if for each input state there is at least one output state that satisfies the specification.

We use standard logical notation for writing specifications: $\wedge$ (conjunction), $\vee$ (disjunction), $\Rightarrow$ (logical implication), $=$ (equality, boolean equivalence), $\neq$ (non-equality, non-equivalence), and **if then else**. The larger operators $==$ and $\Longrightarrow$ are the same as $=$ and $\Rightarrow$, but with lower precedence. We use standard mathematical notation, such as $+ \; - \; \times \; / \; mod$. We use lowercase letters for variables of interest and uppercase letters for specifications.

In addition to the above, we use the following notations: σ (prestate), σ' (poststate), ok ($\sigma' = \sigma$), and $x := e$ ($x' = e \wedge y' = y \wedge \ldots$). The notation ok specifies

that the values of all variables are unchanged. In the assignment $x := e$, x is a state variable (unprimed) and e is an expression (in unprimed variables) in the domain of x.

If R and S are specifications in variables $x, y, \ldots$, then the *sequential composition* of R and S is defined by

$$R; S == \exists x'', y'', \ldots \cdot R'' \wedge S''$$

where R'' is obtained from R by substituting all occurrences of primed variables $x', y', \ldots$ with double-primed variables $x'', y'', \ldots$, and S'' is obtained from S by substituting all occurrences of unprimed variables $x, y, \ldots$ with double-primed variables $x'', y'', \ldots$.

Various laws can be proved about sequential composition. One of the most important ones is the substitution law, which states that for any expression e of the prestate, state variable x, and specification P,

$$x := e; P == (\text{for } x \text{ substitute } e \text{ in } P)$$

Specification S *is refined by* specification P if and only if S is satisfied whenever P is satisfied, that is $\forall \sigma, \sigma' \cdot S \Leftarrow P$. Given a specification, we are allowed to implement an equivalent specification or a stronger one.

Informally, a *bunch* is a collection of objects. It is different from a set, which is a collection of objects in a package. Bunches are simpler than sets; they don't have a nesting structure. See [Hehner 2004] for an introduction to bunch theory. A bunch of one element is the element itself. We use upper-case to denote arbitrary bunches and lower-case to denote elements (an element is the same as a bunch of one element). A, B denotes the union of bunches A and B. $A : B$ denotes bunch inclusion — bunch A is included in bunch B. We use notation $x, ..y$ to mean from (including) x to (excluding) y.

If x is a fresh (previously unused) name, D is a bunch, and b is an arbitrary expression, then $\lambda x : D \cdot b$ is a *function* of a variable (parameter) x with domain D and body b. If f is a function, then Δf denotes the domain of f. If $x : \Delta f$, then fx (f applied to x) is the corresponding element in the range. A function of n variables is a function of 1 variable, whose body is a function of $n - 1$ variables, for $n > 0$. A predicate is a function whose body is a boolean expression. A relation is a function whose body is a predicate. A higher-order function is a function whose parameter is a function.

A *quantifier* is a unary prefix operator that applies to functions. If p is a predicate, then $\forall p$ is the boolean result, obtained by first applying p to all the elements in its domain and then taking the conjunction of those results. Taking the disjunction of the results produces $\exists p$. Similarly, if f is a numeric function, then $\sum f$ is the numeric result, obtained by first applying f to all the elements in its domain and then taking

the sum of those results. We can omit the domain of a variable if it is clear from the context. We can group variables from several quantifications.

A *program* is an implemented specification. A good basis for classical (non-quantum) programming is provided by: ok, assignment, **if then else**, sequential composition, booleans, numbers, bunches, and functions.

Given a specification S, we proceed as follows. If S is a program, there is no work to be done. If it is not, we build a program P, such that P refines S, i.e. $S \Leftarrow P$. The refinement can proceed in steps: $S \Leftarrow \ldots \Leftarrow R \Leftarrow Q \Leftarrow P$.

In $S \Leftarrow P$ it is possible for S to appear in P. No additional rules are required to prove the refinement. For example, it is trivial to prove that

$$x \geq 0 \Rightarrow x' = 0 \Longleftarrow \textbf{ if } x = 0 \textbf{ then } ok \textbf{ else } (x := x - 1; x \geq 0 \Rightarrow x' = 0)$$

The specification says that if the initial value of x is non-negative, its final value must be 0. The solution is: if the value of x is zero, do nothing, otherwise decrement x and repeat.

2.0.2. Probabilistic predicative programming

A *probability* is a real number between 0 and 1, inclusive. A *distribution* is an expression whose value is a probability and whose sum over all values of variables is 1. Given a distribution of several variables, we can sum out some of the variables to obtain a distribution of the rest of the variables.

To generalize boolean specifications to probabilistic specifications, we use 1 and 0 both as numbers and as boolean $true$ and $false$, respectively.[1] If S is an implementable deterministic specification and p is a distribution of the initial state $x, y, ...$, then the distribution of the final state is

$$\sum x, y, ... \cdot S \times p$$

If R and S are specifications in variables $x, y, \ldots$, then the *sequential composition* of R and S is defined by

$$R; S == \sum x'', y'', \ldots \cdot R'' \times S''$$

where R'' is obtained from R by substituting all occurrences of primed variables $x', y', \ldots$ with double-primed variables $x'', y'', \ldots$, and S'' is obtained from S by substituting all occurrences of unprimed variables $x, y, \ldots$ with double-primed variables $x'', y'', \ldots$.

[1] Readers familiar with $\top$ and $\bot$ notation can notice that we take the liberty to equate $\top = 1$ and $\bot = 0$.

If p is a probability and R and S are distributions, then

$$\textbf{if } p \textbf{ then } R \textbf{ else } S == \ p \times R + (1-p) \times S$$

Various laws can be proved about sequential composition. One of the most important ones, the substitution law, introduced earlier, applies to probabilistic specifications as well.

2.0.3. Quantum Predicative Programming

Let $\mathbb{C}$ be the set of all complex numbers with the absolute value operator $|\cdot|$ and the complex conjugate operator *. Then a state of an n-qubit system is a function $\psi : 0, ..2^n \to \mathbb{C}$, such that $\sum x : 0, ..2^n \cdot |\psi x|^2 == \ 1$.

If ψ and ϕ are two states of an n-qubit system, then their *inner product*, denoted by $\langle\psi|\phi\rangle$, is defined by[2]:

$$\langle\psi|\phi\rangle = \sum x : 0, ..2^n \cdot (\psi x)^* \times (\phi x)$$

A *basis* of an n-qubit system is a collection of 2^n quantum states $b_{0,..2^n}$, such that $\forall i, j : 0, ..2^n \cdot \langle b_i|b_j\rangle = (i = j)$. We adopt the following Dirac-like notation for the computational basis: if x is from the domain $0, ..2^n$, then **x** denotes the corresponding n-bit binary encoding of x and $|\mathbf{x}\rangle : 0, ..2^n \to \mathbb{C}$ is the following quantum state:

$$|\mathbf{x}\rangle = \lambda i : 0, ..2^n \cdot (i = x)$$

If ψ is a state of an m-qubit system and ϕ is a state of an n-qubit system, then $\psi \otimes \phi$, the tensor product of ψ and ϕ, is the following state of a composite $m+n$-qubit system:

$$\psi \otimes \phi = \lambda i : 0, ..2^{m+n} \cdot \psi(i \ div \ 2^n) \times \phi(i \ mod \ 2^n)$$

We write $\psi^{\otimes n}$ to mean ψ *tensored with itself* n *times*.

An operation defined on an n-qubit quantum system is a higher-order function, whose domain and range are maps from $0, ..2^n$ to the complex numbers. An *identity* operation on a state of an n-qubit system is defined by

$$I^n = \lambda\psi : 0, ..2^n \to \mathbb{C} \cdot \psi$$

For a linear operation A, the *adjoint* of A, written $A^\dagger$, is the (unique) operation, such that for any two states ψ and ϕ, $\langle\psi|A\phi\rangle = \langle A^\dagger\psi|\phi\rangle$.

[2]We should point out that this kind of function operations is referred to as *lifting*.

The *unitary transformations* that describe the evolution of an n-qubit quantum system are operations U defined on the system, such that $U^\dagger U = I^n$.

In this setting, the *tensor product* of operators is defined in the usual way. If ψ is a state of an m-qubit system, ϕ is a state of an n-qubit system, and U and V are operations defined on m and n-qubit systems, respectively, then the tensor product of U and V is defined on an $m+n$ qubit system by $(U \otimes V)(\psi \otimes \phi) = (U\psi) \otimes (V\phi)$.

Just as with tensor products of states, we write $U^{\otimes n}$ to mean *operation U tensored with itself n times*.

Suppose we have a system of n qubits in state ψ and we measure it. Suppose also that we have a variable r from the domain $0, ..2^n$, which we use to record the result of the measurement, and variables $x, y, \ldots$, which are not affected by the measurement. Then the measurement corresponds to a probabilistic specification that gives the probability distribution of ψ' and r' (these depend on ψ and on the type of measurement) and states that the variables $x, y, \ldots$ are unchanged.

For a general quantum measurement described by a collection $M = M_{0,..2^n}$ of measurement operators, which satisfy the completeness equation (see A), the specification is $\textbf{measure}_M\, \psi\, r$, where

$$\textbf{measure}_M\, \psi\, r == \langle\psi|M_{r'}^\dagger M_{r'}\psi\rangle \times \left(\psi' = \frac{M_{r'}\psi}{\sqrt{\langle\psi|M_{r'}^\dagger M_{r'}\psi\rangle}}\right) \times (\sigma' = \sigma)$$

where $\sigma' = \sigma$ is an abbreviation of $(x' = x) \times (y' = y) \times \ldots$ and means "all other variables are unchanged".

The simplest and the most commonly used measurement in the computational basis is:

$$\textbf{measure}\ \psi\, r == |\psi r'|^2 \times (\psi' = |\mathbf{r}'\rangle) \times (\sigma' = \sigma)$$

In this case the distribution of r' is $|\psi r'|^2$ and the distribution of the quantum state is:

$$\sum r' \cdot |\psi r'|^2 \times (\psi' = |\mathbf{r}'\rangle)$$

which is precisely the mixed quantum state that results from the measurement.

In order to develop quantum programs we need to add to our list of implemented things. We add variables of type quantum state as above and we allow the following three kinds of operations on these variables. If ψ is a state of an n-qubit quantum system, r is a natural variable, and M is a collection of measurement operators that satisfy the completeness equation, then:

1. $\psi := |0\rangle^{\otimes n}$ is a program
2. $\psi := U\psi$, where U is a unitary transformation on an n-qubit system, is a program

3. **measure**$_M\ \psi\ r$ is a program

The special cases of measurements are therefore also allowed.

The *Hadamard* transform, widely used in quantum algorithms, is defined on a 1-qubit system and in our setting is a higher-order function from $0, 1 \to \mathbb{C}$ to $0, 1 \to \mathbb{C}$:

$$H = \lambda\psi : 0, 1 \to \mathbb{C} \cdot \lambda i : 0, 1 \cdot (\psi 0 + (-1)^i \times \psi 1)/\sqrt{2}$$

The operation $H^{\otimes n}$ on an n-qubit system applies H to every qubit of the system. Its action on a zero state of an n-qubit system is:

$$H^{\otimes n}|0\rangle^{\otimes n} = \sum x : 0, ..2^n \cdot |\mathbf{x}\rangle/\sqrt{2^n}$$

On a general state $|\mathbf{x}\rangle$, the action of $H^{\otimes n}$ is:

$$H^{\otimes n}|\mathbf{x}\rangle = \sum y : 0, ..2^n \cdot (-1)^{\mathbf{x}\cdot\mathbf{y}} \times |\mathbf{y}\rangle/\sqrt{2^n}$$

where $\mathbf{x} \cdot \mathbf{y}$ is the inner product of **x** and **y** modulo 2.

3. Quantum Non-locality

In predicative programming, to reason about distributed computation we (disjointly) partition the variables between the processes involved in a computation. Parallel composition is then simply boolean conjunction. For example, consider two processes P and Q. P owns integer variables x and y and Q owns an integer variable z. Suppose $P == \ x := x + 1; y := x$ and $Q == \ z := -z$. Parallel composition of P with Q is then simply

$$\begin{aligned} & P||Q == \ P \wedge Q \\ == & (x := x + 1; y := x) \wedge (z := -z) \\ == & x' = x + 1 \wedge y' = x + 1 \wedge z' = -z \end{aligned}$$

In quantum predicative programming, one needs to reason about distributed quantum systems. Recall that if ψ is a state of an m-qubit system and ϕ is a state of an n-qubit system, then $\psi \otimes \phi$, the tensor product of ψ and ϕ, is the state of a composite $m + n$-qubit system. On the other hand, given a composite $m + n$-qubit system, it is not always possible to describe it in terms of the tensor product of the component m- and n-qubit systems. Such a composed system is *entangled*. Entanglement is one of the most non-classical, most poorly understood, and most interesting quantum phenomena. An entangled system is in some sense both distributed and shared. It is distributed in the sense that each party can apply operations and measurements to only its qubits. It is shared in the sense that the actions of one party affect the outcome of the actions of another party. Simple partitioning of qubits is therefore insufficient to reason about distributed quantum computation.

The formalism we introduce fully reflects the physical properties of a distributed quantum system. We start by partitioning the qubits between the parties involved. For example, consider two parties P and Q. P owns the first qubit of the composite entangled quantum system $\psi = |00\rangle/\sqrt{2} + |11\rangle/\sqrt{2}$ and Q owns the second qubit. A specification is a program only if each party computes with its own qubits. In our example,

$$P == \psi_0 := H\phi_0;\ \textbf{measure}\ \psi_0\ p \quad \text{and} \quad Q == \textbf{measure}\ \psi_1\ q$$

are programs, if p and q are integer variables owned by P and Q, respectively.

Note that we cannot write down expressions for ψ_0 and ψ_1: this is consistent with the laws of quantum mechanics, since ψ is an entangled state. The parties P and Q can access only their own qubits: they could in theory be light years apart.

We define parallel composition of P and Q which share an $n+m$ quantum system in state ψ with the first n qubits belonging to P and the other m qubits belonging to Q as follows. If

$$P == \psi_{0..n-1} := U_P\psi_{0..n-1} \quad \text{and} \quad Q == \psi_{n..n+m-1} := U_Q\psi_{n..n+m-1}$$

where U_P is a unitary operation on an n-qubit system and U_Q is a unitary operation on an m-qubit system, then

$$P\ ||_\psi\ Q == \psi := (U_P \otimes U_Q)\psi$$

Similarly, if

$$P == \textbf{measure}_{M_P}\ \psi_{0..n-1}\ p \quad \text{and} \quad Q == \textbf{measure}_{M_Q}\ \psi_{n..n+m-1}\ q$$

where M_P and M_Q are a collection of proper measurement operators for n- and m-qubit systems, respectively, then

$$P\ ||_\psi\ Q == \textbf{measure}_{M_P \otimes M_Q}\psi\ pq$$

where pq is the number that corresponds to the binary string **pq**.

In our example,

$$\begin{aligned}
&\psi := |00\rangle/\sqrt{2} + |11\rangle/\sqrt{2};\ P\ ||_\psi Q && \text{expand, substitute}\\
==\ &\psi := |00\rangle/\sqrt{2} + |11\rangle/\sqrt{2};\\
&\textbf{measure}\ (H\psi_0)\ p\ ||_\psi\ \textbf{measure}\ \psi_1\ q && \text{compose on } \psi\\
==\ &\psi := |00\rangle/\sqrt{2} + |11\rangle/\sqrt{2};\ \textbf{measure}\ (H \otimes I)\psi\ pq && \text{substitute}\\
==\ &\textbf{measure}\ (H \otimes I)(|00\rangle/\sqrt{2} + |11\rangle/\sqrt{2})\ pq && \text{apply } H \otimes I\\
==\ &\textbf{measure}\ (|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2\ pq && \text{measure}\\
==\ &|(|00\rangle + |01\rangle + |10\rangle - |11\rangle)/2\ pq|^2 \times (\psi' = |\mathbf{p'q'}\rangle) && \text{application}\\
==\ &(\psi' = |\mathbf{p'q'}\rangle)/4
\end{aligned}$$

3.1. Pseudo-telepathy games

We formalize pseudo-telepathy games with n players as follows. For each player i, $0 \leq i < n$, we have a domain D_i from which the inputs to player i are provided and a range R_i of player's possible output results. In addition we may have a promise P: a condition on the inputs to the players. If no promise is given, we set P to 1. The winning condition W can involve inputs as well as outputs for each player. The strategy S is a program, i.e. an implemented specification. The strategy S is winning if

$$P \wedge S \Rightarrow W$$

3.2. Deutsch-Jozsa game

The Deutsch-Jozsa pseudo telepathy game [Brassard et al. 1999, Brassard et al. 2005] is based on a well-known Deutsch-Jozsa algorithm [Deutsch and Jozsa 1992]. A formal analysis of the algorithm is presented in [Tafliovich and Hehner 2006, Tafliovich 2004]. The setting of the game is as follows. Alice and Bob are separated several light years apart and are each presented with a 2^k-bit string. They are promised that either the strings are identical or they differ by exactly half of the bits. To win the game the players must each output a k-bit string, and these strings should be identical if and only if their input strings were identical.

We formalize the game as follows. We partition the space into the world of Alice (variables subscripted A) and the world of Bob (variables subscripted B). Then $D_A = D_B = \{0,1\}^{2^k}$ are the domain of inputs to Alice and Bob, $R_A = R_B = \{0,1\}^k$ are the range of outputs of Alice and Bob, $P = P_0 \vee P_1$, where P_0 states that the inputs are identical, $P_0 == \sum i : 0,..2^k \cdot ((x_A)_i = (x_B)_i) == 2^k$, and P_1 states that the inputs differ by half of the bits, $P_1 == \sum i : 0,..2^k \cdot ((x_A)_i = (x_B)_i) == 2^{k-1}$, is the promise on the inputs, and $W == (x_A = x_B) = (y'_A = y'_B)$ is the winning condition.

We demonstrate the quantum solution by implementing a specification S, so that $P \wedge S \Rightarrow W$:

$S == \psi := \sum z : 0,..2^k \cdot |zz\rangle/\sqrt{2^k}; (S_A \,||_\psi\, S_B)$, where

$S_i == \psi_i := U_i^{\otimes k}\psi_i;\ \psi_i := H^{\otimes k}\psi_i;$ **measure** $\psi_i\ y_i$,

for unitary $U_i|z\rangle = (-1)^{(x_i)_z}|z\rangle$, where $i : A, B$. [3]

To prove the solution correct we show (omitting domains of u, v, z):

S

$== \psi := \sum z \cdot |zz\rangle/\sqrt{2^k};\ (S_A \,||_\psi\, S_B)$ substitute

$== \psi := \sum z \cdot |zz\rangle/\sqrt{2^k};$ composition

measure $H^{\otimes k}(U_A^{\otimes k}\psi_A)\ y_A \,||_\psi$ **measure** $H^{\otimes k}(U_B^{\otimes k}\psi_B)\ y_B$ on ψ

[3] Implementing the initial assignment is easy and we omit the details for the sake of brevity

$$
\begin{aligned}
&== \psi := \sum z \cdot |zz\rangle/\sqrt{2^k}; && \text{substitute and}\\
&\quad \textbf{measure } H^{\otimes 2k}((U_A^{\otimes k} \otimes U_B^{\otimes k})\psi)\ y_A y_B && \text{measure}\\
&== |H^{\otimes 2k}((U_A^{\otimes k} \otimes U_B^{\otimes k})(\sum z \cdot |zz\rangle/\sqrt{2^k}))\ (y_A y_B)'|^2 && \text{apply } U_i, H\\
&== \left|\sum u, v, z \cdot (-1)^{(x_A)_z + (x_B)_z + u\cdot z + v\cdot z}/\sqrt{2^k}^3 \times |uv\rangle\ (y_A y_B)'\right|^2
\end{aligned}
$$

To demonstrate that S is winning, namely that $P \wedge S \Rightarrow W$, it is sufficient to show $P_0 \wedge S \Rightarrow (y'_A = y'_B)$ and $P_1 \wedge S \Rightarrow (y'_A \neq y'_B)$. Proving the first implication:

$$
\begin{aligned}
&\quad P_0 \wedge S && \text{expand}\\
&== \left(\sum i \cdot ((x_A)_i = (x_B)_i) = 2^k\right) \times\\
&\quad \left|\sum u, v, z \cdot (-1)^{(x_A)_z + (x_B)_z + u\cdot z + v\cdot z}/\sqrt{2^k}^3 \times |uv\rangle\ (y_A y_B)'\right|^2 && \text{since } x_A = x_B\\
&== \left|\sum u, v, z \cdot (-1)^{u\cdot z + v\cdot z}/\sqrt{2^k}^3 \times |uv\rangle\ (y_A y_B)'\right|^2 && \text{simplify}\\
&== \left|\sum z \cdot |zz\rangle/\sqrt{2^k}\ (y_A y_B)'\right|^2 && \text{application}\\
&== \sum x : 0, ..2^k \cdot (y'_A = x) \times (y'_B = x) && \text{math}\\
&== (y'_A = y'_B)
\end{aligned}
$$

Similarly, analyzing the amplitudes in the second case, we get:

$$P_1 \wedge S \Rightarrow (y'_A \neq y'_B)$$

3.3. Mermin's game

In a Mermin's game [Mermin 1990] there are three players. Each player i receives a bit x_i as input and outputs a bit y_i. The promise is that the sum of the inputs is even. The players win the game if the parity of the sum of the outputs is equal to the parity of half the sum of the inputs.

We formalize the game as follows: $D_i = R_i = \{0, 1\}$, for $i : 0, 1, 2$. The promise is $P == (x_0 + x_1 + x_2) \mod 2 = 0$. The winning condition is $W == (y'_0 + y'_1 + y'_2) = (x_0 + x_1 + x_2)/2 \mod 2$.

We implement the following quantum strategy. The players share an entangled state $\psi = |000\rangle/\sqrt{1} + |111\rangle/\sqrt{2}$. After receiving the input, each player applies the operation U defined by $U|0\rangle = |0\rangle$ and $U|1\rangle = \sqrt{-1} \times |1\rangle$ to her qubit if the input is 1. The player then applies a Hadamard transform. The qubit is measured in the computational basis and the result of the measurement is the output.

The program is:

$$
\begin{aligned}
&S == \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2};\ S_0\ ||_\psi\ S_1\ ||_\psi\ S_2\\
&S_i == \textbf{if } x_i = 1 \textbf{ then } \psi_i := U\psi_i \textbf{ else } ok;\ \psi_i := H\psi_i;\ \textbf{measure } \psi_i\ y_i
\end{aligned}
$$

where $i : 0, 1, 2$.

To prove the solution is correct we demonstrate:

$$
\begin{array}{lr}
S & \\
== \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2}; & \\
\quad ||_\psi\, i : 0,1,2 \cdot \textbf{ if } x_i = 1 \textbf{ then } \psi_i := U\psi_i \textbf{ else } ok; & \text{conditional,} \\
\qquad\qquad \psi_i := H\psi_i;\ \textbf{measure } \psi_i\ y_i & \text{substitute} \\
== \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2}; & \\
\quad ||_\psi\, i : 0,1,2 \cdot (x_i = 1) \times (\psi_i := H(U\psi_i)) + & \\
\qquad\qquad (x_i = 0) \times (\psi_i := H\psi_i);\ \textbf{measure } \psi_i\ y_i & \text{substitute} \\
== \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2}; & \text{compose} \\
\quad ||_\psi\, i : 0,1,2 \cdot \textbf{measure } H(U^{x_i}\psi_i)\ y_i & \text{on } \psi \\
== \psi := |000\rangle/\sqrt{2} + |111\rangle/\sqrt{2}; & \text{substitute,} \\
\quad \textbf{measure } H^{\otimes 3}((U^{x_0} \otimes U^{x_1} \otimes U^{x_2})\psi)\ y_0y_1y_2 & \text{apply } U \\
== \textbf{measure } H^{\otimes 3}(|000\rangle + (\sqrt{-1})^{x_0+x_1+x_2} \times |111\rangle)/\sqrt{2}\ y_0y_1y_2 &
\end{array}
$$

Finally, the strategy S is winning, since:

$$
\begin{array}{lr}
P \wedge S & \\
== ((x_0 + x_1 + x_2) \mod 2 = 0) \times & \text{Hadamard,} \\
\quad (\textbf{measure } H^{\otimes 3}(|000\rangle + (\sqrt{-1})^{x_0+x_1+x_2} \times |111\rangle)/\sqrt{2}\ y_0y_1y_2) & \text{measure} \\
== (x_0 + x_1 + x_2 = 0)\times & \\
\quad |(|000\rangle + |011\rangle + |101\rangle + |110\rangle)/2\ (y_0y_1y_2)'|^2 + & \\
\quad (x_0 + x_1 + x_2 = 2)\times & \\
\quad |(|001\rangle + |010\rangle + |100\rangle + |111\rangle)/2\ (y_0y_1y_2)'|^2 & \text{application} \\
== y'_0 + y'_1 + y'_2 = (x_0 + x_1 + x_2)/2 \mod 2 &
\end{array}
$$

3.4. Parity Games

In parity games [Brassard et al. 2003, Brassard et al. 2005, Buhrman et al. 2003] there are at least three players. Each player i is given a number $\alpha_i : 0, ..2^l$, or, equivalently, an l-bit binary string. The promise is that $\sum i : 0, ..n \cdot \alpha_i$ is divisible by 2^l. Each player outputs a single bit β_i. The winning condition is that $\sum i : 0, ..n \cdot \beta_i == \sum \alpha_i/2^l \ (mod\ 2)$.

Consider the following strategy. The players share an entangled state $\psi = (|0\rangle^{\otimes n} + |1\rangle^{\otimes n})/\sqrt{2}$. Each player i executes the following program:

$$\psi_i := U\psi_i; \psi_i := H\psi_i;\ \textbf{measure}\ \ \psi_i\ \beta_i$$

where the operator U is defined by

$$U|0\rangle = |0\rangle \text{ and } U|1\rangle = e^{\pi \times \sqrt{-1} \times \alpha_i / 2^l} \times |1\rangle$$

and H is the Hadamard transform.

Again, we can prove that $P \wedge S \Rightarrow W$, where S refers to the parallel execution on the above program after the initialization of the shared entangled state. We omit the proof due to lack of space.

Note that if $n = 3$ and $l = 1$, the parity game is a Mermin game.

4. Conclusion and Future Work

We have presented a formal framework for specifying, implementing, and analyzing quantum pseudo-telepathy games.

Current research focuses on formal reasoning about complexity of distributed quantum algorithms (e.g. [Yimsiriwattana and Jr 2004]). Research in the immediate future will focus on simple proofs and analysis of programs involving communication, both via quantum channels and exhibiting the LOCC (local operations, classical communication) paradigm. Future work involves formalizing quantum cryptographic protocols, such as BB84 [Bennet and Brassard 1984], in our framework and providing formal analysis of these protocols.

References

Abramsky, S. (2004). High-level methods for quantum computation and information. In *LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Science Press.

Abramsky, S. and Coecke, B. (2004). A categorical semantics of quantum protocols. In *LICS '04: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Science Press.

Abramsky, S. and Duncan, R. (2006). A categorical quantum logic. *Mathematical Structures in Comp. Sci.*, 16(3).

Adao, P. and Mateus, P. (2007). A process algebra for reasoning about quantum security. *Electron. Notes Theor. Comput. Sci.*, 170.

Altenkirch, T. and Grattage, J. (2005). A functional quantum programming language. In *LICS '05: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society.

Arrighi, P. and Dowek, G. (2004). Operational semantics for formal tensorial calculus. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*.

Arrighi, P. and Dowek, G. (2005). Linear-algebraic lambda-calculus. In *Proceedings of the 3rd International Workshop on Quantum Programming Languages*.

Bell, J. (1964). On the Einstein-Podolsky-Rosen paradox. *Physics*, 1(3).

Bennet, C. H. and Brassard, G. (1984). Quantum cryptography: Public key distribution and coin tossing. In *IEEE Int. Conf. Computers, Systems and Signal Processing*.

Brassard, G., Broadbent, A., and Tapp, A. (2003). Multi-party pseudo-telepathy. In *Proceedings of the 8th International Workshop on Algorithms and Data Structures*.

Brassard, G., Broadbent, A., and Tapp, A. (2005). Quantum pseudo-telepathy. *Foundations of Physics*, 35:1877–1907.

Brassard, G., Buhrman, H., Linden, N., Méthot, A. A., Tapp, A., and Unger, F. (2006). Limit on nonlocality in any world in which communication complexity is not trivial. *Physical Review Letters*, 96(25).

Brassard, G., Cleve, R., and Tapp, A. (1999). Cost of exactly simulating quantum entanglement with classical communication. *Physical Review Letters*, 83(9):1874–1878.

Brassard, G., Methot, A. A., and Tapp, A. (2004). Minimum entangled state dimension required for pseudo-telepathy. quant-ph/0412136.

Buhrman, H., Hoyer, P., Massar, S., and Roehrig, H. (2003). Combinatorics and quantum nonlocality. *Physical Review Letters*, 91:047903.

Coecke, B. (2004). The logic of entanglement. quant-ph/0402014.

Danos, V., D'Hondt, E., Kashefi, E., and Panangaden, P. (2005). Distributed measurement-based quantum computation. In *Proceedings of the 3rd International Workshop on Quantum Programming Languages*.

Deutsch, D. and Jozsa, R. (1992). Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London*, 439:553–558.

D'Hondt, E. and Panangaden, P. (2004). Quantum weakest precondition. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*.

D'Hondt, E. and Panangaden, P. (2005). Reasoning about quantum knowledge. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science: 25th International Conference*.

Galliard, V., Wolf, S., and Tapp, A. (2003). The impossibility of pseudo-telepathy without quantum entanglement.

Gay, S. J. and Nagarajan, R. (2005). Communicating quantum processes. In *Proceedings of the 32nd ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*.

Hehner, E. (1993). *a Practical Theory of Programming*. Springer, New York, first edition. Current edn. (2007) Available free at `www.cs.utoronto.ca/~hehner/aPToP`.

Hehner, E. (2004). Probabilistic predicative programming. In *Proceedings of the 7th International Conference on Mathematics of Program Construction*, volume 3125 of *Lecture Notes in Computer Science*. Springer.

Jorrand, P. and Lalire, M. (2004). Toward a quantum process algebra. In *Proceedings of the 1st ACM Conference on Computing Frontiers*.

Lalire, M. and Jorrand, P. (2004). A process algebraic approach to concurrent and distributed quantum computation: operational semantics. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*.

Mermin, N. (1990). Quantum mysteries revisited. *Americal Journal of Physics*, 58(8):731–734.

Nielsen, M. A. and Chuang, I. L. (2000). *Quantum Computation and Quantum Information*. Cambridge University Press.

Sanders, J. W. and Zuliani, P. (2000). Quantum programming. In *MPC '00: Proceedings of the 5th International Conference on Mathematics of Program Construction*. Springer-Verlag.

Selinger, P. (2004). Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4).

Tafliovich, A. (2004). Quantum programming. Master's thesis, University of Toronto.

Tafliovich, A. and Hehner, E. (2006). Quantum predicative programming. In *Proceedings of the 8th International Conference on Mathematics of Program Construction*, volume 4014 of *Lecture Notes in Computer Science*. Springer.

Valiron, B. (2004). Quantum typing. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*.

van Tonder, A. (2004). A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33(5).

Yimsiriwattana, A. and Jr, S. J. L. (2004). Distributed quantum computing: A distributed Shor algorithm. `quant-ph/0403146`.

Zuliani, P. (2004). Non-deterministic quantum programming. In *Proceedings of the 2nd International Workshop on Quantum Programming Languages*.

Zuliani, P. (2005). Quantum programming with mixed states. In *Proceedings of the 3rd International Workshop on Quantum Programming Languages*.

A. Quantum Computation

In this section we introduce the basic concepts of quantum mechanics, as they pertain to the quantum systems that we will consider for quantum computation. The discussion of the underlying physical processes, spin-$\frac{1}{2}$-particles, etc. is not our interest. We are concerned with the model for quantum computation only. A reader not familiar

with quantum computing can consult [Nielsen and Chuang 2000] for a comprehensive introduction to the field.

The *Dirac notation*, invented by Paul Dirac, is often used in quantum mechanics. In this notation a vector v (a column vector by convention) is written inside a *ket*: $|v\rangle$. The dual vector of $|v\rangle$ is $\langle v|$, written inside a *bra*. The inner products are *bra-kets* $\langle v|w\rangle$. For n-dimensional vectors $|u\rangle$ and $|v\rangle$ and m-dimensional vector $|w\rangle$, the value of the inner product $\langle u|v\rangle$ is a scalar and the outer product operator $|v\rangle\langle w|$ corresponds to an m by n matrix. The Dirac notation clearly distinguishes vectors from operators and scalars, and makes it possible to write operators directly as combinations of bras and kets.

In quantum mechanics, the vector spaces of interest are the Hilbert spaces of dimension 2^n for some $n \in \mathbb{N}$. A convenient orthonormal basis is what is called a *computational basis*, in which we label 2^n basis vectors using binary strings of length n as follows: if s is an n-bit string which corresponds to the number x_s, then $|s\rangle$ is a 2^n-bit (column) vector with 1 in position x_s and 0 everywhere else. The tensor product $|i\rangle \otimes |j\rangle$ can be written simply as $|ij\rangle$. An arbitrary vector in a Hilbert space can be written as a weighted sum of the computational basis vectors.

Postulate 1 (state space) Associated to any isolated physical system is a Hilbert space, known as the *state space* of the system. The system is completely described by its *state vector*, which is a unit vector in the system's state space.

Postulate 2 (evolution) The evolution of a closed quantum system is described by a *unitary transformation*.

Postulate 3 (measurement) Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators*, which act on the state space of the system being measured. The index m refers to the possible measurement outcomes. If the state of the system immediately prior to the measurement is described by a vector $|\psi\rangle$, then the probability of obtaining result m is $\langle\psi|M_m^\dagger M_m|\psi\rangle$, in which case the state of the system immediately after the measurement is described by the vector $\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}$. The measurement operators satisfy the *completeness equation* $\sum m \cdot M_m^\dagger M_m == I$.

An important special class of measurements is *projective measurements*, which are equivalent to general measurements provided that we also have the ability to perform unitary transformations.

A projective measurement is described by an *observable* M, which is a Hermitian operator on the state space of the system being measured. This observable has a spectral decomposition $M = \sum m \cdot \lambda_m \times P_m$, where P_m is the projector onto the eigenspace of M with eigenvalue λ_m, which corresponds to the outcome of the measurement. The probability of measuring m is $\langle\psi|P_m|\psi\rangle$, in which case immediately after the measurement the system is found in the state $\frac{P_m|\psi\rangle}{\sqrt{\langle\psi|P_m|\psi\rangle}}$.

Given an orthonormal basis $|v_m\rangle$, $0 \leq m < 2^n$, measurement with respect to this basis is the corresponding projective measurement given by the observable $M = \sum m \cdot \lambda_m \times P_m$, where the projectors are $P_m = |v_m\rangle\langle v_m|$.

Measurement with respect to the computational basis is the simplest and the most commonly used class of measurements. In terms of the basis $|m\rangle$, $0 \leq m < 2^n$, the projectors are $P_m = |m\rangle\langle m|$ and $\langle\psi|P_m|\psi\rangle = |\psi_m|^2$. The state of the system immediately after measuring m is $|m\rangle$.

For example, measuring a single qubit in the state $\alpha \times |0\rangle + \beta \times |1\rangle$ results in the outcome 0 with probability $|\alpha|^2$ and outcome 1 with probability $|\beta|^2$. The state of the system immediately after the measurement is $|0\rangle$ or $|1\rangle$, respectively.

Suppose the result of the measurement is ignored and we continue the computation. In this case the system is said to be in a *mixed state*. A mixed state is not the actual physical state of the system. Rather it describes our knowledge of the state the system is in. In the above example, the mixed state is expressed by the equation $|\psi\rangle = |\alpha|^2 \times \{|0\rangle\} + |\beta|^2 \times \{|1\rangle\}$. The equation is meant to say that $|\psi\rangle$ is $|0\rangle$ with probability $|\alpha|^2$ and it is $|1\rangle$ with probability $|\beta|^2$. An application of operation U to the mixed state results in another mixed state, $U(|\alpha|^2 \times \{|0\rangle\} + |\beta|^2 \times \{|1\rangle\}) = |\alpha|^2 \times \{U|0\rangle\} + |\beta|^2 \times \{U|1\rangle\}$.

Postulate 4 (composite systems) The state space of a composite physical system is the tensor product of the state spaces of the component systems. If we have systems numbered 0 up to and excluding n, and each system i, $0 \leq i < n$, is prepared in the state $|\psi_i\rangle$, then the joint state of the composite system is $|\psi_0\rangle \otimes |\psi_1\rangle \otimes \ldots \otimes |\psi_{n-1}\rangle$.

While we can always describe a composite system given descriptions of the component systems, the reverse is not true. Indeed, given a state vector that describes a composite system, it may not be possible to factor it to obtain the state vectors of the component systems. A well-known example is the state $|\psi\rangle = |00\rangle/\sqrt{2} + |11\rangle/\sqrt{2}$. Such a state is called an *entangled* state.