Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 218 (2008) 153–170
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Game Semantics for Quantum Stores

Yannick Delbecque[a,1,2] Prakash Panagaden[a,3]

[a] *School of Computer Science*
McGill University
Montreal, Canada

Abstract

This paper presents a game semantics for a simply-typed λ-calculus equipped with quantum stores. The quantum stores are equipped with quantum operations as commands which give the language enough expressiveness to encode any quantum circuits. The language uses a notion of extended variable, similar to that seen in functional languages with pattern matching, but adapted to the needs of dealing with tensor products. These tensored variables are used to refer to quantum stores and to keep track of the size of the states which they contain. The game semantics is constructed from classical game semantics using intervention operators to encode the effects of the commands. A soundess result for the semantics is given.

Keywords: Game semantics, quantum programing languages, quantum games.

1 Introduction

An important problem in the development of higher-order quantum programming languages is to find an appropriate structure to define a denotational semantics. For example, there was no denotational semantics given in the first presentations of the quantum λ-calculus developed by Selinger and Valiron [15,13]. They proposed in [14] a denotational semantics for the linear part of the quantum λ-calculus; their interpretation is in the category **CPM** of completely positive maps on finite dimensional Hilbert spaces. Working with this restricted language allows them to avoid the problem of finding a structure which can model the possible interactions between the quantum data and the classical data in higher order quantum programming languages. To address this problem, there are many structures to choose from. One can consider for example the biproduct completion of CPM [12,14] or the concept of classical objects [2]. These two examples share a common approach: classical data is encoded using properties of the Hilbert space structure used to describe quantum

[1] This research was supported by a grant from NSERC (Canada)
[2] Email: yannick@delbecque.org
[3] Email: prakash@cs.mcgill.ca

doi:10.1016/j.entcs.2008.10.010

data. In this paper we use a different strategy: quantum data is represented by the classical interactions used to manipulate quantum states and extract information from them with measurements. More precisely, a quantum state is represented as a strategy which makes someone chose the actions according to the laws of quantum mechanics. Our proposed model is built with ideas from game semantics augmented with a new game in which plays describe the behavior of quantum stores.

There are important differences between the quantum λ-calculus of Valiron and Selinger and the language presented in this paper. First, the former language does not allow quantum states to be introduced directly: they can only be referred to using variables of type qbit. Thus, in the type system quantum states are considered as data of type qbit which cannot be duplicated. The quantum store language introduced below is based upon the idea that there should be no harm in duplicating a reference to a qbit, as long as it is not possible to duplicate the qbit itself. With this approach, it is not necessary to assume that variables are used only linearly: each use updates the stored value.

A second difference between the quantum λ-calculus and the quantum store language is the way tensor products of quantum states are dealt with. There is a tensor type-constructor in the quantum λ-calculus which is complemented with a bang operation and typing rules inspired from linear logic. The idea is that the distinction between classical types and quantum types can be reduced to duplicability since quantum data is not duplicable while classical data is. When combined with abstraction, a tensor type constructor allows one to take a program of type $\mathrm{qbit} \otimes \mathrm{qbit} \multimap \mathrm{qbit} \otimes \mathrm{qbit}$ to a program of type $\mathrm{qbit} \multimap (\mathrm{qbit} \multimap \mathrm{qbit} \otimes \mathrm{qbit})$. This may seem problematic, since intuitively this takes a function with two input qbits, which may be in some entangled state, to a program with should be equivalent but using only separated qbits. This is avoided in the quantum store language since there is no tensor type construction. Instead, the quantum stores are equipped with a preparation operation which allows one to add qbits to the current state as necessary. To keep track of the size of the currently held state, we introduce *tensor of variables* to refer to the quantum stores.

2 Quantum mechanics and quantum interventions

A quantum system A is represented as a finite-dimensional Hilbert space H_A. A *quantum state* is represented as a Hermitian positive operator ρ on H_A of unit trace. These operators are called *density matrices*. The set of density matrices on H_A is denoted $\mathrm{D}(H_A)$. The set of positive operators of trace less than one is denoted $\mathrm{SD}(H_A)$; we call such operators *subdensity matrices*.

An important postulate of quantum mechanics is that the states of a composite quantum system AB are described by the density matrices on the tensor product $H_A \otimes H_B$. The evolution of a state over time is described by unitary operation on the Hilbert space: a state ρ is sent to $\mathcal{U}(\rho) = U\rho U^\dagger$. The final ingredient of basic quantum mechanics is the concept of *quantum measurement* by which one extracts classical information about a state by interacting with it. The measurement gives

some information m *and alters the measured state.* This is usually described using projections: a *projective measurement* is a family of projection operators $\{P_m\}$ indexed by the measurement results such that $\sum P_m = 1$ and $P_{m_1} P_{m_2} = 0$ if $m_1 \neq m_2$. If a state ρ is measured, we get the result m with probability $p_m = \mathrm{tr}\left(P_m \rho P_m^\dagger\right)$ and the process leaves the system in state $P_m \rho P_m^\dagger / p_m$. We will denote the projection operator on the canonical basis vector $|k\rangle$ by $[k]$. To simplify the notation, an operation of the form $I_A \otimes M \otimes I_C$ on $H_A \otimes H_B \otimes H_C$, where M is a linear operator on H_B, is denoted by M^B.

Peres introduced a general description of quantum measurements called *intervention operators* [10]. The measurement process is conceived of as a unitary interaction of a measurement apparatus with the quantum system to be measured, followed by a projective measurement on the combined system. Mathematically, Peres proved that this process is some described by what are commonly called *superoperators.* A superoperator is a completely positive trace non-increasing map $\mathcal{E} \colon \mathsf{SD}(H_A) \to \mathrm{SD}(H_B)$. Superoperators are composed as usual, but we use a convenient convention: if the domain of $\mathcal{E}$ does not match the codomain of $\mathcal{F}$ we put $\mathcal{E}\mathcal{F} = 0$. This convention is consistent with the quantum mechanical interpretation of superoperators: an impossible operation is assigned probability zero. A *quantum intervention* on a Hilbert space H_A is a collection of superoperators $\mathcal{E} = \{\mathcal{E}_m \colon \mathrm{SD}(H_A) \to \mathrm{SD}(H_{B_m})\}$ indexed by measurement results m, such that we have $\sum_m \mathrm{tr}\left(\mathcal{E}_m(\rho)\right) = 1$ for any state ρ. If the system is initially in state ρ, performing the quantum intervention yields result m with probability $p_m = \mathrm{tr}\left(\mathcal{E}_m(\rho)\right)$ and leaves the system in state $\mathcal{E}_m(\rho)/p_m$. Note that the space H_{B_m} may depend on the measurement outcome.

3 Simply typed λ-calculus with quantum stores

We now introduce a λ-calculus with quantum stores language (*QSL*) The syntax of QSL is built upon a simply typed λ-calculus with pairing, conditionals and sequential composition. Quantum operations are added using *quantum stores* which have a syntax analogous to classical stores. In a classical higher-order programming language with stores, like idealised ALGOL [11], stores are references to values. They are used through various operations like dereferencing and assignment. The quantum stores we use below are defined according to the following parallel between classical and quantum references:

Classical stores	**Quantum stores**
Dereferencing	Measurement
Assignment	Preparation
Command with side effects	Unitary transformation

In this picture, the quantum counterpart of dereferencing, which classically returns the value stored, is quantum measurement. The counterpart of assignment is state preparation. Note that, while classically it is possible to assign a value to a store multiple times, this is not the case with quantum stores, as a quantum state cannot be destroyed. Instead, preparation expands a given state with a new known quan-

tum state. Classical stores can be equipped with commands with side effects, for example, an integer incrementation command. Unitary operations on the store are the quantum counterpart of classical update operations.

3.1 Syntax

We need to introduce a new syntactic device to accommodate quantum stores. When multiple quantum stores are combined, they can be measured by using a projective measurement on the whole space. Because of this, we must be able to refer to the combined store as a whole, while keeping the possibility to refer to a part of the system. To this end, we introduce tensor of variables in the syntax. An *extended variable* is an expression of the form $x_1 \otimes \cdots \otimes x_n$, where the x_i are variables such that $x_i \neq x_j$ if $i \neq j$. Two extended variables $x_1 \otimes \cdots \otimes x_n$ and $y_1 \otimes \cdots \otimes y_m$ are *disjoint* if $x_i \neq y_j$ for all i, j. Two such extended variables can be joined to form a new extended variable $x_1 \otimes \cdots \otimes x_n \otimes y_1 \otimes \cdots \otimes y_m$. Note that when we use $x_1 \otimes \cdots \otimes x_n$ to refer to an arbitrary extended variable, the case $n = 1$ is also possible. We use the notation $x_1 \otimes \cdots \otimes x_n \sqsubseteq y_1 \otimes \cdots \otimes y_m$ when each of the variables $x_1, \ldots, x_n$ occurs in $y_1 \otimes \cdots \otimes y_n$ and the order of the occurrences is the same in both extended variables. We say in this case that $x_1 \otimes \cdots \otimes x_n$ is a *subvariable* of $y_1 \otimes \cdots \otimes y_m$. To simplify the notation, we use $\overline{x}$ instead of $x_1 \otimes \cdots \otimes x_n$, leaving the number n implicit.

The terms of QSL are defined by

$$\begin{aligned} M, N, P \ ::= \ & \overline{x} \mid * \mid 0 \mid 1 \mid \mathsf{skip} \mid \lambda \overline{x}.\, M \mid MN \mid \\ & \mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ P \mid \langle M, N \rangle \mid \mathsf{fst}\, M \mid \mathsf{snd}\, M \mid M; N \mid \\ & \mathsf{meas}\, x \mid \mathsf{new}\, \overline{x}\, \mathsf{in}\, M \mid UM \mid \mathsf{prep}\, \overline{y}\, \mathsf{with}\, \overline{x}\, \mathsf{in}\, M \end{aligned}$$

where $\overline{x}$ and $\overline{y}$ can be any extended variables and U can be any multiple-qbits unitary transformation. All the classical operations used are standard operations: $\langle M, N \rangle$ is pairing, fst and snd are the two associated projection operations, $M; N$ is sequential composition, and skip is the operation doing nothing. The quantum part of the language consists of operations to manipulate quantum stores: measurement, qbit creation, unitary modification and preparation of extra qbits. The measurement operation $\mathsf{meas}\, x$ measures the qbit x in the quantum register in the canonical basis and returns a boolean value corresponding to the measurement result. For the preparation operation, $\mathsf{prep}\, \overline{y}\, \mathsf{with}\, \overline{x}\, \mathsf{in}\, M$ means that a given quantum store $\overline{x}$ is extended to a larger store by adding extra qbits prepared in the $|0\rangle$ state. In M, the whole extended store is referred to as $\overline{x} \otimes \overline{y}$.

As in any λ-calculus, the λ operation is a binder. Observe that it can be used on extended variables, i.e. terms like $\lambda x \otimes y.\, \mathsf{meas}\, x$ are allowed. The preparation operation is also a binder: $\overline{x}$ is not free in the term $\mathsf{prep}\, \overline{y}\, \mathsf{with}\, \overline{x}\, \mathsf{in}\, M$. The set of free extended variables of M is denoted by $\mathrm{FV}(M)$. A term M is *closed* if it has no free extended variables. We use the notation $M[N/\overline{x}]$ to denote the capture-free substitution (no occurrence of a free variable in N is bound in M) of the term N for every occurrence of $\overline{x}$. For clarity, we use the alternative notation $\mathsf{let}\, x = N\, \mathsf{in}\, M$

Table 1
QSL typing rules

$$\frac{}{\Gamma, \overline{x}\colon A \vdash \overline{x}\colon A} \qquad \frac{}{\Gamma \vdash 0\colon \mathrm{bool}} \qquad \frac{}{\Gamma \vdash 1\colon \mathrm{bool}} \qquad \frac{}{\Gamma \vdash *\colon \top} \qquad \frac{}{\Gamma \vdash \mathsf{skip}\colon \mathrm{com}}$$

$$\frac{\Gamma \vdash M\colon A \Rightarrow B \qquad \Gamma \vdash N\colon A}{\Gamma \vdash MN\colon B} \qquad \frac{\Gamma, \overline{x}\colon A \vdash M\colon B}{\Gamma \vdash \lambda \overline{x}.\, M\colon A \Rightarrow B} \qquad \frac{\Gamma \vdash M_1\colon A_1 \qquad \Gamma \vdash M_2\colon A_2}{\Gamma \vdash \langle M_1, M_2\rangle\colon A_1 \times A_2}$$

$$\frac{\Gamma \vdash M\colon A \times B}{\Gamma \vdash \mathsf{fst}\, M\colon A} \qquad \frac{\Gamma \vdash M\colon A \times B}{\Gamma \vdash \mathsf{snd}\, M\colon B} \qquad \frac{\Gamma \vdash P\colon \mathrm{bool} \qquad \Gamma \vdash M\colon A \qquad \Gamma \vdash N\colon A}{\Gamma \vdash \mathsf{if}\ P\ \mathsf{then}\ M\ \mathsf{else}\ N\colon A}$$

$$\frac{\Gamma \vdash M\colon \mathrm{com} \qquad \Gamma \vdash N\colon A}{\Gamma \vdash M; N\colon A} \quad A = \mathrm{com}\ \text{or}\ \mathrm{bool} \qquad \Gamma, \overline{x}\colon \mathrm{qstore} \vdash \mathsf{meas}\, x_i\colon \mathrm{bool} \quad x_i \sqsubseteq \overline{x}$$

$$\Gamma, \overline{x}\colon \mathrm{qstore} \vdash U\, y_1 \otimes \cdots \otimes y_m\colon \mathrm{com} \quad \begin{array}{l} \overline{y} \sqsubseteq \overline{x} \\ U\colon \mathrm{unitary}, \mathrm{rank}(U) = m \end{array}$$

$$\frac{\Gamma, \overline{x}\colon \mathrm{qstore} \vdash M\colon A}{\Gamma \vdash \mathsf{new}\, \overline{x}\, \mathsf{in}\, M\colon A} \qquad \frac{\Gamma, \overline{x} \otimes \overline{y}\colon \mathrm{qstore} \vdash M\colon A}{\Gamma, \overline{x}\colon \mathrm{qstore} \vdash \mathsf{prep}\, \overline{y}\, \mathsf{with}\, \overline{x}\, \mathsf{in}\, M\colon A}$$

for $(\lambda x.\, M)N$. When multiple variables are bound in this manner successively, we use the notation $\mathsf{let}\, x_1 = N_1, \ldots, x_n = N_n\, \mathsf{in}\, M$ for $(\lambda x_n.\ \ldots\ (\lambda x_1.\, M)\, N_1 \ldots)\, N_n$.

The types of the λ-calculus with quantum stores are the following:

$$A, B\ ::=\ \mathrm{bool} \mid \mathrm{com} \mid \top \mid A \times B \mid A \Rightarrow B \mid \mathrm{qstore}.$$

The type bool is the type of boolean constants, $A \times B$ and $A \Rightarrow B$ are respectively the types of pairs and functions. The type com is the type of *commands* which can be composed using sequential composition. The type qstore is the type of a quantum store. A quantum store does not have a fixed dimension, as the number of qbits it hold can vary in the course of a computation if preparation operations are used.

The typing rules rules for the classical part are given in table 1. The rules for the classical part of the language are the standard rules of a simply typed λ-calculus where extended variables can be used. The rules for involving quantum operations encode the idea that the content of quantum stores can be measured, modified using unitary transformations and that quantum stores can be prepared or extended with an ancilla state. Note that the unitary operation rule allows unitary operations to be applied only to part of a quantum register. An important feature of QSL is that the typing rules do not forbid having multiple references to a quantum store. For example, the typing judgement $x\colon \mathrm{qstore} \vdash \langle \mathsf{meas}\, x, \mathsf{meas}\, x\rangle\colon \mathrm{bool} \times \mathrm{bool}$ is valid. Copying a reference to a qbit is not the same thing as duplicating the qbit. Yet the language does not allow unknown qbit duplication: to duplicate the content of a quantum store x, one would need to prepare a new qbit y and apply an appropriate unitary transformation to the quantum store $x \otimes y$. There is no such unitary transformation.

3.2 Operational semantics

The operational semantics of the classical part of the quantum store language is standard. For the quantum part we use a quantum variant of stores. Note that

we expect that the reduction relation of this language depends on reduction order. There is nothing special in the quantum case in this regard. For example, assuming that x is a classical integer store holding the value 1, the term $\langle x := x + 1, x\rangle$ will reduce in a classical language to either $\langle 2, 2\rangle$ or $\langle 2, 1\rangle$ depending on which component is reduced first.

A *quantum store* Q is a function taking extended variables $x_1 \otimes \cdots \otimes x_n$ taken in a finite domain of extended variables $|Q|$ to a state $|x_1 \dots x_n\rangle_Q \in \left(\mathbb{C}^2\right)^{\otimes n}$. The domain $|Q|$ is assumed to contain only disjoint extended variables. A quantum store holds the state of the quantum registers that are used in a quantum λ-calculus term. We drop the index Q when the context makes it clear to which quantum store a state belongs.

A quantum store Q can be modified in various ways. First, it can be extended by the addition of a new quantum register; since this is similar to the extension of a classical store we use the notation $Q[|x_1 \dots x_n\rangle \mapsto |\varphi\rangle]$ to denote the extension of Q to a store with domain $|Q| \cup \{x_1 \otimes \cdots \otimes x_n\}$ and associating to the new extended variable the state $|x_1 \dots x_n\rangle = |\varphi\rangle$.

Another important operation is *preparation* of extra qbits appended to a cell of a given quantum store Q. If $x_1 \otimes \cdots \otimes x_n \in |Q|$, then $Q[|x_1 \dots x_n y_1 \dots y_m\rangle \mapsto |x_1 \dots x_n\rangle|0 \dots 0\rangle]$ is the quantum store with $x_1 \otimes \cdots \otimes x_n$ removed from $|Q|$ and $x_1 \otimes \cdots \otimes x_n \otimes y_1 \otimes \cdots \otimes y_m$ added, and with associated state $|x_1 \dots x_n y_1 \dots y_n\rangle = |x_1 \dots x_n\rangle|0 \dots 0\rangle$. Note that by definition of quantum store, $\{x_1, \dots x_n\}$ and $\{y_1, \dots y_m\}$ are disjoint.

The final operation that we need is the modification of one register using a unitary operation or a projection. Given a quantum store Q and a linear map A over the Hilbert space associated to the extended variable $x_1 \otimes \cdots \otimes x_n \in |Q|$, we denote by $Q[|x_1 \dots x_n\rangle \mapsto A|x_1 \dots x_n\rangle]$ the quantum store where $|x_1 \dots x_n\rangle$ is replaced by $A|x_1 \dots x_n\rangle$.

A QSL *program* is a pair $Q, \Gamma \vdash M : A$ where Q is a quantum store, $\Gamma \vdash M : A$ is a valid typing judgement such that all the qstore variables in Γ are in $|Q|$. We say that a program Q, M is *closed* if $|\Gamma| \subseteq |Q|$. To simplify the notation, we will often leave the types implicit and write Q, M instead of $Q, \Gamma \vdash M : A$.

A *value* for QSL is a term of the recursively defined form

$$V ::= x_1 \otimes \cdots \otimes x_n \mid 0 \mid 1 \mid * \mid \mathsf{skip} \mid \lambda\overline{y}.\, M \mid \langle M, N\rangle,$$

where $\overline{x}$ can be any extended variable and M is any term with $\overline{y} \in \mathrm{FV}(M)$.

We define the operational semantics of QSL as a big-step probabilistic reduction relation between programs. The notation $Q, M \Downarrow^p Q', V$ means that when M is run with a quantum store in state Q, it reduces with probability p to the value V with the quantum store left in state Q'. When $p = 1$, we omit the probability argument and write simply $Q, M \Downarrow Q', V$. This relation is defined inductively by the rules in table 2. Most of these reduction rules are the usual reduction rules for the simply typed λ-calculus, sequential composition, conditionals and pairing. The reduction rules for the classical part of the language do not affect the quantum stores. The rules involving measurements, preparations or unitary transformations

Table 2
Big-step reduction for the λ-calculus with quantum stores.

$$\frac{}{Q, V \Downarrow Q, V} \qquad \frac{Q, M \Downarrow^p Q', \lambda x.\, M' \qquad Q', M'[N/x] \Downarrow^q Q'', V}{Q, MN \Downarrow^{pq} Q'', V}$$

$$\frac{Q, M \Downarrow^p Q', V}{Q, \mathsf{fst}\,\langle M, N\rangle \Downarrow^p Q', V} \qquad \frac{Q, N \Downarrow^p Q', V}{Q, \mathsf{snd}\,\langle M, N\rangle \Downarrow^p Q', V} \qquad \frac{Q, M \Downarrow^p Q', \mathsf{skip} \qquad Q', N \Downarrow^q Q'', V}{Q, M; N \Downarrow^{pq} Q'', V}$$

$$\frac{Q, P \Downarrow^p Q', 0 \qquad Q', N \Downarrow^q Q'', V}{Q, \mathsf{if}\ P\ \mathsf{then}\ M\ \mathsf{else}\ N \Downarrow^{pq} Q'', V} \qquad \frac{Q, P \Downarrow^p Q', 1 \qquad Q', M \Downarrow^q Q'', V}{Q, \mathsf{if}\ P\ \mathsf{then}\ M\ \mathsf{else}\ N \Downarrow^{pq} Q'', V}$$

$$\frac{}{Q, Uy_1 \otimes \ldots \otimes y_m \Downarrow Q[|x_1 \ldots x_n\rangle \mapsto U|x_1 \ldots x_n\rangle], \mathsf{skip}}$$

$$\frac{}{Q, \mathsf{meas}\, x_j \Downarrow^{\||[0]^{x_j}|x_1 \ldots x_n\rangle\|} Q[|x_1 \ldots x_n\rangle \mapsto [0]^{x_j}|x_1 \ldots x_n\rangle / \||[0]^{x_j}|x_1 \ldots x_n\rangle\|], 0}$$

$$\frac{}{Q, \mathsf{meas}\, x_j \Downarrow^{\||[1]^{x_j}|x_1 \ldots x_n\rangle\|} Q[|x_1 \ldots x_n\rangle \mapsto [1]^{x_j}|x_1 \ldots x_n\rangle / \||[1]^{x_j}|x_1 \ldots x_n\rangle\|], 1}$$

$$\frac{Q[|x_1 \ldots x_n\rangle \mapsto |0 \ldots 0\rangle], M \Downarrow^p Q', V}{Q, \mathsf{new}\ x_1 \otimes \cdots \otimes x_n\ \mathsf{in}\ M \Downarrow Q', V} \quad x_1 \otimes \cdots \otimes x_n \notin |Q|$$

$$\frac{Q[|x_1 \ldots x_n y_1 \ldots y_m\rangle \mapsto |\varphi\rangle|0\rangle], M \Downarrow^p Q', V}{Q[|x_1 \ldots x_n\rangle \mapsto |\varphi\rangle], \mathsf{prep}\ \overline{y}\ \mathsf{with}\ \overline{x}\ \mathsf{in}\ M \Downarrow^p Q', V}$$

change the quantum stores according to quantum mechanics. For example, the rule for measurement says that if x_i is measured with a quantum store in state Q, then the state $|x_1 \ldots x_n\rangle_Q$ where x occurs is projected with the projection $[0]^{x_i}$ or $[1]^{x_i}$, depending on the measurement result, and normalised. Note that this is the only place where there is a probabilistic branching in the reduction. For a unitary transformation operation U, the part of the quantum store Q affected by U is updated to $U|x_1 \ldots x_n\rangle$ and the term reduces to the command skip.

Example 3.1 Consider the following two terms M_1 and M_2 defined respectively by

$$M_1 \colon\ \wedge U x \otimes y \qquad M_2 \colon \mathsf{if}\ \mathsf{meas}\, x\ \mathsf{then}\ (U\, y)\ \mathsf{else}\, \mathsf{skip}$$

where $\wedge U$ denote the controlled version of a unitary operation U. This is defined by $\wedge U|b_1\rangle|b_2\rangle = |b_1\rangle|b_1 \oplus b_2\rangle$, where $\oplus$ is the exclusive-or operation. We have that $x \otimes y \colon \mathrm{qstore} \vdash M_1, M_2 \colon \mathrm{com}$. In a quantum store state Q which assign $|\varphi\rangle$ to $x \otimes y$, M_1 reduce to skip and the state Q is modified by the unitary operation:

$$Q, M_1 \Downarrow Q\left[|xy\rangle \mapsto \wedge U|xy\rangle\right], \mathsf{skip}.$$

The term M_2 also reduce to skip but leaves the quantum store in a different state:

$$Q\left[|xy\rangle \mapsto |\varphi\rangle\right], M_2 \Downarrow^p Q\left[|xy\rangle \mapsto [0]^x|xy\rangle\right], \mathsf{skip}$$
$$Q\left[|xy\rangle \mapsto |\varphi\rangle\right], M_2 \Downarrow^{1-p} Q\left[|xy\rangle \mapsto U^y[1]^x|xy\rangle\right], \mathsf{skip}$$

where $p = \mathrm{tr}\left([0]^x|\varphi\rangle\langle\varphi|\right)$.

Example 3.2 It is possible to program the quantum teleportation protocol [1] in the quantum store language. It is represented as a term $\mathsf{teleport}_{xz}$ which transfers an unknown state from some quantum store x to another quantum store z:

$$
\begin{array}{l}
\mathsf{prep}\, y \otimes z \,\mathsf{with}\, x \,\mathsf{in} \\
\quad H\, y;\ \wedge X y \otimes z; \\
\quad H\, x;\ \wedge X\, x \otimes y; \\
\quad \mathsf{let}\, b_x = \mathsf{meas}\, x,\ b_y = \mathsf{meas}\, y \,\mathsf{in} \\
\qquad \mathsf{if}\, b_x \,\mathsf{then} \\
\qquad\quad \mathsf{if}\, b_y \,\mathsf{then}\, U_{11}\, z \,\mathsf{else}\, U_{10}\, z \\
\qquad \mathsf{else} \\
\qquad\quad \mathsf{if}\, b_y \,\mathsf{then}\, U_{01}\, z \,\mathsf{else}\, U_{00}\, z
\end{array}
$$

where H is the Hadamard tranformation and $U_{00} = I$, $U_{01} = X$, $U_{10} = Z$ and $U_{11} = ZX$ are the four possible correction operations, one of which must be applied to z to change its state to that of the input quantum store x. If follows from the typing rules that

$$x\colon \mathrm{qstore} \vdash \mathsf{teleport}_{xz}\colon \mathrm{com}$$

The command $\mathsf{teleport}_{xz}$ performs the teleportation protocol to transfer the state of the qbit register x to the qbit register z. This can be verified using the operational semantics rules: it is possible to derive that

$$Q, \mathsf{teleport}_{xz} \Downarrow Q\left[|xyz\rangle \mapsto U^z_{b_x b_y}[b_x]^x[b_y]^y \mathrm{cnot}^{xy} H^x |xyz\rangle\right], \mathsf{skip},$$

where we label each unitary transformation and projectors by the subspace associated to the label variables.

Note that it is possible to represent any quantum circuit as a term of QSL. The input is fed to the circuit using a quantum store $\overline{x}$. Some ancilla qbits can be added to the input state using a $\mathsf{prep}\ldots\mathsf{with}\ldots\mathsf{in}$ command. The unitary gates of the circuit are added as unitary commands which are composed using sequential composition. The resulting state can then be measured using meas commands.

4 Denotational semantics

4.1 Probabilistic game semantics

The game semantics presented in this paper is constructed using the definitions of probabilistic games semantics introduced by Danos and Harmer [3]. We give here an overview of the basic definitions and facts of probabilistic game semantics.

Definition 4.1 An *arena* A is a triple $(M_A, \lambda_A, \vdash_A)$ where M_A is a set of *moves*, the function $\lambda_A\colon M_A \to \{\mathrm{O},\mathrm{P}\}\times\{\mathrm{Q},\mathrm{A}\}\times\{\mathrm{I},\mathrm{N}\}$ is a labeling which assigns moves to the two players *Opponent* and *Player*, and tells us which moves are *Questions* and which are *Answers*, and whether they are *Initial* or *Noninitial* moves, and finally $\vdash_A \subseteq M_A \times M_A$ is a relation, called the *enabling relation*, such that

(A1) if $a \vdash_A b$, then $\lambda_A^{\mathrm{OP}}(a) \neq \lambda_A^{\mathrm{OP}}(b), \lambda_A^{\mathrm{QA}}(a) \neq \lambda_A^{\mathrm{QA}}(b)$,

(A2) if $\lambda_A^{\mathrm{IN}}(a) = \mathrm{I}$, then $\lambda_A(a) = \mathrm{OQI}$,

(A3) if $a \vdash b$ and $\lambda_A^{\mathrm{QA}}(b) = \mathrm{A}$ then $\lambda_A^{\mathrm{QA}}(a) = \mathrm{Q}$,

where the functions λ_A^{OP}, λ_A^{QA} and λ_A^{in} are λ_A composed with the projections on the sets $\{\mathrm{O},\mathrm{P}\}$, $\{\mathrm{Q},\mathrm{A}\}$ and $\{\mathrm{I},\mathrm{N}\}$.

We use the convention that M_A^X, where X is some list of superscripts taken from the set of move labels $\{\mathrm{O},\mathrm{P},\mathrm{Q},\mathrm{A},\mathrm{I},\mathrm{N}\}$ denote the set of moves labeled with these labels. Moves in an arena are thus of various types, and the constraints on the enabling relation $\vdash_A$ limit the possible interactions in the arena by limiting which moves can be made at a certain point given the past interactions. The condition (A1) forces that only Player moves to enable Opponent moves and *vice versa*, (A2) asks for all initial moves to be questions by Opponent and finally (A3) says that answers can only be enabled by questions.

A *play* in A is a sequence of moves $s \in M_A^*$. This does not take into account the enabling relation; we define a *justified play* to be a play where each occurrence of a non-initial move b has a pointer to a previous occurrence of a move a with $a \vdash_A b$. We finally need to enforce alternation of the two players. A *legal play* is a justified play where Opponent and Player alternate; we denote the set of legal plays in A by $\mathcal{L}_A$. Note that because all initial moves are Opponent moves, Opponent is always making the first move. The sets of odd and even length legal plays are respectively denoted by $\mathcal{L}_A^{\mathrm{odd}}$ and $\mathcal{L}_A^{\mathrm{even}}$.

Example 4.2 The **bool** arena is defined with $M_{\mathbf{bool}} = \{?, 0, 1\}$ $\lambda_{\mathbf{bool}}(?) = (\mathrm{O},\mathrm{Q},\mathrm{I})$ and $\lambda_{\mathbf{bool}}(0) = \lambda_{\mathbf{bool}}(1) = (\mathrm{P},\mathrm{A},\mathrm{N})$ and with the enabling relation $? \vdash_{\mathbf{bool}} 0, 1$.

Example 4.3 The *empty arena* I is the arena with no moves at all. The only legal play in I is the empty play ε.

Suppose $sa \in \mathcal{L}_A$. Starting from a and following the justification pointers will always lead to an occurrence of an initial move b, which we call the *hereditary justifier* of a in sa. We can see that every legal play will be partitioned in subplays, each one consisting of all occurrences of moves hereditarily justified by a given initial move. These subplays are called *threads*. The *current thread* of a legal play sa ending with an opponent move, denoted by $\lceil sa \rceil$, is the thread of sa where a occurs. If sa ends with a Player move, the current thread is then defined by $\lceil s \rceil a$. We want the current thread to be a legal play, so it is necessary to impose an extra condition on legal plays: a legal play s is *well-threaded* if for every subplay ta ending with a Player move, the justifier of a is in $\lceil t \rceil$. In a well-threaded play, player always plays in the last thread where Opponent played.

Given arenas A, B, the *product* $A \odot B$ and *linear arrow* $A \multimap B$ operations are defined respectively as follows:

- $M_{A\odot B} = M_A + M_B$ (disjoint union)
- $\lambda_{A\odot B} = [\lambda_A, \lambda_B]$ (copairing)
- $m \vdash_{A\odot B} n$ iff $m \vdash_A n$ or $m \vdash_B n$.

- $M_{A\multimap B} = M_A + M_B$
- $\lambda_{A\multimap B} = \left[\langle \overline{\lambda}_A^{\mathrm{OP}}, \lambda_A^{\mathrm{QA}}, \overline{\lambda}_A^{\mathrm{IN}} \rangle, \lambda_B\right]$
- $m \vdash_{A\multimap B} n$ iff $m \vdash_A n$ or $m \vdash_B n$ or $\lambda_B^{\mathrm{IN}}(n) = \lambda_A^{\mathrm{IN}}(m) = \mathrm{I}$.

where $\overline{\lambda}_A^{\mathrm{OP}}$ inverts the roles of the two players and $\overline{\lambda}_A^{\mathrm{IN}}$ makes all moves of A non-initial. The product arena $A \odot B$ is intuitively understood as the arena where at each of Opponent's turn she can choose to play a move in either A or B, and where Player must answer in the last component where Opponent played. In the arena $A \multimap B$, after Opponent makes an initial move in B, at each of his turns Player can choose to play either one of his moves in B or an Opponent move in A.

Given a legal play s in an arena A, let $\mathrm{next}_A(s) = \{a \in M_A | sa \in \mathcal{L}_A\}$ be the set of all moves that can be legally made after the play s.

Definition 4.4 A *probabilistic strategy* for Player is a function $\sigma \colon \mathcal{L}_A^{\mathrm{even}} \to [0,1]$ such that

$$\sigma(\epsilon) = 1 \quad \text{and} \quad \sigma(s) \geq \sum_{b \in \mathrm{next}(sa)} \sigma(sab)$$

The set of *traces* of a strategy σ in A is the set of even length legal plays which are assigned a non-zero probability by σ: it is denoted $\mathcal{T}_\sigma$. A strategy σ is *deterministic* if $\sigma(s) = 1$ for all $s \in \mathcal{T}_\sigma$.

It is possible to describe a probabilistic strategy σ in conditional form. The probability $\sigma(b \mid sa) = \frac{\sigma(sab)}{\sigma(s)}$ is the probability of Player choosing to play b after the play sa.

Composition of strategies is the way interactions between parts of a program are encoded in game semantics. Given two strategies $\sigma \colon A \multimap B$ and $\tau \colon B \multimap C$, we define a new strategy $\sigma ; \tau \colon A \multimap C$ obtained by letting σ and τ "interact" on B. Before giving the definition of composition, it is necessary to formalise this notion of interaction. The set of interactions for A, B, C is

$$\mathcal{I}_{A,B,C} = \{u \in (M_A + M_B + M_C)^* \mid u|_{AB} \in \mathcal{L}_{A \multimap B}, u|_{BC} \in \mathcal{L}_{B \multimap C}, u|_{AC} \in \mathcal{L}_{A \multimap C}\}$$

where $u|_{AB}$ is the sub sequence of u obtained by deleting the moves of C, and similarly for $u|_{BC}$. The case of $u|_{AC}$ is a bit different because deleting from u the moves of B and their associated pointers might leave the moves of A or C that are justified by B-moves without justifiers. In this case, we define the justifiers of $u|_{AC}$ to be as follows: a move a in C justified by a move b in B will be justified by the first move of either A or C we get to by following back the justification pointers from a in u. The set of *witnesses* $\mathrm{wit}(s)$ of $s \in \mathcal{L}_{A \multimap C}$ in an interaction $\mathcal{I}_{A,B,C}$ is the set of interactions $u \in \mathcal{I}_{A,B,C}$ such that $u|_{AC} = s$. The composition of two strategies $\sigma \colon A \multimap B$ and $\tau \colon B \multimap C$ can now be defined as follows:

$$[\sigma ; \tau](s) = \sum_{u \in \mathrm{wit}(s)} \sigma(u|_{AB}) \tau(u|_{BC}).$$

The *identity strategy* (or so-called "copycat strategy") $\mathrm{id}_A \colon A \multimap A$ is neutral with respect to composition. It is defined as the strategy which makes Player copy Opponent moves between corresponding components. Formally, this is defined as the deterministic strategy with trace

$$\mathcal{T}\left(1_A(s)\right) = \left\{ s \in \mathcal{L}_{A_l \multimap A_r} \mid \forall s' \sqsubseteq^{\mathrm{even}} s.\, s'|_{A_r} = s'|_{A_r} \right\}.$$

Using all the structure defined so far it is possible to define a category of arenas and probabilistic strategies. Taking arenas as objects, a morphism $A \to B$ is a strategy in $A \multimap B$. Composition of strategy is the needed composition, with the identity strategies as identity morphisms. It is associative, and it is shown in [3] that probabilistic strategies are closed under composition. This category is also symmetric monoidal. The operation $\odot$ is a tensor product, which acts on morphisms as follows. Given $\sigma\colon A \to C$ and $\tau\colon B \to D$ and $s \in \mathcal{L}^{\text{even}}_{(A\odot B)\multimap(A'\odot B')}$, we set $[\sigma \odot \tau](s) = \sigma(s|_{A\multimap C})\tau(s|_{C\multimap D})$. All coherence isomorphisms are easily defined using variants of the copycat strategy.

Threads have an important role in game semantics as a way to characterize the strategies that encodes programs with side-effects, like stores. This is achieved by forcing Player to use only the limited information available in the current thread instead of using all the information that can be extracted from the whole previous plays, including move made in other threads.

A strategy σ is *well-threaded* if $\mathcal{T}_\sigma$ consists only of well-threaded plays. Note that this condition forces Player to answer in the last thread where Opponent played. Given two well-threaded plays $sab \in \mathcal{L}^{even}_A$ and $ta \in \mathcal{L}^{odd}_A$ with $\lceil sa \rceil = \lceil ta \rceil$, we define $\text{match}(sab, ta)$ to be the unique legal play tab with b justified as in $\lceil sa \rceil$. A well-threaded strategy σ is said to be *thread independent* if $sab \in \mathcal{T}_\sigma$, $t \in \mathcal{T}_\sigma$, $a \in \text{next}(t)$ and $\lceil sa \rceil = \lceil ta \rceil$ implies that

$$\begin{matrix}\sigma(sab)\\ \sigma(s)\end{matrix} = \begin{matrix}\sigma((\text{match}(sab, ta))\\ \sigma(t)\end{matrix}.$$

The meaning of this condition is that if Player plays according to σ, Player chooses his answers with probabilities that only depend on the current thread, i.e. $\sigma(b \mid sa) = \sigma(b \mid ta)$.

The diagonal strategy $\Delta_A\colon A \to A \odot A$ is defined as the deterministic strategy with trace set $\left\{ s \in \mathcal{L}^{even}_{A\multimap A_l\odot A_r} \mid \forall s' \sqsubseteq^{\text{even}} s.s'|_{A_l} \in \text{id}_{A_l} \wedge s'|_{A_r} \in \text{id}_{A_r} \right\}$. This is similar to the definition of the identity strategy: Δ instructs Player to use copying strategies between A and its two copies A_l and A_r. Possible conflicts in A are resolved by separating in different threads moves made according to the left or the right copy plays. There is also a unique strategy $\diamond_A \multimap I$, namely the trivial strategy with trace $\{\varepsilon\}$.

The *pairing* of two thread independent strategies $\sigma\colon A \multimap B$ and $\tau\colon A \multimap C$ is defined by $\langle \sigma, \tau \rangle = \Delta_A; \sigma \odot \tau$. Thus when Player plays using the pair strategy $\langle \sigma, \tau \rangle$, he plays using σ after an initial move in B, and using τ after an initial move in C.

For each arena A, $(A, \Delta_A, \diamond_A)$ is a *comonoid*. It is shown in in [7] that a strategy $\sigma\colon A \multimap B$ is thread independent if and only if σ is a comonoid homomorphism. Using a known fact in category theory[8], this implies that the restriction of the category of arena and probabilistic strategies to thread independent strategies is a Cartesian closed category. Note that projection strategies like $\pi_A\colon B \otimes A \multimap A$ are defined as copying strategies which makes Player copies Opponent's moves between the two A component arenas.

4.2 The quantum store arena

While all the classical operations of QSL have known game semantics interpretations, we need new tools to be able to define the denotational semantics of the quantum store operations. The main idea used to describe a state ρ as a strategy is borrowed from the consistent histories approach to quantum mechanics [4,9,6]: sequences of measurement results are used to describe the evolution of quantum states.

Definition 4.5 The **qstore** arena is the arena with quantum interventions $\mathcal{E}_? = \left\{\mathcal{E}^?_m\right\}$ as questions and natural numbers m as answers. The question $\mathcal{E}_?$ enables its possible measurements results.

A play in this arena is a sequence of moves $\mathcal{E}_{?[1]}m_1 \cdots \mathcal{E}_{?[n]}m_n$ where the quantum interventions $\mathcal{E}_{?[k]}$ may all be different. We need a strategy $[\rho]$ in **qstore** which describes a quantum state ρ.

Definition 4.6 The probabilistic strategy $[\rho]$ in **qstore** associated to a density matrix ρ is defined by $[\rho](\epsilon) = 1$ and $[\rho](\mathcal{E}_{?[1]}m_1 \ldots \mathcal{E}_{?[n]}m_n) = \operatorname{tr}\left(\mathcal{E}^{?[n]}_{m_n} \ldots \mathcal{E}^{?[1]}_{m_1}(\rho)\right)$.

Note that since we use the convention that impossible composition of superoperators yields the zero operator, the above definition assigns probability zero to plays which involve domain inconsistencies. For example, if Opponent asks another question $\mathcal{E}_{?[2]}$ after receiving an answer to $\mathcal{E}_?[1]$, all possible Player answers will have probability zero when the domain of $\mathcal{E}_{?[2]}$ is different than $\mathbf{SD}(H_{m_1})$. When the domain and $\mathbf{SD}(H_m)$ match, the question $\mathcal{E}_{?[2]}$ is answered using the normalised state $\mathcal{E}^{?[1]}_{m_1}(\rho)/\operatorname{tr}\left(\mathcal{E}^{?[1]}_{m_1}(\rho)\right)$.

It is easy to verify this satisfies the definition of probabilistic strategies. Note that the strategy $[\rho]$ is thread dependent: the first question is answered using the probabilities given by $p_{m_1} = \operatorname{tr}\left(\mathcal{E}^{?[1]}_{m_1}(\rho)\right)$, but a second question in a new thread will be answered with the probability distribution given by $\operatorname{tr}\left(\mathcal{E}^{?[2]}_{m_2}\mathcal{E}^{?[1]}_{m_1}(\rho)\right)/p_{m_1}$, i.e. using the updated state $\mathcal{E}^{?[1]}_{m_1}(\rho)/p_{m_1}$. Thus in general the probability distribution used is different in different threads, and is updated according to the laws of quantum mechanics.

Example 4.7 We can define a strategy which describes a unitary operation. This is a strategy $[U]$ in the arena **qstore** $\to$ **com**. Suppose that the superoperator corresponding to U is $\mathcal{U}$. A typical play using $[U]$ is "run $\{\mathcal{U}_0\}_?$ 0 done". The $\{\mathcal{U}_0\}_?$ question in the **qstore** arena change the state used to answers future questions in the arena. Notice that Player does not learn anything about the state in this interaction with Opponent because there is only one possible measurement result. The strategy $[U]$ really describe the effect of U since one can verify that $[\rho];[U] = [\mathcal{U}(\rho)];\mathrm{skip}$ using the definition of composition of strategies.

Example 4.8 We define a strategy which represents performing a projective mea-

surement of the state of a quantum store as follows:

$$\begin{array}{ccc} \mathbf{qstore} & \xrightarrow{\ \mathsf{meas}\ } & \multimap \mathbf{bool} \\ & & ? \\ \{\mathcal{P}_0, \mathcal{P}_1\}_? & & \\ m & & \\ & & m \end{array}$$

where $\mathcal{P}_m$ is the projection superoperator on the canonical basis state $|m\rangle$. The measurement strategy makes Player answer the first question in the output Boolean component by asking about the result of a measurement in the computational basis of the input qbit and copying the answer m to the output component. In contrast to the case of unitary transformations, Player does learn some information about the input state in the part of the exchange happening in the **qstore** arena, and this information is used to provide an answer in the **bool** arena.

4.3 Definition of the denotational semantics

We now use quantum strategies to construct a denotational semantics for the quantum store language. For each type A, we want to define an arena $[\![A]\!]$, and given a term $\Gamma \vdash M : A$, we want a strategy $[\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$.

For types, the definition is given by the following inductive construction :

$$[\![\mathrm{bool}]\!] = \mathbf{bool} \quad [\![\mathrm{com}]\!] = \mathbf{com} \quad [\![\top]\!] = \top \quad [\![\mathrm{qstore}]\!] = \mathbf{qstore}$$
$$[\![A \times B]\!] = [\![A]\!] \odot [\![B]\!] \quad [\![A \Rightarrow B]\!] = [\![A]\!] \multimap [\![B]\!]$$

The arena $\top$ has one possible even-length play: $?*$, and there is thus only one possible strategy aside from the empty one. We denote this strategy $*$. The arena **com** is defined similarly, but with the moves "run" and "done" instead. Intuitively, in the **com** arena Opponent asks Player to run a command, and Player confirms when it is done. The quantum store type is interpreted using the arena **qstore**.

Given a context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, we set $[\![\Gamma]\!]$ to be $[\![A_1]\!] \odot \cdots \odot [\![A_n]\!]$. The interpretation $[\![\Gamma \vdash M : A]\!]$ is defined by induction on the derivation of $\Gamma \vdash M : A$ in what follows.

We begin the definition of $[\![\Gamma \vdash M : A]\!]$ with the base cases of variables and constant terms. The interpretation of $\Gamma, x : A \vdash x : A$ using the projection strategy π_A. The Boolean constants $0, 1$ are interpreted as their corresponding deterministic strategies in **bool**. The denotation of $*$ is the unique non-trivial deterministic strategy $* : [\![\Gamma]\!] \multimap \top$. Similarly, the constant skip is interpreted as the unique non-trivial deterministic strategy skip in **com**.

The strategy $[\![U\, y_1 \otimes \cdots \otimes y_m]\!]$ corresponding to unitary transformation is defined as the strategy $[U] : \mathbf{qstore} \multimap \mathbf{com}$. In the case of measurements, $[\![\mathsf{meas}\, x_i]\!]$ is interpreted as the meas strategy.

We now turn to the inductive cases. The definition of $[\![M_1; M_2]\!]$ follows the standard idea in game semantics: it is defined as the composition $\langle [\![M_1]\!], [\![M_2]\!] \rangle ; \mathsf{seq}$,

where seq is the strategy $\mathbf{com} \odot \mathbf{com} \multimap \mathbf{com}$ defined with the following typical play:

Using this scheme, the commands M_1 and M_2 are successively ran when seq is composed with $\langle [\![M_1]\!], [\![M_2]\!] \rangle$. The other classical operations are also interpreted using the usual game semantics ideas. We refer the reader to [7] for a detailed account.

For quantum store creation using new, suppose that the denotation of $\Gamma, x_1 \otimes \cdots \otimes x_n \colon \mathrm{qstore} \vdash M \colon A$ is already defined. The term $\mathsf{new}\ x_1 \otimes \cdots \otimes x_n\ \mathsf{in}\ M$ is interpreted as the composition $\mathrm{id} \odot [|0\rangle\langle 0|] \,;\, [\![M]\!]$. The strategy $[|0\rangle\langle 0|]$ is used to initiate the state of the new quantum store.

The last case is for the preparation typing rule. The strategy $[\![\mathsf{prep}\,\overline{y}\,\mathsf{with}\,\overline{x}\,\mathsf{in}\,M]\!]$ is defined as the strategy $\mathrm{prep}\,([\![M]\!]) : [\![\Gamma]\!] \odot \mathbf{qstore} \multimap A$ defined with the following idea. Let $\mathcal{F}_0$ be the preparation superoperator taking ρ to $\rho \otimes |0 \ldots 0\rangle\langle 0 \ldots 0|$. Player plays using $\mathrm{prep}\,([\![M]\!])$ by making the moves prescribed by $[\![M]\!]$ except that before playing his first move in the qstore arena, he must initiate an exchange in this arena which forces Opponent to add the $|0 \cdots 0\rangle$ state to the state ρ she uses to answer Player's questions about the state of the quantum store. This is achieved by playing a $\{\mathcal{F}_0\}_?$ quantum intervention question in the **qstore** arena before any other move is played there.

This completes the definition of the denotational semantics.

5 Soundness

To study the relation between the operational and denotational semantics, we need to take quantum stores into account. We use the standard approach used in game semantics of classical stores, described in the last chapter for the language MCdata: we define a strategy $[\![Q, M]\!]$ for each pair Q, M. This strategy is defined as the composition of $[\![M]\!]$ with a strategy $[\![Q]\!]$ representing the state of the quantum registers in Q. For each extended variable $x_1 \otimes \cdots \otimes x_n \in |Q|$, the state $|x_1 \ldots x_n\rangle_Q$ can be described as a strategy $[|x_1 \ldots x_n\rangle]$ in $I \multimap \mathbf{qstore}$. The strategy $[\![Q]\!]$ associated to the quantum store Q is defined as the $\odot$-product of all the strategies $[|x_1 \ldots x_n\rangle]$, $x_1 \otimes \cdots \otimes x_n \in |Q|$.

Lemma 5.1 *(Substitution) For any QSL terms $\Gamma, x \colon A \vdash M \colon B$ and $\Gamma \vdash N \colon A$ with $x \in \mathrm{FV}(M)$, we have that $\Gamma \vdash M\,[N/x] : B$ and $[\![M[N/x]]\!] = \langle \mathrm{id}_{[\![\Gamma]\!]}, [\![N]\!] \rangle ; [\![M]\!]$.*

Proof. A standard structural induction on the construction of M. □

Proposition 5.2 *(Consistency for QSL) Let M and V be two terms of ground type.*

If $Q, M \Downarrow^p Q', V$, *then for all well-opened* $sab \in \mathcal{T}(\llbracket Q', V \rrbracket)$ *we have that*

$$\llbracket Q, M \rrbracket (b \mid sa) = p \llbracket Q, V \rrbracket (b \mid sa).$$

Proof. The proof is a structural induction on the derivation of $Q, M \Downarrow^p Q', V$. We show how to deal with the most interesting cases. In the case of a unitary transformation operation U, suppose that $Q, U\ x_1 \otimes \cdots \otimes x_n \Downarrow Q\left[|x_1 \ldots x_n\rangle \mapsto U|x_1 \ldots x_n\rangle\right], \mathsf{skip}$ holds. By definition of the denotational semantics, we have that $\llbracket Q, U\ x_1 \otimes \cdots \otimes x_n \rrbracket$ is the composition

$$I \quad {}^{\llbracket Q \rrbracket} \quad \circ \llbracket \Gamma \rrbracket \, {}^{\llbracket x_1 \otimes \cdots \otimes x_n \rrbracket} \circ \mathbf{qstore} \quad {}^{[U]} \quad \circ \mathbf{com}$$

A run move in the final **com** arena is answered with the question $\{\mathcal{U}_0\}_?$ in the **qstore** arena and then copied by the projection strategy to the $\llbracket \Gamma \rrbracket$ arena, where an interaction begins with $\llbracket Q \rrbracket$ in which the unitary transformation move $\{\mathcal{U}_0\}_?$ is made, affecting all subsequent interactions in the **qstore** component. The 0 answers that Opponent gives back to Player is copied back to the initial **qstore** arena, and then a "done″ move is made in the **com** arena. In any further interaction with the quantum store strategy $\llbracket Q \rrbracket$ Player will behave as if he is using the strategy $\llbracket Q[|x_1 \ldots x_n\rangle \mapsto U|x_1 \ldots x_n\rangle] \rrbracket$. If Player uses the strategy $\llbracket Q[|x_1 \ldots x_n\rangle \mapsto U|x_1 \ldots x_n\rangle], \mathsf{skip} \rrbracket$, then the behaviour is the same: the initial "run" move is answered with "done" without interacting with the strategy $\left[\left[Q\left[|x'_1 \ldots x'_{n'}\rangle \mapsto U|x'_1 \ldots x'_{n'}\rangle\right]\right]\right]$.

The two rules for quantum measurement operations are dealt with similarly. Suppose that

$$Q, \mathsf{meas}\, x_i \Downarrow^{\mathrm{tr}([0]^{x_i}[x_1 \ldots x_n])} Q\left[|x_1 \ldots x_n\rangle \mapsto [0]^{x_i}|x_1 \ldots x_n\rangle / \|[0]^{x_i}|x_1 \ldots x_n\rangle\|\right], 0.$$

By definition we have that $\llbracket Q, \mathsf{meas}\, x_i \rrbracket$ is the strategy $\llbracket Q \rrbracket ; \llbracket x_i \rrbracket ; \mathrm{meas}$ in $I \multimap \llbracket \Gamma \rrbracket \multimap \mathbf{qstore} \multimap \mathbf{bool}$. Any interaction starting with the question ? in **bool** is answered by measuring in the canonical basis the qbit of the arena **qstore**. The answer to this is given according to $\llbracket Q \rrbracket$ and is 0 with probability $\|[0]^{x_i}|x_1 \ldots x_n\rangle\|$. Any further interaction with $\llbracket Q \rrbracket$ will be made according to $\llbracket Q[|x_1 \ldots x_n\rangle \mapsto [0]^{x_i}|x_1 \ldots x_n\rangle] \rrbracket$, and the answer to the initial question in **bool** is 0. This amounts to saying that $\llbracket Q, \mathsf{meas}\, x_i \rrbracket$ behaves like $\left[\left[Q\left[|x_1 \ldots x_n\rangle \mapsto [0]^0|x_1 \ldots x_n\rangle\right], 0\right]\right]$ with probability $\|[0]^{x_i}|x_1 \ldots x_n\rangle\|$. The other measurement case is similar.

The most interesting induction case is the preparation case. Suppose that the proposition holds when $Q\left[|x_1 \ldots x_n\rangle|y_1 \ldots y_m\rangle \mapsto |\varphi\rangle|0 \ldots 0\rangle\right] \Downarrow^p Q', V$. Assume that $Q\left[|x_1 \ldots x_n\rangle \mapsto |\varphi\rangle\right], \mathsf{prep}\ \overline{y}\ \mathsf{with}\ \overline{x}\ \mathsf{in}\ M \Downarrow^p Q', V$. By definition of $\llbracket \Gamma, \overline{x} \colon \mathrm{qstore} \vdash \mathsf{prep}\ \overline{y}\ \mathsf{with}\ \overline{x}\ \mathsf{in}\ M \colon A \rrbracket$, any play in $\llbracket \Gamma \rrbracket \odot \mathbf{qstore} \multimap \llbracket A \rrbracket$ will be played with player using the strategy $\llbracket M \rrbracket$, except that a preparation move is made in **qstore**. This preparation move is answered by Opponent using the $\llbracket Q\left[|x_1 \ldots x_n\rangle \mapsto |\varphi\rangle\right] \rrbracket$ strategy, which make her pick her answers using the strategy $[|\varphi\rangle\langle\varphi|]$. After the preparation move, Opponent will play as if she is using the strategy $[|\varphi\rangle\langle\varphi||0 \ldots 0\rangle\langle 0 \ldots 0|]$, which is $\llbracket Q\left[|x_1 \ldots x_n\rangle|y_1 \ldots y_m\rangle \mapsto |\varphi\rangle|0 \ldots 0\rangle\right] \rrbracket$. The

overall play is thus just like what would happen if Player uses $[\![M]\!]$ composed with this last strategy. We get the desired result because the induction hypothesis implies that composed strategy dictates the same moves to Players as the strategy $[\![Q', V]\!]$. □

A term $\Gamma \vdash M : A$ is said to be *semi-closed* if $\mathrm{FV}(M)$ contains only variables of type qstore.

Proposition 5.3 *(Adequacy for QSL) Let M be a semi-closed term of ground type. If for all well opened $sab \in \mathcal{T}([\![Q', V]\!])$ we have that $[\![Q, M]\!](b \mid sa) = p\,[\![Q, V]\!](b \mid sa)$, then we must also have that $Q, M \Downarrow^p Q', V$.*

We use the standard proof technique that uses a computability predicate. The usual definition of computability predicate is adapted to quantum stores as follows.

Definition 5.4 (Computability for QSL) Let $\Gamma_1, \Gamma_2 \vdash M : A$, with Γ_1 containing only variable of type qstore. We say M is *computable* if

(i) $\Gamma_1 \vdash M : A$, $A =$ bool, qstore, $\top$ or com and if for all $sab \in \mathcal{T}([\![Q', V]\!])$ we have that $[\![Q, M]\!](b \mid sa) = p\,[\![Q', V]\!](b \mid sa)$, then $Q, M \Downarrow^p Q', V$,

(ii) $\Gamma_1, \overline{x}_1 : A_1, \ldots, \overline{x}_n : A_n \vdash M : A$ is $\Gamma_1 \vdash M[N_1/\overline{x}_1, \ldots, N_n/\overline{x}_n] : A$ is computable for all computable semi-closed terms $\Gamma_1 \vdash N_1 : A_1, \ldots, \Gamma_1 \vdash N_n : A_n$,

(iii) $\Gamma_1 \vdash M : A \multimap B$, M semi-closed and for all semi-closed $\Gamma_1 \vdash N : A$ we have that $\Gamma_1 \vdash MN : B$ is computable,

(iv) $M = \overline{x}$ with $\Gamma_1 \vdash \overline{x}$: qstore and both $\Gamma_1 \vdash \mathsf{meas}\, x_i$: bool and $\Gamma_1 \vdash U\overline{y}$: com with $\overline{y} \sqsubseteq \overline{x}$ are computable.

Proposition 5.3 is a direct consequence of the following lemma.

Lemma 5.5 *All QSL terms are computable.*

Proof. By induction on the construction of M. By the second and third clauses of the definition of computability, we can assume that M is constructed out of semi-closed terms. We explain the most interesting part of the proof, leaving out the cases which are standard classical cases.

For the base case, M must be a constant or a quantum store variable $\overline{x}$. If $M = \overline{x}$ is a quantum store variable, we must apply the last clause of the definition of computability. We need to check that both $\Gamma_1 \vdash \mathsf{meas}\, x_i$: bool and $\Gamma_1 \vdash U\,\overline{y}$: com, $\overline{y} \sqsubseteq \overline{x}$ are computable. In the first case, suppose that $[\![Q, \mathsf{meas}\, x_i]\!]$ makes Player behave as $[\![Q', V]\!]$ for some boolean value V. This means that measuring the qbit i of the quantum store $\overline{x}$ with the quantum register in some state Q gives the boolean result V (without loss of generality, suppose that $V = 0$) with probability p and a quantum register left in state $Q\,[|x_1 \ldots x_n\rangle \mapsto [0]^{x_i}|x_1 \ldots x_n\rangle]$. This implies that $Q, \mathsf{meas}\, x_i \Downarrow^p Q', V$. A similar argument is used to show that $\Gamma_1 \vdash U\,\overline{y}$: com is computable.

For the induction step, we assume that M is constructed out of semi-closed computable terms. In the case of local preparation, consider that $M =$

$\mathsf{prep}\,\overline{y}\,\mathsf{with}\,\overline{x}\,\mathsf{in}\,N$ is a semi-closed term. Assume that $[\![Q, M]\!]$ makes Player behave as if he was using the strategy $[\![Q', V]\!]$, with probability p. Since in the definition of $[\![Q, M]\!]$ Player plays a preparation move before the first question about the state held by $\overline{x} \otimes \overline{y}$ in Q, the answer to this question is given using $[\![Q\left[|x_1 \dots x_n y_1 \dots y_m\rangle \mapsto |x_1 \dots x_n\rangle|0 \dots 0\rangle\right]]\!]$. Thus the strategy $[\![Q', V]\!]$ make player behave as $[\![Q\left[|x_1 \dots x_n y_1 \dots y_m\rangle \mapsto |x_1 \dots x_n\rangle|0 \dots 0\rangle\right], M]\!]$. By induction hypothesis, this implies that $Q\left[|x_1 \dots x_n y_1 \dots y_m\rangle \mapsto |x_1 \dots x_n\rangle|0 \dots 0\rangle\right], M \Downarrow^p Q', V$. Using the operational semantics derivations rules, we get that $Q, M \Downarrow^p Q', V$, which is the desired result. □

To state the soundness result, we need a few extra definitions. A *context* with a hole of type B is a term $C[-]$ with a special free variable "$-$" of type B, i.e. it is possible to derive that $\Gamma, -\colon B \vdash C[-]\colon A$. Capture-free substitution of a term $\Gamma \vdash M\colon B$ in the context $C[-]$ is denoted by $C[M]$. Two semi-closed terms $\Gamma \vdash M_1\colon A$ and $\Gamma \vdash M_2\colon A$ are *contextually equivalent* if for all quantum stores Q, Q' and ground type context $C[-]$ we have $Q, C[M_1] \Downarrow^p Q', V$ if and only if $Q, C[M_2] \Downarrow^p Q', V$. This relation is denoted by $M_1 \sim M_2$.

Proposition 5.6 *Let M_1 and M_2 be two semi-closed terms. If $[\![M_1]\!] = [\![M_1]\!]$, then $M_1 \sim M_2$.*

The proof is a standard argument using consistency and adequacy.

6 Conclusion and future work

Usually game semantics is used to get full-abstraction results by putting appropriate restrictions on the strategies. Here the main goal was instead to introduce a new kind of model for quantum programming languages. While the soundness result we obtained confirms the usefulness of using quantum games to model quantum types, it is a natural next step to seek a full-abstraction result for QSL. The main difficulty is that there is no known characterisation of the probabilistic strategies which can be defined as quantum strategies where the weight assigned to a play is given using the quantum intervention formalism. Gleason's theorem [5] is one result in this spirit: it gives conditions which allow one to know when weights assigned to projection operators P can be described as those computed using a density matrix ρ with the formula $\mathrm{tr}(\rho)$. There is no similar result for quantum interventions. Thus, a full abstraction result here would be a major advance in understanding how to characterize quantum processes. In this case the obstacle has nothing to do with the usual subtleties associated with higher-type languages.

We did not explore fully the categorical properties of the quantum store arenas. For example, one could consider the subcategory of the category of arenas and probabilistic strategies that consist of quantum store arenas and the quantum strategies between them. This category is very different from the other categories which were studied to understand quantum information flow since the qstore arena does not have a fixed dimension while the objects in these other categories are finite dimensional Hilbert space.

References

[1] Bennett, C. H., G. Brassard, C. Crépeau, R. Jozsa, A. Peres and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels*, Physical Review Letters (1993), pp. 1895–1899.

[2] Coecke, B. and D. Pavlovic, *Quantum measurements without sums* (2006).

[3] Danos, V. and R. Harmer, *Probabilistic game semantics*, in: *ACM Transactions On Computational Logic, Special Issue for LICS'00*, Association For Computing Machinery (2002), pp. 359–382.

[4] Gell-Mann, M. and J. Hartle, *Classical equations for quantum systems*, Physical Review D **47** (1993), pp. 3345–3382.

[5] Gleason, A. M., *Measures on the closed subspaces of a hilbert space*, Journal of Mathematics and Mechanics (1957), pp. 885–893.

[6] Griffiths, R., *Consistent histories and quantum reasoning*, Physical Review A **54** (1996), pp. 2759–2774.

[7] Harmer, R., "Games and Full Abstraction For Nondeterministic Languages," Ph.D. thesis, Imperial College (1999).

[8] Jacobs, B., *Semantics of weakening and contraction*, Annals of Pure and Applied Logic (1994), pp. 73–106.

[9] Omnès, R., "The Interpretation of Quantum Mechanics," Princeton Univ. Press, 1994.

[10] Peres, A., *Classical interventions in quantum systems. I. The measuring process*, Physical Review A **61** (2000).

[11] Reynolds, J. C., *The essence of* ALGOL, in: J. W. de Bakker and J. C. van Vliet, editors, *Algorithmic Languages, Proceedings of the International Symposium on Algorithmic Languages* (1981), pp. 345–372.

[12] Selinger, P., *Towards a quantum programming language*, Mathematical Structures in Computer Science **14** (2004), pp. 527–586.

[13] Selinger, P. and B. Valiron, *A lambda calculus for quantum computation with classical control*, Mathematical Structures in Computer Science **16** (2006), pp. 527–552.

[14] Selinger, P. and B. Valiron, *On a fully abstract model for a quantum linear functional language*, in: *Proceedings of the 4th International Workshop on Quantum i Programming Languages, Oxford, July 17-19*, 2006.

[15] Valiron, B., "A functional programming language for quantum computation with classical control," Master's thesis, Departement of Mathematics, University of Ottawa (2004).