Abstract Semantics for a Simple Quantum Programming Language

Martin Churchill, Keble College, Oxford
martin.churchill@keble.oxon.org

Dissertation for the Degree of Master of Science in
Mathematics and the Foundations of Computer Science
at the University of Oxford

Supervisor : Samson Abramsky

November 11, 2007

Abstract

In [3], a simple quantum programming language is presented, together with both denotational and operational semantics. This language follows [24] in that it uses classical control with quantum data. Correspondence results between these semantics are stated; but not proved. We shall firstly prove these results. We shall also expand upon the dynamic logic approach given in [3] and give a full example to show how a formal verification of the Deutch-Josza algorithm can be performed in our framework.

We shall then reformulate the semantics of the language with respect to the abstract categorical quantum axiomatics in [6] and again prove the correspondence results with respect to these abstract semantics. This shall be novel in the following respects: Firstly, it shall be a language explicitly based on the recast axioms of quantum mechanics in [6]. Secondly, it shall provide operational semantics directly from the abstract category, something that is rarely seen. We shall then investigate the sense in which the original concrete semantics is a special case of the abstract semantics. The abstract semantics will use Peter Selinger's CPM construction [23] since they will need to deal with probability distributions over states. Finally, we shall then investigate how it may be possible to develop the logical semantics at our abstract level (using subobjects to represent predicates.)

Contents

1 Introduction

Since its formulation in 1931, Quantum Mechanics has been axiomatised in terms of Hilbert Spaces [11] (intuitively, a fundamental quantum unit (a qubit) can be represented by a pair of complex numbers up to non-zero multiple — a ray in the space $\mathbb{C}^2$.) Combination of systems results in the possibility of quantum entanglement, and the tensor product $\otimes$ is used to model this. Fundamentally, this is not simply the Cartesian product of the two spaces but rather something close to the *linear function space* from one to the other, with $A \otimes B \simeq A \multimap B$. In recent years, it has been noted that it is this correspondence that provides the fundamental quantum behaviour, and a successful attempt has been made to strip this from the complex structure of a Hilbert space (which does, after all, depend on complex numbers, vector spaces, inner products; the important features of which can be stripped down to a small number of categorical axioms.) Thus in [6] Abramsky and Coecke recast Quantum Mechanics into a new light — no longer do we need a rich Hilbert Space, but something weaker: a certain type of category that happens to support the quantum features we require (and in particular, the category of sets and relations also admit these features.) Within this new framework with a small set of axioms, many derived concepts from Quantum Mechanics (and general Linear Algebra) are definable (to quite a deep level — for example, completely positive maps,) and it is possible, for example, to define and prove the teleportation protocol at this level of abstraction [6]. Furthermore, this new formulation directly gives rise to a graphical calculus, i.e. a typed high-level way of reasoning about quantum computation; something that was very much missing (and missed) in Quantum Mechanics of the 20th century. This graphical calculus makes reasoning about such structures far easier and intuitive, and reasoning in this language corresponds directly to proofs about the category (and indeed any particular model of this category, e.g. **FdHilb**. This correspondence has indeed been formalised [16,23].) The fundamental two-dimensionality comes from the orthogonal notions of composition (time) and tensor product (space). In addition, from this category-theoretic perspective, connections with logic are clearer (due to the category-logic correspondence cf. the Curry-Howard isomorphism) and so quantum logic has indeed been reborn with respect to these new semantics.

The graphical language mentioned above attempts to address the problem of a lack of high-level quantum formulation. This has been attempted before, in the vein of programming languages for quantum systems — it's much easier to design algorithms in C, than in the machine code itself. The most successful endeavour in this area has been in Peter Selinger's paper [24], in which a quantum language is presented, along with some (denotational) semantics for that language in terms of superoperators and Hilbert spaces. For a review of the status of quantum programming languages, see [22]. Another example of such a language that is much simpler than that presented in [24] but that admits a corresponding operational and denotational semantics is in Samson Abramsky's talk *A Cook's tour of a simple quantum programming language* [3]. In this dissertation we shall aim to unify the concrete exposition of the quantum programming language in this talk, together with the abstract axiomatics recently proposed by the same author and Bob Coecke, as mentioned above. We shall firstly present the concrete version of the language — as presented in the talk — and furthermore *prove* the stated operational/denotational semantic correspondence. After this, we shall generalise the semantics of the language to the abstract categorical level and again prove the same correspondence. We shall also look at some logical semantics for the language (in terms of a program logic) as is touched on in [3], and also consider bringing this to the abstract level.

In slightly more detail, [3] presents a simple `while` language for a classical computer with access to quantum systems and quantum operations. This language consists of imperative constructs — sequencing, iteration and conditioning — together with commands for updating the classical state space, and applying quantum operations to the qubits (including measurement, leaving the result in a classical register.) Meaning for this language is given in terms of *denotational semantics* (the meaning of a command is given in terms of its subcommands) and *operational semantics* (a virtual machine is built, in the form of a reduction relation.) Due to the inherent probabilities involved in measuring quantum bits, these meanings are in terms of probability distributions. We give these meanings in detail (building up, following [3], from classical to probabilistic to quantum) and show that the resulting function one ends up with from the two different meanings do indeed coincide. We also briefly mention logical semantics — with a modal operator for programs, given meaning by our denotational semantics. This logic can only reason about the classical part of the system; but can hence indirectly reason about the quantum part via the use of measurements.

As a gentle introduction to ideas presented later in the dissertation, we then provide categorical semantics (both operational and denotational) of a purely classical language using ideas of distributive categories and cpo-enrichment; and once again prove the correspondence this time using categorical methods, with commuting diagrams etc. These semantics take place in a distributive O-category. Our goal is then to adapt the meaning of the entire quantum language to the level of the abstract categorical semantics. This involves firstly introducing notions of abstract quantum mechanics (strongly compact closed categories); and then, for notions of probability distributions, mixed states and its abstract counterpart via Peter Selinger's CPM construction [23]. Finally we use biproduct completion to represent classical control. Using these ideas we explicitly give our semantics at the abstract level, and prove a correspondence theorem. We then check that the concrete version is indeed a special case of our abstract investigation.

After this, we look at how we might complete the abstractisation of [3] by looking at an abstract version of the logical semantics. This involves exposing the categorical notion of a *subobject* and the partial order it generates. In particular, we represent the Boolean logical operations by order constructs on this partial order (e.g. conjunction corresponds to least upper bounds) as it does in e.g. powerset domains. We represent the meaning of the modal operator using pullbacks (together with our abstract denotational semantics.) Once again we check then that the key categories satisfy these requirements.

We make a brief comment on originality. As mentioned, section 2 is based on the talk [3] but expands many of the comments and gives further discussion. Its main contributions are to firstly prove the correspondence results that are stated in the talk, and also to provide a worked example of the ideas introduced here (this is done in section 2.5.)

Section 3 is largely material working towards the presentation in section 4. The denotational semantics of the classical language in a distributive O-category is not new, but the author has not seen operational semantics provided in this manner nor a correspondence theorem. Section 3.3 is then aiming towards our specific goals (and in particular the idea of the canonical Cartesian subcategory of the SCCCB is formulated by the author.) Sections marked "preliminaries" are recapping on known material that we require for our work.

The main original achievement of this dissertation comes in section 4. Here we use the quantum categorical semantics in order to give meaning to the quantum programming language. This is a heavy application of Peter Selinger's **CPM** construction. The abstraction of ideas of trace-decreasing maps in the **CPM** construction in section 4.6 is new, but ideas of the trace at the abstract level are well developed.

Section 5 is a brief investigation of how we might raise the logical semantics to the abstract level, and is applying well developed subject areas to our specific project.

On reflection, this dissertation provided quite a nice journey through various areas of study from my MFoCS course (and before.) The concrete exposition combines Quantum Computer Science with Domain Theory, using notions of quantum computation to represent the quantum aspects of our language and using domain theory to represent the recursion aspect present in `while` loops. Looking at a program logic for this language introduced notions apparent in the course *The Logic of Multi-Agent Information Flow.* Of course, proceeding to the abstract level is based upon ideas of categories and owes much to the course *Categories, Proofs and Programs.* When we briefly mention a monad of quantum computation (as given in the talk [3]) this relates our work to the more general interpreters found in the second year Computer Science course *Programming Languages*, lectured by Mike Spivey in 2005. Finally, notions of the subobject lattices in the abstract logical semantics and representing the algebraic Boolean notions using order structure relates to Hilary Priestley's lecture course *Algebraic and Relational Semantics for Non-Classical Propositional Logics.*

2 A Cook's Tour of a Simple Quantum Programming Language

2.1 Preliminaries : Quantum Computation

In order to provide an exposition of a quantum programming language we firstly, of course, need to expose the features of quantum mechanics that we wish to use. A *quantum bit* or *qubit* represents a particle in memory with some quantum quantity such as spin that we use to represent the state of that qubit. Each possible state of the qubit corresponds to a point on the surface of a sphere. Such points are determined precisely by pairs of complex numbers, up to a non-zero complex multiple. That is, the quantum state is described by a ray in the Hilbert space $\mathbb{C}^2$. It will be useful to generalise this as will soon be seen (e.g. to multiple qubits) and so

Definition 2.1.1 *The state space of a quantum system is represented by a finite-dimensional Hilbert space H. A state within this state space corresponds to a ray (a one dimensional subspace) of H, typically represented by a vector of unit norm.*

For the qubit case, then, we write $|0\rangle$ and $|1\rangle$ for the basis vectors of $\mathbb{C}^2$ and as such a qubit can be represented by $\alpha_0|0\rangle + \alpha_1|1\rangle$ for some $\{\alpha_0, \alpha_1\} \in \mathbb{C} \times \mathbb{C}$ identifying such states that differ only by a global complex multiple.

Clearly in our quantum computer we will wish to deal with more than one qubit. Initially one might expect the state of two qubits to be the Cartesian product of the states of the two qubits, i.e. the direct sum $\oplus$ of the Hilbert spaces. However, quantum theory dictates that this indeed is not the case. Different qubits can be *entangled*, amounting to the bizarre fact that the state space of two systems is given by the *tensor product* of the two subsystems. It is this fact which gives much of quantum computing its power, since e.g. with 8 qubits the state space has dimension 2^8 as opposed to merely 8 for classical computing (since $dim(A \otimes B) = dim(A).dim(B)$ while $dim(A \oplus B) = dim(A) + dim(B)$.)

Definition 2.1.2 *If we have quantum systems of state spaces H_1 and H_2 respectively, then the combined system is represented by the state space $H_1 \otimes H_2$ where $\otimes$ represents the linear-algebraic tensor product.*

Unfortunately all is not as perfect as it seems however: for classical bits, we can read, write and modify arbitrarily. This is very far from the truth in the quantum world. The only operations we can perform on quantum systems are *unitary* ones (adjoint = inverse). In the qubit case, this corresponds precisely to rotations. In particular all of these operations are invertible.

Definition 2.1.3 *The basic data transformations on quantum systems (= Hilbert spaces) are unitary transformations.*

In addition, we cannot read qubits arbitrarily. We must be able to read them in some way, however, as otherwise we would not be able to exploit their power. This is done via *quantum measurements.* Given some qubit q in state $\alpha_0|0\rangle + \alpha_1|1\rangle$ we can choose to ask the question of whether the state of q is $|0\rangle$ or $|1\rangle$ (of course, it may be in neither.) We will get answer to this (by measuring the qubit) that will read either a $|0\rangle$ or $|1\rangle$, and which it is will depend *probabilistically* on whether the state of q is "closer" to being in state $|0\rangle$ or $|1\rangle$. This closeness measure is performed using the inner product, which in the case of qubits on the surface of a sphere gives the intuitive result. Furthermore, if the result of the measurement was that the state is closer to $|0\rangle$ than $|1\rangle$, the state of q *will become* $|0\rangle$, and vice versa. That is, *the act of measuring the qubit destroys the state of the qubit.* This very restricted form of measurement however can be used in cunning ways to exploit quantum weirdness to yet achieve factoring algorithms, search algorithms and secure key distribution. Of course, we can generalise the above to arbitrary bases and Hilbert spaces.

Definition 2.1.4 *Given an orthonormal basis B of H, we can* measure *the state of a system with respect to that basis. Given each vector in the basis b_i the state of the system becomes b_i with probability $\langle b_i|q\rangle$ (making use here of the inner product,) and the result is returned as to which i has actually occurred.*

So, as an example, if one were to measure the same qubit twice one would get the same result; although beforehand it is impossible to predict which result this is. There is also the concept of a *non-degenerate measurement* that is not discussed here.

It is this combination of features then — quantum states, applying unitaries and measurements — that allow us to do interesting things with quantum systems. Examples include teleporting quantum bits over arbitrary distances via. entanglement, coding two bits in terms of one qubit, factorising numbers in less-than-exponential time, performing blind search in square-root time and exposing a (provably) secure key distribution system for use e.g. in one-time pads. See many standard texts or e.g. [11] for a full exposition of this.

In the remainder of this section, we describe a simple quantum programming language and give both operational and denotational semantics for it. We then prove a correspondence result between these semantics. This is done using an incremental method: firstly just pure classical computations; then adding in probabilities and nondeterminism (which we need to model due to measurements); and finally adding in the quantum components of the language. This is based on the talk *A Cook's tour of a simple quantum programming language* by Samson Abramsky [3], expanding on many remarks, proving the correspondence theorems (these proofs are absent from the talk [3]) and providing an example.

2.2 Classical Language

As a building block, we firstly expose the classical part of our language and give some semantics for it.

2.2.1 Syntax

We assume a syntax of arithmetic expressions aexp, Boolean expressions bexp and program variables var. We then define the syntax of commands

$$C ::= \mathsf{var} := \mathsf{aexp} \mid \mathtt{skip} \mid C_1;C_2 \mid \mathtt{if}\ \mathtt{bexp}\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2 \mid \mathtt{while}\ \mathtt{bexp}\ \mathtt{do}\ C$$

These are to have the standard C-like meanings, with one simplification: variables are thought here to be synonymous for locations in memory (addresses) not the usual abstracted level above them (this is fine though, since we don't manipulate references.) Thus, in our semantics below, the state of the system will be a mapping from variables to possible values.

Here aexp and bexp are generic arithmetic and Boolean expressions — a typical example of a bexp may be variable = value but we do not go into this much detail here. Instead, we assume a syntactic form of these expressions and assume that some meaning (of the appropriate type, to be explained) will be given for them.

Finally, we note that if can be simulated with the while construct and an ancillary variable, but we include it here as it will make both our language and our proofs more structured.

2.2.2 Operational Semantics

We now give an operational meaning to our language. We define a *configuration* to be a pair (C, s) where C is a command and s is a state, i.e. a mapping from variables to values (we assume countably many variables and values.) We write S for the set of states (thus S is countable.) A *terminal* state is a state of the form $(\mathtt{skip}, s)$. Note that we shall sometimes assume that the state set is finite, i.e. that the machine has a finite amount of memory (this is not an unrealistic assumption, cf. actual computers.) Though making less of a difference at this stage, this follows [24] in being required for the abstract quantum semantics.

Given state $s : \mathsf{var} \to V$, variable v and value $x \in V$ we define $s[v \mapsto x]$ as the map that sends v to x and u to $s(u)$ for all other variables u. For each arithmetic expression a we assume a primitive denotation $[\![a]\!] : S \to V$. For each Boolean expression b we assume a primitive denotation $[\![b]\!] : S \to \{\mathsf{tt}, \mathsf{ff}\}$.

We define a one-step relation → between configurations by structural induction on the command component of the source of the arrow. This is made explicit in the following set of rules:

$$(v := e,\, s) \to (\mathtt{skip},\, s[v \mapsto [\![e]\!](s)])$$

$$(\mathtt{skip};C,\, s) \to (C,\, s)$$

$$\frac{(C_1,\, s) \to (C_1',\, s')}{(C_1;\, C_2,\, s) \to (C_1';\, C_2,\, s')}$$

$$\frac{[\![b]\!](s) = \mathsf{tt}}{(\mathtt{if}\ b\ \mathtt{then}\ C_1\ \mathtt{else}\ C_2,\, s) \to (C_1,\, s)}$$

$$\frac{[\![b]\!](s) = \mathsf{ff}}{(\texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2,\ s) \rightarrow (C_2,\ s)}$$

$$\frac{[\![b]\!](s) = \mathsf{tt}}{(\texttt{while}\ b\ \texttt{do}\ C,\ s) \rightarrow (C;\ \texttt{while}\ b\ \texttt{do}\ C,\ s)}$$

$$\frac{[\![b]\!](s) = \mathsf{ff}}{(\texttt{while}\ b\ \texttt{do}\ C,\ s) \rightarrow (\texttt{skip},\ s)}$$

Note that the above semantics are completely deterministic — given any configuration $(C,\ s)$ there is at most one configuration $(C',\ s')$ such that $(C,\ s) \rightarrow (C',\ s')$. That is, the transition relation is in fact a partial function on configurations, only undefined on $(C,\ s)$ when $C = \texttt{skip}$. We define the relation $\vec{\rightarrow}$ to be the reflexive transitive closure of $\rightarrow$. Then given any (C, s) either $(C, s)\vec{\rightarrow}(\texttt{skip}, s')$ for a unique s' or (C, s) does not reduce to a `skip` command (in which case we say (C, s) *loops*. This is caused inherently by the `while` construct, for example the program `while true do skip` will loop in this manner.) We can then define the partial function $\mathcal{O}[\![C]\!] : S \rightharpoonup S$ for any command C by

$$\mathcal{O}[\![C]\!](s) = \begin{cases} s' & \text{if } (C,\ s)\ \vec{\rightarrow}\ (\texttt{skip},\ s') \\ \mathsf{undefined} & \text{if } (C, s) \text{ loops} \end{cases}$$

Before we proceed we make the following note/lemma:

Proposition 2.2.1 *If* $(C, s)\vec{\rightarrow}(C', s')$ *then* $\mathcal{O}[\![C]\!](s) = \mathcal{O}[\![C']\!](s')$.

Proof This is clear since if (C', s') loops then so does (C, s) since reduction is deterministic; and if $(C', s')\vec{\rightarrow}(\texttt{skip}, s'')$ then $(C, s)\vec{\rightarrow}(\texttt{skip}, s'')$ by transitivity.

□

2.2.3 Denotational Semantics

We shall now, given any command C, define its denotational meaning $\mathcal{D}[\![C]\!] : S \rightharpoonup S$ compositionally. We do this by induction on C.

$$\begin{array}{lcl}
\mathcal{D}[\![\texttt{skip}]\!] & = & id_s \\
\mathcal{D}[\![v := e]\!](s) & = & s[v \mapsto [\![e]\!](s)] \\
\mathcal{D}[\![C_1; C_2]\!] & = & \mathcal{D}[\![C_2]\!] \circ \mathcal{D}[\![C_1]\!] \\
\mathcal{D}[\![\texttt{if}\ b\ \texttt{then}\ C_1\ \text{else}\ C_2]\!](s) & = & \mathsf{if}\ [\![b]\!](s) = \mathsf{tt}\ \mathsf{then}\ \mathcal{D}[\![C_1]\!](s)\ \mathsf{else}\ \mathcal{D}[\![C_2]\!](s) \\
\mathcal{D}[\![\texttt{while}\ b\ \texttt{do}\ C]\!] & = & \mathsf{lfp}[\lambda f : S \rightharpoonup S.\lambda s : S.\ \mathsf{if}\ [\![b]\!](s) = \mathsf{tt}\ \mathsf{then}\ (f \circ \mathcal{D}[\![C]\!])(s)\ \mathsf{else}\ s]
\end{array}$$

Here we are using the fact that the set of partial functions $[S \rightharpoonup S]$ is a complete partial order (with $f \sqsubseteq g$ if $f \subseteq g$ as partial function (graphs)) and that the function given inside the brackets in the while case (an endofunction on $[S \rightharpoonup S]$) is continuous, and hence has a least fixed point (with respect to the ordering on partial functions.) The function in question is continuous since it is a standard combination of continuous functions. For details of this (and the general fixpoint construction, details of which we will use here) see the *Domain Theory* lecture notes [4].

2.2.4 Correspondence Theorem

We now seek to show that the above semantics coincide.

Theorem 2.2.2 *For any command C we have $\mathcal{D}[\![C]\!] = \mathcal{O}[\![C]\!]$*

Proof We show this by induction on C.

In the case that $C = \texttt{skip}$ we note that $\mathcal{D}[\![C]\!]$ is the identity function and so we need to show that $\mathcal{O}[\![C]\!](s) = s$. That is, we need to show that $(C,\ s) \vec{\rightarrow} (\texttt{skip},\ s)$. Well we know that this is the case since $(C,\ s) = (\texttt{skip},\ s)$ and $\vec{\rightarrow}$ is reflexive.

In the case that $C = C_1; C_2$ we need to show that $\mathcal{O}[\![C]\!] = \mathcal{O}[\![C_2]\!] \circ \mathcal{O}[\![C_1]\!]$ since by inductive hypothesis the right hand side is $\mathcal{D}[\![C_2]\!] \circ \mathcal{D}[\![C_1]\!] = \mathcal{D}[\![C]\!]$. We show this by extensionality, i.e. that for any s, $\mathcal{O}[\![C]\!](s) = \mathcal{O}[\![C_2]\!](\mathcal{O}[\![C_1]\!](s)))$

Firstly suppose that $\mathcal{O}[\![C_1]\!](s) = s'$. Then we know that $(C_1,\ s) \vec{\rightarrow} (\texttt{skip},\ s')$. That is $(C_1,\ s) \rightarrow (C_1',\ s_1) \rightarrow \ldots \rightarrow (\texttt{skip},\ s')$. From the operational semantics for ; in the non-skip case it follows that $(C_1; C_2,\ s) \rightarrow (C_1'; C_2,\ s_1) \rightarrow \ldots \rightarrow (\texttt{skip};C_2,\ s')$ — we can show this formally by induction if we wish. Finally $(\texttt{skip};C_2,\ s') \rightarrow (C_2,\ s')$ by the skip;- operational semantic. We conclude from this that $(C,\ s)$ $\vec{\rightarrow}$ $(C_2,\ s')$. There are then two possibilities: $\mathcal{O}[\![C_2]\!](s') = s''$ or $\mathcal{O}[\![C_2]\!](s')$ is undefined. In the former case, we have $(C_2,\ s') \vec{\rightarrow} (\texttt{skip},\ s'')$ and so by transitivity of $\vec{\rightarrow}$ it follows that $(C,\ s) \vec{\rightarrow} (\texttt{skip},\ s'')$ and so $\mathcal{O}[\![C]\!](s) = s'' = \mathcal{O}[\![C_2]\!](s') = (\mathcal{O}[\![C_2]\!] \circ \mathcal{O}[\![C_1]\!])(s)$ as required. In the latter case, there is no such s'' such that $(C_2,\ s') \vec{\rightarrow} (\texttt{skip},\ s'')$ and as such by the operational semantics there is no s'' such that $(\texttt{skip};C_2,\ s')$ $\vec{\rightarrow}$ $(\texttt{skip},\ s'')$. Since $(C,\ s) \vec{\rightarrow} (\texttt{skip};C_2,\ s')$ and the relation $\rightarrow$ is deterministic it follows that there is no s'' such that $(C,\ s) \vec{\rightarrow} (\texttt{skip},\ s'')$ and as such $\mathcal{O}[\![C]\!](s)$ is undefined. This is equal then to $(\mathcal{O}[\![C_2]\!] \circ \mathcal{O}[\![C_1]\!])(s)$ as $\mathcal{O}[\![C_2]\!](s)$ is undefined.

Secondly if $\mathcal{O}[\![C_1]\!](s)$ is undefined then there is no s' such that $(C_1,\ s) \vec{\rightarrow} (\texttt{skip},\ s')$. As such $(C = C_1; C_2,$ $s)$ can never reduce to $(\texttt{skip};C_2,\ s')$ and so $(C,\ s)$ can never reduce to $(C_2,\ s')$ and so certainly never to $(\texttt{skip},\ s'')$. Another way of putting this is that any terminating reduction path from $(C_1; C_2, s)$ to $(\texttt{skip},\ s')$ must contain a terminating reduction path from (C_1, s) which we have assumed not to exist. Either way, we find that $\mathcal{O}[\![C]\!](s)$ is undefined and hence equivalent to $(\mathcal{O}[\![C_2]\!] \circ \mathcal{O}[\![C_1]\!])(s)$

In the case that $C = v := e$ we need to show that $\mathcal{O}[\![C]\!](s) = \mathcal{D}[\![C]\!](s) = s[v \mapsto [\![e]\!](s)]$ i.e. that $(C,\ s) \vec{\rightarrow}$ $(\texttt{skip},\ s[v \mapsto [\![e]\!](s)])$. But this is clear since $(C,\ s) \rightarrow (\texttt{skip},\ s[v \mapsto [\![e]\!](s)])$ and $\rightarrow \subseteq \vec{\rightarrow}$.

In the case that $C = \texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2$ we once again need to show that for all s, $\mathcal{O}[\![C]\!](s) = \mathcal{D}[\![C]\!](s)$. If $s = \texttt{tt}$ then the RHS is $\mathcal{D}[\![C_1]\!](s)$ which is $\mathcal{O}[\![C_1]\!](s)$ by inductive hypothesis. Finally we note that this is $\mathcal{O}[\![C]\!](s)$ by our above lemma 2.2.1, since $(C, s) \rightarrow (C_1, s)$. We can show the result in the case $s = \mathsf{ff}$ in a similar manner.

In the case that $C = \texttt{while}\ b\ \texttt{do}\ C_1$ we need to show that $\mathcal{O}[\![C]\!] = \mathcal{D}[\![C]\!] = \mathsf{lfp}(h)$ for our given function h. To do this we shall firstly show that $\mathcal{O}[\![C]\!]$ is *a* fixpoint of this function, i.e. that $\mathcal{O}[\![C]\!] \sqsupseteq \mathcal{D}[\![C]\!]$. We shall then show that $\mathcal{D}[\![C]\!] \sqsupseteq \mathcal{O}[\![C]\!]$ by a direct induction argument. By information extensionallity (antisymmetry of our order relation) this tells us that $\mathcal{D}\llbracket C\rrbracket = \mathcal{O}\llbracket C\rrbracket$ as required. This argument is very much of the flavour of [4].

To show the first inequality, we note that if $\llbracket b\rrbracket(s) = \texttt{tt}$ then $\mathcal{O}\llbracket C\rrbracket(s) = \mathcal{O}\llbracket C_1; C\rrbracket(s)$ by noting that $(C,\ s) \rightarrow (C_1; C,\ s)$ and using the above note 2.2.1. We know that $\mathcal{O}\llbracket C_1; C\rrbracket(s) = (\mathcal{O}\llbracket C\rrbracket \circ \mathcal{O}\llbracket C_1\rrbracket)(s)$ by correspondence for composition above, and then by inductive hypothesis it follows that $\mathcal{O}\llbracket C\rrbracket(s) = (\mathcal{O}\llbracket C\rrbracket \circ \mathcal{D}\llbracket C_1\rrbracket)(s)$. If $\llbracket b\rrbracket(s) = \mathsf{ff}$ then $\mathcal{O}\llbracket C\rrbracket(s) = \mathcal{O}\llbracket \texttt{skip}\rrbracket(s) = s$ by using the above note again. Hence we find that $\mathcal{O}\llbracket C\rrbracket(s) = \mathsf{if}\ \llbracket b\rrbracket(s) = \texttt{tt}\ \mathsf{then}\ (\mathcal{O}\llbracket C\rrbracket \circ \mathcal{D}\llbracket C_1\rrbracket)(s)\ \mathsf{else}\ s$ i.e. that $\mathcal{O}\llbracket C\rrbracket$ is indeed a fixpoint of h where $h = \lambda f : S \rightharpoonup S.\lambda s : S.\ \mathsf{if}\ \llbracket b\rrbracket(s) = \texttt{tt}\ \mathsf{then}\ (f \circ \mathcal{D}\llbracket C_1\rrbracket)(s)\ \mathsf{else}\ s$. It follows that since $\mathcal{D}\llbracket C\rrbracket$ is the least such fixpoint we have $\mathcal{O}\llbracket C\rrbracket \sqsupseteq \mathcal{D}\llbracket C\rrbracket$ in the ordering on partial functions.

To show the second inequality, $\mathcal{O}\llbracket C\rrbracket \sqsubseteq \mathcal{D}\llbracket C\rrbracket$ we need to show that if $\mathcal{O}\llbracket C\rrbracket(s) = s'$ then $\mathcal{D}\llbracket C\rrbracket(s) = s'$ by making the ordering on functions explicit. Well suppose $\mathcal{O}\llbracket C\rrbracket(s) = s'$. Then by definition we have $(C,\ s) \overset{\rightarrow}{\rightarrow} (\texttt{skip},\ s')$. Note that the shape of this reduction will be

$$(C,\ s = s_0) \rightarrow (C_1; C,\ s_0) \overset{\rightarrow}{\rightarrow}(\texttt{skip};C,\ s_1) \rightarrow (C,\ s_1) \overset{\rightarrow}{\rightarrow} (C,\ s_2) \overset{\rightarrow}{\rightarrow} (C,\ s_n) \rightarrow (\texttt{skip},\ s_n = s')$$

for some $n \geq 0$ where $\llbracket b\rrbracket(s_i) = \texttt{tt}$ for $i < n$ and $\llbracket b\rrbracket(s_n) = \mathsf{ff}$. Note we also have that $s_{i+1} = \mathcal{O}\llbracket C_1\rrbracket(s_i)$ which is $\mathcal{D}\llbracket C'\rrbracket(s_i)$ by inductive hypothesis.

We claim that for any i, $h^{i+1}\bot s_{n-i} = s_n$. We show this by induction on i. For $i = 0$ we have LHS $= h\bot s_n = s_n$ which holds since $h\bot s_n =$ if $\llbracket b\rrbracket(s_n) = \texttt{tt}$ then $(\bot \circ \mathcal{D}\llbracket C_1\rrbracket)(s_n)$ else s_n. Since $\llbracket b\rrbracket(s_n) = \mathsf{ff}$ it follows that $h\bot s_n = s_n$ as required.

If $i = k+1$ we have $h^{i+1}\bot s_{n-i} = h^{k+2}\bot s_{n-k-1} = h(h^{k+1}\bot)s_{n-k-1}$. Since $k \geq 0$ we have $n-k-1 \leq n-1$ and so $\llbracket b\rrbracket(s_{n-k-1}) = \texttt{tt}$. As such $h(h^{k+1}\bot)s_{n-k-1} = ((h^{k+1}\bot) \circ \mathcal{D}\llbracket C_1\rrbracket)s_{n-k-1} = ((h^{k+1}\bot) \circ \mathcal{O}\llbracket C_1\rrbracket)s_{n-k-1}$ by outer inductive hypothesis on commands. This is $h^{k+1}\bot(\mathcal{O}\llbracket C_1\rrbracket(s_{n-k-1}))$ — however $\mathcal{O}\llbracket C_1\rrbracket(s_{n-k-1}) = s_{n-k}$ by looking at our reduction above. It follows then that $h^{i+1}\bot s_{n-i} = h^{k+1}\bot s_{n-k} = h^i\bot s_{n-(i-1)}$ which is s_n by our current inductive hypothesis, and so our claim is proved.

Finally, as a special case of the above with $i = n$ we have $h^{n+1}\bot s_0 = s_n$ and since $\mathcal{D}\llbracket C\rrbracket = \mathsf{lfp}(h) \sqsupseteq h^{n+1}\bot$ it follows that $\mathcal{D}\llbracket C\rrbracket(s_0) = s_n$. But $s_0 = s$ and $s_n = s'$ and so we conclude that $\mathcal{D}\llbracket C\rrbracket(s) = s'$ as required. And so in general $\mathcal{O}\llbracket C\rrbracket(s) = s'$ implies $\mathcal{D}\llbracket C\rrbracket(s) = s'$ and so $\mathcal{D}\llbracket C\rrbracket \sqsupseteq \mathcal{O}\llbracket C\rrbracket$ in the ordering of partial functions. Together with our previous result it follows that $\mathcal{D}\llbracket C\rrbracket = \mathcal{O}\llbracket C\rrbracket$ in the while case, as required.

□

2.2.5 Logical Semantics

We shall now define a program logic for our language. We can firstly define a set of logical formulas

$$\phi ::= \top \mid \phi \wedge \phi \mid \neg\phi \mid \langle C\rangle\phi$$

Intuitively, here we have propositional connectives together with a modal operator $\langle C\rangle$ (for commands C) where $\langle C\rangle\phi$ is to represent *after performing C (on the state,) ϕ holds.* Formally we define for any ϕ the semantics $\llbracket\phi\rrbracket \subseteq S$ by

$$\begin{array}{lcl}
[\![\top]\!] & = & S \\
[\![\phi \wedge \psi]\!] & = & [\![\phi]\!] \cap [\![\psi]\!] \\
[\![\neg\phi]\!] & = & S - [\![\phi]\!] \\
[\![\langle C\rangle\phi]\!] & = & \{s \in S | \mathcal{D}[\![C]\!](s) \in [\![\phi]\!]\}
\end{array}$$

We note that (following the original talk [3]) there are no atomic formulas in the above apart $\top$ — this is common in transition-based languages. However, in state-based systems such as this it makes sense to add state-dependent atomic formulae, and as such we choose to include them here. We feel that the obvious nature of an atomic formula specific to this state set would be formulae of the kind $\mathsf{variable} = \mathsf{value}$ and the semantics of this statement would be precisely the set of states s such that $s(\mathsf{variable}) = \mathsf{value}$.

$$\begin{array}{l}
\phi :== \ldots \mid v = x \\
[\![v = x]\!] = \{s \in S | s(v) = x\}
\end{array}$$

We can then define *weakest precondition semantics* $\mathsf{WP}[\![C]\!] : \mathcal{P}(S) \to \mathcal{P}(S)$ by $\mathsf{WP}[\![C]\!](U) = \langle C\rangle U$. Thus if $U \subseteq S$ then $\mathsf{WP}[\![C]\!](U)$ gives the weakest precondition (as a set of states) that we need to guarantee that after applying C to the current state, we end up in U. Note here we are identifying syntax and semantics, viewing subsets and formulae interchangeable but only at a notational level in the obvious manner.

Finally we note that we may also, for example, wish to have atomic formulae such as $\mathsf{variable}_1 = \mathsf{variable}_2$ but note that (under assumption of a finite value space) this is in fact (very crudely) derivable. Similarly we may wish to allow ourselves a greater number of arithmetic expressions than in our computable language, in fact we can use anything available to us in the *meta*language (unless of course we wish to provide some implementation of a verifier.) For now, however, we stick to our single atomic formula type as above.

This can be shown to be equivalent to our other forms of semantics — in fact this is very clear. Formulating this with the above intuition becomes

Theorem 2.2.3 *$\mathcal{D}[\![C]\!](s) = s'$ iff $s \in \mathsf{WP}[\![C]\!](\{s'\})$*

Proof $s \in \mathsf{WP}[\![C]\!](\{s'\})$ iff $s \in \langle C\rangle\{s'\}$ iff $s \in \{s \in S|\mathcal{D}[\![C]\!](s) \in \{s'\}\}$ iff $\mathcal{D}[\![C]\!](s) \in \{s'\}$ iff $\mathcal{D}[\![C]\!](s) = s'$

□

Note that weakest precondition semantics are equivalent to working with Hoare triples. A Hoare triple consists of a precondition formula, a command and a postcondition formula ($\phi C\psi$). We say then that $\phi C\psi$ holds if whenever C is applied to a state s satisfying ϕ the result satisfies ψ, i.e. $s \in [\![\phi]\!] \Rightarrow \mathcal{D}[\![C]\!](s) \in [\![\psi]\!]$. Now $WP[\![C]\!]$ can be seen as a mapping on formulae, sending ϕ to the weakest formula ψ such that $\psi C\phi$. That is, $\psi C\phi$ holds iff $[\![\psi]\!] \subseteq WP[\![C]\!]([\![\phi]\!])$ (ψ is a precondition if it is implies the weakest precondition.) This notation transforms the above result into

Theorem 2.2.4 *$\mathcal{D}[\![C]\!](s) = s'$ iff $\{s\}\, C\, \{s'\}$*

when once again we view $\{s\}$ as a logical formula that picks out the state in question.

Moreover, the set of formulas true for a command characterise it up to equivalence w.r.t the other forms of the semantics. That is

Theorem 2.2.5 *$\mathcal{D}[\![C]\!] = \mathcal{D}[\![C']\!]$ iff for any ϕ, $[\![\langle C\rangle\phi]\!] = [\![\langle C'\rangle\phi]\!]$*

Proof The left-to-right direction is obvious. Showing right-to-left amounts to showing that the formulas we have available to us are expressive enough. We do this, again using the fact that we have a finite state space.

If we have formulas of form $\mathsf{value} = \mathsf{variable}$ then for any state s we can define a formula ψ_s such that $[\![\psi_s]\!] = \{s\}$ (assuming we have a finite state space — see above) simply using conjunction. Given a state s we simply form the formula consisting of the conjunction of all formulae of the form $v = s(v)$ for each of the finite number of variables. Then the result easily follows: Given any s we need to show that $\mathcal{D}[\![C]\!](s) = \mathcal{D}[\![C']\!](s)$. Applying hypothesis to ψ_s we get $[\![\langle C\rangle\psi_s]\!] = [\![\langle C'\rangle\psi_s]\!]$. For any C, $[\![\langle C\rangle\psi_s]\!] = \{s' : \mathcal{D}[\![C]\!](s') \in \{s\}\} = \{s' : \mathcal{D}[\![C]\!](s') = s\}$. If for any s we have $\{s' : \mathcal{D}[\![C]\!](s') = s\} = \{s' : \mathcal{D}[\![C']\!](s') = s\}$ it follows that for any s, $s \in \{s' : \mathcal{D}[\![C]\!](s') = \mathcal{D}[\![C]\!](s)\} = \{s' : \mathcal{D}[\![C']\!](s') = \mathcal{D}[\![C]\!](s)\}$ implying that $\mathcal{D}[\![C]\!](s) = \mathcal{D}[\![C']\!](s)$ as required.

□

The relation to Hoare tripples here was inspired by ideas in [15].

2.3 Probabilities and Nondeterminism

Since we are going to be working towards a quantum programming language — which will have inherently probabilistic semantics based on the laws of quantum mechanics (measurements) — we need to now find a way of dealing with probabilities in our syntax and semantics. To this end we introduce a new command

$C ::= \ldots \mid \texttt{cointoss}\ p\ \texttt{in}\ v$

where p is a probability (a real number between 0 and 1) and v is some variable. The idea is that this command assigns v to 0 with probability p and to 1 with probability $1 - p$. Thus, we seek to update the semantic framework to allow us to model commands that have such a nondeterministc effect, and we begin with the operational semantics.

2.3.1 Operational Semantics

We shall now update our transition relation $\rightarrow$, which will no longer necessarily be deterministic. Each transition $(C, s) \rightarrow (C', s')$ will now be labeled with a probability p such that given any (C, s) the sum of all p such that $(C, s) \rightarrow^p (C', s')$ is at most 1.

We define our operational semantics on configurations (C, s) as follows — firstly, in the above semantics whenever we have specified (C, s) reducing uniquely to (C', s') then this transition still holds with certainty, i.e. with probability 1 (thus satisfying the above summation condition.) Finally we add the new operational rules for the cointoss construct:

$$(\texttt{cointoss}\ p\ \texttt{in}\ v,\ s) \rightarrow^p (\texttt{skip},\ s[v \mapsto 0])$$

$$(\texttt{cointoss}\ p\ \texttt{in}\ v,\ s) \rightarrow^{1-p} (\texttt{skip},\ s[v \mapsto 1])$$

We note that the `cointoss` probabilities also sum to 1 as required.

We define $\mathsf{DProb}(S)$ to be the set of discrete sub-probability distributions on S, i.e. functions $\mu : S \rightarrow [0, 1]$ such that $\Sigma_s \mu(s) \leq 1$. Program meanings will now have the form $S \rightarrow \mathsf{DProb}(S)$. We define $\mathsf{Comp}(C, s, s')$

to be the set of all sequences $(C, s) = (C_0, s_0) \rightarrow^{p_1} \ldots \rightarrow^{p_k} (C_k, s_k) = (\texttt{skip}, s')$ and if c is such a sequence we define $p(c)$ to be the product Πp_i. Then given any command C we define $\mathcal{O}[\![C]\!] : S \rightarrow \mathsf{DProb}(S) = S \rightarrow (S \rightarrow [0, 1]) = S \times S \rightarrow [0, 1]$ as follows:

$$\mathcal{O}[\![C]\!](s, s') = \Sigma \{p(c) | c \in \mathsf{Comp}(C, s, s')\}$$

Note that interestingly it is possible that the family over which we sum here could be infinite — for example, in the case where a program loops (using while) until some random variable becomes true. In this case there are an infinite number of branches that can lead ultimately to the same configuration (albeit only a countable number.) In this case we need to check that the above sum does converge. We note that any finite sum of elements in the above set is at most 1 — this is clear, by the probabilities in our operational semantics. As such given the infinite sum of probabilities $\Sigma \alpha_i$ the sum of the first m of these is at most 1 for any m. As such the limit (least upper bound) of this sequence of partial summations must also be at most 1 (since if it was strictly more than 1 it could not be the *least* upper bound.) This is of course assuming the least upper bound exists — but it does, since we can appeal to the classical result of first year real analysis that any bounded monotonic increasing sequence converges. This same issue also arises below in definition of denotational composition; but the same solution can be used to overcome this. (Note that something that cannot help us however is our assumption of finite state sets — even under the presence of a finite number of states there can still be an infinite number of paths to any particular state. Indeed, this is precisely the issue here.)

To make all of this explicit, we define, for any C, $\mathsf{Comp}_n(C, s, s')$ to be the set of all chains $(C, s) = (C_0, s_0) \rightarrow^{p_1} \ldots \rightarrow^{p_k} (C_k, s_k) = (\texttt{skip}, s')$ with length at most n. If c is such a sequence we define $p(c)$ to be the product Πp_i. Then given any command C we define $\mathcal{O}[\![C]\!]_n : S \rightarrow \mathsf{DProb}(S)$ as $\mathcal{O}[\![C]\!]_n(s, s') = \Sigma \{p(c) | c \in \mathsf{Comp}_n(C, s, s')\}$. We then define $\mathcal{O}[\![C]\!](s, s')$ to be the limit of the sequence $\mathcal{O}[\![C]\!]_n(s, s')$. To justify this, we note that $\mathsf{Comp}_n(s, s') \subseteq \mathsf{Comp}_{n+1}(s, s')$ and so since probabilities are positive $\mathcal{O}[\![C]\!]_n(s, s') \leq \mathcal{O}[\![C]\!]_{n+1}(s, s')$, that is this sequence is monotonic increasing. Also this sequence is bounded above, since $\mathcal{O}[\![C]\!]_n(s, s') \leq 1$ for any n (we can show this by induction on n by using the fact that the probabilities at each single reduction step sum to at most one.) As such, once again the sequence converges. Hence $\mathcal{O}[\![C]\!]$ is the pointwise supremum (limit) of the sequence $\mathcal{O}[\![C]\!]_n$.

It is clear that both the infinite sum and limit process given the same result, and for notational ease we shall deal with the former in most cases — we will only need to result to the fine-grained description of the latter for the while case.

Finally we note that we could here generalise proposition 2.2.1 to stating that if $(C, s) \rightarrow (C_i, s_i)$ then $\mathcal{O}[\![C]\!](s) = \Sigma p_i \mathcal{O}[\![C_i]\!](s_i)$.

2.3.2 Denotational Semantics

We use the pointwise ordering on probability distributions and state transformers to make $S \rightarrow \mathsf{DProb}(S)$ into a (continuous) domain (which we will use for the fixpoint theorem for *while*.) We note that any partial function $f : S \rightharpoonup S$ can be embedded in $S \rightarrow \mathsf{DProb}(S)$ by setting $\hat{f}(s)(s') = 1$ if $f(s) = s'$ or 0 otherwise. We now define the denotational semantics of a command C compositionally, with $\mathcal{D}[\![C]\!] : S \rightarrow \mathsf{DProb}(S)$ or equivalently $S \times S \rightarrow [0, 1]$.

$$\begin{array}{lcl}
\mathcal{D}[\![\texttt{skip}]\!] & = & \widehat{id} \\
\mathcal{D}[\![v := e]\!] & = & \widehat{f} \text{ where } f = \lambda s.s[v \mapsto [\![e]\!](s)] \\
\mathcal{D}[\![C_1; C_2]\!](s, s'') & = & \Sigma_{s'}(\mathcal{D}[\![C_1]\!](s, s').\mathcal{D}[\![C_2]\!](s', s'')) \\
\mathcal{D}[\![\texttt{cointoss } v \texttt{ in } p]\!](s, s') & = & \begin{cases} p & \text{if } s' = s[v \mapsto 0] \\ 1 - p & \text{if } s' = s[v \mapsto 1] \\ 0 & \text{otherwise} \end{cases} \\
\mathcal{D}[\![\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2]\!](s) & = & \textsf{if } [\![b]\!](s) = \textsf{tt then } \mathcal{D}[\![C_1]\!](s) \textsf{ else } \mathcal{D}[\![C_2]\!](s) \\
\mathcal{D}[\![\texttt{while } b \texttt{ do } C]\!] & = & \textsf{lfp}[\lambda f : S \to \textsf{DProb}(S).\lambda s : S.\ \textsf{if } [\![b]\!](s) = \textsf{tt then } (\mathcal{D}[\![C]\!]; f)(s) \textsf{ else } \widehat{id}(s)]
\end{array}$$

In the final construct $f; g$ is defined as $\lambda(s, s'').\Sigma_{s'}(f(s, s').g(s', s''))$ as in the composition case. Note here that this sum could be an infinite one. However, assuming the state set is countable we see that there are only countably many s', and since the probabilities are all non-negative we can define this sum to be the ω-limit of the chain of finite approximates as before.

Once again we use domain theory for the meaning of the `while` case. We assign $[0, 1]$ with the continuous cpo-structure inherited from ordering on the real numbers (a linear order) and so $0 = \bot$ and note that least upper bounds of increasing chains exist (since bounded monotone increasing sequences of the real numbers converge.) We then assign $\textsf{DProb}(S) = S \to [0, 1]$ with the cpo-structure inherited from the codomain, using a pointwise ordering. Then we perform a similar pointwise lifting to give $[S \to \textsf{DProb}(S)]$ a cpo structure. Finally, we note that construct inside the lfp expression is indeed continuous with respect to this structure as in e.g. [4] using composition and selection over continuous functions.

2.3.3 Correspondence Result

We now prove the usual correspondence result:

Theorem 2.3.1 *For any command C we have $\mathcal{O}[\![C]\!] = \mathcal{D}[\![C]\!]$.*

Proof Once again we use structural induction on C. In the case that $C = \texttt{skip}$ we note that $\textsf{Comp}(C, s, s')$ is empty unless $s = s'$ in which case it is singleton, and for the the single nullary path $c \in \textsf{Comp}(C, s, s)$ we have $p(c) = 1$ (the empty product.) Hence $\mathcal{O}[\![C]\!](s, s') = 1$ if $s = s'$ and 0 otherwise. That is, $\mathcal{O}[\![C]\!](s, s') = \mathcal{D}[\![C]\!](s, s')$ as required.

In the case that $C = C_1; C_2$ we again consider $\textsf{Comp}(C, s, s')$. Any reduction from (C, s) to $(\texttt{skip}, s')$ consists of firstly a reduction from (C_1, s) to $(\texttt{skip}, s'')$ followed by a reduction from (C_2, s'') to $(\texttt{skip}, s')$. Hence (up to slight relabeling of nodes) we have $\textsf{Comp}(C, s, s') = \bigcup_{s''}(\textsf{Comp}(C_1, s, s'') \otimes \textsf{Comp}(C_2, s'', s'))$ and we note that this union is disjoint. The set formed by the $\otimes$ operation is defined to be the set consisting of any path from $(C_1; C_2, s)$ to $(\texttt{skip}; C_2, s'')$ in $\textsf{Comp}(C_1, s, s'')$ linked to a path (C_2, s'') to $(\texttt{skip}, s')$ in $\textsf{Comp}(C_2, s'', s')$ in the obvious way.

$\mathcal{O}[\![C]\!](s, s')$ is the sum of the probabilities of the paths then in $\bigcup_{s''}(\textsf{Comp}(C_1, s, s'') \otimes \textsf{Comp}(C_2, s'', s'))$. It is clear that this sum is $\Sigma_{s'', c_1, c_2}(p(c_1).p(c_2) | c_1 \in \textsf{Comp}(C_1, s, s'') \wedge c_2 \in \textsf{Comp}(C_2, s'', s'))$, since the probability of a path in this set is just the product of the probabilities of its two subpaths. This expression is $\Sigma_{s''}(\Sigma(p(c) | c \in \textsf{Comp}(C_1, s, s'')).\Sigma(p(c) | c \in \textsf{Comp}(C_2, s'', s'))) = \Sigma_{s''}(\mathcal{O}[\![C_1]\!](s, s'').\mathcal{O}[\![C_2]\!](s'', s'))$. This is $\Sigma_{s''}(\mathcal{D}[\![C_1]\!](s, s'').\mathcal{D}[\![C_2]\!](s'', s'))$ by inductive hypothesis which is simply $\mathcal{D}[\![C]\!]$. Note that all of the above applies to infinite sets of paths.

The case where C is assignment follows exactly as in skip. $\mathsf{Comp}(C, s, s')$ is empty unless $s' = s[v \mapsto [\![e]\!](s)]$ in which case it is singleton, and the single path $c \in \mathsf{Comp}(C, s, s')$ has one probability transition with probability 1. Hence $\mathcal{O}[\![C]\!](s, s') = 1$ if $s' = s[v \mapsto [\![e]\!](s)]$ and 0 otherwise. This is precisely the definition of $\mathcal{D}[\![C]\!](s, s')$.

If C is a conditional statement then we consider the two cases $[\![b]\!](s) = \mathsf{tt}$ or $[\![b]\!](s) = \mathsf{ff}$. In the first case $\mathcal{D}[\![C]\!](s) = \mathcal{D}[\![C_1]\!](s)$ and so we only need to show that $\mathcal{O}[\![C]\!](s) = \mathcal{O}[\![C_1]\!](s)$ by inductive hypothesis. Well this is clear since $\{p(c) | c \in \mathsf{Comp}(C, s, s')\} = \{p(c) | c \in \mathsf{Comp}(C_1, s, s')\}$ since paths in LHS are in a bijection with paths in the RHS (with an additional step at the beginning) and this bijection preserves probabilities (since this additional step has probability 1 attached.) The case when $[\![b]\!](s) = \mathsf{ff}$ is entirely similar.

If $C =$ cointoss p in v then we need to show that $\mathcal{O}[\![C]\!](s)$ is the probability distribution represented by $\{(p, s\,[v \mapsto 0]), (1-p, s\,[v \mapsto 1])\}$. We note that (C, s) reduces to (skip, $s\,[v \mapsto 0]$) with probability p and to (skip, $s\,[v \mapsto 1]$) with probability $1 - p$. Hence $\mathsf{Comp}(C, s, s')$ is empty unless either $s' = s\,[v \mapsto 0]$ or $s' = s\,[v \mapsto 0]$ in which case the set is singleton with probabilities p and $1 - p$ respectively. Thus, $\mathcal{O}[\![C]\!](s, s\,[v \mapsto 0]) = p$ and $\mathcal{O}[\![C]\!](s, s\,[v \mapsto 1]) = 1 - p$ and $\mathcal{O}[\![C]\!](s, s') = 0$ otherwise. This is precisely as required by the denotational semantics.

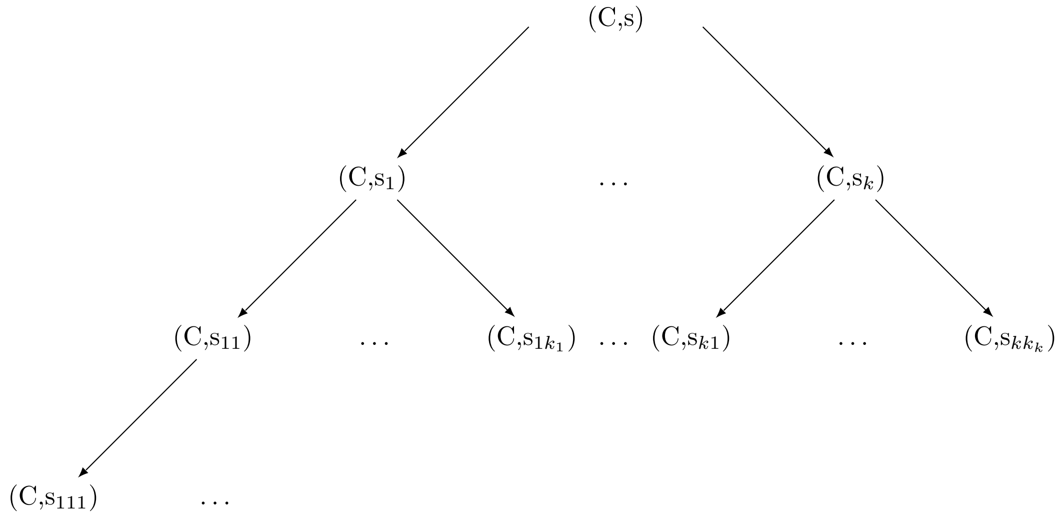
It remains to show the while case with $C =$ while b do C_1. This shall be done in a similar manner to before, showing that $\mathcal{O}[\![C]\!] \sqsupseteq \mathcal{D}[\![C]\!]$ by showing that $\mathcal{O}[\![C]\!]$ is a fixpoint of the function h, and then showing $\mathcal{O}[\![C]\!] \sqsubseteq \mathcal{D}[\![C]\!]$ by a direct induction argument.

For the former, we once again note that if (C, s) reduces to (C', s') with probability 1 then $\mathcal{O}[\![C]\!](s) = \mathcal{O}[\![C']\!](s')$. This is clear by our definition of the operational semantic function. In the case that $[\![b]\!](s) = \mathsf{tt}$ we find that $(C, s) \rightarrow^1 (C_1; C, s)$ and so $\mathcal{O}[\![C]\!](s) = \mathcal{O}[\![C_1; C]\!](s) = \mathcal{O}[\![C_1]\!]; \mathcal{O}[\![C]\!](s) = \mathcal{D}[\![C_1]\!]; \mathcal{O}[\![C]\!](s)$ by inductive hypothesis. In the case that $[\![b]\!](s) = \mathsf{ff}$ we find that $(C, s) \rightarrow^1 (\mathtt{skip}, s)$ and so $\mathsf{Comp}(C, s, s')$ is as in the identity case and we find that $\mathcal{O}[\![C]\!](s) = \widehat{id}(s)$. Either way, we find the $\mathcal{O}[\![C]\!]$ is a fixpoint of the function $h = \lambda g.\lambda s.$ if $[\![b]\!](s) = \mathsf{tt}$ then $(\mathcal{D}[\![C_1]\!]; g)(s)$ else $\widehat{id}(s)$ as required and so it is greater than the least fixpoint of h, i.e. $\mathcal{O}[\![C]\!] \sqsupseteq \mathcal{D}[\![C]\!]$.

For the other inequality, we need to show that $\mathcal{O}[\![C]\!] \sqsubseteq \mathcal{D}[\![C]\!]$ i.e. $\mathcal{O}[\![C]\!](s) \sqsubseteq \mathcal{D}[\![C]\!](s)$ for any s, as probability distributions. This amounts to showing that $\mathcal{O}[\![C]\!](s, s') \leq \mathcal{D}[\![C]\!](s, s')$ as real numbers. We shall show that for all n there is some m such that $\mathcal{O}[\![C]\!]_n(s, s') \leq (h^m \bot)(s, s')$ where h is as above in the definition of the while semantics. From this it shall follow that for all n $\mathcal{O}[\![C]\!]_n \leq \mathcal{D}[\![C]\!]$ and so the limit of the chain $\mathcal{O}[\![C]\!]_n$ is at most $\mathcal{D}[\![C]\!]$, i.e. precisely that $\mathcal{O}[\![C]\!] \sqsubseteq \mathcal{D}[\![C]\!]$.

We show $\forall n$ $\mathcal{O}[\![C]\!]_n(s, s') \leq (h^{n+1} \bot)(s, s')$. Let us consider the shape of $\mathsf{Comp}_n(C, s, s')$ for our specific C = while b do C_1. The shape of a deterministic while reduction branch (as in the case without nondeterminism above) is $(C,\ s) = (C,\ s_0) \rightarrow (C_1; C,\ s_0) \rightarrow \ldots \rightarrow (\mathtt{skip};C,\ s_1) \rightarrow (C,\ s_1) \rightarrow \ldots \rightarrow (C,\ s_2) \rightarrow \ldots \rightarrow (C,\ s_m) \rightarrow (\mathtt{skip},\ s_m)$ for some $m \geq 0$ with $m \leq n$ where $[\![b]\!](s_i) = \mathsf{tt}$ for $i < m$ and $[\![b]\!](s_i) = \mathsf{ff}$ for $i = m$. In the nondeterministic case, this needs arbitrary dimensional addressing, and so we consider the reduction *tree* starting at (C, s) where (C, s) reduces to $(C, s_1) \ldots (C, s_k)$ and then each (C, s_j) reduces to

$(C, s_{j1}) \ldots (C, s_{jk_j})$ for some k_j (see diagram).



We note that in the above $[\![b]\!](s) = \mathsf{ff}$ for all nodes along the bottom row that lead immediately to $(\texttt{skip},s)$ and that for all other nodes s, $[\![b]\!](s) = \mathsf{tt}$. We note that all terminal nodes are at depth (i.e. have an address with length at most) n. Writing $p(s_{ij})$ for the probability of the path from (C, s_i) to (C, s_{ij}) in the tree above (i.e. the product of the single steps in this path) it follows that $p(s_{ij}) \leq \mathcal{O}[\![C_1]\!](s_i, s_{ij})$ by analysis of the branch given above and definition of operational semantics. Note that then $\mathcal{O}[\![C_1]\!](s_i, s_{ij}) = \mathcal{D}[\![C_1]\!](s_i, s_{ij})$ by inductive hypothesis.

Given a node (C, s_i) in the above tree we define $P(s_i)$ to be the sum of the probabilities below the node s_i in the tree — thus $\mathcal{O}[\![C]\!]_n(s)(s') = P(s)$ where s is labeled with the empty address and referring to the top node of the tree. Our claim then amounts to showing that $P(s) \leq (h^{n+1}\bot)(s, s')$. We shall show this by a backwards induction going up the tree — given any node s_i in the tree we write $d(s_i)$ for the maximum distance (in terms of nodes in our tree above where all nodes are of the form (C, s_j)) between (C, s_i) and its underlying $(\texttt{skip}, s')$. We claim that for all nodes in the tree s_i we have $P(s_i) \leq (h^{d(s_i)+1}\bot)(s_i, s')$. We show this by induction on $d(s_i)$.

In the terminal case, with $d(s_i) = 0$ then $s_i = s'$ and $P(s_i) = 1$ since we consider the single transition $(C, s_i) \to (\texttt{skip}, s_i)$ with probability 1. On the other hand since $s_i = s'$ we have $[\![b]\!](s_i) = \mathsf{ff}$ and so $(h\bot)(s_i, s') = \hat{id}(s_i, s') = 1$ and so we have our result.

In the case that $d(s_i) = k+1$ then we consider $P(s_i)$. Clearly by inspection $P(s_i) = \Sigma_j(p(s_{ij}).P(s_{ij}))$ (see above for notation.) Now $(h^{k+2}\bot)(s_i, s') = (h(h^{k+1}\bot))(s_i, s')$. Since $[\![b]\!](s_i) = \mathsf{tt}$ this is $(\mathcal{D}[\![C_1]\!]; (h^{k+1}\bot))(s_i, s')$ $= \Sigma_j(\mathcal{D}[\![C_1]\!](s_i, s_{ij}).(h^{k+1}\bot)(s_{ij}, s'))$. (We note this by definition of the ; operator and our tree — in particular that it contains precisely such notes s_{ij} in our graph since the tree described is maximal by assumption. Also, this is a finite sum since our tree is finite.) By our above note $p(s_{ij}) \leq \mathcal{D}[\![C_1]\!](s_i, s_j)$ and by inductive hypothesis $(h^{k+1}\bot)(s_{ij}, s') \leq P(s_{ij})$ since s_{ij} has depth k. Hence it follows that $(h^{k+2}\bot)(s_i, s') \leq \Sigma_j(p(s_{ij}).P(s_{ij})) = P(s_i)$ as required.

The culmination of this induction states that $P(s) \leq (h^{d(s)+1}\bot)(s, s') \leq (h^{n+1}\bot)(s, s')$ since $d(s) = m \leq n$ and we are dealing with an increasing chain. This states precisely that $\mathcal{O}[\![C]\!]_n(s, s') \leq (h^{n+1}\bot)(s, s') \leq$

$\mathcal{D}[\![C]\!](s,s')$, and since this holds for any n it follows that since $\mathcal{O}[\![C]\!](s,s')$ is a *least* upper bound we have $\mathcal{O}[\![C]\!](s,s') \leq \mathcal{D}[\![C]\!](s,s')$ as required.

It follows then from the above that $\mathcal{O}[\![C]\!](s) = \mathcal{D}[\![C]\!](s)$ in the while case, and so our result is proved.

□

2.3.4 Logical Semantics

We can extend our logic from before replacing our modal operator with

$$\phi ::= \ldots \mid \langle C\rangle_q\phi$$

where q is some rational number. Intuitively, $\langle C\rangle_q\phi$ means "after C, ϕ probably (to the degree q) holds" i.e. that the probability of ϕ holding after C as at least q. Note that from this we can get "at most" using negation — $\langle C\rangle_{1-q}\neg\phi$ holds iff the probability of $\neg\phi$ occurring after C is at least $1-q$, i.e. precisely if the probability of ϕ occurring after C is at most q.

We can then define the semantics as

$$[\![\langle C\rangle_q\phi]\!] = \{s \in S \mid P(\mathcal{D}[\![C]\!](s) \in [\![\phi]\!]) \geq q\}$$

Here if $x \in \mathsf{DProb}(S)$ and $U \subseteq S$ then $P(x \in U) := \Sigma_{s\in U}(x(s))$. Thus given a probability distribution x and set U, $P(x \in U)$ represents the proportion of the distribution x that lies within the set U.

We can then define once again our *weakest precondition semantics*. This time we have $\mathsf{WP}[\![C]\!] : \mathcal{P}(S) \times [0,1] \to \mathcal{P}(S)$ with $\mathsf{WP}[\![C]\!](U,q) = \langle C\rangle_q U$. Once again $\mathsf{WP}[\![C]\!]([\![\phi]\!],q)$ represents the weakest predicate ψ that we need to guarantee that, after running C, then the state satisfies ϕ with a probability at least q.

Once again we can expose the notion of Hoare triples (quadruples?) at this level, with $(\phi C\psi)_q$ meaning "if ϕ holds, then after C, ψ holds with probability (at least) q". Note then $(\phi C\psi)_q$ iff $\phi \Rightarrow \mathsf{WP}[\![C]\!](\psi,q)$ i.e. iff $[\![\phi]\!] \subseteq \mathsf{WP}[\![C]\!]([\![\psi]\!],q)$ i.e. iff $s \in [\![\phi]\!] \Rightarrow s \in \langle C\rangle_q[\![\psi]\!]$ i.e. iff $s \in [\![\phi]\!] \Rightarrow \Sigma_{s'\in[\![\psi]\!]}\mathcal{D}[\![C]\!](s,s') \geq q$. This is all intuitive and what we expect from our above meaning.

We then have a correspondence formulation for this. The natural way of expressing this is once again

Theorem 2.3.2 *$\mathcal{D}[\![C]\!](s,s') \geq q$ iff $(\{s\}\,C\,\{s'\})_q$*

Proof By the note immediately above, the right hand side (with the singleton set giving a trivial summation) holds precisely iff $\mathcal{D}[\![C]\!](s,s') \geq q$ as required.

□

Finally we can once again provide a completeness result:

Theorem 2.3.3 *$\mathcal{D}[\![C]\!] = \mathcal{D}[\![C']\!]$ iff for all ϕ, q we have $[\![\langle C\rangle_q\phi]\!] = [\![\langle C'\rangle_q\phi]\!]$.*

Proof The one direction $\Rightarrow$, as before is trivial. The other direction $\Leftarrow$ runs exactly as before, choosing to set $q = 1$ and using our ϕ_s for any state s as in the deterministic case.

□

2.4 A Quantum Language

We now introduce quantum features into the programming language. We treat qubits as a type of variable (as we have treated classical storage locations as variables.) The state of a qubit is a ray in a 2-dimensional Hilbert space, and we use the tensor product to represent multiple qubits, as above. Thus for k qubits, we have a space of dimension 2^k, and we represent basis vectors by $|b\rangle$ for $b \in \{0,1\}^k$ and so the general form of the state of such a block is $\Sigma \alpha_b |b\rangle$ (with $\Sigma|\alpha_b|^2 = 1$ for normalisation.) Note that such a state cannot be decomposed into a k-tuple of states of the components, but rather represents a superposition/entanglement between such states.

As before, there are just two kinds of physical operations on qubits, and we will represent these by commands in our language. The first is that one may apply a *unitary* linear map to quantum states. In particular these are invertible, and preserve angles and lengths (isometries.) There are "complete" finite bases of unitaries consisting only of unary and binary operations that between them can be used to implement any other unitary (via composition and tensor product [11]) and so we shall assume a small finite set that provides such a basis (however in practice decomposing arbitrary unitaries into their primitive components is not an easy problem.) The second is that one may apply measurements. We consider only measurements in the computational basis, since measurements in other bases can be derived from this by applying a suitable unitary. Explicitly, if we measure a qubit i from a system in state $\Sigma\alpha_b|b\rangle$ then with probability $p_0 = \Sigma\left\{|\alpha_b|^2 : b_i = 0\right\}$ we get the result 0, and the state collapses to $\frac{1}{\sqrt{p_0}}\Sigma\left\{\alpha_b|b\rangle : b_i = 0\right\}$; and with probability $p_1 = \Sigma\left\{|\alpha_b|^2 : b_i = 1\right\}$ we get the result 1, and the state collapses to $\frac{1}{\sqrt{p_1}}\Sigma\left\{\alpha_b|b\rangle : b_i = 1\right\}$. Note that this is just realising equations in the preliminaries section (with respect to the inner product on the Hilbert space) and adding an extra normalisation step (we normalise qubits since we are inherently dealing with *rays.*) Note also that due to the normalisation of the qubit representative states we have $p_0 + p_1 = 1$.

We shall represent our quantum part of the state space by the *QRAM model* as in the literature. Here we have access to a block of quantum bits and can apply unitaries and measurements to certain (pairs of) qubits. Hence e.g. when we apply a unitary U to blocks 2 and 3 we are applying $I \otimes U \otimes \ldots \otimes I$ to the quantum space — here $I : Q \to Q$ is the identity operation that we $\otimes$ in parallel with the single application of U.

In our language we follow the "classical control, quantum data" QRAM paradigm as in [24], and thus are directly extending the presented language above, using our classical control structure but adding the ability to manipultae quantum data.

2.4.1 Syntax

We add two additional constructs to our language. Firstly, as well as program variables `var` we have a set of quantum variables `qvar`. We also assume a set of unitaries `uni`. We then extend our syntax as follows:

C ::= $\ldots$ | `apply uni to qvar, qvar` | `measure qvar in var`

The idea here is that `apply` U `to` q_1, q_2 applies U to the qubits q_1 and q_2; and `measure` q_1 `in` v measures the qubit q_1 in the computational base, and stores the result (either 0 or 1) in the (classical) variable v. Note that here we just consider binary unitaries, treating unitary ones as a special case (once again tensoring with the identity I.)

2.4.2 Operational Semantics

Configurations are now of the form (C, s, ϕ) where (C, s) are as before and $\phi \in Q^k$ where Q is our qubit space $\mathbb{C}^2$ and $Q^k = Q \otimes \ldots \otimes Q$ with the $\ldots$ abbreviating all but $k-2$ of the Q symbols (where k is the number of qubits we have.) Thus Q^k represents our QRAM Hilbert space, and ϕ a normalised representative of a ray in that space.

The transitions from the probabilistic language carry over essentially unchanged. For each transition $(C, s) \rightarrow^p (C', s')$ in our probabilistic language we add a transition $(C, s, \phi) \rightarrow^p (C', s', \phi)$ in our quantum language for each ϕ. Thus, the quantum state is simply carried as a "silent passenger". We simply need to define the rules for the new constructs.

Firstly, for each quantum state $\phi \in Q^k$ with $\phi = \Sigma \alpha_b |b\rangle$ we define $p^i_j(\phi) = \Sigma \left\{|\alpha_b|^2 : b_i = j\right\}$ and $P^i_j(\phi) = \frac{1}{\sqrt{p^i_j}}.\Sigma \left\{\alpha_b |b\rangle : b_i = j\right\}$. Thus $p^i_j(\phi)$ is the probability that we will get the result j on measuring the ith qubit in state ϕ and $P^i_j(\phi)$ is the collapsed state that will result from such a measurement outcome. Given binary unitary U as above we define $U_{i,j}$ to be the result of applying U to qubits indexed i and j and tensoring the map with the identity id on all other qubits.

We now define our new operational rules as follows:

$$(\texttt{apply } U \texttt{ to } q_1, q_2,\, s,\, \phi) \rightarrow^1 (\texttt{skip},\, s,\, U_{i,j}(\phi))$$

$$(\texttt{measure } q \texttt{ in } v,\, s,\, \phi) \rightarrow^{p^q_0(\phi)} (\texttt{skip},\, s\,[v \mapsto 0],\, P^q_0(\phi))$$

$$(\text{measure } q \text{ in } v,\, s,\, \phi) \rightarrow^{p^q_1(\phi)} (\texttt{skip},\, s\,[v \mapsto 1],\, P^q_1(\phi))$$

Note that since $p^q_1(\phi) + p^q_0(\phi) = 1 \leq 1$ this still satisfies the probability limit rules we require above for our operational semantics.

Once again, we assign meanings to commands as probabilistic state transformers. Probabilistic branching is caused by measurement, once again the operational semantics provide a finitely-branching Markov process. There is a slight technical subtlety here for representation reasons — before with a classical state we could always assume that a probability distribution consists of effectively a countable number of (probability, state) pairs up to certain summing conditions. This is not the case when states are rays in a Hilbert space, and so we need to define $\mathsf{DProb}(S \times Q^k)$ as discrete probability subdistributions with *countable support* i.e. the number of non-zero probability states is countable. Note that certainly from the operational semantics the number of states reachable from a configuration is indeed countable, since it's a finitely branching Markov process. Note then that while $\mathsf{DProb}(S \times Q^k)$ does have least upper bounds of increasing chains (pointwise) this is not true of general pairwise-bounded subsets (i.e. it is not a 2cpo — e.g. the three subdistributions $\{(x, 0.5)\}$, $\{(y, 0.5)\}$ and $\{(z, 0.5)\}$ are pairwise bounded but there is no subdistribution that is an upper bound to all three due to the summation condition.)

We can hence define the operational meaning as before, now using the set $\mathsf{Comp}_n(C, (s, \phi), (s', \phi'))$ for our space of paths. Meanings of programs now exactly as before are represented by functions $S \times Q^k \rightarrow \mathsf{DProb}(S \times Q^k)$.

2.4.3 Denotational Semantics

As above, we need to define meanings as $S \times Q^k \to \mathsf{DProb}(S \times Q^k)$. Once again we can use our previous semantics, carrying ϕ along as a silent passenger: if $f : S \to \mathsf{DProb}(S)$ we can define $f' : S \times Q^k \to \mathsf{DProb}(S \times Q^k)$ by $f'(s, \phi, s', \phi') = \delta_{\phi,\phi'}.f(s, s') \in [0, 1] \subseteq \mathbb{R}$ where $\delta_{a,b} = 1$ if $a = b$ and 0 otherwise. Once again we just need to define the semantics for the new constructs. We do this in the obvious manner:

$$\mathcal{D}[\![\texttt{apply}\ U\ \texttt{to}\ q_1, q_2]\!](s, \phi)(s', \phi') = \begin{cases} 1 & \text{if } s = s' \wedge \phi' = U_{i,j}(\phi) \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{D}[\![\texttt{measure}\ q\ \texttt{in}\ v]\!](s, \phi)(s', \phi') = \begin{cases} p_0^q(\phi) & \text{if } s' = s\,[v \mapsto 0] \wedge \phi' = P_0^q(\phi) \\ p_1^q(\phi) & \text{if } s' = s\,[v \mapsto 1] \wedge \phi' = P_1^q(\phi) \\ 0 & \text{otherwise} \end{cases}$$

2.4.4 Correspondence Result

Once again we have our usual correspondence result.

Theorem 2.4.1 *For any command C we have $\mathcal{O}[\![C]\!] = \mathcal{D}[\![C]\!]$.*

Proof This is straightforward: All of the previous cases were dealt with as before, just carrying the quantum state along as a silent passenger. We only need to look at the new commands, which are atomic.

The case of unitary application is exactly the same as that of updating the classical state since there is a single transition of probability 1 ending in `skip` and changing the state parameter.

The case of measurements in a similar manner is the same as the `cointoss` construct as above — it is a single atomic command that changes the state parameter to one of two results with different probabilities.

□

2.4.5 Logical Semantics?

Meanings of formulas are now subsets of $S \otimes Q^k$. Our probabilistic modality $\langle C \rangle_q$ still makes good sense for the quantum programming language with respect to our new quantum denotational semantics; the purely logical constructs have the same interpretation and our classical state predicates still make sense, regarding the first component of the overall state.

However it is not clear which atomic formulas we should use for the quantum part of our system — for example, talking about a specific qubit seems to contradict no-deleting theorems. A very crude mechanism could be then formulas of the type "the overall quantum value is x" for $x \in Q^k$ but this seems to cause problems with decoherence, since no quantum system is entirely independent physically. For verifying e.g. the teleportation protocol, we'd need a formula of the type "old value = new value" — in the classical case we could express this by copying information and then comparing the old value with the new value, but we cannot physically do this in the quantum case because of e.g. no cloning.

We chose then to avoid this problem, and not have any additional quantum atomic formulas. This is not a problem though — we can still reason about results of quantum measurements (since we have formulas of the form "after command C, measurement result v is 0".) Thus, while our atomic formulas are not quantum, the commands we deal with are, and we can still reason about the quantum part indirectly via classical measurements in the modal operator. Thus we satisfy ourselves with this restriction, and pause only to refer to recent work by Alexandru Baltag and Sonja Smets [7] for ideas regarding quantum dynamic logic. Note that physically we cannot observe the state of quantum bits, only through the results of measurements; precisely as in our logic here (thus while we could introduce other atomic formulas regarding the quantum state, they would not be physically testable.)

Finally, we note then that the completeness theorem would clearly only be true for the classical part of the system, exactly as in the previous probabilistic case (since we can only express formulas about the classical part.) In particular our logic in question would not be capable of differentiating between states that differ only in the quantum component. Note that the correspondence result does hold, however, exactly as above (the weakest-precondition construction does not make any use of the atomic formulae available.)

2.4.6 A Monad of Quantum States

Finally we note that we can describe a monad of the notions above, in the sense of functional program side effects, as in [25]. We can define a type constructor T_k as

$$T_k(X) = Q^k \Rightarrow \mathsf{DProb}(X \times Q^k)$$

which, given a classical space X, gives us the notion of carrying along the quantum state by the side of it (and introduces nondeterminism.) Thus, a program with output $T_k(S)$ for input type S ($S \Rightarrow T_k(S)$) then has type (isomorphic to via currying) $S \times Q^k \Rightarrow \mathsf{DProb}(S \times Q^k)$. We can define the return and composition operators for this monad easily, and so it can be slotted into monadic functional programs (such as the interpreters in [25]) in a natural way. Indeed, the interpreters of [25] can be made into our interpreters by the very use of these constructs: we can define the new constructs of unitaries and measurements as mappings $S \Rightarrow T_k(S)$ in the natural manner (ignoring its parameter S — hence these could be considered mappings $1 \Rightarrow T_k(1)$ where 1 is the unit terminal type.) This gives a very nice unification between ideas here and the very general framework for interpreters in [25], which indeed provides an abstraction of the above in an orthogonal direction to that of the rest of this dissertation.

2.5 Application : The Deutch-Josza algorithm

In this section we discuss how the above could be used to represent, for example, the Deutch-Josza algorithm, as presented in [11]. We outline the program, its (denotational) semantics, and how the logical semantics can be used to perform formal verification.

The Deutch-Josza algorithm takes as its input a predicate, i.e. a function $f : B^n \to B$ where B represents the Booleans $\{0, 1\}$, that is (as a precondition) either balanced ($|f^{-1}(\{0\})| = |f^{-1}(\{1\})|$) or constant ($|f^{-1}(\{0\})| = 0$ or $|f^{-1}(\{1\})| = 0$.) The algorithm then calculates (in constant time) whether the function f is indeed balanced or constant (note that this would take $\frac{n}{2} + 1$ checks in a naive classical setup.)

The Algorithm

We now explain the algorithm itself. Given input function $f : B^n \to B$ we define $\tilde{f} : B^n \times B \to B^n \times B$ by $\tilde{f}(i, 0) = (i, f(i))$ and $\tilde{f}(i, 1) = (i, \neg f(i))$ where $\neg 0 = 1$ and $\neg 1 = 0$. We note that given any f, $\tilde{f}$ is bijective since given any (i,j) we have either $f(i)=j$ or $f(i)=\neg j$. The former case implies $(i,0)$ is the unique element mapped to our output by $\tilde{f}$, and the latter case implies $(i,1)$ is.

The input format of this function f is in the form of a unitary $U_{\tilde{f}}$ representing the bijective extension $\tilde{f}$, so that $U_{\tilde{f}}|i\rangle = |\tilde{f}(i)\rangle$. Note that given f the operator U_f is a reasonable representation of f — it is in fact, when viewed as a matrix, the graph of f (with a 1 in location (i,j) if $f(i)=j$ and 0s otherwise.) Also in the case that f is bijective (as in the case $\tilde{f}$ above) then U_f consists of a invertible map that is just rearranging the computational basis vectors (and so certainly a unitary.)

We then apply this unitary $U_{\tilde{f}}$ to a certain special input $\Sigma|i\rangle \otimes (|0\rangle - |1\rangle)$. Note that $U_{\tilde{f}}(\Sigma|i\rangle \otimes |0\rangle)$ gives us $\Sigma|i, f(i)\rangle$ (which encodes all of the information of the entire function in the output, but we cannot read this without performing a measurement, which only returns one $|i, f(i)\rangle$ pair at random which is not much help.) However note also that $U_{\tilde{f}}(|i\rangle, |0\rangle - |1\rangle) = |i, f(i)\rangle - |i, \neg f(i)\rangle = (-1)^{f(i)}|i\rangle \otimes (|0\rangle - |1\rangle)$. Combining these we find that $U_{\tilde{f}}(\Sigma|i\rangle \otimes (|0\rangle - |1\rangle)) = \Sigma((-1)^{f(i)}|i\rangle) \otimes (|0\rangle - |1\rangle)$.

In the case that f is constant, this is $\pm\Sigma|i\rangle \otimes (|0\rangle - |1\rangle)$. In the case that f is balanced, we have $\Sigma(-1)^{f(i)} = 0$ and so $\Sigma_{ij}(-1)^{f(i)}.\langle i|j\rangle = 0$ so $(\Sigma_i(-1)^{f(i)}\langle i|).(\Sigma_j|j\rangle) = 0$ i.e. $\Sigma((-1)^{f(i)}|i\rangle)$ is orthogonal to $\Sigma|i\rangle$. Hence, if we measure the first part of the system with respect to a basis including the vector $\Sigma|i\rangle$ then if f is constant then the result of the measurement must be $\Sigma|i\rangle$ with certainty; and if f is balanced then the result of the measurement cannot possibly be $\Sigma|i\rangle$. Hence if we measure with respect to this basis and observe the result is $\Sigma|i\rangle$ then f is constant, and otherwise f is balanced. This has all been achieved by preparing states, applying a single unitary and then performing a single measurement.

2.5.1 A Simple Quantum Program

We now look at how we might code the above algorithm in our simple quantum programming language. We immediately note that the simplifications of our language has immediately lost the quantum speedup factor that the algorithm provides. For example, in this constant time algorithm we perform “a single measurement” and a “single unitary” but these measurements and unitaries are on a space of n qubits, and in our language we can only perform big compound tensor measurements and unitaries by decomposing them into their unary/binary components, which will of course be linear in n. Nonetheless, we continue, as this could still work in systems when we can perform e.g. arbitrary unitaries and measurements in constant time, and this wouldn’t be a huge complication to our language.

We note in the above that the input format of f is the unitary $U_{\tilde{f}}$. In our language, then we need to write a procedure to apply $U_{\tilde{f}}$ to the qubits it requires (a procedure, but only in the very simple sense of a “copy rule.”) In order to do this then we need to decompose $U_{\tilde{f}}$ into its primitive binary unitary operations (which indeed we have never specified.) This should be reasonably straightforward, however, as, as has been mentioned, the unitary just consists of a rearranging of the computational basis vectors and so can be implemented by switching pairs one at a time (which we can do using the symmetry unitary operation.) This decomposition can thus be created on the fly by classical analysis of f (hence the note above about losing our quantum speedup.)

The second thing we need access to are the state $\Sigma|i\rangle$ and $|0\rangle - |1\rangle$. Well assuming we have access to the standard Hadamard unitary H (with matrix $\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$) we can construct the latter easily from our computational basis by applying H to $|1\rangle$. Of course, we also need to generate $|1\rangle$ but we can do this by applying a computational basis measurement to some qubit and then applying the 180 degree rotation if we end up with $|0\rangle$. Likewise, we can use H to generate $\Sigma|i\rangle$ by applying $H^{\otimes n}$ to $|0\rangle^{\otimes n}$.

Finally, to measure with respect to our basis including $\Sigma|i\rangle$ we can use the inverse to the Hadamard basis $(H^\dagger)^{\otimes n}$ to map the space to itself, sending $\Sigma|i\rangle$ to $|0\rangle$, and then measure in the computational base (we then wouldn't need to perform the reverse translation back to our original state as we're not interested in the quantum output of the measurement.)

In the case $n = 2$ the code would then be as follows (an explanation of each line is to be given shortly):

```
1. measure $q_1$ in $v_1$;
2. if $v_1 = 1$ then apply $rot_{180}$ to $q_1$;
3. measure $q_2$ in $v_1$;
4. if $v_1 = 1$ then apply $rot_{180}$ to $q_2$;
5. measure $q_3$ in $v_1$;
6. if $v_1 = 0$ then apply $rot_{180}$ to $q_3$;
(6a. $v_1$ := 0)
7. apply $H$ to $q_1$;
8. apply $H$ to $q_2$;
9. apply $H$ to $q_3$;
10. run our procedure for applying $U_{\tilde{f}}$ to $q_1, q_2, q_3$;
11. apply $H^\dagger$ to $q_1$;
12. apply $H^\dagger$ to $q_2$;
13. measure $q_1$ in $v_1$;
14. measure $q_2$ in $v_2$
```

Note we have some abbreviations in the above (e.g. applying unitaries to single qubits — in our syntax above we would tensor with the identity; or having an if statement with an empty else clause — in our syntax above we would just have a `skip` statement in the else clause.) Also we note that $H^\dagger = H$, but the notation above is used to make explicit how we are rotating backwards to take the measurement in the required basis.

In the above 1. and 2. sets $q_1 := |0\rangle$, 3. and 4. set $q_2 := |0\rangle$ and 5. and 6. set $q_3 := |1\rangle$ in the manner described above. Then 7. and 8. set both qubits q_1 and q_2 to $|0\rangle + |1\rangle$ (leaving $q_1 \otimes q_2$ in $|00\rangle + |01\rangle + |10\rangle + |11\rangle$) and 9. sets qubit q_3 to $|0\rangle - |1\rangle$. 11. and 12. rotate the system sending $\Sigma|i\rangle \mapsto |00\rangle$ and then 13. and 14. measure in the computational basis, giving us the required measurement as described above. Line 6a. is clearly redundant in our program, but will make reasoning below when we look at the meaning of the above program slightly easier to write out.

Thus, in the above if at the end of the program $v_1 = 0$ and $v_2 = 0$ then we conclude that f was constant, and otherwise infer that it was balanced. Note that the code for greater values of n is similar, with extra lines acting similarly on the other qubits. Our reasoning above then informally verifies correctness of this program; we can do so formally using our semantics.

2.5.2 (Denotational) Semantics

The denotation of the above then gives us a function $S \times (Q \otimes Q \otimes Q) \to \mathsf{DProb}(S \times (Q \otimes Q \otimes Q))$ where we can now consider our state space S as a mapping from our two variables $\{v_1, v_2\}$ to our bit-space $\{0,1\}$ (so $S \cong 2 \to 2 \cong 4$.) This is given then as the composition of the meanings of each of the individual lines in the program above, and we note that since the meaning of sequence ; is associative (easily checked,) the above has a well-defined meaning as a program.

We assume that the meaning of the program indicated by the subroutine for applying $U_{\tilde{f}}$ is the function $\hat{g}$ where $g = id \times U_{\tilde{f}}$ in the sense of the $\hat{-}$ construct from our probabilistic meanings.

The meaning of line 1. then, by our denotational semantics, is a mapping a of the type indicated. If we let $\phi = \Sigma\phi_j|j\rangle$ then applying a to (s, ϕ) returns the distribution containing of a pair: the states $S_i = (s[v_1 \mapsto i], \phi_{i00}|i00\rangle + \phi_{i01}|i01\rangle + \phi_{i10}|i10\rangle + \phi_{i11}|i11\rangle)$ for some probabilities p_i that are irrelevant to this argument ($i \in \{0,1\}$.) This is simply by the meaning of the denotational semantics in the measurement case. We now consider this followed by command 2. above. Command 2. is a single action that is deterministic, and in the definition of the denotational semantics for ; we have a single component in the sum and the result of applying $\mathcal{D}[\![2]\!]$ to the above gives a dual distribution with elements $(\mathcal{D}[\![2]\!]'(S_i), p_i)$ where $\widehat{\mathcal{D}[\![2]\!]'} = \mathcal{D}[\![2]\!]$. Since 2. is a conditional we look at the result of the predicate $v = 0$ on $s[v \mapsto i]$ to see whether we do anything. In the first case $i = 0$ we do nothing, and in the second case we apply rot_{180} to the first qubit of our Hilbert space (by semantics of unitaries and the convention above. Note also we are clearly assuming primitive Boolean expressions of the form $i = j$ and assuming their natural semantics interacting with assignment.) The result of this conditional rotation leaves us in the distribution with elements $((s[v_1 \mapsto i], \phi_{i00}|000\rangle + \phi_{i01}|001\rangle + \phi_{i10}|010\rangle + \phi_{i11}|111\rangle), p_i) = ((s[v_1 \mapsto i], \Sigma_{j,k}\phi_{ijk}|0jk\rangle), p_i)$ for $i \in \{0,1\}$ (we have ignored normalisation of quantum states at this stage; we shall deal with these all at once shortly.)

We then apply line 3. to the above. This once again expands the tree, looking at the result of line 3 on each element of the distribution and pasting each of these trees at the tips of the tree above. Thus the result of applying line 3. leaves us in the distribution with four elements $((s[v_1 \mapsto j], \Sigma_k\phi_{ijk}|0jk\rangle), p_i)$ for $i, j \in \{0,1\}$. As above, the result of applying line 4. to this leaves us in the distribution with four states $((s[v_1 \mapsto j], \Sigma_k\phi_{ijk}|00k\rangle), p_i)$ for $i, j \in \{0,1\}$. In a similar manner, applying lines 5 and 6 to this leave us in the distribution with eight states $((s[v_1 \mapsto k], \phi_{ijk}|001\rangle), p_i)$ for $i, j, k \in \{0,1\}$. We then note that line 6a. is deterministic and sends all elements in the state to that element with the classical state alteration, leaving us in states $((s[v_1 \mapsto 0], \phi_{ijk}|001\rangle), p_i)$ for $i, j, k \in \{0,1\}$. Finally we recall that we haven't normalised the quantum parts of the state as we go along, the normalised form of a quantum state $\phi_{ijk}|001\rangle$ is of course just $|001\rangle$. Hence when we have normalised all of the 8 states in the distribution are all equal, and in fact we have a distribution with one element in, namely $((s[v_1 \mapsto 0], |001\rangle), 1)$. That is, given our initial state (s, ϕ) we have $\mathcal{D}[\![(1;2;3;4;5;6;6a)]\!](s, \phi) = \{((s[v_1 \mapsto 0], |001\rangle), 1)\}$ — thus lines 1 through 6a have indeed set $(q_1, q_2, q_3) = (|0\rangle, |0\rangle, |1\rangle)$ (and also $v_1 = 0$) with certainty, as we require.

Now applying line 7 to this (again using composition in the case where we have a singleton distribution as the result of the first command) leaves us in state $((s[v_1 \mapsto 0], (|0\rangle + |1\rangle) \otimes |01\rangle), 1)$ by the semantics of unitary application and the notion of what the Hadamard gate does. Applying this two more times for the lines 8. and 9. we end up in the state $((s[v_1 \mapsto 0], (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)), 1) = ((s[v_1 \mapsto 0], (|00\rangle + |01\rangle + |10\rangle + |11\rangle) \otimes (|0\rangle - |1\rangle)), 1)$ as we require.

We now use our assumptions of $U_{\tilde{f}}$ with line 10. to leave us in the state (probability distribution) $((s[v_1 \mapsto 0], (\Sigma(-1)^{f(i)}|i\rangle) \otimes (|0\rangle - |1\rangle)), 1)$. Applying lines 11 and 12 then leave us in the probability distribution $((s[v_1 \mapsto 0], (H^\dagger \otimes H^\dagger)(\Sigma(-1)^{f(i)}|i\rangle) \otimes (|0\rangle - |1\rangle)), 1)$. Applying lines 13 and 14 then leave us in states $((s[v_2 \mapsto j][v_1 \mapsto i], |ij\rangle \otimes (|0\rangle - |1\rangle)), p_{ij})$ for probabilities p_{ij} *and furthermore* our reasoning above (together with reasoning about the measurement/rotation) ensures that in the case that f is balanced, $p_{00} = 0$ and in the case that f is constant $p_{00} = 1$. Hence, f is constant iff in all possible (= non-zero possibility) states in the resulting distribution, $v_1 = v_2 = 0$; and f is balanced if $v_1 = v_2 = 0$ occurs in no states of the resulting distribution. So in any event we can examine the value of (v_1, v_2) and can a correct result regarding the nature of f. A more formal way of saying this is using our logical semantics.

2.5.3 Logical semantics

Finally it would be nice to also incorporate the logical semantics into this — i.e. having a logical formula representing whether the function is balanced or constant and then checking at the end that this formula corresponds with the values of v_1 and v_2. Once again this comes into contact with the details of the representation of f, and indeed representing f in the logical language. One way this could be done is that to define the formula $f(i) = j$ we could assume a program C_f that sets v_1 to the value of $f(v_1)$ (note that a program representing f is something we assumed above) and then define $f(i) = j$ as $\langle v_1 = i; C_f\rangle_1(v_1 = j)$ using our atomic formula and modal operator. Once we have a definition of $f(i) = j$ since there are only finite possibilities we can define a (rather long) formula for *f balanced* or *f constant* and it is clear that the definition of this formula will be correct. Then, the specification formula can be given as $\psi = (\mathsf{bal}(f) \vee \mathsf{con}(f)) \Rightarrow (\mathsf{con}(f) \Leftrightarrow \langle C\rangle_1(v_1 = 0 \wedge v_2 = 0)) \wedge (\mathsf{bal}(f) \Leftrightarrow \langle C\rangle_1\neg(v_1 = 0 \wedge v_2 = 0))$ where C is our program (command) above.

Formal verification of the program above then corresponds to *validity* of the above formula — a formula ψ is *valid* if for all (s, ϕ), $s \models \psi$ where $s \models \psi$ means $s \in [\![\psi]\!]$. Assuming the subroutine for applying $U_{\tilde{f}}$ we can assert this: To show that $(s, \phi) \models \psi$ by logical semantics it suffices to show that if $(s, \phi) \models \mathsf{bal}(f) \vee \mathsf{con}(f)$ then $(s, \phi) \models \mathsf{con}(f) \Leftrightarrow \langle C\rangle_1(v_1 = 0 \wedge v_2 = 0)$ and that $(s, \phi) \models \mathsf{bal}(f) \Leftrightarrow \langle C\rangle_1\neg(v_1 = 0 \wedge v_2 = 0)$. We thus assume that $(s, \phi) \models \mathsf{bal}(f) \vee \mathsf{con}(f)$, which is equivalent to f being either balanced or constant (by correctness of $\mathsf{bal}(f)$ and $\mathsf{con}(f)$.)

To show that $(s, \phi) \models \mathsf{con}(f) \Leftrightarrow \langle C\rangle_1(v_1 = 0 \wedge v_2 = 0)$ we need to show that $(s, \phi) \models \mathsf{con}(f)$ iff $(s, \phi) \models \langle C\rangle_1(v_1 = 0 \wedge v_2 = 0)$ by logical semantics. This is that f is constant iff $(s, \phi) \models \langle C\rangle_1(v_1 = 0 \wedge v_2 = 0)$. Now the latter holds iff (by our probabilistic modal operator semantics) the portion of the distribution $\mathcal{D}[\![C]\!](s, \phi)$ lying in the set $[\![v_1 = 0 \wedge v_2 = 0]\!]$ comes to a total probability of at least 1. Given the nature of probability distributions (and the fact that our program will terminate,) this is if all nonzero states of $\mathcal{D}[\![C]\!](s, \phi)$ satisfy $[\![v_1 = 0 \wedge v_2 = 0]\!]$. By our reasoning above in the denotational semantics, this is indeed precisely iff f is constant, and so we are done.

Likewise showing that $(s, \phi) \models \mathsf{bal}(f) \Leftrightarrow \langle C\rangle_1\neg(v_1 = 0 \wedge v_2 = 0)$ amounts to showing that f is balanced iff all nonzero states of $\mathcal{D}[\![C]\!](s, \phi)$ satisfies $\neg(v_1 = 0 \wedge v_2 = 0)$ i.e. precisely iff no such state satisfies $v_1 = 0 \wedge v_2 = 0$. Again by our reasoning above this does indeed hold iff f is balanced, and so we are done here too.

By combining the above and using our logical semantics, we see that $(s, \phi) \models \psi$ for any (s, ϕ), i.e. ψ is indeed a valid formula of our logic, i.e. the program C in question satisfies its specification formula.

3 Towards Abstraction

We shall now begin to work towards providing an abstract version of the above semantics, where are semantic meanings lie in a special kind of a *category* [2].

3.1 Abstracting the Classical Language

We now work towards providing an abstract version of the semantics of the classical language alone. This will familiarise ourselves with working at a categorical level for semantics of programming languages, giving both denotational and operational semantics and showing that they coincide. Our classical language as above shall be described by the following syntax:

$$C ::= \texttt{var} := \texttt{aexp} \mid \texttt{skip} \mid C_1;C_2 \mid \texttt{if bexp then } C_1 \texttt{ else } C_2 \mid \texttt{while bexp do } C$$

3.1.1 Semantic Preliminiaries

We assume the reader is familliar with categories and their standard constructs. We shall give classical programs meanings in a *distributive O-category.* A distributive O-category will be a category with a terminal object, products and a weak form of coproducts; together with a domain structure on each of the hom-sets. It is known that full coproducts and distributivity is incompatible with cpo-enrichment [19] and in particular the canonical example **Cpo** does not have full coproducts, hence we require a weaker form here. Note that as hom-sets are cpo-enriched, each has a bottom element $\bot : A \to B$.

Definition 3.1.1 *An arrow $f : A \to B$ is strict if $f.\bot_{C\to A} = \bot_{C\to B}$*

We then define a *cpo coproduct* of two objects to almost be a coproduct of those two objects, but we only require that $[f, g]$ is the unique *strict* arrow satisfying $[a_1, a_2].q_i = a_i$. Thus given f and g then $[f, g]$ is still (uniquely) well-defined.

We will need to use the usual coproduct laws regarding cpo coproducts. In particular we need the following:

Proposition 3.1.2 *With a cpo coproduct structure we have $[a.b, c.d] = [a, c].(b + d)$.*

Proof To do this it suffice to show that $[a, c].(b + d)$ satisfies $[a, c].(b + d).q_1 = a.b$ and $[a, c].(b + d).q_2 = c.d$ and $[a, c].(b + d)$ is strict. For the first we note that $[a, c].(b + d) = [a, c].[q_1.b, q_2.d]$ and $[a, c].[q_1.b, q_2.d].q_1 = [a, c].q_1.b = a.b$. Similarly we have $[a, c].[q_1.b, q_2.d].q_2 = [a, c].q_2.d = b.d$. Finally we note that $[a, c].[q_1.b, q_2.d]$ is strict since each $[f, g]$ is strict and so $[a, c].[q_1.b, q_2.d].\bot = [a, c].\bot = \bot$.

□

From this we infer that e.g. + is still functorial, since $(a+b).(c+d) = [q_1.a, q_2.b].(c+d) = [q_1.a.c+q_2.b.d] = a.c + b.d$.

Definition 3.1.3 *A* distributive O-category *is a cpo-enriched category with a terminal object, products and cpo coproducts. Furthermore, we require that the canonical map $[id \times q_1, id \times q_2] : (A \times B) + (A \times C) \to A \times (B + C)$ has a left inverse $\mathsf{dist} : A \times (B + C) \to (A \times B) + (A \times B)$ so that* $\mathsf{dist}.[id \times q_1, id \times q_2] = id$.

Finally, we require that composition, pairing and copairing are continuous (and composition is strict) when viewed as functions on the cpo-product of homsets.

Strictness of composition states that $\perp.\perp = \perp$ which in the presence of a terminal object implies that $\perp.s = \perp$ (as will be seen — note that this is the domain theoretic definition of strictness, rather than our internal definition of strictness above.) Ideas of distributive categories here are discussed in [13], with the additional cpo structure imposed for our recursion from ideas of [18] and [4] (we shall comment on examples of these categories shortly.)

The products and cpo coproducts give us our classical structure (together with the terminal object, e.g. Booleans can be represented by $T + T$) distributivity gives us classical control (for the conditional) and the cpo-enrichment allows us to give meanings to while loops via recursion [4] (for example **Set** does not allow recursion but **Pfn** does as partiality is needed to deal with non-termination.)

We assume a value object V and assuming we have a finite memory (otherwise we require Cartesian Closure) we define the state space S to be the n-fold product $V \times \ldots \times V = V^n$ where n is the number of classical registers in memory. We also assume that each variable var is just a value between 1 and n, i.e. an address in this memory.

We assume the following primitive denotations: for every aexp a, $[\![a]\!] : S \to V$. Furthermore we assume (for reasons to become clear) that $[\![a]\!]$ is strict, i.e. $[\![a]\!].\perp = \perp$ when $\perp : T \to S$ (this does not seem to unreasonable to assume.)

For every bexp b, we assume a primitive $[\![b]\!] : S \to T + T$. Furthermore, we assume that $[\![b]\!]$ is truly *Boolean* in that for any $s : T \to S$, $[\![b]\!].s : T \to T + T$ is equal to either q_1 or q_2 (note that if in the category the only arrows $T \to T + T$ are q_1 and q_2 we get this for free, i.e. if the Boolean object has two elements.) Hence the denotation $[\![b]\!]$ is actually fully defined by a function from "elements" of S (arrows $T \to S$) to the Boolean set $\{q_1, q_2\}$. This is an example of how the classical part of our abstract semantics shall in fact not be quite as abstract as they initially appear.

Finally given a variable v ($\in \{1 \ldots n\}$) and $a : S \to V$ we define the variable update map $[v \mapsto a] : S \to S = V^n$ to be $\langle \pi_1, \ldots, \pi_{v-1}, a, \pi_{v+1}, \ldots, \pi_n \rangle$ from our product construction. We firstly show a small lemma concerning $[v \mapsto a]$:

Proposition 3.1.4 *If a is strict (as we have assumed for our primitive* aexp *denotations) then $[v \mapsto a]$ is also strict, i.e. $[v \mapsto a].\perp = \perp$ for $\perp : T \to S = V^n$.*

Proof Both of these sides are mappings $T \to V^n$. Given any object A write $\perp_A$ for $\perp \in \mathbf{C}(T, A)$ with respect to the cpo structure. We need to show then that $[v \mapsto a].\perp_{V^n} = \perp_{V^n}$

Firstly we note that $\perp_{V^n} = \langle \perp_V, \perp_V, \ldots, \perp_V \rangle$ since any map $f : T \to V^n$ can be decomposed using the product structure, and so $f = \langle f_1, \ldots, f_n \rangle \sqsupseteq \langle \perp_V, \perp_V, \ldots, \perp_V \rangle$ by monotonicity of pairing and so the right hand side of this is least with respect to $\sqsupseteq$. Thus in particular $\pi_i.\perp_{V^n} = \perp_V$. Hence

$$\begin{aligned}
[v \mapsto a].\perp_{V^n} &= \langle \pi_1, \ldots, \pi_{v-1}, a, \pi_{v+1}, \ldots, \pi_n \rangle.\perp_{V^n} \\
&= \langle \pi_1.\perp_{V^n}, \ldots, \pi_{v-1}.\perp_{V^n}, a.\perp_{V^n}, \pi_{v+1}.\perp_{V^n}, \ldots, \pi_n.\perp_{V^n} \rangle \\
&= \langle \perp_V, \ldots, \perp_V \rangle = \perp_{V^n}
\end{aligned}$$

as required.

□

3.1.2 Operational Semantics

An *element* of an object A is defined to be an arrow $T \to A$. A *configuration*, as before, is defined to be a pair (C,s) where C is a command and s is an element of S, i.e. a mapping $T \to S$. We define our operational semantics as a relation $\to$ on configurations as as follows:

$$(v := e,\, s) \to (\texttt{skip},\, [v \mapsto [\![e]\!]].s)$$

$$(\texttt{skip};C,\, s) \to (C,\, s)$$

$$\frac{(C_1,\, s) \to (C_1',\, s')}{(C_1;\, C_2,\, s) \to (C_1';\, C_2,\, s')}$$

$$\frac{[\![b]\!].s = q_1}{(\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2,\, s) \to (C_1,\, s)}$$

$$\frac{[\![b]\!].s = q_2}{(\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2,\, s) \to (C_2,\, s)}$$

$$\frac{[\![b]\!].s = q_1}{(\texttt{while } b \texttt{ do } C,\, \mathrm{s}) \to (C;\, \texttt{while } b \texttt{ do } C,\, s)}$$

$$\frac{[\![b]\!].s = q_2}{(\texttt{while } b \texttt{ do } C,\, \mathrm{s}) \to (\texttt{skip},\, s)}$$

Once again the relation is deterministic and so given any (C, s) there is at most one (C', s') such that $(C, s) \to (C', s')$.

Note that while we cannot use these operational semantics to give us a "semantic arrow" i.e. an arrow $S \to S$ in the category, we can define a semantic function as a mapping of element-homsets as follows: Given C define $\mathcal{O}[\![C]\!]$:$\mathbf{C}(T, S)$ $\to$$\mathbf{C}(T, S)$ by, as before, $\mathcal{O}[\![C]\!](s) = s'$ if $(C, s)\vec{\to}(\texttt{skip}, s')$ and $\mathcal{O}[\![C]\!](s) = \bot$ $\in$$\mathbf{C}(T, S)$ if there is no such s'. We make a further lemma:

Proposition 3.1.5 *For any command C we have $\mathcal{O}[\![C]\!](\bot) = \bot$.*

Proof If $(C, \bot)$ loops the the result clearly holds by definition of $\mathcal{O}[\![C]\!]$. If $(C, \bot)\vec{\to}(C, s')$ then we must have $s' = [v \mapsto a_1].[v \mapsto a_2].\ldots.[v \mapsto a_m].\bot$ since this is the only way of updating the state part of the configuration. But since each $[v \mapsto a_i]$ is strict we see that this expression is just $\bot$ (applying strictness right-to-left.)

□

3.1.3 Denotational Semantics

We now define a compositional denotational semantics of the language as an arrow $S \to S$ in the category.

$$
\begin{array}{lcl}
\mathcal{D}\llbracket \texttt{skip} \rrbracket & = & id_S \\
\mathcal{D}\llbracket C_1 ; C_2 \rrbracket & = & \mathcal{D}\llbracket C_2 \rrbracket . \mathcal{D}\llbracket C_1 \rrbracket \\
\mathcal{D}\llbracket v := e \rrbracket & = & [v \mapsto \llbracket e \rrbracket] \\
\mathcal{D}\llbracket \texttt{ if } b \texttt{ then } C_1 \texttt{ else } C_2 \rrbracket & = & [\mathcal{D}\llbracket C_1 \rrbracket, \mathcal{D}\llbracket C_2 \rrbracket].(\pi_1 \times \pi_1).\mathsf{dist}.(id_S \times \llbracket b \rrbracket).\langle id, id \rangle \\
\mathcal{D}\llbracket \texttt{while } b \texttt{ do } C_1 \rrbracket & = & \mathsf{lfp}[\lambda f : S \to S.([f.\mathcal{D}\llbracket C_1 \rrbracket, id].(\pi_1 + \pi_1).\mathsf{dist}.(id \times \llbracket b \rrbracket).\langle id, id \rangle)
\end{array}
$$

Here $\mathsf{dist} : S \times (T+T) \to (S \times T) + (S \times T)$ is as above in our definition of a distributive O-category, and we use distributivity to model conditionals internally: Given S we can copy it to $S \times S$ and then apply our conditional $\llbracket b \rrbracket$ to the latter, leaving us in $S \times (T+T)$. Distributivity allows us to view this as $(S \times T)+(S \times T)$ which is isomorphic to $S + S$. Finally we can use the coproduct structure to act conditionally on this using $[\mathcal{D}\llbracket C_1 \rrbracket, \mathcal{D}\llbracket C_2 \rrbracket]$.

$$\begin{tikzcd}
\mathrm{S} \arrow[r, "\langle id, id \rangle"] & \mathrm{S} \times S \arrow[r, "id \times \llbracket b \rrbracket"] & \mathrm{S} \times (T+T) \arrow[d, "\mathsf{dist}"'] \\
\mathrm{S} & \mathrm{S} + \mathrm{S} \arrow[l, "{[\mathcal{D}\llbracket C_1 \rrbracket, \mathcal{D}\llbracket C_2 \rrbracket]}"'] & (\mathrm{S} \times T) + (S \times T) \arrow[l, "\pi_1 + \pi_1"']
\end{tikzcd}$$

We use a similar structure for the conditional element of the `while` command. Note in said `while` case we are using the fact that the category in question is an O-category. As such, each hom-set (in particular $\mathbf{C}(S,S)$) is enriched with a cpo-structure. Furthermore, enrichment requires that the composition function $\mathbf{C}(S,S) \times \mathbf{C}(S,S) \to \mathbf{C}(S,S)$ is continuous with respect to this cpo-structure. We have similar continuous maps for pairing and copairing. Also constant maps are automatically continuous (see e.g. the domain theory course [4].) From this we infer that the function inside the lfp operator in the while case is indeed continuous, and hence admits the canonical least fixed point (the function can be written as $\mathsf{comp}.\langle \mathsf{copair}.\langle \mathsf{comp}.\langle id, \mathsf{konst}_{\mathcal{D}\llbracket C_1 \rrbracket} \rangle, \mathsf{konst}_{id} \rangle, \mathsf{konst}_f \rangle$ where f is the arrow $(\pi_1 + \pi_1).\mathsf{dist}.(id \times \llbracket b \rrbracket).\langle id, id \rangle$. Here we have decomposed our function into primitive functions on hom-sets that we know to be continuous from basic Domain Theory and our assumptions (konst_f represents the constant function sending all inputs to f.) We have here used category product notation for this, but we are working externally and this is just a notational convenience.

Note that from $\mathcal{D}\llbracket C \rrbracket : S \to S$ we can define a function on hom-sets $\mathcal{D}'\llbracket C \rrbracket : \mathbf{C}(T,S) \to \mathbf{C}(T,S)$ by $\mathcal{D}'\llbracket C \rrbracket(s) = \mathcal{D}\llbracket C \rrbracket.s$ giving us the same type of meaning as our operational semantics, allowing a correspondence theorem.

3.1.4 Correspondence

Our correspondence result shall be as follows:

Theorem 3.1.6 *For any command C, $\mathcal{D}'\llbracket C \rrbracket = \mathcal{O}\llbracket C \rrbracket$*

Note that proving this initial result will be a good indication as to whether these abstract operational semantics are going to work. The above is an equation of *hom-set functions* as we cannot create an arrow from the operational semantics directly since the arrows are abstract and not necessarily functions. In fact, the correspondence result above asserts existence of this possibility — i.e. that there is an internal arrow representing the operational semantics, namely $\mathcal{D}\llbracket C \rrbracket$. Expanding the above definitions, the correspondence theorem states that given any command C and any state s then a) $\mathcal{D}\llbracket C \rrbracket.s = \bot$ iff (C, s) does not terminate b) $\mathcal{D}\llbracket C \rrbracket.s = s' \neq \bot$ iff $(C, s)\vec{\rightarrow}(\texttt{skip}, s')$

Proof We need to show that $\mathcal{D}'\llbracket C \rrbracket = \mathcal{O}\llbracket C \rrbracket$ for any command C. Again we prove this by structural induction on C. Note that in each case it amounts to showing by extensionality that $\mathcal{D}'\llbracket C \rrbracket(s) = \mathcal{O}\llbracket C \rrbracket(s)$ for arbitrary state s.

We firstly consider the case $C = \texttt{skip}$. Then $\mathcal{D}'\llbracket C \rrbracket(s) = \mathcal{D}\llbracket C \rrbracket.s = id.s = s$. Also $\mathcal{O}\llbracket C \rrbracket(s) = s$ since $(C, s)\vec{\rightarrow}(\texttt{skip}, s)$ by reflexivity. Hence $\mathcal{D}'\llbracket C \rrbracket(s) = \mathcal{O}\llbracket C \rrbracket(s)$ as required.

We secondly consider the case $C = C_1; C_2$. Then $\mathcal{D}'\llbracket C \rrbracket(s) = \mathcal{D}\llbracket C_2 \rrbracket.\mathcal{D}\llbracket C_1 \rrbracket.s = \mathcal{O}\llbracket C_2 \rrbracket.\mathcal{O}\llbracket C_1 \rrbracket(s)$ by inductive hypothesis. We only need to show that $\mathcal{O}\llbracket C_2 \rrbracket.\mathcal{O}\llbracket C_1 \rrbracket(s) = \mathcal{O}\llbracket C \rrbracket(s)$.

If (C_1, s) loops then $\mathcal{O}\llbracket C_1 \rrbracket(s) = \bot$ and so LHS is $\mathcal{O}\llbracket C_2 \rrbracket(\bot) = \bot$ by our lemma above (this is where we need the fact that $\llbracket a \rrbracket$ is strict — so that $\mathcal{O}\llbracket C_2 \rrbracket$ is, since if C_1 fails to terminate then C_2 cannot suddenly redeem this to make the composite command C terminate.) Also RHS $= \bot$ since if (C_1, s) loops then so must (C, s) by operational semantics.

If $(C_1, s)\vec{\rightarrow}(\texttt{skip}, s')$ then $\mathcal{O}\llbracket C_1 \rrbracket(s) = s'$. In the case that (C_2, s') loops then LHS $= \mathcal{O}\llbracket C_2 \rrbracket(s') = \bot$ which is also RHS as (C, s) must loop. In the case that $(C_2, s')\vec{\rightarrow}(\texttt{skip}, s'')$ then $\mathcal{O}\llbracket C_2 \rrbracket(s') = s'' = \mathcal{O}\llbracket C \rrbracket(s)$ by operational semantics as in the concrete case, as required.

Thirdly we consider the case $C = v \texttt{ := } e$. Then $\mathcal{D}'\llbracket C \rrbracket(s) = [v \mapsto \llbracket e \rrbracket].s$. Also $(C, s) \rightarrow (\texttt{skip}, [v \mapsto \llbracket e \rrbracket].s)$ and so $\mathcal{O}\llbracket C \rrbracket(s) = [v \mapsto \llbracket e \rrbracket].s = \mathcal{D}'\llbracket C \rrbracket(s)$, as required.

Fourthly we consider the case $C =$ **if** b **then** C_1 **else** C_2. Now either $\llbracket b \rrbracket.s = q_1$ or $\llbracket b \rrbracket.s = q_2$. We assume firstly the former case.

We firstly claim that in this case, $\mathcal{D}'\llbracket C \rrbracket(s) = \mathcal{D}'\llbracket C_1 \rrbracket(s)$. This amounts to showing that $\mathcal{D}\llbracket C \rrbracket.s = \mathcal{D}\llbracket C_1 \rrbracket.s$ and we prove this immediately below.

Let's assume that $\llbracket b \rrbracket.s = q_1$. We claim firstly that $\Delta_S.s = (s \times s).\Delta_T$ where Δ_A is the natural diagonal $\langle id, id \rangle : A \rightarrow A \times A$. Well this follows simply by naturality of said diagonal.

$$\begin{tikzcd}
\mathrm{T} \arrow[r, "\Delta_T"] \arrow[d, "s"'] & \mathrm{T} \times T \arrow[d, "s \times s"] \\
\mathrm{S} \arrow[r, "\Delta_S"] & \mathrm{S} \times S
\end{tikzcd}$$

It follows then that $\mathcal{D}\llbracket C\rrbracket.s = [\mathcal{D}\llbracket C_1\rrbracket, \mathcal{D}\llbracket C_2\rrbracket].(\pi_1 + \pi_1).\mathsf{dist}.(id_S \times \llbracket b\rrbracket).(s \times s).\Delta_T = [\mathcal{D}\llbracket C_1\rrbracket, \mathcal{D}\llbracket C_2\rrbracket].(\pi_1 + \pi_1).\mathsf{dist}.(s \times \llbracket b\rrbracket.s).\Delta_T = [\mathcal{D}\llbracket C_1\rrbracket, \mathcal{D}\llbracket C_2\rrbracket].(\pi_1 + \pi_1).\mathsf{dist}.(s \times q_1).\Delta_T$ by assumption.

We claim that $(\pi_1 + \pi_1).\mathsf{dist}.(s \times q_1).\Delta_T : T \to S + S = q_1.s$. Hence $\mathcal{D}\llbracket C\rrbracket.s = [\mathcal{D}\llbracket C_1\rrbracket, \mathcal{D}\llbracket C_2\llbracket].q_1.s = \mathcal{D}\llbracket C_1\rrbracket.s$ as required.

To see this, firstly we show that $(s \times q_1).\Delta_T = [\langle id, q_1.\mathsf{ignore}_S\rangle, \langle id, q_2.\mathsf{ignore}_S\rangle].q_1.s$ where ignore_S is the unique arrow $S \to T$. Well using the coproduct equation the right hand side is $\langle id, q_1.\mathsf{ignore}_S\rangle.s$. Using the fact that $\langle f, g\rangle.s = \langle f.s, g.s\rangle$ this is $\langle s, q_1.\mathsf{ignore}_S.s\rangle$. Since id_T is the unique arrow $T \to T$ we know $\mathsf{ignore}_S.s = id_T$ so $\langle s, q_1.\mathsf{ignore}_S.s\rangle = \langle s, q_1.id_T\rangle = \langle s, q_1\rangle$. Finally using the law $(f \times g).\langle j, h\rangle = \langle f.j, g.h\rangle$ this is $(s \times q_1).\langle id, id\rangle = (s \times q_1).\Delta_T$ as required.

Hence the LHS of the equation we wish to show is $(\pi_1 + \pi_1).\mathsf{dist}.[\langle id, q_1.\mathsf{ignore}_S\rangle, \langle id, q_2.\mathsf{ignore}_S\rangle].q_1.s = (\pi_1{+}\pi_1).\mathsf{dist}.[(id{\times}q_1).\langle id, \mathsf{ignore}_S\rangle, (id{\times}q_2).\langle id, \mathsf{ignore}_S\rangle].q_1.s = (\pi_1{+}\pi_1).\mathsf{dist}.[(id{\times}q_1), (id{\times}q_2)].(\langle id, \mathsf{ignore}_S\rangle + \langle id, \mathsf{ignore}_S\rangle).q_1.s = (\pi_1 + \pi_1).(\langle id, \mathsf{ignore}_S\rangle + \langle id, \mathsf{ignore}_S\rangle).q_1.s = (\pi_1.\langle id, \mathsf{ignore}_S\rangle + \pi_1.\langle id, \mathsf{ignore}_S\rangle).q_1.s = (id + id).q_1.s = [q_1.id, q_2.id].q_1.s = q_1.id.s = q_1.s$ as required, using product/coproduct laws and our distributivity isomorphism.

Hence $\mathcal{D}'\llbracket C\rrbracket(s) = \mathcal{D}\llbracket C\rrbracket.s = [\mathcal{D}\llbracket C_1\rrbracket, \mathcal{D}\llbracket C_2\rrbracket].q_1.s = \mathcal{D}\llbracket C_1\rrbracket.s = \mathcal{D}'\llbracket C_1\rrbracket(s)$ as required. This can all be summarised in the diagram below, with $\mathcal{D}\llbracket C\rrbracket.s$ representing the clockwise path around the diagram (starting from T) and $\mathcal{D}\llbracket C_i\rrbracket.s$ the anticlockwise path.

$$\begin{tikzcd}
 & & \mathrm{S} & & = & & \mathrm{S} \\
 & & \mathrm{T} \arrow[u, "s"] \arrow[r, "\Delta_T"] \arrow[d, "s"] & \mathrm{T} \times T \arrow[dd, "s \times q_i"] & = & \mathrm{T} \times T \arrow[d, "s \times s"] & \| \\
\mathrm{S} \arrow[dd, "\mathcal{D}\llbracket C_i\rrbracket"] & = & \mathrm{S} \arrow[d, "q_i"] & & & \mathrm{S} \times S \arrow[d, "id \times \llbracket b\rrbracket"] & \mathrm{S} \arrow[l, "\Delta_S"'] \\
 & & \mathrm{S} + \mathrm{S} \arrow[r, "\cong"] \arrow[d, "{[\mathcal{D}\llbracket C_1\rrbracket, \mathcal{D}\llbracket C_2\rrbracket]}"] & \mathrm{S} \times (T + T) & = & \mathrm{S} \times (T + T) & \\
\mathrm{S} & = & \mathrm{S} & & & &
\end{tikzcd}$$

Since $(C, s) \to (C_1, s)$ under these circumstances, it follows that $\mathcal{O}\llbracket C\rrbracket(s) = \mathcal{O}\llbracket C_1\rrbracket(s)$ (considering the cases $s = \perp$ and $s \neq \perp$ respectively and noting unique reduction paths.) From these pieces we can put together our required result using induction: $\mathcal{O}\llbracket C\rrbracket(s) = \mathcal{O}\llbracket C_1\rrbracket(s) = \mathcal{D}'\llbracket C_1\rrbracket(s) = \mathcal{D}'\llbracket C\rrbracket(s)$ using the inductive hypothesis at the central step.

We can repeat all of the above reasoning in the case that $[\![b]\!].s = q_2$ and obtain our required result that $\mathcal{D}'[\![C]\!](s) = \mathcal{O}[\![C]\!](s)$.

We finally consider the while case, $C = \texttt{while}\ b\ \texttt{do}\ C_1$. We show that $\mathcal{O}[\![C]\!](s) = \mathcal{D}'[\![C]\!](s)$ in two cases, $\mathcal{O}[\![C]\!](s) \neq \bot$ and then $\mathcal{O}[\![C]\!](s) = \bot$.

Firstly we assume the former. Then $(C, s)\vec{\rightarrow}(\texttt{skip}, s')$ where $s' = \mathcal{O}[\![C]\!](s) \neq \bot$. We need to show that $\mathcal{D}[\![C]\!].s = s'$. This shall be done in a similar manner to showing $\mathcal{O}[\![C]\!] \sqsubseteq \mathcal{D}[\![C]\!]$ in the concrete case. If $(C, s) \rightarrow (\texttt{skip}, s')$ then we have $(C, s) = (C, s_0) \rightarrow (C, s_1) \rightarrow \ldots \rightarrow (C, s_n) \rightarrow (\texttt{skip}, s_n)$ where $[\![b]\!].s_i = q_1$ for $i < n$ and $[\![b]\!].s_n = q_2$. We show by induction, as before, that for any i we have $(h^{i+1}\bot).s_{n-i} = s_n$. For the case $i = 0$ we have LHS $= (h\bot).s_n = ([\bot.\mathcal{D}[\![C_1]\!], id].(\pi_1 \times \pi_1).\mathsf{dist}.(id \times [\![b]\!]).\langle id, id\rangle).s_n = [\bot.\mathcal{D}[\![C_1]\!], id].(\pi_1 \times \pi_1).\mathsf{dist}.(s_n \times q_2).\Delta_T$ by same reasoning as before since we know that $[\![b]\!].s_n = q_2$. By the reasoning in the if case, this is $[\bot.\mathcal{D}[\![C_1]\!], id].q_2.s_n = id.s_n = s_n$ as required.

If $i = k + 1$ we have $(h^{i+1}\bot).s_{n-i} = (h^{k+2}\bot).s_{n-k-1} = h(h^{k+1}\bot).s_{n-k-1}$. Since $k \geq 0$ we have $n - k - 1 \leq n - 1$ and so $[\![b]\!](s_{n-k-1}) = q_1$. As such $(h(h^{k+1}\bot)).s_{n-k-1} = ((h^{k+1}\bot).\mathcal{D}[\![C_1]\!]).s_{n-k-1}$ by using the same reasoning resulting in the commutative diagram above. But $(h^{k+1}\bot).\mathcal{D}[\![C_1]\!].s_{n-k-1} = (h^{k+1}\bot).\mathcal{O}[\![C_1]\!](s_{n-k-1})$ by inductive hypothesis. But $\mathcal{O}[\![C_1]\!](s_{n-k-1}) = s_{n-k}$ by looking at our reduction path, and so this is $(h^{k+1}\bot).s_{n-k}$. By inductive hypothesis on i we see that this is s_n and we conclude that $(h^{i+1}\bot).s_{n-i} = s_n$ as required.

We conclude then that with $i = n$ in the above claim that $(h^{n+1}\bot).s_0 = s_n$ i.e. $(h^{n+1}\bot).s = s'$. We can then repeat the induction above in exactly the same manner to show that for any i, $(h^{i+2}\bot).s_{n-i} = s_n$. It follows that $(h^{n+2}\bot).s = s'$.

Consider the chain $((h^i\bot).s)_i$. We know that this chain is monotonic increasing since $(h^i\bot)_i$ is monotonic increasing and so is the constant chain $(s)_i$. It follows that since $((h^i\bot).s)_{n+1} = ((h^i\bot).s)_{n+2} = s'$ that the limit of this chain is s' (by induction we can show that $(h^j\bot).s = s'\forall j \geq n + 1.$). The limit of this chain is $\bigsqcup(h^i\bot).\bigsqcup(s)$ by continuity of composition which is $\mathcal{D}[\![C]\!].s$, and so we conclude that $\mathcal{D}[\![C]\!].s = s'$, as required.

We have just shown in the above that if $\mathcal{O}[\![C]\!](s) \neq \bot$ then $\mathcal{O}[\![C]\!](s) = \mathcal{D}'[\![C]\!].s$. We now need to show that if $\mathcal{O}[\![C]\!](s) = \bot$ then $\mathcal{D}'[\![C]\!](s) = \bot$. To do this we shall show that $\mathcal{D}'[\![C]\!](s) \sqsubseteq \mathcal{O}[\![C]\!](s)$ in any case, which implies in the case that $\mathcal{O}[\![C]\!](s) = \bot$ we have $\mathcal{D}'[\![C]\!](s) = \bot$ also by antisymmetry of the cpo order relation. Showing that $\mathcal{D}'[\![C]\!](s) \sqsubseteq \mathcal{O}[\![C]\!](s)$ for all s amounts to showing that $\mathcal{D}'[\![C]\!] \sqsubseteq \mathcal{O}[\![C]\!]$ when viewed as elements in the cpo-function space.

To do this, we firstly show that $\mathcal{D}'[\![C]\!]$ is the least fixpoint of the function g where $g : (\mathbf{C}(T, S) \rightarrow \mathbf{C}(T, S)) \rightarrow (\mathbf{C}(T, S) \rightarrow \mathbf{C}(T, S))$ is given by

$$g = \lambda f : \mathbf{C}(T, S) \rightarrow \mathbf{C}(T, S).\lambda s \in \mathbf{C}(T, S).\ \mathsf{if}\ [\![b]\!].s = q_1\ \mathsf{then}\ f(\mathcal{D}[\![C_1]\!].s)\ \mathsf{else}\ s$$

Firstly we note that g is continuous, by Standard Observations. To show that $\mathcal{D}'[\![C]\!]$ is indeed this least fixed point we use the result above together by induction by showing that the approximates are the same, i.e. that $(g^k\bot)(s) = (h^k\bot).s$ where $h : \mathbf{C}(S, S) \rightarrow \mathbf{C}(S, S)$ given in the semantics of the while case. We show this by induction on k.

The case $k = 0$ amounts to $\bot.s = (\bot \in \mathbf{C}(T, S))(s) = \bot$. Since comp is strict we note that $\bot_{B\rightarrow C}.\bot_{A\rightarrow B} = \bot_{A\rightarrow C}$. We use presence of the terminal object to obtain our result:

$$\begin{tikzcd}
A \arrow[r, "s"] \arrow[dr, "\perp"] & B \arrow[d, "\perp"] \arrow[dr, "\perp"] & \\
& T \arrow[r, "\perp"] & C
\end{tikzcd}$$

From the above we infer that $\perp.s = \perp.\perp = \perp$ as required (taking in our special case $A = T$ and $B = C = X$.)

For the inductive step $k = n+1$ we have $(h^{k+1}\perp).s = [(h^k\perp).\mathcal{D}[\![C_1]\!], id].(\pi_1 \times \pi_1).\mathsf{dist}.(id \times [\![b]\!]).\langle id, id\rangle.s$. In the case that $[\![b]\!].s = q_1$ in the usual manner this is $(h^k\perp).\mathcal{D}[\![C_1]\!].s$ which is $(g^k\perp)(\mathcal{D}[\![C_1]\!].s)$ by inductive hypothesis which is $(g^{k+1}\perp)(s)$ by definition of g. The case where $[\![b]\!].s = q_2$ both sides are equal to just s and so hence equal (using again the correspondance results for conditionals above.)

Thus for any k, $(h^k\perp).s = (g^k\perp)(s)$. Thus $\mathcal{D}'[\![C]\!](s) = \mathcal{D}[\![C]\!].s = (\bigsqcup h^k\perp).(\bigsqcup s) = \bigsqcup(h^k\perp).s = \bigsqcup(g^k\perp)(s) = (\bigsqcup g^k\perp)(s) = \mathsf{lfp}(g)(s)$ and so $\mathcal{D}'[\![C]\!] = \mathsf{lfp}(g : \mathbf{C}(T,S) \rightarrow \mathbf{C}(T,S))$ by generalisation over s. Furthermore, $\mathcal{O}[\![C]\!]$ is a fixed point of the function g by definition of operational semantics exactly as in the concrete case above. It follows that $\mathcal{D}'[\![C]\!] \sqsubseteq \mathcal{O}[\![C]\!]$ as required, since $\mathcal{D}'[\![C]\!]$ is the least such fixed point. Hence in particular if $\mathcal{O}[\![C]\!](s) = \perp$ then $\mathcal{D}'[\![C]\!](s) = \perp$, as required. This completes the `while` and final case of the induction.

□

3.1.5 Comments on Distributive O-Categories

We have shown above how to give operational and denotational semantics for a classical language in a distributive O-category assuming the relevant primitives. Some examples of O-categories include e.g. **Rel** (the category of sets and relations,) **Pfn** (the category of sets and partial functions) and **Cpo** (the category of domains and continuous functions.) The cpo-structure on the former two comes from inclusion and the latter componentwise.

Of these, **Rel** has both products and coproducts, but these in fact coincide (both being the Cartesian Product of the two sets — this makes sense since **Rel** and $\mathbf{Rel}^{\mathsf{op}}$ are isomorphic as categories.) Because of this, we in fact see that the terminal object T is also an initial object, and as such there is only one "element" $T \rightarrow A$ (and this must hence be $\perp$). As such our operational semantics are not going to work. In **Rel** the terminal object is in fact a *zero object*, and we will learn more of these creatures later.

We can also order **Pfn** by subset and find that this too is an O-category. We can define products in **Pfn** in the same way for **Set**. Unfortunately this causes a problem: let $f : A \rightharpoonup B$ be the empty partial function and $g : A \rightharpoonup A$ be the constant map. Then $\langle f, g\rangle : A \rightharpoonup B \times A$ must be undefined since $\pi_1.\langle f, g\rangle = f$ and π_1 would clearly send any defined element of $B \times A$ to a defined element of B. But then $\pi_2.\langle f, g\rangle$ is also undefined, but this is required to be $g = id_A$ by definition of the product. We clearly need to move to the more general case of **Cpo**.

In **Cpo** the way to define a product of two objects is not obvious, since there are two options: We could chose to either to a) adjoin a new bottom element or b) coincide the two bottom elements in question (the *smash product* [4]). The latter causes the product operation not in fact to be a product, for precisely the same reasons as above. It is option (a) that allows us to solve this problem, and this does indeed give us a product structure. For the coproduct, again we can chose to either a) coincide $inl\bot$ and $inr\bot$ or b) add a further $\bot$ lying beneath them. (a) causes problems since then $[f, g]$ is not well-defined unless f and g are both strict. For (b) we're ok, except we lose the unicity property of the coproduct (How do we define $[f, g](\bot)$?). However this is not a major problem: we define $[f, g](\bot) = \bot$ and we're done, as $[f, g]$ is indeed the unique strict map h s.t. $h.q_1 = f$ and $h.q_2 = g$ and so $[f, g]$ is indeed a cpo coproduct as we require.

We finally need to check distributivity. The canonical map $(A \times B) + (A \times C) \to A \times (B + C)$ in **Cpo** is not an isomorphism: its inverse must send both $(a, \bot)$ and $(b, \bot)$ to $\bot$. Rather than being an isomorphism, in **Cpo** this map is an *embedding.* But it does have the left inverse property we require — there is a map dist such that $\mathsf{dist}.[(id \times q_1), (id \times q_2)] = id$ [13].

Finally, we note that in **Cpo** we do indeed have a terminal object, the one-object domain $\{\bot\}$ since there is a unique continuous map $S \to T$ that sends everything to $\bot$. Also elements of X are represented by (continuous) mappings $\{\bot\} \to X$ in a healthy way — it is a good job we do not require strictness of arrows, in which case $\{\bot\}$ would in fact be a zero object, as in **Rel**.

Also we claim without proof here that composition, pairing and copairing are indeed continuous operations in **Cpo** (see e.g. [4] for this and a full discussion of results in this subsection.) Hence **Cpo** is indeed a distributive O-category as we require.

We now turn to providing a categorical account of the probabalistic and quantum parts of the language; and to do this we need to firstly expose categorical quantum mechanics.

3.2 Preliminaries : Abstract Quantum Mechanics

We now introduce the ideas of categorical quantum mechanics that we'll need for lifting our entire quantum language to the abstract categorical level. The categorical axiomatisation of quantum mechanics introduced in [6] can be seen as consisting of two levels, the *multiplicative* level and the *additive* level. The multiplicative level provides a representation of time and space, operations, preparations of states, projections etc. We can then use an additive operator $\oplus$ to sum together these different "slices of probability" and indeed use *scalars* to represent the probability of each slice (which arise from the categorical structure alone.) Axiomatics for the multiplicative level (in which we model deterministic slices) are given in terms of *dagger compact closed categories* and these are then summed together using *biproducts.*

In order to understand what the features of the abstract quantum category we need, we firstly give an incremental definition of our dagger compact closed categories with biproducts, building upon the notion of a *symmetric monoidal category.* For notions of this see e.g. [2], but in brief we have a bifunctor $\otimes$: $\mathbf{C} \times \mathbf{C} \to \mathbf{C}$ that has natural associativity and symmetry isomorphisms together with a monoidal identity I such that any composition of these natural isomorphisms are equal (coherence.) As a motivating example fitting in with our concrete semantics, the category **FdHilb** is a model of this with $\otimes$ representing the tensor product and I representing the field of scalars $\mathbb{C}$. In fact, **FdHilb** is an example of all of the structures in this section, as indeed is the category **Rel** of sets and relations (thus providing an example of a category in which we can do quantum mechanics which is not **FdHilb**.) Definitions in this section are described in more detail in e.g. [6] and [23], along with relating results.

3.2.1 Monoidal Scalars

Firstly we will wish to model *scalars* in our category. A scalar is an element of the field I; elements of objects are given as mappings from the monoidal unit I; and thus scalars are represented by arrows $I \to I$. We can then multiply scalars using composition, and it is shown in [17] that multiplication of such scalars is commutative. Given an arrow $f : A \to B$ and scalar s we then define scalar multiplication of arrows $s \bullet f$ as

$$\begin{tikzcd}
\mathrm{A} \arrow[d, "\cong"'] & \mathrm{B} \arrow[d, "\cong"'] \\
\mathrm{A} \otimes I \arrow[r, "f \otimes s"] & \mathrm{B} \otimes I
\end{tikzcd}$$

Scalars (and in particular *positive ones*) will represent probability weightings in our semantics.

3.2.2 Star-autonomous Categories

Definition 3.2.1 *A symmetric monoidal category is* star-autonomous *if there exists a full and faithful contravariant functor* $(-)^*$ *supporting a natural (hom-set) isomorphism* $C(A \otimes B, C^*) \cong C(A, (B \otimes C)^*)$*.*

Note in the above we find that the category is closed (as in the sense of adjunctions / monoidal closure) when setting $A \multimap B = (A \otimes B^*)^*$.

Note that if a symmetric monoidal category represents semantics for linear logic, then the star-autonomy provides a form of negation: a full and faithful contravariant functor represents contrapositives — proofs from A to B correspond precisely to proofs from $\sim B = B^*$ to $\sim A = A^*$. Then the closure operator $\multimap$ can represent implication: $A \multimap B$ holds iff (A and $\sim B$) fails. In particular, a star-autonomous category is symmetric monoidal closed, with the closure being expressed in the more fine-grained notion of negation. Given a symmetric monoidal closed category, we have an equivalent definition for the fact that the implication can be expressed in terms of negation:

Definition 3.2.2 *A symmetric monoidal closed category is* star-autonomous *iff there exists an object F such that the canonical map $A \to (A \multimap F) \multimap F$ is an isomorphism for any A.*

This latter definition is useful in showing the interesting fact that if a symmetric monoidal closed category is in fact a Cartesian Closed Category; and the category is also star-autonomous with respect to this structure; then the entire category collapses and necessarily must be a preorder [1].

3.2.3 Compact Closed Categories

A compact closed category is a star-autonomous category (= symmetric monoidal + negation) for which the multiplicative AND $\otimes$ and OR $\parr$ operators coincide, where $A \parr B := (A^* \otimes B^*)^*$ — that is, $\parr$ is the deMorgan dual of $\otimes$. Thus compact closure is a "degenerate" special case of a star-autonomous category.

Definition 3.2.3 *A* compact closed *category is a star-autonomous category for which we have a natural isomorphisms* $(A \otimes B)^* \cong A^* \otimes B^*$ *and* $I^* \cong I$*.*

Compact closed categories give us much of the multiplicative structure of Categorical Quantum Mechanics. For this purpose, a different (equivalent) definition is useful.

Definition 3.2.4 *A symmetric monoidal category is* compact closed *if each object A has a dual object A^*, a unit $\eta_A : I \to A^* \otimes A$ and a counit $\epsilon_A : A \otimes A^* \to I$ such that $(\epsilon_A \otimes 1_A).(1_A \otimes \eta_A) = 1_A$ and $(1_{A^*} \otimes \epsilon_A).(\eta_A \otimes 1_{A^*}) = 1_{A^*}$ with natural monoidal isomorphisms left implicit.*

The equational axioms for units and counits is the fundamental "yanking" rule of abstract Quantum Mechanics [6,10]. Note that in such a category we can extend $(-)^*$ functorially and also define *names* and *conames* of morphisms as performed in e.g. [6].

3.2.4 Dagger Structure

To perform quantum mechanics at the multiplicative level we will also require the notion of the *adjoint* of a function, allowing us to define e.g. unitaries and conjugation.

Definition 3.2.5 *A* dagger category *is a category $\boldsymbol{C}$ together with a a contravariant involutive identity-on-objects $(-)^\dagger$ endofunctor on $\boldsymbol{C}$.*

In such a category, we say an arrow is *unitary* if $f^\dagger.f = f.f^\dagger = id$, and an endoarrow is *self-adjoint* if $f = f^\dagger$.

Definition 3.2.6 *A* strongly compact closed category *is a compact closed category $\boldsymbol{C}$ together with a dagger operation as above, such that $(f \otimes g)^\dagger = f^\dagger \otimes g^\dagger$; the symmetric monoidal isomorphisms are unitary and such that the counit is the adjoint to the unit (up to a symmetry natural isomorphism.)*

It is these strongly (or dagger) compact closed categories that can provide a basis for the multiplicative level of quantum mechanics (and that these levels are then probabilistically summed together using biproducts, to come next.) In a dagger compact closed category we can derive many operations, such as conjugation (using adjoints together with the compact closure * operator,) inner products, traces, unitaries, Bell states etc. The correctness of a deterministic branch of the teleportation protocol can then be easily proved using information flow techniques and the graphical language [6].

3.2.5 Biproducts

In Linear Logic, we can add in classical "additive" operators & and $\oplus$ representing the Cartesian and Cocartesian constructs. We can make sense of this in any symmetric monoidal category that *in addition* has products, coproducts, an initial object and a terminal object (so in the case where we have the additive structures only, we have a representation of standard intuinistic logic.) At the multiplicative level we required coinciding AND and OR operators; and this situation mirrors itself at the additive level. A category has *biproducts* if, intuitively, it has coinciding products and coproducts.

Definition 3.2.7 *A* zero object *in a category is both initial and terminal.*

Note then given any two objects A and B there is a unique zero arrow $A \to 0 \to B$. We write 0 for this arrow, in the same way that we write 1 for the identity map.

Similarly, one can perform this treatment for the binary case:

Definition 3.2.8 *A* biproduct *of A_1 and A_2 is an object $A_1 \oplus A_2$ together with morphisms $p_i : A_1 \oplus A_2 \to A_i$ and $q_i : A_i : A_1 \oplus A_2$ such that the pair (p_1, p_2) forms a product cone; (q_1, q_2) forms a coproduct cone; and $p_i.q_j = \delta_{ij}$ where $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $j \neq i$.*

Addition for arrows in this category is to be defined shortly. We have the following equivalent definition, that is more succinct but uses more complex notions:

Definition 3.2.9 *In a category, any map $f : A + B \to C \times D$ can be written as a matrix. A category has* biproducts *if it has a zero object, products and coproducts and furthermore any map $\Sigma A_i \to \Pi A_i$ represented by the "identity matrix" (in the sense of standard notation of linear algebra adapted to our abstract purposes, with 1 representing the identity and 0 representing the zero arrow) is an isomorphism.*

The implicit matrix representation in the above definition shows a nice generalisation of linear algebra matrix multiplication and indeed shows that matrix usage in linear algebra works because we the domain is a coproduct and the codomain is a product. Indeed biproducts as an abstraction of ideas from linear algebra continue in this manner: using the biproduct structure we can define addition of arrows in the same hom-set ($f + g = \Delta.(f \oplus g).\nabla$) and this addition enriches the category with the structure of a commutative monoid, with the zero map as the identity. We then have $q_1.p_1 + q_2.p_2 = id$. Furthermore, any map then $A \oplus B \to C \oplus D$ can be written as a matrix; and composition of such matrices correspond to matrix multiplication in the usual sense (which then uses the commutative addition.)

It shall later be useful to find a converse to this construction.

Definition 3.2.10 *Given any category $\boldsymbol{C}$ that is enriched over commutative monoids, we can define the* biproduct completion *of $\boldsymbol{C}$ to be the "free" biproduct category generated from $\boldsymbol{C}$. Objects are this category are finite tuples of objects from $\boldsymbol{C}$, and arrows $f : \langle A_1, \ldots, A_n \rangle \to \langle B_1, \ldots, B_m \rangle$ consist of $n \times m$ matrices whose (i, j)th component is a map $A_i \to B_j$ in C. Composition is just matrix multiplication (using the addition structure assumed of $\boldsymbol{C}$). Given a commutative monoid enriched category $\boldsymbol{C}$ the outcome of the construction is denoted $\boldsymbol{C}^{\oplus}$.*

Note fundamentally that $\mathbf{C}$ is a (full) subcategory of $\mathbf{C}^{\oplus}$, sitting inside it as the singleton tuples, and that the category $\mathbf{C}^{\oplus}$ has biproducts (given by concatenation of object tuples.)

Finally, we note that if we have a compact closed category with biproducts it follows that $(A \oplus B) \otimes C \cong (A \otimes C) \oplus (B \otimes C)$ and so we automatically have a distributivity natural isomorphism (see e.g. [6] for these results.) We also have distributivity laws of composition over addition (and we shall use this law heavily — it forces all arrows in our category to be *linear* in terms of linear algebra.) In addition, then the dagger structure preserves the biproducts (i.e. addition) "up to isomorphism". We would like it to preserve them directly, however, which requires an additional coherence condition.

3.2.6 Strongly Compact Closed Categories with Biproducts

The coherence condition we shall require has many equivalent forms, such as $\langle f, g \rangle^\dagger = [f^\dagger, g^\dagger]$ or $f^\dagger + g^\dagger = (f + g)^\dagger$ or $p_i^\dagger = q_i$. And so

Definition 3.2.11 *A* strongly compact closed category with biproducts *is a strongly compact closed category with biproducts such that for $i = 1, 2$ we have $p_i^\dagger = q_i : A_i \to A_1 \oplus A_2$ for all objects A_1, A_2.*

Using this biproduct structure we can now interact with the multiplicative level and introduce classical communication into our abstract presentation of quantum mechanics. This is all laid out explicitly in [6], where the authors formulate and prove correctness of the teleportation protocol at this abstract level (using the biproduct to provide the classical part of the communication — quantum data and classical control, as in [24].)

In the above incremental definition, some of the axiomatics "overwrite each other". An equivalent single-stroke definition of an SCCCB can be given as follows for ease of use [5]:

Definition 3.2.12 *A strongly compact closed category with biproducts is a symmetric monoidal category that has: A monoidal involutive assignment $A \mapsto A^*$ on objects; an identity-on-objects, contravariant, monoidal, involutive functor $f \mapsto f^\dagger$; and a unit $\epsilon_A : I \to A^* \otimes A$ with $\epsilon_{A^*} = \sigma_{A^*;A}.\epsilon_A$; and a biproduct structure; such that $(\epsilon_A^\dagger.\sigma_{A;A*} \otimes id_A).(id_A \otimes \epsilon_A) = id_A$; $p_i = q_i^\dagger$ for the biproducts; and also such that the symmetric monoidal natural isomorphisms are unitary with respect to †.*

Once again in the above some of the monoidal natural isomorphisms have been left implicit; and as such the types of identities have to be included. We note that by the coherence conditions for the natural monoidal isomorphisms any way of inserting the natural isomorphisms that make the expression type-valid will represent the same arrow in the category.

For full details of how this provides an axiomatisation of quantum mechanics see many of the recent papers such as [5,6,10].

3.3 Program Semantics in an SCCCB

We now seek to look at how we might provide our program semantics in an SCCCB.

3.3.1 Classical Semantics in an SCCCB

In our classical abstract semantics above, we strongly used the Cartesian and Cocartesian nature of the category in question. We intend to provide semantics for these classical constructs in our biproduct strongly compact closed category, where these constructs are not available to us (the coproduct is since we have $\oplus$, but for products we will need to use $\otimes$ which is only endowed with a symmetric monoidal structure and does not have the stronger Cartesian requirements. We cannot use $\oplus$ for both our product and coproduct as e.g. then distributivity is likely to fail.) We shall perform a trick that allows us to extract a distributive (Cartesian) category out of any SCCCB, which we can use to adapt our above semantics to this level. This shall use the monoidal tensor $\otimes$ as the product operation, with monoidal unit I taking the place of the terminal object.

Considering I in this sense, we firstly "unpack" V to be defined as the object $k.I = I \oplus \ldots \oplus I$ where k is the number of values each of our variables can take. Here we are using the fact that $\oplus$ is a coproduct. We then define S to be $V^{\otimes m}$ where we have m variables in play. Hence $S \cong k^m.I$ using distributivity of $\otimes$ over $\oplus$. Importantly, then, our state object here will be of the form $k.I$ for some k (i.e. *finite dimensional.*)

In the above, we defined an element of an object A to be a mapping $T \to A$. An element will now be of the form $I \to A$, with I now representing our selector object. Furthermore, we assume that A is of the form $k.I$ for some k, as with S above. Since we wish to be representing only classical computations here, where $\otimes$

just represents the product, we consider only elements $I \to k.I$ that are *classical*, i.e. coproduct injections q_i for some i. Thus an element of $S = k^m.I$ will be precisely one of k^m coproduct injections, i.e. one of the k^m values intuitively within S. Our meaning arrows will then also be classical in that they will send classical elements (coproduct injections) to other classical elements (coproduct injections.) We are effectively working here in the *classical subcategory* of an SCCCB, and it is equivalent to the concrete category of functions on finite sets.

Definition 3.3.1 *Given any SCCCB* $\boldsymbol{C}$ *we can construct the* classical subcategory *of the* $\boldsymbol{C}$ *with objects of type* $I \oplus \ldots \oplus I$ *and arrows* $n.I \to m.I$ *are those defined from a function* $f : \{1 \ldots n\} \to \{1 \ldots m\}$ *in the obvious manner (permuting the coproduct possibilities, i.e. arrows* $[q_{f(1)}, \ldots, q_{f(n)}]$*.)*

Proposition 3.3.2 *This category is distributive, with* $\otimes$ *now being Cartesian and the monoidal unit* I *being the terminal object.*

Proof This is obvious, since arrows $m.I \to n.I$ are just functions from m to n, $m.I \otimes n.I$ has dimension mn and the set mn in in **Set** is the Cartesian product of the sets m and n. Explicitly, we have projections $p_1p_2.I \to p_i.I$ sending (x_1, x_2) to x_i when viewed in this function form. Likewise given $f : m.I \to k.I$ and $g : m.I \to n.I$ we can define an $m.I \to kn.I$ in the obvious way using multiplication/pairing sending i to $f(i).g(i)$.

□

From the above we can extract a natural copying map in the subcategory, $\langle id, id \rangle : m.I \to m^2.I$ sending i to $i.i$. Hence we do have an arrow $S \to S \otimes S$ that does perform copying, but only on classical arrows. Assuming then that our primitive expressions are given classical denotations, we can hence adapt the general treatment of semantics in a distributive category to this one, interpretting classical programs in (this subcategory) of the SCCCB as a classical endoarrow on S. Of course, to deal with while loops we need to incorporate cpo-enrichment into this framework, and for example having a classical $\perp$ arrow will cause problems with coproducts. However, when we proceed into our full quantum language, we will not require that $\perp$ be calssical (in fact it will necessarily not be) and so we will consider this no further.

Finally, perhaps an alternative route here could have been to just assume that V be a classical object in the sense of [9] i.e. an object together with abstract cloning and deleting maps satisfying cocommutativity and coassociativity (rather than specifically unfolding it,) but we chose our route for now as it fits in to the previous and next sections more smoothly. One disadvantage in the above approach is that we have lost the abstractness we had previously obtained. However the *quantum* semantics will remain fully abstract in this sense, and we have effectively had to construct the classical semantics on top of this in a concrete manner since the quantum semantics assumptions themselves do not provide suitable classical structure.

3.3.2 Towards Quantum Semantics in an SCCCB

The next issue then is to incorporate the quantum features into our above structure. Once again we define our classical state space $S = k^m.I$. We define the type of qubit in this category to be $I \oplus I$ and then note that the quantum system we have access to consists of q qubits, thus the space Q^q which we denote H. Thus the compound system containing both our classical space and quantum space is $X = S \otimes H$. Note that this is $k^m.I \otimes H \cong k^m.H$ and so this can also be viewed as the k^m-fold coproduct of H. This is explicitly representing the *quantum data + classical control* paradigm since the classical control is determined entirely by which of the k^m components of the coproduct we are in, and then in each component we have a space H representing the quantum part of the structure. Note currently our types of qubits and bits coincide, which seems unfortunate, but this will soon be remedied.

We can give semantics for classical update to the state, once again carrying the quantum part of the state as a silent passenger — if we have $f : S \to S$ from our classical denotation then we can run $f \otimes id$ as an endoarrow on the compound space X. (Note e.g. arithmetic expressions still only have type $S \to V$ as in the concrete case.) We can define the denotation of applying a unitary also easily (simply $id \otimes u$ where u represents the primitive unitary denotation acting on the whole quantum space H.) In a similar manner we can define measurements $H \to H$ applying the projector $q_j.p_j$ to the relevant qubit. In an SCCCB the object I represents the scalars, and so elements of I (i.e. mappings $I \to I$) can represent our probabilities for which we annotate our trees.

And this is where we run into our problem: the resulting denotation is not simply a mapping $k^m.H \to k^m.H$ representing a pure function on the compound system of the classical and quantum state spaces; but rather a probabilistic function of this nature — quantum measurements introduce probabalistic nondeterminism. Hence representing programs by endoarrows on $k^m.H$ is not going to work. The natural solution to this is for our programs to be represented as arrows $(S \otimes H) \to k.(S \otimes H)$ for some k — the values $\{1 \dots k\}$ represent the k different non-zero outcomes and the probability is encoded in each element of H as a monoidal scalar. This approach would most likely work if we didn't have while loops; but in the presence of while loops it causes a problem — the reason being if there is a measurement within a while loop then the type that the while loop is denoted by depends on the run-time content of that while loop. If the program is to branch for an undetermined amount of time depending on some quantum measurement variable, then the value of k in the above cannot be determined until run-time. The main solution then requires infinite coproducts. Requiring such entities in our category immediately leads us into infinite-dimensional vector spaces (an infinite coproduct in **FdHilb** is an infinite-dimensional vector space, but **FdHilb** is specifically the category of *finite* dimensional vector spaces.) This is certainly undesirable. The solution shall be to have an object FH that represents probability distributions over H and we then define our space X to be $S \otimes FH = k^m.FH$ effectively distributing the probability/nondeterminism *underneath* the classical structure. To make all of this more explicit, we turn to mixed states and Peter Selinger's CPM construction.

3.4 Preliminaries : Mixed States and the CPM(-)$^\oplus$ Construction

3.4.1 The Mixed State Formulation of Quantum Mechanics

In general quantum mechanics, the state is often not known but rather a *probability distribution* over states is, and we shall call such a distribution over states a *mixed state.* A typical mixed state then, is of the form $\{(\lambda_1, \phi_1), \dots, (\lambda_n\, \phi_n)\}$ where $\Sigma\lambda_i = 1$ and the probability that the system is actually in state ϕ_i is given by λ_i. Note that a mixed state is just a *perspective* — in particular, a particular quantum particle can be viewed as different mixed states depending on the observer (e.g. if observer A has measured ϕ to be in state $|0\rangle$ and thus knows the state; while observer B does not know this information and is in a uniform mixed state.) Measurements (of the whole system) then map mixed states to pure states (mixed states with $n = 1 = \lambda_1$.)

See the Quantum Computer Science lecture course notes [11] for further details of this section.

In the Hilbert space formulation of quantum mechanics, we can represent a mixed state as a *density matrix.* If we represent the pure state by a ray in a Hilbert space H then we represent the mixed state by a ray in $H \otimes H$, i.e. a matrix $H \rightarrow H$. Given a pure state u in H its density matrix in $H \otimes H$ is $u \otimes u_*$ with the latter symbol denoting conjugation. (Viewing u as a mapping $I \rightarrow H$ this is equivalent to the matrix $u.u^\dagger$ as a mapping $H \rightarrow H$.) Note that we lose no information in this representation, since $u \in H$ is uniquely determined by $u \otimes u_* \in H \otimes H$. Furthermore, probability distributions over pure (in fact in general potentially mixed) states are given by weighted sums after this construction. So a mixed state that could be in u or v with equal probability is represented by $\frac{1}{2}(u \otimes u_*) + \frac{1}{2}(v \otimes v_*)$. Note that some information *is* lost in this notation, for example if the state could be in either $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ or $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ with equal probability the density matrix is the same as that of being in the state $|0\rangle$ or $|1\rangle$ with equal probability. However, in terms of that which can be *observed* the two mixed states are the same (that is, in terms of the operations and measurements we can perform upon them.) Furthermore, mixed states generated in this way are precisely represented by positive matrices in $H \otimes H$ that have a trace of 1 (a trace of exactly one if the probabilities need to add up to 1; a trace of at most 1 if the probabilities need to sum to at most 1, as will soon be the case for us due to possibilities of nontermination.) We can define the action of operations on mixed states (mapping mixed states to mixed states) by noting that if we have $U : H \rightarrow H$ then the corresponding mixed state operation $H \otimes H \rightarrow H \otimes H$ is $A \mapsto UAU^\dagger$ (viewing our $H \otimes H$ matrices as linear maps $H \rightarrow H$ using the presence of compact closure in this particular concrete case.) Alternatively given a $U : H \rightarrow H$ we can define are equivalent operation on mixed states $H \otimes H \rightarrow H \otimes H$ by $U \otimes U_*$, the latter symbol representing conjugation (these two ideas are easily seen equivalent in the graphical language of abstract quantum mechanics, and we shall later give an exposition of this.) This tells us how measurements and unitary operators should act on mixed states.

A mapping from probability distributions to probability distributions can be represented as a *completely positive map* on H, i.e. a map from $H \otimes H^*$ to itself that preserves positivity (i.e. factorisation into $f.f^\dagger$) in a strong manner (in order to map representations of probability distributions onto representations of probability distributions. Thus, these maps will also be trace-decreasing, but we do not deal with this quite yet.)

The next step is to generalise these ideas to the abstract level.

3.4.2 The CPM Construction

It is easy to define a notion of positivity at the categorical level, simply if a map factorises into $h^\dagger.h$. Then using compact closure it is also possible to define what it means for an arrow $A \otimes A^* \rightarrow B \otimes B^*$ to be *completely positive*, and this is performed in [23]. Then [23] proceeds to construct, from any strongly compact closed category **C**, the category **CPM(C)** — a category whose objects are of type $A \otimes A^*$ and whose arrows are the completely positive maps. This category inherits the dagger compact closed structure from the category it is generated from.

Definition 3.4.1 *Given an SCCCB* ***C*** *we define* ***CPM(C)*** *to be the category whose objects are objects of* ***C*** *and whose arrows* $f : A \rightarrow B$ *are completely positive maps* $f : A \otimes A^* \rightarrow B \otimes B^*$ *in* ***C****. Note that we have a functor* $F : \boldsymbol{C} \rightarrow \boldsymbol{CPM(C)}$ *taking* $F : h \mapsto h \otimes h*$.

For various equivalent definitions of complete positivity for arrows in these categories, see [23]. Thus "elements" $I \to A \otimes A^*$ in **CPM(C)** correspond to the density operators, and general arrows to completely positive maps manipulating density operators; we get the above result in the case that $\mathbf{C} = \mathbf{FdHilb}$.

Theorem 3.4.2 *CPM(C) is a strongly compact closed category.*

The above result also comes from [23]. However, while **CPM(C)** inherits the strong compact closure from **C**, it does not inherit the additive structure, i.e. the biproducts (since the functor F squares, so fundamentally fails to be linear.) However, if the original category has the biproduct structure as described above, then it has addition (commutative monoid enrichment) and hence so does **CPM(C)** since addition preserves complete positivity. Hence we can construct the biproduct completion and form a strongly compact closed category with biproducts.

Definition 3.4.3 ***CPM(C)***$^{\oplus}$ *is then defined to be the biproduct completion of* ***CPM(C)*** *arising from the addition on completely positive maps inherited by the addition in* **C**. *Note then that, as above, objects of* ***CPM(C)***$^{\oplus}$ *are tuples of objects of* ***CPM(C)*** *and arrows are matrices.*

So objects of $\mathbf{CPM(C)}^{\oplus}$ intuitively consist of tuples of (probability distributions of) Hilbert spaces. We shall find this idea very useful, as will shortly be seen. It is important to note that the biproducts in $\mathbf{CPM(C)}^{\oplus}$ are "free" and do *not* correspond to biproducts in **C** (it is this that will provide us with a distinction between bits and qubits — the former is given by $I \oplus I$ in $\mathbf{CPM(C)}^{\oplus}$, i.e. $\langle I, I \rangle$ written as a tuple-object, and the latter is given by $I \oplus I$ in **C**, i.e. $\langle I \oplus I \rangle$ written as a tuple-object.)

Finally we note that we can inherit the tensor product into $\mathbf{CPM(C)}^{\oplus}$ acting on matrices in the same manner it does for concrete matrices in linear algebra and that the tensor and biproduct operators distribute over each other in the natural manner. In fact we obtain a full strongly compact closed category with biproducts, following results in [23]. For example, adjoints of matrices use the adjoints available in **CPM(C)** and also transpose, in the same way as concrete linear algebra. All of these interact in the required way once again inheriting from the compact closed structures etc. in **C** (and hence in **CPM(C)**) from our canonical inclusion functor $F : \mathbf{C} \to \mathbf{CPM(C)}^{\oplus}$.

We are now ready to use ideas of mixed states and the **CPM** construction together with our earlier comments to give a full abstract treatment of the programming language in question.

4 Abstracting the Full Quantum Language

In the concrete version of the quantum language above, our semantic function is a map into $\mathsf{DProb}(S \times H)$ where H is our quantum value space $H = Q^k$ and we have k quantum bits in our system. It should be noted that this discrete sub-probability distribution is a set of triples (probability, element of S, element of H). We might as well include all such elements of S here with corresponding probabilities zero. Hence, this distribution can be seen as a mapping from S to $\mathsf{DProb}(H)$ (the fact that these are *sub-distributions* here is important). Explicitly from a triple set A we obtain this function by $f(s) = \{(p, h) | (s, p, h) \in A\}$. Since we can assume S to be finite (a finite number of registers, each of which can take a finite number of values, as above) this is equivalently a set of k elements of $\mathsf{DProb}(H)$ where $k = |S|$. Each sub-probability distribution can be modeled as a density matrix $H \to H$ and so this can be seen as k elements of $H \to H = H \otimes H^*$, i.e.

a tuple of mixed states (cf. Peter Selinger's paper [24].) Hence the meaning of a program can be seen as a map into $k.(H \otimes H^*)$.

Our language, we recall, looks like

C ::= `var := aexp` | `skip` | C_1;C_2 | `if bexp then` C_1 `else` C_2 | `while bexp do` C| `apply uni to qvar, qvar` | `measure qvar in var`

built on top of the assumed primitives `aexp`, `bexp` and `uni`.

4.1 Semantic Preliminaries

As expected, we assume a biproduct strongly compact closed category $\mathbf{C}$, and shall perform our semantics in $\mathbf{CPM(C)}^{\oplus}$. In order to deal with while loops we will once again need cpo-enrichment, although not of $\mathbf{C}$ but of $\mathbf{CPM(C)}^{\oplus}$. We note that a sufficient (and indeed necessary) condition for cpo-enrichment of $\mathbf{CPM(C)}^{\oplus}$ is cpo-enrichment of $\mathbf{CPM(C)}$ (with the ordering inherited to matrices pointwise.) Furthermore we will require to specify the way in which the cpo-enrichment is coherent with respect to the SCCCB structure.

Let $\mathbf{C}$ be a sufficient SCCCB (with sufficiency to be defined shortly.) Within the category $\mathbf{C}$, we define the quantum part of our structure $H = Q^q$ where q is the number of qubits we have available to us and $Q = I \oplus I$. We construct the category $\mathbf{CPM(C)}^{\oplus}$ in the above described manner. As an object in $\mathbf{CPM(C)}^{\oplus}$ we define $V = k.I$ and $S = V^m = k^m.I$ where coproducts here are the free coproducts generated by the $(-)^{\oplus}$ construction, so V is a k-tuple of I and S is a k^m tuple of I (with each component representing a classical possibility.) We can then construct $F(H)$ as an object in $\mathbf{CPM(C)}^{\oplus}$ representing mixed states over H (so here F is the canonical functor embedding $\mathbf{C}$ into $\mathbf{CPM(C)}$ as above, and here we identify $\mathbf{CPM(C)}$ as a subcategory of $\mathbf{CPM(C)}^{\oplus}$ as singleton objects.) We then define X to be $S \otimes F(H)$, i.e. a classical component together with a mixed state component. Once again using distributivity this can be written $X = k^m.F(H)$, i.e. our state space consists of k^m probability distributions over quantum states, i.e. a probability distribution over quantum states for each classical possibility, as described intuitively above. Note here that we are using the *bi*products in a fundamental way — initially it makes sense to have the classical state space as the coproduct $k^m.I$ but in the presence of quantum superpositions/probability distributions we are using this as a *product*, encoding a tuple of multiple probability distributions that do indeed sum to 1.

To summarise, we firstly construct the type *qubit* in $\mathbf{C}$ by letting $Q = I \oplus I$ via biproducts in $\mathbf{C}$. We then use tensors in $\mathbf{C}$ to produce $H = Q^q$. We then map the whole thing to $\mathbf{CPM(C)}^{\oplus}$ using our functor. Finally we use biproducts in $\mathbf{CPM(C)}^{\oplus}$ (from the free biproduct completion, distinct from our qubit biproduct) to sum together the classical possibilities.

4.1.1 Classical Atomic Formula

Once again we assume primitive denotations $[\![a]\!] : S \to V$ and $[\![b]\!] : S \to (I \oplus I)$ that are classical in the above sense (note that this is naturally the free biproduct structure, i.e. the object $\langle I, I \rangle$ in the category.) Note that insisting that $[\![b]\!]$ be classical is saying that $[\![b]\!].q_i = q_j$, i.e. that $[\![b]\!]$ is a mapping from $i \in \{1, k^m\}$ to the $j \in \{\mathsf{tt}, \mathsf{ff}\}$, and so for the classical part of the structure we are reduced to the concrete case, as described above.

We note that $\mathbf{CPM(C)}^{\oplus}$ is an SCCCB and $S = k^m.I$ and so as above we can define a map $\Delta_S : S \to S{\otimes}S$ that acts as a copying operation in the classical subcategory. Given this map, we define an operation $\Delta : X \to X \otimes S$ for extracting the classical information from the overall state by setting $\Delta = \mathsf{sym}.(\Delta_S \otimes id_{F(H)})$ (we recall that $X = S \otimes F(H)$ and so here $\mathsf{sym} : S \otimes S \otimes F(H) \to S \otimes F(H) \otimes S.$). Note that then if $f : I \to X = S \otimes F(H)$ is of the form $q_i \otimes \phi$ then $\Delta.f = \Delta_S.q_i \otimes id.\phi = q_i \otimes q_i \otimes \phi = q_i \otimes f$ as required, so this does indeed extract the classical component. Note here we have implicitly used the isomorphism $I \cong I \otimes I$ without mentioning it, and we will continue to do so (along with natural associativity morphisms etc.) for readability.

We firstly look at updating a variable in this new setting. Given a variable $v \in \{1 \ldots m\}$ and classical $a : S \to V$ we define $[v \mapsto a] : S \to S$ as follows. We firstly clone the classical information with $\Delta_S : S \to S \otimes S$. We then apply $id \otimes a$ taking us to $S \otimes V$, the old state space and the new variable. Now S is $V^m = V \otimes \ldots \otimes V$ and by using symmetry isomorphisms we assume that the final V is the variable we wish to change. Thus we then apply $id \otimes \ldots \otimes id \otimes f$ where $f : V \otimes V \to V$ is effectively the right projection, forgetting the old value and remembering the new one instead. To give this projection explicitly we note that $V \otimes V = k.I \otimes V \cong k.V$ and so we perform this isomorphism and then run $[id, \ldots, id] : k.V \to V$ forgetting the old value.

$$\begin{tikzcd}
\mathrm{S} \arrow[r, "\Delta_S"] & \mathrm{S} \otimes S \arrow[r, "id \otimes a"] & \mathrm{S} \otimes V \arrow[d, "id \otimes \ldots \otimes id \otimes f"'] \\
& & \mathrm{S}
\end{tikzcd}$$

We can then extend this to a mapping $X \to X$ with $X = S \otimes F(H)$ by using $[v \mapsto a] \otimes id$.

Once again we shall exploit distributivity (regarding $\otimes$ and $\oplus$) in order to represent the conditionals present in the if statement and while loop. We go from X to $X \otimes S$ using Δ and then to $X \otimes (I \oplus I)$ by applying our Boolean map, which is isomorphic to $X \oplus X$ and we can then use $[C_1, C_2]$ to branch on which possibility we're in leaving us in X again. Once again we shall use these ideas together with cpo-enrichment to represent *while*.

4.1.2 Quantum Atomic Formula

We shall also assume primitive meanings of unitaries aas unitary arrows $u : Q^2 \to Q^2$. Given such a u together with quantum variables x and y we define $\mathsf{app}_{u,x,y} : H \to H$ as $perm^{-1}_{x,y}.(u \otimes id_{Q^{q-2}}).\mathsf{perm}_{x,y}$ where $\mathsf{perm}_{x,y} : Q^q \to Q^q$ is some isomorphism made from the monoidal natural isomorphisms that reorders the qubits, sending the qubit in location x to the first component of the giant product, and y to the second. We then use $id \otimes F(\mathsf{app}_{u,x,y}) : S \otimes F(H) \to S \otimes F(H)$ which updates the quantum part (lifted to a mixed state level via F) and leaves the classical part as it is.

We can similarly define the meaning of projectors. We can clearly define a map $P^i_j : H \to H$ applying the computational basis projection $q_j.p_j$ to the ith qubit, and from this a map $X \to X$ exactly as above. As mentioned before we will represent probabilities for measurements as scalars, i.e. endoarrows on the monoidal identity $I \to I$.

4.1.3 Scalars

Definition 4.1.1 *A scalar $s : I \to I$ in $\mathbf{C}$ is* positive *if it is of the form $h^\dagger.h$ for some $h : I \to A$.*

For scalar arithmetic we will need to perform there will be various requirements we will need regarding the scalars in our category — it is these non-zero positive scalars that shall represent our probabilities and we need to be able to do some arithmetic with them (e.g. normalisation.) These requirements then are that the nonzero positive scalars are closed under addition and admit nonzero positive square-roots and inverses. The square root of a scalar h is some scalar s such that $h = s.s$. We also require that (on positive scalars) addition and multiplication are monotonic with respect to $\sqsubseteq$. Assuming these, we also have the following facts:

Firstly, for a positive scalar s we have $s^\dagger = s$ since $s^\dagger = (h^\dagger.h)^\dagger = h^\dagger.h^{\dagger\dagger} = h^\dagger.h = s$.

Secondly, positive scalars are closed under multiplication. This is clear since if a and b are positive then $a.b = r.r.s.s = r.r^\dagger.s.s^\dagger = r.s.s^\dagger.r^\dagger = (s.r)^\dagger.(s.r)$. Note in particular if a scalar has a (positive) square root then it must be positive.

Thirdly, if a and b are positive scalars then $a + b \sqsupseteq a$. This is clear since $b \sqsupseteq 0$ and so by monotonicity $a + b \sqsupseteq a + 0 = a$.

We now need to make a note about embedding probabilities into our CPM category. Given a positive scalar $s : I \to I$ in $\mathbf{C}$ we can embed it into a scalar in $\mathbf{CPM(C)}$ (i.e. a completely positive arrow $I \otimes I^* \to I \otimes I^*$) in two ways. Firstly we can exploit the isomorphism $I \to I \otimes I^*$ as follows:

$$\begin{tikzcd}
\mathrm{I} \arrow[r, "s"] & \mathrm{I} \\
\mathrm{I} \otimes I^* \arrow[u, "\cong"] & \mathrm{I} \otimes I^* \arrow[u, "\cong"]
\end{tikzcd}$$

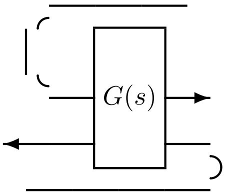
Thus, given any scalar $s : I \to I$ in $\mathbf{C}$ we write $G(s) : I \otimes I^* \to I \otimes I^* = F(I) \to F(I)$ for the above arrow in $\mathbf{CPM(C)}$. We need to check that for all positive s, $G(s)$ is indeed completely positive and hence a valid arrow (scalar) in $\mathbf{CPM(C)}$.

Proposition 4.1.2 *For any positive scalar s, $G(s)$ is completely positive.*

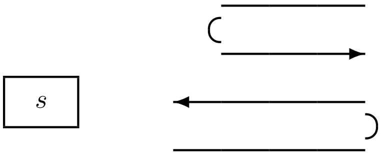
Proof To do this we firstly note that $G(s) = s \bullet id_{I \otimes I^*}$ using scalar multiplication. This is asserting commutativity of the diagram

$$\begin{tikzcd}
\mathrm{I} \arrow[r, "s"] & \mathrm{I} \\
\mathrm{I} \otimes I^* \arrow[u, "\cong"] \arrow[d, "\cong"] & \mathrm{I} \otimes I^* \arrow[u, "\cong"'] \arrow[d, "\cong"'] \\
\mathrm{I} \otimes I^* \otimes I \arrow[r, "id \otimes s"] & \mathrm{I} \otimes I^* \otimes I
\end{tikzcd}$$

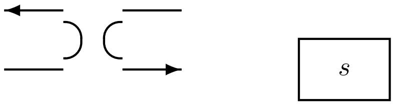
which clearly holds by naturality of the monoidal unit. To show that $G(s)$ is indeed completely positive we exploit Corollary 4.13(b) of [23]. This states that a map $f : A \otimes A^* \to B \otimes B^*$ is completely positive iff $(id_{A^* \otimes B} \otimes \epsilon_B).(id_A \otimes f \otimes id_B).(\eta_A \otimes id_{A^* \otimes B})$ is a positive arrow (in the sense of factoring into $h^\dagger.h$ as above.) In the diagrammatic notion of [23] or [10] we require that the following is positive:



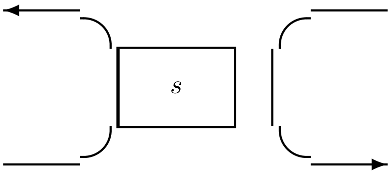
Well noting the above that $G(s) = s \bullet id_{I \otimes I^*}$ we find that $(id_{A^* \otimes B} \otimes \epsilon_B).(id_A \otimes G(s) \otimes id_B).(\eta_A \otimes id_{A^* \otimes B}) = s \bullet (id_{I^* \otimes I} \otimes \epsilon_I).(id_I \otimes id \otimes id_I).(\eta_I \otimes id_{I^* \otimes I})$ (exploiting the fact that we can move scalars around in time and space, here working in the category $\mathbf{C}$.)



This simplifies immediately to $s \bullet (id_{I^* \otimes I} \otimes \epsilon_I).(\eta_I \otimes id_{I^* \otimes I})$ which again by reasoning in a diagrammatic language can be simplified to $s \bullet \epsilon_I.\eta_I$



By properties of scalars this is $\epsilon_I.s.\eta_I = \epsilon_I.h^\dagger.h.\eta_I = (h.\eta_I)^\dagger.(h.\eta_I)$ as required.



Hence $G(s)$ is indeed a completely positive map, and so it is a valid scalar in **CPM(C)**.

□

The second way of embedding scalars into our CPM category is to exploit the functor F by considering $F(s) = s \otimes s_* : F(I) \to F(I)$ as above and in [23]. Note then we can use either F or G to map scalars in **C** to scalars in **CPM(C)** (and hence also to scalars in $\mathbf{CPM(C)}^{\oplus}$ by identifying **CPM(C)** with the full subcategory of $\mathbf{CPM(C)}^{\oplus}$ consisting of 1-tuple objects.)

We now turn to ideas of square roots.

Proposition 4.1.3 *If $s = r.r$ for positive scalars s and r then $F(r) = G(s)$*

Proof We firstly note that the following diagram commutes by compact closure.

$$\begin{tikzcd}
\mathrm{I} \arrow[r, "s"] & \mathrm{I} \\
\mathrm{I}^* \arrow[u, "\cong"] \arrow[r, "s^*"] & \mathrm{I}^* \arrow[u, "\cong"']
\end{tikzcd}$$

Secondly we show that for any positive scalar s, $s^* = s_*$. If s is positive then so is s^* since $s^* = (h^\dagger.h)^* = h^*.(h^*)^\dagger$. Then we know by our comment above that $s^* = s^{*\dagger} = s_*$ as required.

Finally we use the above to prove our result $G(r.r) = F(r)$ with commutativity of the following diagram (the middle box using properties of scalars.) $G(s)$ is given by the clockwise circuit from the southwestmost object to the southeastmost object.

$$\begin{tikzcd}
\mathrm{I} \arrow[r, "s"] \arrow[d, equal] & \mathrm{I} \arrow[d, equal] \\
\mathrm{I} \arrow[r, "r.r"] & \mathrm{I} \\
\mathrm{I} \otimes I \arrow[u, "\cong"] \arrow[r, "r \otimes r"] & \mathrm{I} \otimes I \arrow[u, "\cong"] \\
\mathrm{I} \otimes I^* \arrow[u, "\cong"] \arrow[r, "r \otimes r_*"] & \mathrm{I} \otimes I^* \arrow[u, "\cong"]
\end{tikzcd}$$

□

Hence if s admits a positive square root, we have $F(\sqrt{s}) = G(s)$.

We are now ready to explicitly state the assumptions we assume of our category in question.

4.2 The Semantic Category

Definition 4.2.1 *A* normalising strongly compact closed O-category with biproducts *(SCCOCB) is a strongly compact closed category with biproducts (in the sense of former definitions) such that **CPM(C)** is cpo-enriched. We require that composition and addition are continuous with respect to this ordering and* $0 = \bot$*. We also require that the nonzero positive scalars are closed under addition, multiplication (which we get for free,) inverses, square roots. Finally we require that additon and multiplication are monotonic with respect to* $\sqsubseteq$ *on the positive scalars.*

We pause to make a few notes. Firstly, we're going to be working mostly in the category $\mathbf{CPM(C)}^{\oplus}$. We note that this consists of tuple objects from **CPM(C)**, and that **CPM(C)** sits inside $\mathbf{CPM(C)}^{\oplus}$ as the full subcategory generated from the single-tuple objects. We have already seen that G sends positive scalars from **C** into positive scalars in **CPM(C)**, and in fact it's clear that the map G is surjective (and preserves addition etc. by naturality of the isomorphisms) and so positive scalars in **C** correspond precisely to the scalars in **CPM(C)**, which in turn is precisely the scalars in $\mathbf{CPM(C)}^{\oplus}$ (since the monoidal unit in this category is the single-tuple object $\langle I \rangle$.) Thus, we have all of the above structure of the scalars in the category $\mathbf{CPM(C)}^{\oplus}$.

In the abstract classical case we required that $\bot.\bot = \bot$ (and as such $\bot.a = \bot$) — we get this now for free since $\bot = 0$.

Also, given the ordering on **CPM(C)** we can extend it to the ordering on $\mathbf{CPM(C)}^{\oplus}$ just by componentwise comparison of arrows. It is clear then that pairing and copairing in the category $\mathbf{CPM(C)}^{\oplus}$ are continuous functions on homsets (using the diagonal lemma from [4].) Furthermore, we recall that we define composition in $\mathbf{CPM(C)}^{\oplus}$ from addition and composition in **CPM(C)**. Since addition and composition in

CPM(C) are continuous, it follows that composition in $\mathbf{CPM(C)}^{\oplus}$ is continuous. These ideas are of course needed to give the denotational meaning of a while loop.

Finally we note that we will eventually need to weaken our assumptions to a category where the ordering is only complete for *trace-decreasing* maps — but it makes sense to make these modifications afterwards and for now use the stronger assumption above (this stronger assumption holds for **Rel** but not for **FdHilb**.)

4.3 Denotational Semantics

We can now use this to define our denotational semantics (as arrows $X \to X$ in $\mathbf{CPM(C)}^{\oplus}$ where **C** is a normalising SCCOCB) as follows:

$$\begin{array}{lcl}
\mathcal{D}[\![\texttt{skip}]\!] & = & id_X \\
\mathcal{D}[\![C_1; C_2]\!] & = & \mathcal{D}[\![C_2]\!].\mathcal{D}[\![C_1]\!] \\
\mathcal{D}[\![v := e]\!] & = & [v \mapsto [\![e]\!]] \otimes id \\
\mathcal{D}[\![\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2]\!] & = & [C_1, C_2].\mathsf{iso}.(id \otimes [\![b]\!]).\Delta \\
\mathcal{D}[\![\texttt{apply } u \texttt{ in } q_1, q_2]\!] & = & id \otimes F(\mathsf{app}_{u,x,y}) \\
\mathcal{D}[\![\texttt{measure } q \texttt{ in } v]\!] & = & ([v \mapsto 0] \otimes F(P^q_0)) + ([v \mapsto 1] \otimes F(P^q_1)) \\
\mathcal{D}[\![\text{while } b \text{ do } C]\!] & = & \mathsf{lfp}[\lambda f : X \to X.([f.\mathcal{D}[\![C]\!], id].\mathsf{iso}.(id \otimes [\![b]\!]).\Delta]
\end{array}$$

In the `if` case the isomorphism is the natural one from $X \otimes (I \oplus I) \to X \oplus X$ that we have in any SCCCB.

In the while case we use the cpo-enrichment and note that this is a continuous function, since it can be written as a combination of copair, comp and constant functions once again as in the classical abstract case; and these functions are continuous by reasoning in the previous section.

In the measurement case $P^i_j : H \to H$ applies $q_j.\pi_j$ to the ith qubit in parallel $\otimes$ with identity operations on the other qubits. Note that perhaps surprisingly we do not deal with the probability scalings here, and neither do we deal with normalisation. This is because, quite pleasantly, they cancel out in our structure, as will be seen. It is fortunate this is the case — otherwise we would have internalisation issues (e.g. representing normalisation inside the category, which requires having an arrow calculating square roots, which could be a problem since all of our maps have to be linear since composition and addition must commute.) An alternative equivalent perspective here is that we are normalising but not to 1 but rather to the "probability at this point in the tree" as the ideas of [24].

Once again, from the denotational semantics we can derive $\mathcal{D}'[\![C]\!]$:$\mathbf{CPM(C)}^{\oplus}(I, S)\times$ $\mathbf{C}(I, H) \to \mathbf{CPM(C)}^{\oplus}(I, X)$ by setting $\mathcal{D}'[\![C]\!](s, \phi) = \mathcal{D}[\![C]\!].(s \otimes F(\phi))$, taking a classical arrow and a quantum state and outputting the resulting probability distribution after C is applied.

4.4 Operational Semantics

We now give operational semantics in this domain. A configuration, then, is a combination (C, s, ϕ) where $s : I \to S$ is a classical arrow (i.e. q_i for some $i \leq k^m$) in $\mathbf{CPM(C)}^{\oplus}$, and $\phi : I \to H$ in the category **C**.

Once again we define a one-step relation $\to$ between configurations, with each relation tagged with a probability, i.e. a positive scalar $I \to I$ in the category **C**. Also it once again must be the case that the trees generated from this relation is finitely branching, and indeed they are — our reduction rules are as follows:

$$(v := e,\, s,\, \phi) \to^1 (\texttt{skip},\, [v \mapsto [\![e]\!]].s,\, \phi)$$

$$(\texttt{skip};C,\, s,\, \phi) \rightarrow^{1} (C,\, s,\, \phi)$$

$$\frac{(C_1,\, s,\, \phi) \rightarrow^{p} (C_1',\, s',\phi')}{(C_1;\, C_2,\, s,\, \phi) \rightarrow^{p} (C_1';\, C_2,\, s',\, \phi')}$$

$$\frac{[\![b]\!].s = q_1}{(\texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2,\, s,\, \phi) \rightarrow^{1} (C_1,\, s,\, \phi)}$$

$$\frac{[\![b]\!].s = q_2}{(\texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2,\, s,\, \phi) \rightarrow^{1} (C_2,\, s,\, \phi)}$$

$$\frac{[\![b]\!].s = q_1}{(\texttt{while}\ b\ \texttt{do}\ C,\, s,\, \phi) \rightarrow^{1} (C;\, \text{while}\ b\ \text{do}\ C,\, s,\, \phi)}$$

$$\frac{[\![b]\!].s = q_2}{(\texttt{while}\ b\ \texttt{do}\ C,\, s,\, \phi) \rightarrow^{1} (\texttt{skip},\, s,\, \phi)}$$

$$(\texttt{apply}\ U\ \texttt{to}\ q_1, q_2,\, s,\, \phi) \rightarrow^{1} (\texttt{skip},\, s,\, \mathsf{app}_{u,q_1,q_2}.\phi)$$

$$(\texttt{measure}\ q\ \texttt{in}\ v,\, s,\, \phi) \rightarrow^{p_0^q(\phi)} (\texttt{skip},\, [v \mapsto 0]\,.s,\, \sqrt{p_0^q(\phi)^{-1}} \bullet P_q^0.\phi)$$

$$(\texttt{measure}\ q\ \texttt{in}\ v,\, s,\, \phi) \rightarrow^{p_1^q(\phi)} (\texttt{skip},\, [v \mapsto 1]\,.s,\, \sqrt{p_1^q(\phi)^{-1}} \bullet P_q^1.\phi)$$

Measurements work as follows: once again we define projections $P_j : Q \rightarrow Q = q_j.\pi_j$ and extend this to $P_j^i : H \rightarrow H$ acting on qubit i in the natural way. For $\phi : I \rightarrow H$ we can define the scalar $p_j^i(\phi)$ to be $\phi^\dagger.P_j^i.\phi$. Finally we note that the result of the measurement needs to be *normalised*, i.e. divided by the square root of the probability as in the concrete case. We will shortly see that this does indeed perform normalisation. We will embed the probabilities into $\mathbf{CPM(C)}$ using G and the fact that $G(p_j^i(\phi)).F(\sqrt{p_j^i(\phi)^{-1}}\bullet P_j^i.\phi) = F(P_j^i.\phi)$ representing the idea that *probabilities and normalisation canceling out* as mentioned above.

Once again by taking the reflexive transitive closure of our relation we obtain reduction trees labeled with probabilities, looking exactly as in the concrete quantum case. We now need to use these operational semantics to form the semantic function, as above. This is where we use the **CPM** construction, for creating probabilistic weightings of states.

Given a command C and states $s, s' : I \rightarrow S$ and $\phi, \phi' : I \rightarrow H$ we define $\mathsf{Comp}(C, s, s', \phi, \phi')$ to be the set of all reductions $(C, s, \phi) \rightarrow^{p_1}\rightarrow \ldots \rightarrow^{p_n} (\texttt{skip}, s', \phi')$ and given such a $c \in \mathsf{Comp}(C, s, s', \phi, \phi')$ we define $p(c)$ to be $p_1.\ldots.p_n$ c.f. multiplication (note that multiplication of scalars $p : I \rightarrow I$ are commutative.)

We then define Prob :$\mathbf{CPM(C)}^{\oplus}(I, S)\times\mathbf{CPM(C)}^{\oplus}(I, S)\times\mathbf{C}(I, H)\times\mathbf{C}(I, H) \rightarrow \mathbf{CPM(C)}^{\oplus}(I, I)$ by $\mathsf{Prob}(s, s', \phi, \phi') = G(\Sigma\,\{p(c)|c \in \mathsf{Comp}(C, s, s', \phi, \phi')\})$. We then define $\mathcal{O}[\![C]\!](s, \phi) = \Sigma_{s',\phi'}(\mathsf{Prob}(s, s', \phi, \phi').(s'\otimes F(\phi'))$ using addition from the biproduct, summing over all (s', ϕ') such that $(C, s, \phi)\overset{\rightarrow}{\rightarrow}(\texttt{skip}, s', \phi')$. Note that if (C, s, ϕ) never terminates to any solution this summation results in the zero distribution, i.e. the $\perp$ element. This gives us $\mathcal{O}[\![C]\!]$:$\mathbf{CPM(C)}^{\oplus}(I, S)\times\mathbf{C}(I, H) \rightarrow\mathbf{CPM(C)}^{\oplus}(I, X)$.

Note that in passing from scalars in $\mathbf{C}$ to scalars in $\mathbf{CPM(C)}$ we *do not* square the scalars (we use the functor G rather than F.) This is more natural (since G preserves addition while F does not) and allows normalisation and probability scalings to cancel out (as in Peter Sellinger's paper [24]) which is vital due to internalisation issues. We need to show that the probabilities with which we annotate are indeed positive, so that $G(s)$ is a valid completely positive arrow in $\mathbf{CPM(C)}$.

We note immediately that the identity is positive ($id = id.id = id.id^\dagger$) and that the p^i_j from above are indeed positive. To see this we note that $p^i_j(\phi) = \phi^\dagger.P^i_j.\phi = \phi^\dagger.(id \otimes \ldots \otimes (q_j.\pi_j) \otimes \ldots \otimes id).\phi = \phi^\dagger.(id \otimes \ldots \otimes q_j \otimes \ldots \otimes id).(id \otimes \ldots \otimes \pi_j \otimes \ldots \otimes id).\phi = \phi^\dagger.(id \otimes \ldots \otimes \pi^\dagger_j \otimes \ldots \otimes id).(id \otimes \ldots \otimes \pi_j \otimes \ldots \otimes id).\phi = \phi^\dagger.(id \otimes \ldots \otimes \pi_j \otimes \ldots \otimes id)^\dagger.(id \otimes \ldots \otimes \pi_j \otimes \ldots \otimes id).\phi = (id \otimes \ldots \otimes \pi_j \otimes \ldots \otimes id).\phi)^\dagger.((id \otimes \ldots \otimes \pi_j \otimes \ldots \otimes id).\phi)$ as required.

We now return to our definition of $\mathcal{O}[\![C]\!]$ above and seek to refine it — as in the concrete case, there is a chance that e.g. $\mathsf{Comp}(C, s, s', \phi, \phi')$ could be infinite; or indeed that the number of states (s', ϕ') reachable from (s, ϕ) is infinite (since while our classical state space is finite, certainly our quantum space will not be.) In the concrete case, the probability is capped by 1 by our definition of the operational semantics, but in the abstract case the fact that this converges is not as apparent. However we can define this infinite sum if we wish using the cpo-structure by taking the least upper bound of the partial approximates.

Explicitly, we once again define $\mathsf{Comp}_n(C, s, s', \phi, \phi')$ as the set of reduction paths from (C, s, ϕ) to ($\mathtt{skip}$,s',ϕ') with at length at most n, and if c is such a path we write $p(c)$ for the probability weighting of that path, i.e. the product of all of the probabilities along this path (given by composition of scalars, which we know to be commutative and associative.) We then define $\mathcal{O}[\![C]\!]_n$ as a function from $\mathbf{CPM(C)}^\oplus(I, S)\times$ $\mathbf{C}(I, H)$ to $\mathbf{CPM(C)}^\oplus(I, X)$. To define this we firstly define $\mathsf{Prob}_n(C, s, s', \phi, \phi') = G(\Sigma\,\{p(c)|c \in \mathsf{Comp}_n(C, s, s', \phi, \phi')\})$ — we note that this is a summation using the biproduct structure, and furthermore that this is a finite sum since $\mathsf{Comp}_n(C, s, s', \phi, \phi')$ is guaranteed to be finite since the reduction tree we have will be a finitely branching one — in fact each node has at most two direct children, so this set will be bounded by 2^n. This fact also shows us that there will only be finitely many s', ϕ' such that $(C, s, \phi)\vec{\rightarrow}(\mathtt{skip}, s', \phi')$ in at most n steps. Hence we define

$$\mathcal{O}[\![C]\!]_n(s, \phi) = \Sigma\,\{\mathsf{Prob}_n(C, s, s', \phi, \phi').(s' \otimes F(\phi'))|(C, s, \phi)\vec{\rightarrow}(\,\mathtt{skip}, s', \phi')\}$$

summing over such (s', ϕ'). We note that functions of this kind (from $\mathbf{CPM(C)}^\oplus(I, S)\times$ $\mathbf{C}(I, H)$ to $\mathbf{CPM(C)}^\oplus(I, X)$ in fact forms a domain, with pointwise ordering (since our underlying *codomain* hom-sets are domains.) Hence we define $\mathcal{O}[\![C]\!]$ to be the least upper bound of the chain $(\mathcal{O}[\![C]\!]_n)_n$ (or equivalently just set $\mathcal{O}[\![C]\!](s, \phi) = \bigsqcup(\mathcal{O}[\![C]\!]_n(s, \phi))$ as before. For this to make sense though we need to check that this is indeed a chain.

To do this, we need to show that $\mathcal{O}[\![C]\!]_{n+1}(s, \phi) \sqsupseteq \mathcal{O}[\![C]\!]_n(s, \phi)$. Since we know that

$$\mathsf{Comp}_n(C, s, s', \phi, \phi') \subseteq \mathsf{Comp}_{n+1}(C, s, s', \phi, \phi')$$

this amounts to requiring that if p is a probability (arising from a product of probabilities from the reduction tree) then $a + p \sqsupseteq p$ in the cpo-structure ordering. Fortunately we have this since a and p are guaranteed to be positive and we have our monotonicity assumption of $+$.

Thus we have once again $\mathcal{O}[\![C]\!]$ and $\mathcal{D}'[\![C]\!]$ as mappings $\mathbf{CPM(C)}^\oplus(I, S)\times$ $\mathbf{C}(I, H) \rightarrow \mathbf{CPM(C)}^\oplus(I, X)$ and once again we can seek to show that they conincide.

4.5 Correspondence

Once again our correspondence result is

Theorem 4.5.1 *For any command C, $\mathcal{D}'[\![C]\!](s,\phi) = \mathcal{O}[\![C]\!](s,\phi)$ for all ϕ and for classical s.*

We adapt and combine the proofs of the previous sections (concrete + nondeterministic and abstract + deterministic) to arrive at this conclusion. Note that if we consider the domains of $\mathcal{D}'[\![C]\!]$ and $\mathcal{O}[\![C]\!]$ to only consider classical arrows, again this amounts to $\mathcal{O}[\![C]\!] = \mathcal{D}'[\![C]\!]$

Proof Once again we prove this by induction on C.

In the case that $C = \mathtt{skip}$ once again $\mathcal{D}'[\![C]\!](s,\phi) = \mathcal{D}[\![C]\!].(s \otimes F(\phi)) = id.(s \otimes F(\phi)) = s \otimes F(\phi)$. Also $(C, s, \phi)\overrightarrow{\to}^1(\mathtt{skip}, s, \phi)$ and so $\mathsf{Comp}(C, s, s', \phi, \phi')$ is nonempty only if $s = s'$ and $\phi = \phi'$ and the set $\mathsf{Comp}(C, s, s, \phi, \phi)$ contains a single empty computation branch whose probability is 1, the empty product. As such $\mathsf{Prob}(s, s, \phi, \phi) = 1 = id$ and so $\mathcal{O}[\![C]\!](s) = 1.(s \otimes F(\phi)) = s \otimes F(\phi) = \mathcal{D}'[\![C]\!](s,\phi)$ as required.

We secondly consider the case $C = C_1; C_2$. We need to show that $\mathcal{O}[\![C]\!](s,\phi) = \mathcal{D}'[\![C]\!](s,\phi)$. By definition the right hand side is $\mathcal{D}[\![C]\!].(s \otimes F(\phi)) = \mathcal{D}[\![C_2]\!].\mathcal{D}[\![C_1]\!].(s \otimes F(\phi)) = \mathcal{D}[\![C_2]\!].\mathcal{D}'[\![C_1]\!](s,\phi) = \mathcal{D}[\![C_2]\!].\mathcal{O}[\![C_1]\!](s,\phi)$ by inductive hypothesis. Hence it suffices to show that $\mathcal{O}[\![C]\!](s,\phi) = \mathcal{D}[\![C_2]\!].\mathcal{O}[\![C_1]\!](s,\phi)$.

Let (C_1, s, ϕ) reduce to $(\mathtt{skip}, s_1, \phi_1) \dots (\mathtt{skip}, s_n, \phi_n)$ with probabilities $p_1 \dots p_n$ (note that this covers all cases since if (C_1, s, ϕ) does not terminate then $n = 0$. In this case our final probability is the empty sum 0, which coincides with $\perp$.) Then $\mathcal{O}[\![C_1]\!](s,\phi) = \Sigma(p_i.(s_i \otimes F(\phi_i)))$. Hence our expression above is $\mathcal{D}[\![C_2]\!].\Sigma(p_i.(s_i \otimes F(\phi_i)))$ which is $\Sigma(p_i.\mathcal{D}[\![C_2]\!].(s_i \otimes F(\phi_i)))$ by distributivity and properties of scalars. This is $\Sigma(p_i.\mathcal{D}'[\![C_2]\!].(s_i, \phi_i))$ which is once again $\Sigma(p_i.\mathcal{O}[\![C_2]\!].(s_i, \phi_i))$ by inductive hypothesis. Hence it suffices to show that $\mathcal{O}[\![C]\!](s,\phi)$ is this expression.

Well let (C_2, s_i, ϕ_i) reduce to $(\mathtt{skip}, s_{1i}, \phi_{1i}) \dots (\mathtt{skip}, s_{m_i i}, \phi_{m_i i})$ with probabilities $p_{1i}, \dots, p_{m_i i}$. We note that then by studying the operational semantics that (C, s) reduces to

$$(s_{11}, \phi_{11}) \dots, (s_{m_1 1}, \phi_{m_1 1}), \dots, (s_{1n}, \phi_{1n}), \dots, (s_{m_n n}, \phi_{m_n n})$$

with probabilities

$$p_{11}.p_1, \dots, p_{m_1 1}.p_1, \dots, p_{1n}.p_n, \dots, p_{m_n n}.p_n$$

It follows then that $\mathcal{O}[\![C]\!](s,\phi) = \Sigma\Sigma p_{ij}.p_j.(s_{ij} \otimes F(\phi_{ij})$. By definition, this is $\Sigma(p_i.\mathcal{O}[\![C_2]\!].(s_i, \phi_i))$ as required. Note that the above can be extended to the infinite case just using an infinite sum here in the sense of the above (using continuity together with finite distributivity to get infinite distributivity, etc.)

In the case that $C =$ v := e then we need to show that $\mathcal{O}[\![C]\!](s,\phi) = \mathcal{D}'[\![C]\!](s,\phi) = ([v \mapsto [\![e]\!]] \otimes id).(s \otimes F(\phi)) = [v \mapsto [\![e]\!]]\, .s \otimes F(\phi)$. This clearly holds since (C, s, ϕ) reduces only to $(\mathtt{skip}[v \mapsto [\![e]\!]]\, .s, \phi)$ with probability 1.

The case $C = \mathtt{apply}\ u\ \mathtt{to}\ q_1, q_2$ is similar. We need to show that $\mathcal{O}[\![C]\!](s,\phi) = \mathcal{D}'[\![C]\!](s,\phi) = (id \otimes F(app_{u,x,y})).(s \otimes F(\phi)) = s \otimes F(\mathsf{app}_{u,x,y}).F(\phi) = s \otimes F(\mathsf{app}_{u,x,y}.\phi)$. This does indeed hold again by definition of operational semantics since (C, s, ϕ) reduces uniquely to $(\mathtt{skip},\ s,\ \mathsf{app}_{u,x,y}.\phi)$

We now consider measurement $C = \text{measure } q \text{ in } v$. Firstly $\mathcal{D}'[\![C]\!](s,\phi) = \Sigma_j([v \mapsto j] \otimes F(P^q_j)).(s \otimes F(\phi)) = \Sigma_j([v \mapsto j]\, .s \otimes F(P^q_j.\phi))$. Secondly $\mathcal{O}[\![C]\!](s,\phi) = \Sigma_j(G(p^q_j(\phi)) \bullet ([v \mapsto j]\, .s \otimes F(\sqrt{p^q_j(\phi)}^{-1} \bullet P^q_j.\phi))) = \Sigma_j(G(p^q_j(\phi)) \bullet ([v \mapsto j]\, .s \otimes F(\sqrt{p^q_j(\phi)}^{-1}) \bullet F(P^q_j.\phi))) = \Sigma_j(G(p^q_j(\phi)).F(\sqrt{p^q_j(\phi)}^{-1}) \bullet ([v \mapsto j]\, .s \otimes F(P^q_j.\phi)))$. It remains only to show that $G(p^q_j(\phi)).F(\sqrt{p^q_j(\phi)})^{-1} = 1$ (i.e. that the probabilistic weight and the normalisation do indeed cancel out) and we have in fact already done this, since we have already commented that $G(s) = F(\sqrt{s})$ and so $G(p^q_j(\phi)).F(\sqrt{p^q_j(\phi)})^{-1} = F(\sqrt{p^q_j(\phi)}.F(\sqrt{p^q_j(\phi)})^{-1} = id$, as required.

We now consider the conditional case, with $C = \texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2$. We note that $(C,s,\phi) \rightarrow^1 (C_1,s,\phi)$ in the case that $[\![b]\!].s = q_1$ and to (C_2,s,ϕ) in the case that $[\![b]\!].s = q_2$. It follows then that $\mathcal{O}[\![C]\!](s,\phi) = \mathcal{O}[\![C_i]\!](s,\phi)$ if $[\![b]\!].s = q_i$ as in the concrete case above. This is $\mathcal{D}'[\![C_i]\!](s,\phi)$ by inductive hypothesis, which is $\mathcal{D}[\![C_i]\!].(s \otimes F(\phi))$ by definition. Hence it suffices to show that $\mathcal{D}[\![C]\!].(s \otimes F(\phi)) = \mathcal{D}[\![C_i]\!].(s \otimes F(\phi))$, as before.

Well we know the left hand side is $[C_1,C_2].\mathsf{iso}.(id \otimes [\![b]\!]).\Delta.(s \otimes F(\phi))$. We firstly recall our above note that $\Delta.(s \otimes F(\phi)) = (s \otimes F(\phi) \otimes s)$ since s is classical, and so in fact the left hand side is $[C_1,C_2].\mathsf{iso}.(s \otimes F(\phi) \otimes [\![b]\!].s)$ $= [C_1,C_2].iso.(s \otimes F(\phi) \otimes q_i)$ by assumptions on $[\![b]\!]$. We claim that $\mathsf{iso}.(s \otimes F(\phi) \otimes q_i) = q_i.(s \otimes F(\phi))$. Then $\mathcal{D}[\![C]\!].(s \otimes F(\phi)) = [\mathcal{D}[\![C_1]\!], \mathcal{D}[\![C_2]\!]].q_i.(s \otimes F(\phi)) = \mathcal{D}[\![C_i]\!].(s \otimes F(\phi)) = \mathcal{D}'[\![C_i]\!](s \otimes F(\phi))$ as required.

To see that $\mathsf{iso}.(s \otimes F(\phi) \otimes q_i) = q_i.(s \otimes F(\phi))$ we write r for $s \otimes F(\phi)$ and seek to show that $\mathsf{iso}.(r \otimes q_i) = q_i.r$.

Firstly, we note that the following diagram commutes by naturality/coherence:

$$\begin{tikzcd}
\mathrm{I} \arrow[r, "\cong"] \arrow[d, "\Delta"'] & \mathrm{I} \otimes I \arrow[d, "id \otimes \Delta"'] \\
\mathrm{I} \oplus I \arrow[r, "\cong"] \arrow[d, "\cong"'] & \mathrm{I} \otimes (I \oplus I) \\
(\mathrm{I} \otimes I) \oplus (I \otimes I) \arrow[ur, "\cong"] &
\end{tikzcd}$$

Note that we use Δ here for the codiagonal $A \to A \oplus A$ rather than the classical cloning operation we have above.

With this in mind (and the fact that $(r \oplus r).\Delta = \Delta.r$,) the diagram below shows as required that $\mathsf{iso}.(r \otimes q_i) = q_i.r$ (clockwise is RHS, anticlockwise is LHS) exploiting distributivity isomorphisms (in the case q_1 wlog). (We use here the fact that $q_0 = (1 \oplus 0).\Delta$ where here $\Delta : I \to I \oplus I$ and similarly $q_1 = (0 \oplus 1).\Delta$. To see this we note that $\Delta = \langle id, id \rangle$ and so $(0 \oplus 1).\langle id, id \rangle = \langle 0, 1 \rangle = q_1$ by biproduct equations.)

$$\begin{tikzcd}
 & \mathrm{I} \arrow[d, "\cong"'] \arrow[r, phantom, "="] & \mathrm{I} \arrow[dd, "r"'] \\
 & \mathrm{I} \otimes I \arrow[dl, "r \otimes q_1"'] \arrow[d, "id \otimes \Delta"'] & \\
\mathrm{X} \otimes (I \oplus I) \arrow[d, "\cong"'] & \mathrm{I} \otimes (I \oplus I) \arrow[l, "r \otimes (0 \oplus 1)"'] \arrow[d, "\cong"'] & \mathrm{X} \arrow[ddd, "\Delta"'] \\
(\mathrm{X} \otimes I) \oplus (X \otimes I) \arrow[d, "\cong"'] & (\mathrm{I} \otimes I) \oplus (I \otimes I) \arrow[l, "r \otimes 0 \oplus r \otimes 1"'] \arrow[d, "\cong"'] & \\
\mathrm{X} \oplus X & \mathrm{I} \oplus I \arrow[l, "0 \oplus r"'] \arrow[d, "r \oplus r"'] & \\
 & \mathrm{X} \oplus X \arrow[ul, "0 \oplus 1"] \arrow[r, phantom, "="] & \mathrm{X} \oplus X
\end{tikzcd}$$

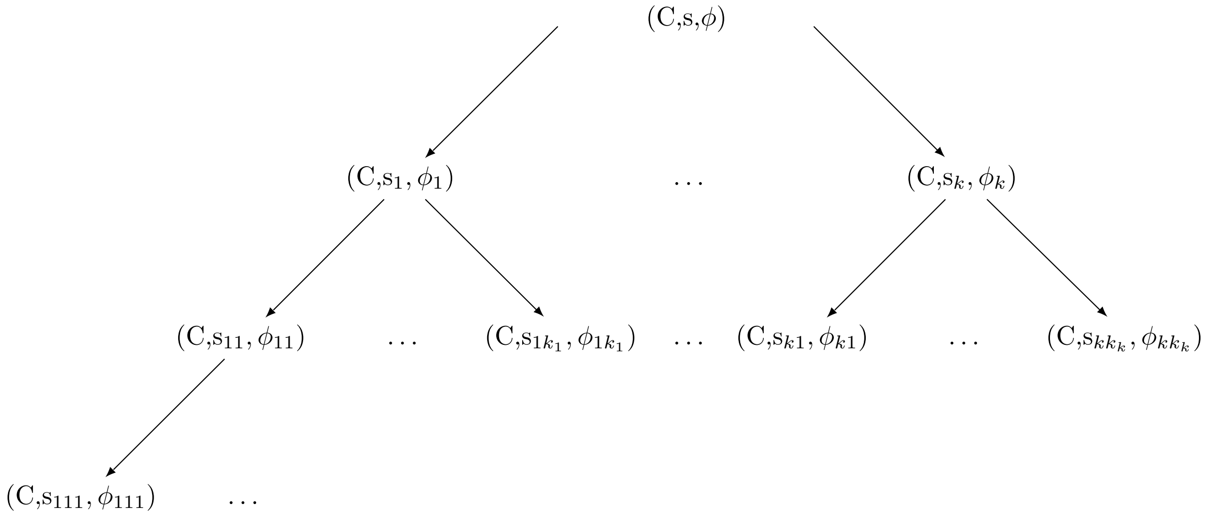
Finally we show the `while` b `do` C_1. We need to show that (for any s, ϕ) $\mathcal{O}[\![C]\!](s, \phi) = \mathcal{D}'[\![C]\!](s, \phi)$ and once again we do this by showing $LHS \sqsupseteq RHS$ and $RHS \sqsupseteq LHS$ in the pointwise ordering inherited from the cpo structure.

We firstly show that $\mathcal{O}[\![C]\!] \sqsupseteq \mathcal{D}'[\![C]\!]$. We do this by showing that $\mathcal{D}'[\![C]\!]$ is the fixed point of the function $g = \lambda f.\lambda(s, \phi).$ if $[\![b]\!].s = q_1$ then $f(\mathcal{D}[\![C_1]\!].(s \otimes F(\phi)))$ else $(s \otimes F(\phi))$. We note once again that this function is continuous. We claim then that for all k, $(h^k\bot).(s \otimes F(\phi)) = g^k(\bot)(s, \phi)$. We emphasise the types for clarity at this stage: here h is an endomap on $(\mathbf{CPM(C)}^{\oplus}(X, X))$ and so $h^k\bot$ is an arrow in this hom-set. Conversely, g is an endomap on the function space $\mathbf{CPM(C)}^{\oplus}(I, S) \otimes \mathbf{C}(I, H) \rightarrow \mathbf{CPM(C)}^{\oplus}(I, X)$ and so $g^k\bot$ is a map of the previously expressed type.

In the base case we have $\bot.(s \otimes F(\phi)) = \bot = (\bot)(s, \phi)$ since $\bot = 0$. For the inductive step $k = n + 1$ we have $(h^{k+1}\bot).(s \otimes F(\phi)) = ([(h^k\bot).\mathcal{D}[\![C_1]\!], id].(\pi_1 \oplus \pi_1).\mathsf{dist}.(id \times [\![b]\!]).\Delta.(s \otimes F(\phi)))$. In the case that $[\![b]\!].s = q_1$ (by our reasoning in the if case) this is $(h^k\bot).\mathcal{D}[\![C_1]\!].(s \otimes F(\phi))$ which is $(g^k\bot)(\mathcal{D}[\![C_1]\!].(s \otimes F(\phi)))$ by inductive hypothesis which is $(g^{k+1}\bot)(s, \phi)$ by definition of g. The case where $[\![b]\!].s = q_2$ both sides are equal to just $(s \otimes F(\phi))$ and so hence equal. Hence for all k, $(g^k\bot)(s, \phi) = (h^k\bot).(s \otimes F(\phi))$ and so $\mathcal{D}'[\![C]\!](s, \phi) = (\bigsqcup(h^k\bot)).(s \otimes F(\phi)) = \bigsqcup((h^k\bot).(s \otimes F(\phi))) = \bigsqcup(g^k\bot(s, \phi))$ by the above which is $\bigsqcup(g^k\bot)(s, \phi)$ by definition of limits of functions. Finally this is $\mathsf{lfp}(g)(s, \phi)$ as required, and so $\mathcal{D}'[\![C]\!]$ is indeed the least fixed point of the function g. We now see that $\mathcal{O}[\![C]\!] \sqsupseteq \mathcal{D}'[\![C]\!]$ since $\mathcal{O}[\![C]\!]$ is too a fixed point of g, and this is clear from the operational semantics as before.

For the other inequality, we need to show that $\mathcal{O}[\![C]\!] \sqsubseteq \mathcal{D}'[\![C]\!]$ i.e. $\mathcal{O}[\![C]\!](s, \phi) \sqsubseteq \mathcal{D}'[\![C]\!](s, \phi)$ for any s, ϕ using the cpo ordering. We shall show that for all n there is some m such that $\mathcal{O}[\![C]\!]_n(s, \phi) \sqsubseteq (h^m \bot)(s \otimes F(\phi))$ where h is the endofunction above (given in the definition of the while case semantics.) From this it shall follow that for all n, $\mathcal{O}[\![C]\!]_n \leq \mathcal{D}[\![C]\!]$ and so the limit of the chain $\mathcal{O}[\![C]\!]_n$ is at most $\mathcal{D}[\![C]\!]$, i.e. precisely that $\mathcal{O}[\![C]\!] \sqsubseteq \mathcal{D}[\![C]\!]$.

We show $\forall n \mathcal{O}[\![C]\!]_n(s, \phi) \sqsubseteq (h^{n+1}\bot)(s, \phi)$. Let us consider the shape of $\mathsf{Comp}_n(C, s, s', \phi, \phi')$ for our specific while case. The shape of a deterministic while reduction branch (as in the case without nondeterminism above) is $(C,\, s,\, \phi) = (C,\, s_0,\, \phi_0) \rightarrow (C_1; C,\, s_0,\, \phi_0) \rightarrow \ldots \rightarrow (\texttt{skip};C,\, s_1,\, \phi_1) \rightarrow (C,\, s_1,\, \phi_1) \rightarrow \ldots \rightarrow (C,\, s_2,\, \phi_2) \rightarrow \ldots \rightarrow (C,\, s_m,\, \phi_m) \rightarrow (\texttt{skip},\, s_m,\, \phi_m) = (\texttt{skip}, s', \phi')$ for some $m \geq 0$ with $m \leq n$ where $[\![b]\!].s_i = q_1$ for $i < m$ and $[\![b]\!].s_i = q_2$ for $i = m$. As before, we consider the reduction tree starting at (C, s, ϕ) where (C, s, ϕ) reduces to $(C, s_1, \phi_1) \ldots (C, s_k, \phi_k)$ and then each (C, s_j, ϕ_j) reduces to $(C, s_{j1}, \phi_{j1}) \ldots (C, s_{jk_j}, \phi_{jk_j})$ for some k_j each time (see diagram).


(C,s,ϕ)
(C,s$_1$, ϕ_1)
...
(C,s$_k$, ϕ_k)
(C,s$_{11}$, ϕ_{11})
...
(C,s$_{1k_1}$, ϕ_{1k_1})
...
(C,s$_{k1}$, ϕ_{k1})
...
(C,s$_{kk_k}$, ϕ_{kk_k})
(C,s$_{111}$, ϕ_{111})
...

We note that in the above $[\![b]\!].s = q_2$ for all nodes that lead immediately to $(\texttt{skip}, s)$ and that for all other nodes s, $[\![b]\!].s = q_1$. We note that all terminal nodes are at depth (i.e. have an address with length at most) n. Writing $p(s_{ij})$ for the probability of the path from (C, s_i) to (C, s_{ij}) in the tree above (i.e. the product of the single steps in this path) it follows that $p(s_{ij}) \sqsubseteq \mathcal{O}[\![C_1]\!](s_i, s_{ij})$ by analysis of the tree and definition of operational semantics. Note that then $\mathcal{O}[\![C_1]\!](s_i, s_{ij}) = \mathcal{D}'[\![C_1]\!](s_i, s_{ij})$ by inductive hypothesis.

Given a node (C, s_i, ϕ_i) in the above tree and node below it (s_j, ϕ_j) we define $P(s_i, \phi_i, s_j, \phi_j)$ to be the sum of the probabilities along the path between them in the tree — thus $\mathcal{O}[\![C]\!]_n(s, \phi) = \Sigma P(s, \phi, s', \phi').(s' \otimes \phi')$ where s, ϕ is labeled with the empty address and referring to the top node of the tree, and the summation is over terminal nodes below that. Our claim then amounts to showing that $\Sigma P(s, \phi, s', \phi').(s' \otimes F(\phi')) \leq (h^{n+1}\bot)(s, \phi)$. We shall show this by a backwards induction going up the tree — given any node s_i, ϕ_i in the tree we write $d(s_i, \phi_i)$ for the maximum distance (in terms of nodes in our tree above where all nodes are of the form (C, s_j, ϕ_j)) between (C, s_i, ϕ_i) and its underlying $(\texttt{skip}, s', \phi')$. We claim that for all nodes in the tree s_i, ϕ_i we have $\Sigma P(s_i, \phi_i, s', \phi').(s' \otimes F(\phi')) \sqsubseteq (h^{d(s_i,\phi_i)+1}\bot)(s_i, \phi_i)$. We show this by induction on $d(s_i, \phi_i)$.

In the terminal case, with $d(s_i, \phi_i) = 0$ then $(s_i, \phi_i) = (s', \phi')$ and $P(s_i, \phi_i, s_i, \phi_i) = 1$ since we consider the single transition $(C, s_i) \rightarrow (\texttt{skip}, s_i)$ with probability 1. Hence LHS is simply $s_i \otimes F(\phi_i)$ On the other hand since $s_i, \phi_i = s', \phi'$ we have $[\![b]\!].s_i = q_2$ and so $(h\bot)(s_i, \phi_i) = id.(s_i \otimes F(\phi_i)) = (s_i \otimes F(\phi_i))$ and so we have our result.

In the case that $d(s_i, \phi_i) = k + 1$ then we consider $\Sigma P(s_i, \phi_i, s', \phi').(s' \otimes F(\phi'))$, which we denote $f(i)$. Clearly by inspection this is equal to $\Sigma_j(p(s_{ij}).f(ij))$ by examination of our operational semantic tree.

Now $(h^{k+2}\bot)(s_i, \phi_i) = (h(h^{k+1}\bot))(s_i, \phi_i)$. Since $[\![b]\!].s_i = q_1$ this is $((h^{k+1}\bot).\mathcal{D}[\![C_1]\!])(s_i \otimes F(\phi_i))$ by using our usual reasoning (as in e.g. the if case before.)

Now $\mathcal{D}[\![C_1]\!].(s_i \otimes F(\phi_i)) = \mathcal{O}[\![C_1]\!](s_i, \phi_i)$ by inductive hypothesis. It is clear by definition of the operational semantics, monotonicity of addition and our reduction tree that that $\mathcal{O}[\![C_1]\!](s_i, \phi_i) \sqsupseteq \Sigma_j(p(s_{ij}).(s_{ij} \otimes F(\phi_{ij})))$. Hence $(h^{k+2}\bot)(s_i, \phi_i) \sqsupseteq (h^{k+1}\bot).(\Sigma_j(p(s_{ij}).(s_{ij} \otimes F(\phi_{ij}))))$. By linearity and scalars this is $\Sigma_j(p(s_{ij}).(h^{k+1}\bot).(s_{ij} \otimes F(\phi_{ij})))$. By our current inductive hypothesis this is $\Sigma_j(p(s_{ij}).f(ij))$ as required, which is $f(i)$ as required by our above note, hence we are done.

The culmination of this induction states that $\mathcal{O}[\![C]\!]_n(s, \phi) \sqsubseteq (h^{d(s,\phi)+1}\bot)(s, \phi) \sqsubseteq (h^{n+1}\bot)(s, \phi)$ since $d(s) = m \leq n$ and we are dealing with an increasing chain. This states precisely that $\mathcal{O}[\![C]\!]_n(s, \phi) \sqsubseteq (h^{n+1}\bot)(s, \phi) \sqsubseteq \mathcal{D}[\![C]\!](s, \phi)$, and since this holds for any n it follows that since $\mathcal{O}[\![C]\!](s, \phi)$ is a *least* upper bound we have $\mathcal{O}[\![C]\!](s, \phi) \sqsubseteq \mathcal{D}[\![C]\!](s, \phi)$ as required.

It follows then from the above that $\mathcal{O}[\![C]\!](s) = \mathcal{D}[\![C]\!](s)$ in the while case, and so our result is proved.

□

4.6 A Further Refinement: The category $\mathbf{SUP(C)}^{\oplus}$

We now seek to strengthen our above result by weakening the hypothesis, i.e. the requirements of the category in question. The reason for this is that our above assumptions require the category $\mathbf{CPM(C)}^{\oplus}$ to be cpo-enriched, which as mentioned amounts to requiring $\mathbf{CPM(C)}$ to be cpo-enriched. In the case $\mathbf{C} = \mathbf{FdHilb}$ we have our natural ordering on probabilities that can be extended to all completely positive maps (as will be described shortly,) but it naturally does not yield upper bounds of chains as such — for example the chain $[1, 2, 3, \ldots]$ clearly has no upper bound. What we do have however is that, as in the concrete case, we are only dealing with probabilities that are at most 1 and such probabilities do yield least upper bounds. We will thus proceed as follows: We shall assume our ordering $\sqsubseteq$ on the whole category $\mathbf{CPM(C)}$ and assume that e.g. addition is monotonic with respect to it exactly as above. The only change to the assumptions above is that we only require least upper bounds for chains where each component is *trace-decreasing*, as in [24]. In this update, we now also require that the quantum part of configurations be normalised.

Definition 4.6.1 *Any "element" in the CPM category $I \rightarrow A \otimes A^*$ can be written as the lambda abstraction of a map $A \rightarrow A$. Given such an $s = \Lambda(f)$ then we define $tr(s)$ to be the scalar $\epsilon_A.(id_{A^*} \otimes f).\eta_A : I \rightarrow I$ (we note by properties of compact closure etc. that this is equal to $\epsilon_A.s$ and so perhaps this is a more succinct definition.)*

We note we shall use here the standard results that the trace of a scalar is that scalar, and that trace preserves addition (this is clear by definition of trace, since composition does.)

Note that this coincides with treatment given in e.g. [10]. Further details can be found in e.g. [6] and we note that this is a direct abstraction of the notion of trace from linear algebra (sum of the diagonal elements when expressed as a matrix.)

We now generalise this to tuples of **CPM**-objects using the free biproduct closure.

Definition 4.6.2 *Let $f : I \to A$ where A is some object in **CPM(C)**$^{\oplus}$. Then since objects in **CPM(C)**$^{\oplus}$ are tuples $\langle A_1, \dots A_n \rangle = A_1 \oplus \dots \oplus A_n$, f can be written as $\langle f_1, \dots, f_n \rangle$ with $f_i : I \to A_i$. Since A_i is an object in **CPM(C)** and so is of the form $X \otimes X^*$ we can take the trace of each f_i. We then define $tr(f) = \Sigma(tr(f_i))$.*

Definition 4.6.3 *A map in the **CPM**$^{\oplus}$ category $f : A \to B$ is said to be* trace-decreasing *if for any $s : I \to A$ we have $tr(f.s) \sqsubseteq tr(s)$.*

We note that trace-decreasing arrows are closed under composition etc. and clearly the identity is trace decreasing, hence:

Definition 4.6.4 *The category **SUP(C)**$^{\oplus}$ is a subcategory of **CPM(C)**$^{\oplus}$ whose objects are the same and arrows are those that are also trace-decreasing (*superoperators *in the literature e.g. in [24], hence the name.)*

Note the notation may be slightly misleading here, since **SUP(C)**$^{\oplus}$ is not meant to refer to the biproduct completion of **SUP(C)** (the category of completely positive trace-decreasing maps — in fact this does not make sense, as **SUP(C)**$^{\oplus}$ is not necessarily closed under addition, e.g. the set $[0, 1]$ of the real numbers is not.)

We now have identified three principal subcategories of **CPM(C)**$^{\oplus}$ that we use — the canonical classical subcategory, the full subcategory **CPM(C)** of single-tuple objects, and this **SUP(C)**$^{\oplus}$ subcategory of trace-decreasing maps.

Our requirement, then, is that **CPM(C)** is order-enriched as before, and that the derived ordering in **CPM(C)**$^{\oplus}$ as above is complete for homsets of **SUP(C)**$^{\oplus}$.

Definition 4.6.5 *A* weak normalising SCCOCB *consists of a strongly compact closed category with biproducts with an order relation on **CPM(C)** that is complete when extended to homsets of **SUP(C)**$^{\oplus}$. We require that composition and copairing are continuous with respect to this ordering and $0 = \bot$. We also require that the nonzero positive scalars are closed under addition, multiplication, inverses and square roots. Finally we require that additon and multiplication are monotonic with respect to $\sqsubseteq$ on the positive scalars.*

Note that trace-decreasing completely positive maps are closed under composition (as noted above) but we need to check that trace-decreasing completely positive maps are closed under copairing for the above to make sense.

Proposition 4.6.6 *If $a : J \to A$ and $b : K \to A$ are trace-decreasing arrows, then so is $[a, b] : J \oplus K \to A$.*

Proof Let $c : I \to J \oplus K$. Then $c = \langle c_1, c_2 \rangle$ with $c_1 : I \to J$ and $c_2 : I \to K$ since $\oplus$ is also a product. $\langle c_1, c_2 \rangle = (c_1 \oplus c_2).\Delta$ with $\Delta : I \to I \oplus I$ by biproducts. Dually $[a, b] = \nabla.(a \oplus b)$. Thus $tr([a, b].c) = tr([a, b].\langle c_1, c_2 \rangle) = tr(\nabla.(a \oplus b).(c_1 \oplus c_2).\Delta) = tr(\nabla.(a.c_1 \oplus b.c_2).\Delta) = tr(a.c_1 + b.c_2) = tr(a.c_1) + tr(b.c_2) \sqsubseteq tr(c_1) + tr(c_2) = tr(c)$. We use here the primitive definition of $+$ and also the fact that $tr(c) = tr(c_1) + tr(c_2)$ by the concrete definition of traces in the category **CPM(C)**$^{\oplus}$ (if $J = \langle J_1, \dots, J_n \rangle$ and $K = \langle K_1, \dots K_m \rangle$ then $c_1 = \langle c_{11}, \dots, c_{1n} \rangle$ and $c_2 = \langle c_{21}, \dots, c_{2m} \rangle$. Then by definition of coproducts $J \oplus K = \langle J_1, \dots, J_n, K_1, \dots, K_n \rangle$ and $c = \langle c_{11}, \dots, c_{1n}, c_{21}, \dots, c_{2m} \rangle$ and we see that $tr(c) = \Sigma c_{1i} + \Sigma c_{2i} = tr(c_1) + tr(c_2)$.)

□

Finally we note that any normalising SCCOCB in the sense of our previous treatment is also a weak strongly normalising SCCOCB in the above sense, assuming the technical condition that the limit of a chain of trace-decreasing arrows is trace-decreasing (in this case $\mathbf{CPM(C)}^{\oplus}$ is cpo-enriched and then so is $\mathbf{SUP(C)}^{\oplus}$ since it is closed under least upper bounds; and we note then that composition and copairing in $\mathbf{SUP(C)}^{\oplus}$ is continuous since composition and addition in $\mathbf{CPM(C)}$ is continuous, in the same manner as before.)

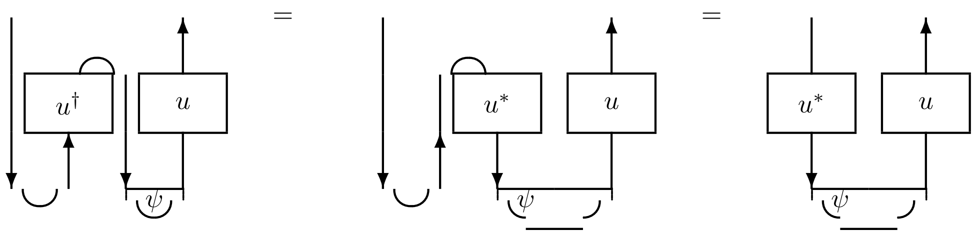
We need to check that the above treatment still makes sense under this weakening of the hypothesis.

4.6.1 Atomic Preliminaries

We firstly make the following observation, justifying a previous comment about applying unitaries to mixed states. Given $\psi : I \to A \otimes A^*$ we write $\psi_\diamond$ for the corresponding map $A \to A$, i.e. $\psi_\diamond = (id_A \otimes \epsilon_A).(\psi \otimes id_A)$. Note then that the $(-)_\diamond$ construct is the inverse to the name construct $\ulcorner - \urcorner$, which we also notate here as $\Lambda(-)$ for obvious reasons.

Proposition 4.6.7 *For $\psi : I \to A \otimes A^*$ and $u : A \to A$ we have $F(u).\psi = \Lambda(u.\psi_\diamond.u^\dagger)$*

Proof Now given any s, $\Lambda(s)$ is given by $(id_{A^*} \otimes s).\eta_A$ and so the right hand side is given by $(id_{A^*} \otimes \epsilon_A \otimes id_A).(id_{A^*} \otimes u^\dagger \otimes id_{A^*} \otimes u).(\eta_A \otimes \psi)$ by expanding the definition of the $(-)_\diamond$ construct. We can use diagrammatic compact closure logic to show that this is equal to the LHS (noting $F(u) = u \otimes u_*$).

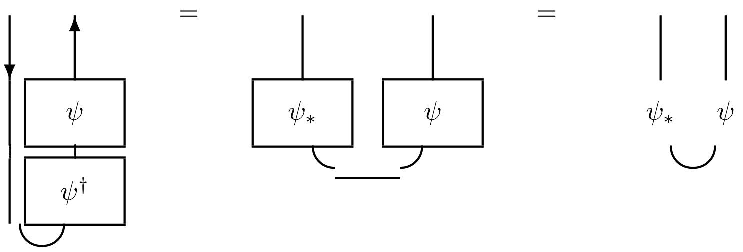


□

We also note that

Proposition 4.6.8 *For $\psi : I \to X$ we have $F(\psi)_\diamond = \psi.\psi^\dagger$.*

Proof To see this we show that $\Lambda(\psi.\psi^\dagger) = F(\psi)$. Again we reason using information flow laws and the fact that η_I is the isomorphism $I \to I^* \otimes I$.

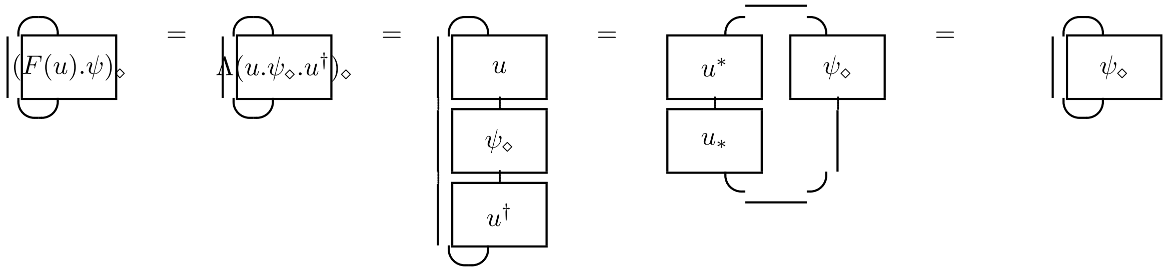


□

We next show that if u is a unitary then $F(u)$ is trace decreasing. In particular then this tells us that that $\mathsf{app}_{u,x,y}$ is trace decreasing, since it is of the form $id \otimes \ldots \otimes u \otimes \ldots \otimes id$ which is a unitary since u is.

Proposition 4.6.9 *If u is a unitary then $F(u)$ is trace decreasing.*

Proof Well $tr(F(u).\psi)$ is as follows, and we reason using the above lemma 4.6.7 and the fact that $\Lambda(-)$ and $(-)_\diamond$ are inverse constructs:



One step uses the fact that $u^*.u_* = u^*.(u^\dagger)^* = (u.u^\dagger)^* = id^* = id$. The rightmost diagram is $tr(\psi)$ as required.

□

We now show a similar proposition for the projectors.

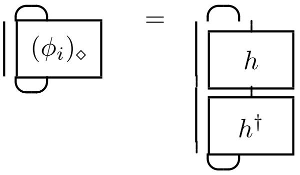
Proposition 4.6.10 $F(P^q_0) + F(P^q_1)$ *is trace-decreasing.*

Proof We note that by linearity $tr((F(P^q_0) + F(P^q_1)).\phi) = tr(F(P^q_0).\phi + F(P^q_1).\phi) = tr(F(P^q_0).\phi) + tr(F(P^q_1).\phi)$.

Now, P^q_i are projectors $A \oplus B \to A \oplus B$ for some A and B and they can be written as $1 \oplus 0$ and $0 \oplus 1$ respectively. As such $F(P^q_0) : (A \oplus B) \otimes (A \oplus B)^* \to (A \oplus B) \otimes (A \oplus B)^*$ is $(1 \oplus 0) \otimes (1 \oplus 0)^*$. By distributivity this can be seen as the map $1 \otimes 1^* \oplus 1 \otimes 0^* \oplus 0 \otimes 1^* \oplus 0 \otimes 0^*$ mapping the space $A \otimes A^* \oplus A \otimes B^* \oplus B \otimes A^* \oplus B \otimes B^*$ to itself. This is clearly the map $1 \oplus 0 \oplus 0 \oplus 0$, and so $F(P^q_0)$ is in fact a projector onto $A \otimes A^*$. Similarly $F(P^q_1)$ is the projector onto $B \otimes B^*$.

Any $\phi : I \to A \otimes A^* \oplus A \otimes B^* \oplus B \otimes A^* \oplus B \otimes B^*$ can be written as $\langle \phi_1, \phi_2, \phi_3, \phi_4 \rangle = \Sigma \phi_i.q_i$ with $q_i : I \to I \oplus I \oplus I \oplus I$. Then we have $tr(F(P^q_0).\phi) + tr(F(P^q_1).\phi) = tr(\phi_1.q_1) + tr(\phi_4.q_4)$. We need to show that this is at most $tr(\phi)$. Well $tr(\phi) = tr(\Sigma \phi_i.q_i) = \Sigma tr(\phi_i.q_i)$ and so we need to show that $tr(\phi_1.q_1) + tr(\phi_4.q_4) \sqsubseteq \Sigma tr(\phi_i.q_i)$. We claim that each $tr(\phi_i)$ is positive, from which our result easily follows by monotonicity of addition.

Well since ϕ_i is an element in the category **CPM(C)** it must be the name of a positive map, and thus $tr(\phi_i)$ is



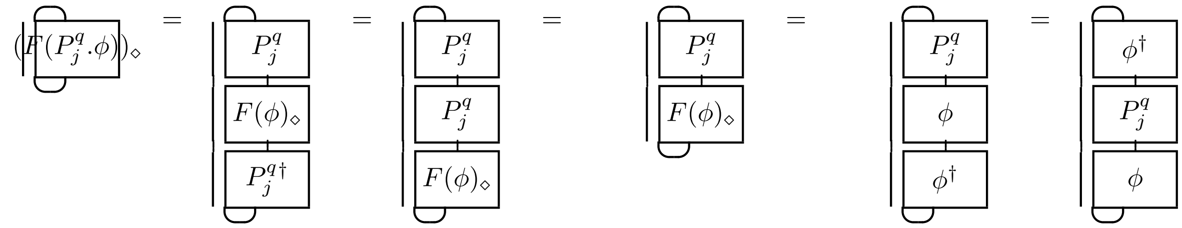
which is $(\epsilon.(id \otimes h)).(\epsilon.(id \otimes h))^\dagger$ and as such positive. Thus $tr(\phi_2.q_2 + \phi_3.q_3)$ is positive and so $tr(\phi) = \Sigma tr(\phi_i.q_i) \sqsupseteq tr(\phi_1.q_1) + tr(\phi_4.q_4)$ as required.

□

We show one final proposition before proceeding to our results:

Proposition 4.6.11 *$tr(F(P^q_j.\phi)) = p^q_j(\phi)$ with our projector notation as above (so $p^q_j(\phi) = \phi^\dagger.P^q_j.\phi$.)*

Proof Well $tr(F(P^q_j.\phi)) = tr(F(P^q_j).F(\phi))$ is as the diagram on the left. We use our above lemmas 4.6.7 and 4.6.8, compact closure logic and the fact that P^q_j is a projector and so idempotent.



Finally, since the trace of a scalar is that scalar, the rightmost diagram is $\phi^\dagger.P^q_j.\phi$ as required.

□

4.6.2 Denotational Semantics

For the denotational semantics, the cpo-structure is used in the while case. In order for the definition of the while semantic to make sense we need to know that it concerns the least fixed point of a continuous function on homsets of $\mathbf{SUP(C)}^\oplus$, i.e. that h maps trace decreasing functions to trace decreasing functions (i.e. h is an endomap of $\mathbf{SUP(C)}^\oplus$) and that it is a continuous such endomap.

We make a quick lemma that will be useful in showing that this is the case.

Definition 4.6.12 *An arrow $f : X \to X = k.I \otimes F(H)$ is* part classical trace decreasing *if for any $q_i \otimes \phi : I \to k.I \otimes F(H)$ we have $tr(f.(q_i \otimes \phi)) \sqsubseteq tr(q_i \otimes \phi)$.*

Proposition 4.6.13 *If an arrow f is part classical trace decreasing, then it is trace decreasing in general.*

Proof Let $s : I \to X$. Then since $X = k.I \otimes F(H) = k.F(H)$ we can write s as $\langle s_1, \ldots, s_k \rangle$ using products. Then $s = \Sigma\langle 0, \ldots, 0, s_i, 0, \ldots, 0\rangle = \Sigma q_i \otimes s_i$. Then $tr(f.s) = tr(f.\Sigma q_i \otimes s_i) = tr(\Sigma f.(q_i \otimes s_i)) = \Sigma tr(f.(s_i \otimes q_i)) \sqsubseteq \Sigma tr(q_i \otimes s_i) = tr(q_i \otimes s_i) = tr(s)$ as required. Here we once again use our assumed monotonicity of addition, trace distributing over addition etc.

□

Here we have noted that $q_i \otimes \phi : I \to X$ is in fact the map $\langle 0, \ldots, 0, \phi, , \ldots, 0\rangle : I \to k.F(H)$ with ϕ occuring in the ith component (by the definition of tensor in the biproduct category) and so $tr(q_i \otimes \phi) = tr(\phi)$.

We now seek to show that the denotational semantics of while still makes sense as above. Clearly the above meaning of $\mathcal{D}[\![C]\!]$ for all non-while commands can remain the same under these new assumptions. We firstly show that for all such commands, $\mathcal{D}[\![C]\!]$ is trace-decreasing i.e. is an element of $\mathbf{SUP(C)}^\oplus$.

Proposition 4.6.14 *For any non-while command C, $\mathcal{D}[\![C]\!] : X \to X$ is trace-decreasing.*

Proof We show this by induction on C. Clearly by the above lemma it will suffice to show that $\mathcal{D}[\![C]\!]$ is part classical trace decreasing.

In the case $C = \texttt{skip}$ then $\mathcal{D}[\![C]\!] = id$ which is trace-preserving since $tr(id.A) = tr(A) \sqsubseteq tr(A)$ by reflexivity.

In the case $C = v := e$ then $\mathcal{D}[\![C]\!] = [v \mapsto [\![e]\!]] \otimes id$. Now $tr(([v \mapsto [\![e]\!]] \otimes id).(q_i \otimes \phi)) = tr([v \mapsto [\![e]\!]] .q_i \otimes \phi)$. Since $[v \mapsto [\![e]\!]]$ is a classical arrow (see above), we see that this is $tr(q_j \otimes \phi)$ for some j, which is $tr(\phi)$ as above. Hence $tr(\mathcal{D}[\![C]\!].(q_i \otimes \phi)) = tr(q_j \otimes \phi) = tr(\phi) = tr(q_i \otimes \phi)$ and so $\mathcal{D}[\![C]\!]$ is trace-preserving.

In the case $C = C_1; C_2$ then $tr(\mathcal{D}[\![C]\!].a){=}tr(\mathcal{D}[\![C_2]\!].\mathcal{D}[\![C_1]\!].a) \leq tr(\mathcal{D}[\![C_1]\!].a) \sqsubseteq tr(a)$ by inductive hypothesis applied to C_2 and then C_1 respectively.

In the case $C = \texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2$ we know that $tr(\mathcal{D}[\![C]\!].s) = tr(\mathcal{D}[\![C_1]\!].s)$ or $tr(\mathcal{D}[\![C_2]\!].s)$ by the correspondence result, which is at most $tr(s)$ by inductive hypothesis.

For the unitary application case. Then $\mathcal{D}[\![C]\!].(q_i \otimes \phi) = (id \otimes F(app)).(q_i \otimes \phi) = q_i \otimes F(app).\phi$. Thus $tr(q_i \otimes \phi) = tr(\phi) \sqsubseteq tr(F(app).\phi) = tr(\mathcal{D}[\![C]\!].(q_i \otimes \phi))$ as required using our above lemma 4.6.9.

For the measurement case we note that $tr(\mathcal{D}[\![C]\!].(q_i \otimes \phi)) = tr(([v \mapsto 0] \otimes F(P_0^q) + [v \mapsto 1] \otimes F(P_1^q)).(q_i \otimes \phi)) = tr(([v \mapsto 0] \otimes F(P_0^q) + [v \mapsto 1] \otimes F(P_1^q)).(q_i \otimes \phi)) = tr((id \otimes F(P_0^q) + id \otimes F(P_1^q)).(q_i \otimes \phi))$ since $[v \mapsto j] \otimes id$ is trace-decreasing by the assignment case above. This is $tr((id \otimes (F(P_0^q) + F(P_1^q))).(q_i \otimes \phi)) = tr(q_i \otimes (F(P_0^q) + F(P_1^q)).\phi) = tr((F(P_0^q) + F(P_1^q)).\phi) \sqsubseteq tr(\phi)$ since we have shown above that $F(P_0^q) + F(P_1^q)$ is trace-decreasing 4.6.10.

□

We now need to show that the denotational semantics in the while case makes sense. To do this we need to show firstly that h maps trace-decreasing maps to trace-decreasing maps. To do this it suffices to show that if f is trace-decreasing then hf is part classical trace-decreasing. Well $tr((hf).(q_i \otimes \phi)) = tr(q_i \otimes \phi)$ or $tr(f.\mathcal{D}[\![C_1]\!].(q_i \otimes \phi))$ by our correspondance depending on whether $[\![b]\!].q_i = q_1$ or q_2. In the former case we are clearly done. In the latter case we note that by assumption f is trace decreasing and also that $\mathcal{D}[\![C_1]\!]$ is (by inductive hypothesis embedding the above proposition) , and so we are done.

Finally we need to show that the function h is a continuous endofunction of $\mathbf{SUP(C)}^{\oplus}$. Well $hf = [f.\mathcal{D}[\![C_1]\!], id].\mathsf{dist}.(id \otimes [\![b]\!]).\Delta$. Since composition and copairing are continuous, it suffices to show that $g = \mathsf{dist}.(id \otimes [\![b]\!]).\Delta$ is trace-decreasing, since then we can write h as

$$\mathsf{comp}.\langle \mathsf{copair}.\langle \mathsf{comp}.\langle id, \mathsf{konst}_{\mathcal{D}[\![C_1]\!]}\rangle, \mathsf{konst}_{id}\rangle, \mathsf{konst}_g\rangle$$

That is, we can express h as a combination of constant maps and continuous functions on our cpo-space and as such it too must be continuous ($\mathcal{D}[\![C_1]\!]$, id and g are trace-decreasing; comp and copair are continuous maps of $\mathbf{SUP(C)}^{\oplus}$; and the composition of continuous maps is also continuous.) Well to show that $g : X \to X \oplus X$ is trace-decreasing it of course suffices to show that g is part classical trace decreasing, and from our proofs above we see that $g.(q_i \otimes \phi) = q_j.(q_i \otimes \phi)$ for some $j \in \{1, 2\}$. This is of course $q_{ij} \otimes \phi$ and thus $tr(g.(q_i \otimes \phi)) = tr(q_{ij} \otimes \phi) = tr(\phi) = tr(q_i \otimes \phi)$ as required.

Hence h is indeed a valid continuous endofunction on $\mathbf{SUP(C)}^{\oplus}$ and so taking its least fixed point in the while case of the denotational semantics is indeed a valid thing to do.

4.6.3 Operational Semantics

For the operational semantics we also use a least-upper-bound construct, and we need to show that $\mathcal{O}[\![C]\!]_n(s,\phi)$ is trace-decreasing, since we need to find its least fixed point $\mathcal{O}[\![C]\!](s,\phi)$. In order to do this we need to add the additional constraint on configurations (s,ϕ) that ϕ is normalised. This is of course perfectly acceptable, and perhaps we should have assumed this to begin with — elements are represented by rays in a Hilbert space, and we usually deal with their normalised representative. Normalisation of ϕ amounts to $tr(F(\phi)) = 1$ — indeed a qubit $c = (a,b)$ is normalised in **FdHilb** if $a^2 + b^2 = 1$ which is precisely iff $tr(c \otimes c_*) = 1$. We need to show that normality is preserved by the operational semantics.

Proposition 4.6.15 *If $(C,s,\phi) \to (C',s',\phi')$ and ϕ is normalised, then so is ϕ'.*

Proof Clearly we only need to examine the cases where C is a unitary application or a measurement, since otherwise $\phi' = \phi$.

In the former case, $tr(F(\phi')) = tr(F(u.\phi))$ where u is some unitary. The RHS is $tr(F(u).F(\phi)) \sqsubseteq tr(F(\phi))$ by our proposition above that $F(u)$ is trace decreasing.

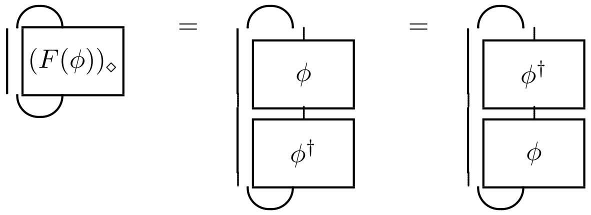
In the latter case, we note that $tr(F(\phi')) = tr(F(\frac{1}{\sqrt{p^q_j(\phi)}}.P^q_j.\phi))$. Since we can factor scalars out of the trace construct this is $F(\frac{1}{\sqrt{p^q_j(\phi)}}).tr(F(P^q_j.\phi))$ and since F squares scalars this is $\frac{1}{p^q_j(\phi)}.tr(F(P^q_j.\phi))$. By our above atomic preliminary 4.6.11 this is $\frac{1}{p^q_j(\phi)}.p^q_j(\phi)$ which is of course 1 by definition of inverses.

□

We next make the following note: In the first version above we did not require that probabilities branching out of a configuration sum to at most one in the operational semantics (since we assumed arbitrary cpo-enrichment for completely positive maps.) Now we have the assumption that ϕ is normalised, we note that this is indeed the case — when the computation is deterministic this clearly holds, and so we only need to check the measurement case:

Proposition 4.6.16 $p^q_0(\phi) + p^q_1(\phi)$ *= 1*

Proof This is $\phi^\dagger.P^q_0.\phi + \phi^\dagger.P^q_1.\phi = \phi^\dagger.(P^q_0 + P^q_1).\phi = \phi^\dagger.id.\phi$ by the biproduct structure, which is $\phi^\dagger.\phi$. We know that this is 1 since $\phi^\dagger.\phi = tr(F(\phi)) = 1$ since ϕ is normalised. To see the former equality, we note that $tr(F(\phi))$ is



We firstly use our above lemma 4.6.7, and then compact closure, and finally wenote that the right hand side is our required expression since the trace of a scalar is a scalar.

□

Thus given any $(C, s, \phi)\vec{\rightarrow}(C_1, s_1, \phi_1), \ldots, (C_n, s_n, \phi_n)$ we know that the sum of all of the probabilities of the branches $(C, s, \phi)\vec{\rightarrow}(C_i, s_i, \phi_i)$ sum to at most one, as before (using monotonicity of composition for positive scalars.)

We now seek to show our main result, i.e. that that $\mathcal{O}[\![C]\!]_n(s, \phi)$ is trace-decreasing (so we can find its least upper bound to define $\mathcal{O}[\![C]\!](s, \phi)$.) We note that the type of $\mathcal{O}[\![C]\!]_n(s, \phi)$ is $I \to X$.

Proposition 4.6.17 *If $a : I \to X$ satisfies $tr(a) \sqsubseteq 1$ then a is trace-decreasing.*

Proof To see this, we note that such an arrow is trace-decreasing if $tr(a).r \sqsubseteq tr(r)$ for any $r : I \to I$ (r must have domain I by definition of *trace decreasing*, and must have codomain I since a has domain I.) Hence if $tr(a) \sqsubseteq 1$ then $tr(a).r \sqsubseteq r$ by montonoicity of composition. Since r is a scalar this means $tr(a.r) \sqsubseteq r$ since scalars disribute over trace. Finally this means that $tr(a.r) \sqsubseteq tr(r)$ since the trace of a scalar is that scalar, as required. In the above we use the fact that composition is monotonic, but of course we only know that this is the case for positive scalars. However we know r to be a positive scalar since r is a scalar in the category **CPM(C)** and as such is (completely) positive.

□

Hence it suffices to show that $tr(\mathcal{O}[\![C]\!]_n(s, \phi)) \sqsubseteq 1$.

Proposition 4.6.18 $tr(\mathcal{O}[\![C]\!]_n(s, \phi)) \sqsubseteq 1$.

Proof Well $\mathcal{O}[\![C]\!]_n(s, \phi) = \Sigma(p(s', \phi').(s' \otimes F(\phi')))$ and so $tr(\mathcal{O}[\![C]\!]_n(s, \phi)) = tr(\Sigma(p(s', \phi').(s' \otimes F(\phi')))) = \Sigma(p(s', \phi').tr(s' \otimes F(\phi')))$ since we can move scalars out of the trace construct. Now we know that s' is classical i.e. of the form q_i for some i and so as before we have $tr(s' \otimes F(\phi')) = tr(F(\phi'))$. Hence $tr(\mathcal{O}[\![C]\!]_n(s, \phi)) = \Sigma(p(s', \phi').tr(F(\phi')))$. Now since ϕ is normalised and for all ϕ' here we have $\phi\vec{\rightarrow}\phi'$ we know that ϕ' is normalised by our above lemma 4.6.15, and so $tr(F(\phi')) = 1$ and so the above expression is $\Sigma(p(s', \phi'))$. This is at most 1 by the above lemma 4.6.16, and so we have $tr(\mathcal{O}[\![C]\!]_n(s, \phi)) \sqsubseteq 1$ as required.

□

Hence $\mathcal{O}[\![C]\!]_n(s, \phi)$ is indeed trace-decreasing and so the infinite sum makes sense as a limit of a cpo-process in the $\mathbf{SUP(C)}^{\oplus}$ subcategory.

4.6.4 Correspondence

The proof of the correspondence theorem is largely unchanged (of course now it is only valid for normalised ϕ) — the only alteration is that the function g in the while case (for which $\mathcal{D}'[\![C]\!]$ is the least fixed point) is now an endofunction on the space $\mathbf{CPM(C)}^{\oplus}(I, S) \times \mathbf{C}(I, H)^* \to \mathbf{SUP(C)}^{\oplus}(I, X)$ where $\mathbf{C}(I, H)^*$ consists of normalised ϕ (note that because the second input is assumed to be normalised the output of g is guaranteed to be trace-decreasing in each of the two cases.) This is of course not surprising — the definitions of the semantics have not changed, we have merely ensured that they make sense under our new assumptions.

4.7 Examples of (weak) Normalising SCCOCBs

To recall then, we define our semantics in an SCCCB **C** such that **CPM(C)** is order-enriched and this order is complete for $\mathbf{SUP(C)}^{\oplus}$, satisfying some further laws regarding interaction and scalars. We now need to check that these requirements are reasonable, which to us means that they are satisfied by **FdHilb** and **Rel**.

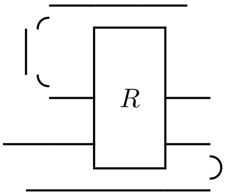
Firstly to see that **FdHilb** provides a valid model (a weak normalising SCCOCB), we appeal to such results from the paper [24]. Clearly **FdHilb** is a strongly compact closed category with biproducts. For the order relation on **CPM(FdHilb)** we use the *Lowner partial order* on completely positive maps [24]. This states that $A \sqsubseteq B$ if and only if $B - A$ is a positive matrix. This is a partial order that has a bottom element (which is 0 as we require) and in fact supports least upper bounds for trace-decreasing completely positive maps. Furthermore, the inherited ordering is complete for homsets of $\mathbf{SUP(FdHilb)}^{\oplus}$ as we require — a proof of this is given in [24], although using different notation (that the category **Q** is cpo-enriched with this ordering, which is the same as $\mathbf{SUP(FdHilb)}^{\oplus}$.) Composition and coparing are continuous (details of this is given in [24]. We note that functions are continuous with respect to this ordering if they are continuous with respect to the standard Euclidean topology [24].)

We note that a scalar a is positive if it it can be expressed $b.b^{\dagger}$, i.e. if it can be written in the form $\Sigma a_i.a_i^{\dagger}$ for complex numbers a_i. Scalars in **FdHilb** are complex numbers, and the complex numbers expressible in this form are precisely the positive reals. Thus the positive scalars are indeed the positive real numbers. As such, positive (nonzero) scalars are closed under addition, multiplication, inverses and square roots. Furthermore then for scalars b and a, $b - a$ is positive if and only if $b \geq a$ in the usual ordering on real numbers — and so composition and addition are indeed monotonic.

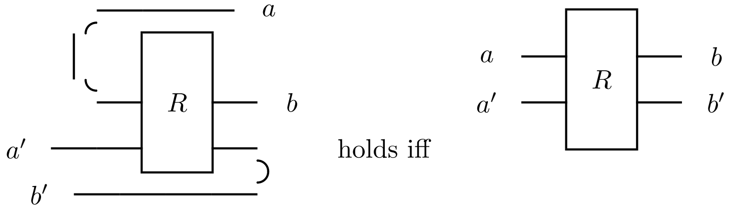
We now show that **Rel** too satisfies our requirements. As has been mentioned, **Rel** is indeed a strongly compact closed category with biproducts. We need to endow **CPM(Rel)** with an order-relation and show that is complete for trace-decreasing maps. There is a natural ordering on homsets of **Rel** itself — namely set inclusion. We show that this partial order is in fact a cpo on **CPM(Rel)**, and so **Rel** is a normalising SCCOCB in the sense of the former stronger definition.

Proposition 4.7.1 *$\perp$ is completely positive and the union of a chain of completely positive relations is also completely positive.*

Proof We once again appeal to corollary 4.13(b) of [23], which tells us that R is completely positive if R' is positive, where $R' = (id \otimes \epsilon_B).(id \otimes R \otimes id).(\eta_A \otimes id)$.



(Note that in the above diagram now the arrows have been removed, since $A = A^*$ for any object A and so wires are "bidirectional.") We recall from [23] that in **Rel** a relation is positive if it is symmetric and partial reflexive ($aRb \Rightarrow aRa$). We note that $(a, b)R'(a', b')$ iff $(a, b, *)(id \otimes \epsilon_B).(id \otimes R \otimes id).(\eta_A \otimes id)(*, a', b')$ revealing the previously suppressed monoidal isomorphism and writing $*$ as the single element of the monoidal unit $\{*\}$ in **Rel** (where the monoidal operation is the Cartesian product.) We can then write this out fully and determine that (by definition of the equality relation η_A) this holds iff $(a, a')R(b, b')$. This can alternatively be shown quickly using the graphical language:



For the $\perp$ case we need to show that the empty set $\emptyset$ satisfies $\emptyset' = (id \otimes \epsilon_B).(id \otimes \emptyset \otimes id).(\eta_A \otimes id)$ is positive. Well $(a, b)\emptyset'(a', b')$ iff $(a, a')\emptyset(b, b')$ which of course never holds and so $\emptyset' = \emptyset$ is the empty relation. The empty relation is clearly positive (symmetric and partial reflexive) and so $\emptyset'$ is positive and so $\emptyset$ is completely positive, as required.

Now consider a chain of completely positive relations $R_1, \ldots$. We let R be the least upper bound (union) of this chain. We need to show that R is completely positive, i.e. that R' is positive. Well suppose $(a, b)R'(a', b')$. Then $(a, a')R(b, b')$ by reasoning in the diagrammatic language as above. Then for some j, $(a, a')R_j(b, b')$ since R is the union of all such j. Then $(a, b)R'_j(a', b')$ by reasoning in the diagrammatic language once again. Since R_j is completely positive it follows that R'_j is positive and so we obtain both $(a, b)R'_j(a, b)$ and $(a', b')R'_j(a, b)$. Since $R_j \subseteq R$ and composition is monotonic with respect to inclusion, it follows that $R'_j \subseteq R'$. Hence we find that $(a, b)R'(a, b)$ and $(a', b')R'(a, b)$ concluding that R' is indeed a positive relation and so R is indeed completely positive.

□

So it seems that **CPM(Rel)** is complete as it is, and so we don't in particular need to consider trace decreasing operators — thus unlike **FdHilb**, **Rel** is a normalising SCCOCB in the sense of our earlier, stronger definition. Also $\perp$ is indeed the zero map (i.e. the empty set) and composition and addition is continuous (the sum of two arrows in **Rel** is their union.)

A scalar in **Rel** is an arrow $I \to I$, i.e. a relation between I and I, where $I = \{*\}$. There are two such relations — the empty relation and the relation relating $*$ with $*$ — we shall label these 0 and 1 respectively. A relation is positive if it is symmetric and partial reflexive ($xRy \Rightarrow xRx$) [23]. Both scalars 0 and 1 in **Rel** satisfy this property (0 satisfies both vacuously, and 1 is clearly both symmetric and reflexive.) Thus the positive scalars are indeed closed under addition. Likewise the scalars have square-roots ($1 = 1.1$, $0{=}0.0$) and also non-zero elements admit inverses ($1 = 1.1$). Finally, in the ordering we have $1 \sqsupset 0$ and so addition and multiplication are indeed monotonic, since $1 + 1 = 1$ and $1.1 = 1$.

4.8 Concrete Semantics as a Special Case

We now investigate how the concrete semantics given in the initial talk [3] and presented above compare with the abstract semantics given immediately above with respect to the special case $\mathbf{C} = \mathbf{FdHilb}$.

As noted above, since S is finite, $\mathsf{DProb}(S \times H)$ can be represented by $|S|$ elements of $\mathsf{DProb}(H)$ and $\mathsf{DProb}(H)$ corresponds to density operators of H in our category. To justify this, we consider the two directions: by comments in e.g. [11] any density operator can be written as a weighted sum of elements of the form $s \otimes s_*$ and as such can be represented as a weighted sum of elements in the image of the functor $F : \mathbf{FdHilb} \to \mathbf{CPM(FdHilb)}$ which corresponds to an element of $\mathsf{DProb}(H)$. For the other direction, we note that we have already specified that elements in $\mathsf{DProb}(H)$ must have *countable support*, that is only have a countable number of $s \in S \times H$ such that $f(s) \neq 0$ for $f \in \mathsf{DProb}(H)$. As such the probability distributions can be represented by an $\omega-$sum of finite sub-distributions. Each of these distributions can be represented by a density operator (by using F and addition) and we note that since our **SUP** category of density operators is *cpo-enriched* we can take their infinite sum (again appealing to the idea that we have a monotonic increasing sequence of partial approximates to the infinite sum and can hence take their limit.) Then by cpo-enrichment this is also a density operator (i.e. an element of H in the **SUP**-category.)

Note then in the concrete case we consider mappings $S \times H \to \mathsf{DProb}(S \times H)$ i.e. maps from a classical element of S, an element of H to a probability distribution. Under the above note, this is taking an element of S, an element of H and mapping it to $|S|$ density matrices over H. In the abstract case, we consider a map from $\mathbf{CPM(C)}^{\oplus}(I, S) \times \mathbf{C}(I, H) \to \mathbf{CPM(C)}^{\oplus}(I, X = S \otimes F(H))$. We have a correspondence here: The first parameter is the classical element (literally — it corresponds to one of k^m coproduct injections,) the second is a ray in the Hilbert space H and the result is a k^m tuple of density matrices over H, corresponding to $|S|$ probability distributions H as above (note that the output is indeed trace-decreasing, by our reasoning above, and so is indeed a density operator.)

And so the types of our denotations in the specialised abstract and concrete case coincide — this is no great surprise given our intentions. We also can easily see that the denotations themselves coincide with respect to this correspondence. As an example, we examine the measurement case, showing that the abstract case corresponds to the concrete exact formula we give above.

To recall, our formulae in the concrete case for calculating probabilities and results of measurements (in the computational basis) were as follows: To measure qubit i from $\Sigma \alpha_b |b\rangle$ then with probability $p_j = \Sigma \left\{ |\alpha_b|^2 | b_i = j \right\}$ we get the result j, and the state collapses to $\frac{1}{\sqrt{p_j}} \Sigma \left\{ \alpha_b |b\rangle | b_i = j \right\}$.

Clearly it suffices to show then that $p^i_j(\phi)$ is the former expression and $P^i_j(\phi)$ is the latter (without the normalisation constant.) We note $\phi : I \to H = Q^q$ in $\mathbf{C}$ and that $Q^q = (I \oplus I)^q \cong 2^q.I$. Any map $I \to k.I$ can be written as a tuple $\langle \phi_1, \ldots, \phi_k \rangle$ since we are mapping into a product, or alternatively $\Sigma(\phi_i.q_i)$ with $q_i : I \to 2^q.I$ since we have biproducts. Note that we are effectively using the biproduct decomposition as a basis — this idea is more formally presented in [6]. Note here that each ϕ_i is a scalar for each $i \in 2^k$, i.e. we can consider each i as a string of k binary bits, and so using the *ket* notation ϕ can be written as $\phi_1.|0\ldots0\rangle + \ldots + \phi_{2^k}.|1\ldots1\rangle$ where each ket $|r\rangle$ represents $q_r : I \to H$, i.e. a basis element. Note then applying P^i_j to this ϕ does indeed leave us in the state $\Sigma \left\{ \phi_b |b\rangle : b_i = j \right\}$ since $P^i_j(\phi) = P^i_j(\phi_1.|0\ldots0\rangle + \ldots + \phi_{2^k}.|1\ldots1\rangle = \Sigma(\phi_r.P^i_j|r\rangle) = \Sigma \left\{ \phi_b |b\rangle : b_i = j \right\}$ using the biproduct axiom for interaction between q_i and π_j ($\pi_j.q_i = \delta_{ij}$.)

Similarly then $p^i_j(\phi) = \phi^\dagger.P^i_j.\phi = (\langle 0\ldots0|\phi^\dagger_1 + \ldots + \langle 1\ldots1|\phi^\dagger_{2^k}).\Sigma \left\{ \phi_b |b\rangle | b_i = j \right\} = \Sigma \left\{ \phi_b.\phi^\dagger_b | b_i = j \right\}$ using orthonormality of different components of projections/injections, as required.

5 Abstract Logical Semantics

Having successfully abstracted the operational and denotational semantics using the SCCCB concepts developed in [6] and our additions, we now seek to give an investigation into how we might present the *logical semantics* at this abstract level. This shall be delivered via the notion of subobjects (the categorical abstraction of subsets,) and their order structure, as in the categorical analysis of logic in [14]. However, while [14] works in the structure of a topos (a complete, cocomplete category with exponenents and subobject classifiers — an abstraction of the strong structure of **Set** to a categorical level,) we work in the quite different structure of an SCCCB. Here subobjects correspond to subspaces rather than subsets and in the **FdHilb** case we find ourselves in the domain of quantum logic and [8]. In such logic, starting from the idea of measurable physical variables (self-adjoint operators, as with quantum measurements) we find that the propositions in question are represented by subspaces of a Hilbert space — and the lack of distributivity for subspaces makes the derived implication operator behave oddly (e.g. modus ponens fails.)

Our aim shall be to go as far as defining the meanings of our logical formula in the abstract case; and as such the requirements of the category will be intuitive and relate to what we think we need in order to give a reasonable definition of the logical semantics (conjunction, modal operator etc.) As such whether we require orthomodularity (negation + modularity axiom) of our structure or the weaker orthocomplementation (just negation) makes no difference to our aims here, and we shall stick with the latter as their is no real reason why we need the orthomodularity axiom in order to understand negation.

5.1 Subobjects

In our concrete logical semantics, we have semantically identified formulas with subsets of our state set. To expose the equivalent notion in category theory then, requires the notion of a *subobject.* This is a standard notion presented e.g. in [12].

In the category **Set**, subsets are represented by injective maps (any subset of a set is the range of some injective map into that set, and furthermore such a map is an isomorphism from the domain onto the subset.) The abstract version of an injective map is a monic arrow (see e.g. [2]). However, the same subset in **Set** can be represented by many injective maps in this manner — for example the subset of even natural numbers can be represented by an injective map from the even numbers (id) or from the odd numbers ($+1$). Given two injective maps a, b with codomain A they represent the same subset if $a = b.f$ for some $f : \mathsf{dom}(a) \to \mathsf{dom}(b)$, or if the symmetric scenario occurs. This motivates the following definition

Definition 5.1.1 *Let A be an object in a category $\boldsymbol{C}$. We define a preorder $\leq$ on the class of all monic arrows with codomain A by $u \leq v$ iff $u = v.f$ for some f. This partial order gives rise to an equivalence relation $\equiv$ with $u \equiv v$ iff $u \leq v \wedge v \leq u$. A* subobject *of A is then an equivalence class of $\equiv$.*

$$\begin{tikzcd}
\mathrm{P} \arrow[r, "v"] & \mathrm{A} \\
\mathrm{Q} \arrow[u, "f"] \arrow[ur, "u"] &
\end{tikzcd}$$

Note that $\leq$ is then a partial order on subobjects. This partial order precisely represents the subset relation in **Set** and intuitively generalises this concept (and also that of e.g. subgroups etc.) As another example, in a category that is a poset with a greatest element v, the partial order of the subobjects of v is the partial order inherent in the category itself (since all arrows in such a category are monic.)

Given an element of A, $x : I \to A$, intuitively that element x satisfies subobject ϕ of A if $\phi.f = x$ for some $f : I \to \mathsf{dom}(\phi)$, that is precisely if $x \leq \phi$ in our ordering extended to general arrows.

The partial order on subobjects when viewed from a logical perspective gives us entailment. We can also use, for example, greatest lower bounds for conjunction. For a full treatment of the interaction between logical formulae and partial order lattices see e.g. [20] — we shall use these ideas to express the purely logical part of our language.

5.1.1 Subobject Order Structure

We recall the definition of our logical formulae and their semantics in the concrete case

$$\phi ::= \top \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{variable} = \mathsf{value} \mid \langle C \rangle_q \phi$$

The top formulae $\top$ represents truth and requires that the subobject ordering has a greatest element with respect to the $\leq$ structure (since $\phi \Rightarrow \top$ for all ϕ and so $[\![\top]\!] \supseteq [\![\phi]\!]$ for all ϕ.) Thus $\top$ is represented by the abstract version of the universal total set in the domain we are working in, the greatest element in the ordering — and in our case, this will always be the subobject of X represented by the arrow id_X.

Conjunction $\wedge$ requires that the subobject poset has greatest lower bounds. To see this, e.g. in **Set** the greatest lower bound of two sets (with respect to the subset ordering) is the intersection of these sets, a semantic representation of conjunction.

In order to define the meaning of negation from an order perspective we need to assume also that we have a disjunction construct. This states that any two elements of the subobject poset have a *least upper bound* — this is the dual to the idea of conjunction immediately above. A *lattice* is defined to be a poset with least upper bounds and greatest lower bounds. To define semantics of negation we shall also need a false element $\bot$ that is least in the subset ordering, a dual to the truth value $\top$ above, represented by the empty set (since no elements satisfy this formula.) A lattice with both a top and bottom element is a *bounded lattice*.

Looking at **Set** as a motivating example, we find that the meaning of negation is the complement of the meaning of its principal subformula. By complement of a set X in abstract terms we mean a set Y that is disjoint from X but, together with X, covers the whole set (universe) in question. Thus we shall require for every element of the ordering x an element $\neg x$ such that $x \wedge \neg x = 0$ and $x \vee \neg x = 1$. We also require that the $\neg$ operator is involutive ($\neg\neg x = x$) and contravariant with respect to our ordering on subobjects (as in the case with our negation operator in star-autonomous categories) — we require that if $a \leq b$ then $\neg b \leq \neg a$. A bounded lattice with such an operator is known as an *orthocomplemented lattice*. In quantum logics a slightly richer structure is traditionally used (an orthomodular lattice, in which if $x \leq y$ then $y = x \vee (\neg x \wedge y)$) — however this assumption is not needed to give us an intuitive notion of negation for our subspaces.

Note that we cannot require distributivity between conjunction and disjunction for our structure if we wish **FdHilb** to be a canonical example — distributivity fails in **FdHilb** [8,11], and is a major difference between "quantum" logic and "classical" logic.

Thus to represent our (purely logical) part of the logical semantics, we require an lattice with a greatest and least element, together with negation in the sense of the above. That is, we require that the subobject partial orders in our category (or at least for our state space X) form an orthocomplemented lattice.

We now turn to the modal operator.

5.1.2 Pullbacks for the Modal Operator

For the modal operator $\langle C\rangle\phi$, we wish to find the *preimage* of the subobject $\llbracket\phi\rrbracket$ under $\mathcal{D}\llbracket C\rrbracket$. In category theory [2] the abstract notion of a preimage is that of a *pullback*.

Definition 5.1.2 *A* pullback *of a pair of arrows $f : A \to C$ and $g : B \to C$ is an object P and arrows $g' : P \to A$ and $f' : P \to B$ such that $f.g' = g.f'$; and if $i : X \to A$ and $j : X \to B$ are such that $f.i = g.j$ then there is a unique $k : X \to P$ such that $i = g'.k$ and $f = f'.k$*

$$\begin{tikzcd}
P \arrow[r, "f'"] \arrow[d, "g'"'] & B \arrow[d, "g"] \\
A \arrow[r, "f"] & C
\end{tikzcd}$$

This is a standard style category-theoretic universal construction. Pullbacks are a generalisation of products (and indeed equalisers, [18].)

We consider using this in our specific case: to find a denotation for $\langle C\rangle\phi$ from $\llbracket\phi\rrbracket : A \to X$ and $\mathcal{D}\llbracket C\rrbracket : X \to X$. Taking the pullback of this setup gives us (pre, g) that is the limit of the diagram

$$\begin{tikzcd}
P \arrow[r, "pre"] \arrow[d, "g"'] & X \arrow[d, "\mathcal{D}\llbracket C\rrbracket"'] \\
A \arrow[r, "\llbracket\phi\rrbracket"] & X
\end{tikzcd}$$

Then the mapping $pre : P \to X$ represents the inverse image subobject, and the mapping $g : P \to A$ witnesses the fact that $\mathcal{D}\llbracket C\rrbracket.pre \leq \llbracket\phi\rrbracket$, i.e. that taking the pre-image and applying $\mathcal{D}\llbracket C\rrbracket$ to it leaves us in a state where ϕ holds (as we desire.) The fact that (pre, g) is limiting in the above sense shows that it is maximal — if $j : Q \to X$ also satisfies $\mathcal{D}\llbracket C\rrbracket.j \leq \llbracket\phi\rrbracket$ then we have $j \leq pre$:

$$\begin{tikzcd}
Q \arrow[r, phantom, "="] \arrow[d, equal] \arrow[dr, "\leq"] & Q \arrow[dr, "j"] & \\
Q \arrow[dr, "\leq"] & P \arrow[r, "pre"] \arrow[d, "\leq"'] & X \arrow[d, "\mathcal{D}\llbracket C \rrbracket"'] \\
& A \arrow[r, "\llbracket \phi \rrbracket"] & X
\end{tikzcd}$$

Finally we need to check that the preimage of a subobject is also a subobject. To recall, in the diagram below:

$$\begin{tikzcd}
P \arrow[r, "f'"] \arrow[d, "g'"'] & B \arrow[d, "g"'] \\
A \arrow[r, "f"] & C
\end{tikzcd}$$

in our semantics we take g to be $\mathcal{D}\llbracket C \rrbracket$ and f to be $\llbracket \phi \rrbracket$; f' to be our result and g' to be the arrow witnessing the fact that $\mathcal{D}\llbracket C \rrbracket.f' \leq \llbracket \phi \rrbracket$. Hence this amounts to showing that

Proposition 5.1.3 *If f is a subobject arrow in the above pullback square, then f' is also.*

Proof We shall do this in two stages. The first stage, answering exercise 1.8.2. of [18], amounts to showing that f monic implies f' monic. The second stage is showing that the transition from f to f', mapping monic arrows with codomain X to monic arrows with codomain X, preserves the equivalence relation $\equiv$, i.e. it is a valid operation taking subobjects to subobjects.

To show the former we assume $f'.a = f'.b$. Then clearly $g.f'.a = g.f'.b$. Then by commutativity of the pullback square we have $f.g'.a = f.g'.b$. Then by the fact that f is assumed monic we have $g'.a = g'.b$. This implies commutativity of the diagrams

$$\begin{tikzcd}
Q \arrow[r, phantom, "="] \arrow[d, equal] \arrow[dr, "a"] & Q \arrow[dr, "g'.a = g'.b"] & \\
Q \arrow[dr, "f'.a = f'.b"] & P \arrow[r, "g'"] \arrow[d] & X \arrow[d, "f"'] \\
& A \arrow[r, "g"] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
Q \arrow[r, phantom, "="] \arrow[d, equal] \arrow[dr, "b"] & Q \arrow[dr, "g'.a = g'.b"] & \\
Q \arrow[dr, "f'.a = f'.b"] & P \arrow[r, "g'"] \arrow[d] & X \arrow[d, "f"'] \\
& A \arrow[r, "g"] & X
\end{tikzcd}$$

but by unicity there is only one k in the above such that

$$\begin{tikzcd}
Q \arrow[r, phantom, "="] \arrow[d, phantom, "||"] \arrow[dr, "k"] & Q \arrow[dr, "g'.a = g'.b"] & \\
Q \arrow[dr, "f'.a = f'.b"] & P \arrow[r, "g'"] \arrow[d, "f'"] & X \arrow[d, "f"] \\
& A \arrow[r, "g"] & X
\end{tikzcd}$$

and so it follows that $a = k = b$ as required.

To show that the operation $f \mapsto f'$ preserves the relation $\equiv$ on monics it is sufficient to show that it preserves the operation $\leq$ since then $f_1 \equiv f_2 \Rightarrow f_1 \leq f_2 \wedge f_2 \leq f_1 \Rightarrow f_1' \leq f_2' \wedge f_2' \leq f_1' \Rightarrow f_1' \equiv f_2'$.

To show this suppose $f_1 \leq f_2$. Then $f_1 = f_2.r$. Hence the following trapezium commutes

$$\begin{tikzcd}
P \arrow[r, "g'"] \arrow[d, "f_1'"'] & B \arrow[r, "r"] \arrow[d, "f_1"'] & D \arrow[dl, "f_2"'] \\
A \arrow[r, "g"] & C &
\end{tikzcd}$$

and so $(f_1', r.g')$ is a (f_2, g) pullback-shape cone. Then since (f_2', g'') is the pullback of (f_2, g) we have

$$\begin{tikzcd}
P \arrow[r, phantom, "="] \arrow[d, phantom, "||"] \arrow[dr, "!k"] & P \arrow[dr, "r.g'"] & \\
P \arrow[dr, "f_1'"] & Q \arrow[r, "g''"] \arrow[d, "f_2'"'] & D \arrow[d, "f_2"'] \\
& A \arrow[r, "g"] & C
\end{tikzcd}$$

Hence it follows in the above that $f_1' = f_2'.k$ i.e. $f_1' \leq f_2'$ as required.

□

Thus, the operation sending $[\![\phi]\!]$ to the pullback of $[\![\phi]\!]$ under $\mathcal{D}[\![C]\!]$ is indeed a well defined one mapping subobjects to subobjects. (Indeed, is this result that allows the subobject assignment $\mathsf{Sub} : \mathbf{C} \rightarrow \mathbf{Set}$ — sending an object of $\mathbf{C}$ to its set of subobjects — to be extended to a contravariant functor. If $f \in\mathbf{C}(X, Y)$

then we can define $\mathsf{Sub}(f) \in \mathbf{Set}(\mathsf{Sub}(Y), \mathsf{Sub}(X))$ as a function that takes a subobject of Y and pulls it back under $f : X \to Y$ to a subobject of X. This assignment on arrows is indeed functorial [12].)

Hence to deal with the modal operator we need to assume *pullbacks* of our category in question.

5.2 Subobjects and Pullbacks in our Categories

To recall, to understand the meaning of logical formulae in our structure we require pullbacks, and the fact that the subobject poset has the structure of an orthocomplemented lattice.

5.2.1 Subobject Structure and Pullbacks in FdHilb and Rel

FdHilb

We show that **FdHilb** does indeed satisfy these conditions. Subobjects in **FdHilb** correspond to subspaces of that vector space (to see this we note that a monic arrow in **FdHilb** is an injective one — with the same proof as **Set** — and the image of a space under a linear map is indeed a subspace.) Exhibiting semantics of logic as subspaces is linked to traditional quantum logic, as in e.g. Gleason's Theorem [11]. In such a structure we do indeed have conjunction (given by the intersection of two subspaces, which is also a subspace) and disjunction (given by the direct sum of the two subspaces, which is also a subspace, and indeed the least upper bound of the subspaces.) Furthermore we have top and bottom subspaces, represented by the original space and the zero subspace $\{\mathbf{0}\}$ respectively. Negation is given by the orthogonal complement of the subspace, $A^{\perp} = \{\psi \in H | \forall \phi \in A, \langle \psi | \phi \rangle = 0\}$ and this does indeed satisfy the order-theoretic requirements above (indeed, the intersection between A and $A^{\perp}$ is the zero subspace, and their direct sum is the top subspace; we also have involution and contravariance with respect to inclusion.) As mentioned, distributivity fails in **FdHilb** — for a counterexample see e.g. [11]. It is this lack of distributivity that has helped to shape traditional quantum logic, and again relates to the earlier point that finding a logic to express the quantum part of the state is a nontrivial matter.

Note that finally we do have pullbacks in **FdHilb** since (informally) the linear preimage of a subspace is also a subspace. More formally, we know that **FdHilb** has both products and equalisers (the equaliser of f and g is the kernel of $f - g$, which is indeed a vector space,) and so it has all finite limits (e.g. appealing to theorem 1.9.7 of [18]) and in particular pullbacks (also, **FdHilb** has coproducts and coequalisers and so also has all finite colimits.)

Rel

In **Rel**, given any relation $R : A \to B$ we can construct a map $R' : P(A) \to P(B)$ sending a set $C \subseteq A$ to those $b \in B$ such that cRb for some $c \in C$. An arrow R is monic in **Rel** iff the function R' is injective.

To see this, given any $x \in P(A)$ we can construct $x' : \{*\} \to A$ relating $*$ to those elements of A in x. Hence if R is monic then $R'(x) = R'(y)$ iff $R.x' = R.y'$ implying $x' = y'$ and so $x = y$. Conversely, if R' is injective then $R.H = R.G$ implies that for all x we have $(R.G)'(x) = (R.H)'(x)$ i.e. $R'.H'(x) = R'.G'(x)$ so $G'(x) = H'(x)$ and so by generalisation $G' = H'$ and so $G = H$. We use here the fact that the $R \mapsto R'$ construct is clearly compositional.

Hence a monic arrow with codomain A represents an injective map with codomain $P(A)$, i.e. subobjects on A represent predicates on subsets of A, i.e. families of subsets of A (the same type one finds in general topology.) For subobjects we have $S \leq R$ iff

$$\begin{tikzcd}
\mathrm{B} \arrow[r, "R"] & \mathrm{A} & & \mathrm{P(B)} \arrow[r, "R'"] & \mathrm{P(A)} \\
\mathrm{C} \arrow[u, "T"] \arrow[ur, "S"] & & & \mathrm{P(C)} \arrow[u, "T'"] \arrow[ur, "S'"] &
\end{tikzcd}$$

i.e. intuitively if the predicate S' on $P(A)$ is contained in the predicate R'. Thus, the ordering on these predicates is the natural one we expect. Since we are dealing with once again a power-set structure, we find we have our orthocomplemented lattices (in fact in this case Boolean Algebra) constructions since subobjects of A correspond precisely to subsets of $P(A)$. Hence we have our order-structure required in **Rel**.

We only now need to check we have the modal structure, i.e. the pullbacks. Unfortunately the category **Rel** is not equipped with pullbacks and is hence unable naively to deal with the modal structure. To see this, if **Rel** were to have pullbacks then it would have equalisers [18]. Consider the scalars $0 = \emptyset$ and $1 = \{(*,*)\}$ in **Rel** of type $\{*\} \to \{*\}$. An equaliser of 0 and 1 is an arrow $r : A \to \{*\}$ such that $r;0 = r;1$ and given any $k : B \to \{*\}$ with $k;0 = k;1$ we have a unique $h : B \to A$ s.t. $h;r = r$. Assume such an equaliser (A, r) exists. Then $r;0 = 0$ and $r;1 = r$ and so $r = 0$. Then consider $k : \{*\} \to \{*\} = 0$. Then $0;0 = 0;1$ and so there is a unique $h : \{*\} \to A$ such that $h;0 = 0$ since $r = 0$ is the equaliser. But there are in fact two such h — 0 and 1.

It seems, then, that whilst **Rel** does have all of the *quantum* structure for performing our semantics, it does not have the full logical structure (note the fact that we need pullbacks has nothing to do with the quantum structure itself; merely to do with the modal operator for the logical semantics.) However, while **Rel** does not have pullbacks, we note that it represents the relations over a category with lots of very strong structure, **Set** (**Set** is a topos [14], a category containing strong structure e.g. powersets; and **Rel** is the category of relations over this topos.) Perhaps it would be possible then to weaken our requirement from pullbacks to something that **Rel** does satisfy as a result of this strong structure, and this could indeed be a further direction to head. However pullbacks do very naturally seem to be the thing to use to represent the modal operator in our logic, as we have seen.

5.2.2 General SCCCB Structure

Here we investigate whether any of our required structure on the subobject poset comes immediately from the SCCCB structure alone, and any extra conditions we may need for it to do so.

Top

We note that there is automatically a top element in any subobject poset. The subobject generated by the (clearly monic arrow) $id : X \to X$ is clearly the top subobject of X, by commutativity of the diagram

$$\begin{tikzcd}
X \arrow[r, "id"] & X \\
A \arrow[u, "f"] \arrow[ur, "f"] &
\end{tikzcd}$$

Bottom

Likewise in presence zero object we have a $\perp$ subobject $0 : Z \to X$. 0 is monic since $0.x = 0.y$ implies $y = x$ since x and y must both have type $A \to Z$ for some A and Z is terminal. Then for any subobject f we have commutativity of the following diagram:

$$\begin{tikzcd}
X \arrow[r, "f"] & X \\
Z \arrow[u, "0"] \arrow[ur, "0"] &
\end{tikzcd}$$

since Z is initial and hence 0 is the unique arrow $Z \to X$. Thus, $f \geq 0$ as required.

Disjuntion

Note that in the case of **FdHilb** the least upper bound of subspaces A and B is given by $A \oplus B$. Since we have biproducts generally available, it seems clear that we should investigate whether we can in general use them to give us our least upper bounds.

We temporarily assume that if a and b are monic so is $[a, b]$. Then given two subobjects a and b of X we have a subobject $[a, b]$ that is greater than both of them

$$\begin{tikzcd}
A \arrow[r, "a"] \arrow[d, "q_1"'] & X \\
A \oplus B \arrow[ur, "{[a,b]}"] & B \arrow[u, "b"'] \arrow[l, "q_2"']
\end{tikzcd}$$

And furthermore given any subobject c above them both, $[a, b]$ is above c

$$\begin{tikzcd}
A \arrow[r, phantom, "="] \arrow[d, phantom, "||"] & A \arrow[dr, "f"] & \\
A \arrow[r, "a"] \arrow[d, "q_1"'] & X & C \arrow[l, "c"'] \\
A \oplus B \arrow[ur, "{[a,b]}"] \arrow[d, phantom, "||"] & B \arrow[l, "q_2"'] \arrow[u, "b"'] \arrow[ur, "g"] & \\
A \oplus B \arrow[r, phantom, "="] & A \oplus B \arrow[r, phantom, "="] & A \oplus B \arrow[uu, "{[f,g]}"]
\end{tikzcd}$$

We just need to check then that $[a, b]$ is monic assuming a and b are. Unfortunately this is not generally the case, e.g. in **FdHilb**: an arrow in **FdHilb** is monic iff it is injective (with a similar proof to that in the category **Set** with the monoidal unit substituted for the terminal object.) A linear transformation is injective if the dimension of the output is at least the dimension of the input; i.e. as long as the image of the basis vectors are linearly independent. This is if the columns of the matrix representation are linearly independent. Then given $a : A \to B$ and $b : C \to B$ with matrix representations m_a and m_b respectively we note that the matrix of $[a, b] : A \oplus C \to B$ is given by the matrix of m_a placed horizontally next to the matrix of m_b. Since the union of two linearly independent sets is not necessarily linearly independent, it does not follow that a monic and b monic implies $[a, b]$ monic.

This problem can, in fact, be solved in the category **FdHilb**. Consider an object A and a map $f : B \to A$ that we would like to represent a subobject, but cannot for f fails to be monic. Then the matrix representation of f contains some columns that are linearly dependent. We can hence take a maximal linearly independent subset of these columns (which has the same *span* as the original vectors.) This gives us a matrix $f' : C \to A$ where C is a subspace of B. Then the maps f' and f have the same range (since their column vectors have precisely the same span, and so the image of the basis vectors are the same):

$$\begin{tikzcd}
A \arrow[r, "f"] & X \\
B \arrow[u, "c"] \arrow[ur, "f'"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "d"'] & X \\
B \arrow[ur, "f'"] &
\end{tikzcd}$$

i.e. $f \equiv f'$ in the ordering on subobjects (if f were to be monic.) Since we have the ability to construct a monic f' out of any f by the reasoning above in **FdHilb**, we can define the union of two subobjects a and b to be $[a, b]'$ and find that a slightly expanded version of our commuting diagram above still holds (due to the range equivalence via c and d.) Of course, we have not made any comments here as to whether this monicisation occurs in general SCCCBs, but with this extra assumption we could define disjunctions of subobjects this way. We note that **FdHilb** is not the only place this can be done — indeed, in a topos any arrow can be decomposed into an epic arrow followed by a monic arrow [14]. However **FdHilb** is not a topos — if it were, it would admit a distributive subobject structure (the subobject structure of any object in a topos is automatically a bounded distributive lattice, and is almost a Boolean algebra except we do not necessarily have $a \vee \neg a = \top$. That is, general topoi admit intuinistic logic [14].)

Conjunction

For conjunction, we note that if the category in question has pullbacks then we can use the pullback of one subobject under another to get their intersection (and this will result in a subobject by reasoning above:)

$$\begin{tikzcd}
\mathrm{A} \arrow[r, "a"] & \mathrm{X} \\
\mathrm{P} \arrow[u, "p_a"] \arrow[r, "p_b"] & \mathrm{B} \arrow[u, "b"]
\end{tikzcd}$$

Thus $a \wedge b = a.p_a = b.p_b$. Since p_a and p_b are monic we see that P is (the source of a) subobject of both A and B. The universal property shows that this is the greatest of the lower bounds. Hence if our category does have pullbacks then we do indeed get the conjunction part of the structure for free (and since we also have the top object for free as above, in any category with pullbacks the subobject structure is automatically a *semilattice*.)

5.3 Subobjects and our Semantics

5.3.1 Subobjects and the CPM(-)$^{\oplus}$ construction

We now proceed to examine how the logical semantics make sense with respect to our specific categories, objects and formulae. The above intuitions imply that we might represent formulae by subobjects of our state space $X = S \otimes F(H)$. The first obvious issue relates to how this makes sense: In the concrete quantum case our formulae were predicates on $S \times Q^q$ and the probabilistic nature of the system was only inherent in our modal operator, which converted probabilistic ideas into Boolean ones with formulae "is the probability of this happening greater than rational number q?". Converting this directly into our abstract metalanguage implies that we should be working with subobjects of $S \otimes H$, but this doesn't make sense as S is an object in the **CPM(C)**$^{\oplus}$ category while H is not. In definition of our operational semantics we split the classical and quantum parts, so maybe this is wise here. There is, however, no canonical way of decomposing a subset of $S \times H$ losslessly into a subset of S and a subset of H (this is clear visually considering S and H to be one-dimensional.) Unfortunately, then, we seem to have reached a problem.

Or have we? What if we were to treat predicates as subobjects of X in **CPM(C)**$^{\oplus}$? Is this fundamentally flawed, or is there a perspective for which this does make sense? We note that here the logical semantics would now directly talk about probability distributions directly, unlike our concrete treatment. A natural place to investigate this idea is the **FdHilb** case. Here an element of X consists of a number of subdistributions (one for each of the k^m possibilities of the classical state space.) A subspace of X then represents of some predicate on such distributions. If our $\mathbf{CPM(C)}^{\oplus}$ category has orthocomplemented subobject lattices and pullbacks, then we can have logical and modal operations on such predicates as we would like. The main unaddressed issue is then the nature of the atomic formulae.

An atomic formula of the kind "the classical value is i" could be represented by the arrow $q_i \otimes id : S \otimes F(H) \to S \otimes F(H)$, or better the subobject generated by $q_i : F(H) \to k^m.F(H) = X$. We note that q_i is indeed monic. An element of X then satisfies this subobject if it is equal to $q_i.\phi$, i.e. is $\langle 0, \ldots, 0, \phi, 0, \ldots, 0\rangle$. Intuitively this means "all of the possible states with a nonzero possibility have classical value i" and if ψ is normalised/total with $tr(\psi) = 1$ then $\psi \leq q_i$ does indeed correspond to the classical value being i with certainty for element $\psi : I \to X$. We can then generalise this to $q_i : n.F(H) \to k^m.F(H)$ meaning the value of the classical state is in the set $\{1, \ldots, n\}$. This allows us to provide semantics to all formulas over distributions of the type "the classical part of any non-zero state in the distribution satisfies this classical predicate."

What then, about formulas of the type "we know with (at least) probability q that the classical state is in one of these possibilities", as in the concrete case? Well unfortunately this is slightly trickier. We can use addition to sum together predicates of the above to see if two certain classical predicates occur with e.g. exactly the same probability, or with some exact particular weighting that we chose to give. However we cannot do so with ranges of probability, which is what we need here. To do this we would require an arrow that has range only elements with trace greater than q, but no such map can exist since all maps must be linear and hence send 0 to 0 etc. This is a necessary limitation we have that is similar to those encountered before: we are seeking to internalise concepts and we can only internalise concepts that are linear because of the constraints of the categorical structure (and in particular since e.g. **FdHilb** needs to be an example of such a structure.)

However note all is not lost here: in many program specifications, formulas are likely to need to be of the form "this thing has definitely happened" as opposed to "this thing has probably (to degree q) happened" — this is the very definition of correctness, which is generally what is sought by such formal methods. Hence we are (mostly) satisfied with our logic at this level dealing with probability distributions, but only being able to talk about the probability distribution being certainly in a specific state. Note this is far from our notions of completeness given with regards to our concrete semantics.

5.3.2 Abstract Logical Semantics

Thus, we can express formulas of the form

$$\phi ::= \top \mid \phi \wedge \phi \mid \neg\phi \mid \mathsf{variable} = \mathsf{value} \mid \langle C \rangle \phi$$

Intuitively, a distribution satisfies $\mathsf{variable} = \mathsf{value}$ iff the classical part of all non-zero states in the distribution satisfy this; and a distribution satisfies $\langle C \rangle \phi$ if after applying $\mathcal{D}[\![C]\!]$ to the distribution, ϕ holds. We formally give the semantics of the formulae as subobjects of X. For the first three formulae types we use the logical structure (e.g. $[\![\phi \wedge \psi]\!] = \mathsf{glb}([\![\psi]\!], [\![\phi]\!])$.) For the atomic formula we use the subobject $q_{x,v} : F(H) \to X$ and we define $[\![\langle C \rangle \phi]\!]$ as the pullback of ϕ under $\mathcal{D}[\![C]\!]$, as above.

Note here we have defined formulas as subobjects of X, but we have not specified the category in question: we could require subobjects of X in $\mathbf{CPM(C)}^{\oplus}$, $\mathbf{SUP(C)}^{\oplus}$ or even $\mathbf{C}^{\oplus}$. Well there is no need for our subobject injection mappings to be trace-decreasing, or even completely positive. We can clearly view $\mathbf{CPM(C)}^{\oplus}$ as sitting inside the category $\mathbf{C}^{\oplus}$ (commutative monoid enrichment does not require complete positivity.) Then given any subobject f of X in the category $\mathbf{C}^{\oplus}$ we have defined above what it means for an element $I \to X$ to satisfy f, and this still holds for completely positive elements. Likewise $\mathcal{D}[\![C]\!]$ is certainly an arrow in the category $\mathbf{C}^{\oplus}$ also, as is $q_{x,v}$, and furthermore $q_{x,v}$ is monic in this category. Thus we define the meaning of a formula to be a subobject of X in the category $\mathbf{C}^{\oplus}$. To interpret our logical semantics we require that the category $\mathbf{C}^{\oplus}$ has pullbacks and orthocomplemented subobject lattices.

But there is more: The category $\mathbf{C}$ has biproducts by assumption, and so $\mathbf{C}^{\oplus}$ is the biproduct completion of a category with biproducts.

Proposition 5.3.1 *If the addition in* $\boldsymbol{C}$ *is from a biproduct structure, then we have a categorical equivalence* $\boldsymbol{C}^{\oplus} \cong \boldsymbol{C}$

Proof To show equivalence [2] we must exhibit a full and faithful functor $F : \mathbf{C} \to \mathbf{C}^{\oplus}$ that is essentially surjective. Well as mentioned $\mathbf{C}$ sits as a full subcategory of $\mathbf{C}^{\oplus}$ — explicitly this functor maps object A to the singleton $< A >$ and an arrow f to the 1 by 1 matrix with element f. By definition of composition and identities in $\mathbf{C}^{\oplus}$ this is clearly a functor, and it is clearly full and faithful. Hence we need only show that F is essentially surjective, i.e. that for every object B in $\mathbf{C}^{\oplus}$ there is an object A in $\mathbf{C}$ such that $F(A) \cong B$.

Well consider the object $B = \langle B_1, \ldots, B_n \rangle$ in $\mathbf{C}^{\oplus}$. Then A is a biproduct of $B_1, \ldots, B_n$. But so is $\langle B_1 \oplus \ldots \oplus B_n \rangle$ since $\mathbf{C}$ sits inside $\mathbf{C}^{\oplus}$ as the full subcategory of singleton objects. Hence they must be isomorphic, by standard results [2]. Thus $B \cong F(B_1 \oplus \ldots \oplus B_n)$, as required. (Explicitly then, the homotopic-inverse functor to F if the functor G sending $\langle B_1, \ldots, B_n \rangle$ to $\langle B_1 \oplus \ldots \oplus B_n \rangle$.)

□

Thus the category $\mathbf{C}^{\oplus}$ is isomorphic to the category $\mathbf{C}$ up to internal isomorphism. We do not get a genuine categorical isomorphism here (an isomorphism of categories "on the nose," as it were) since for one thing the biproducts in the category $\mathbf{C}$ are not guaranteed to be "free" and thus may yield additional equalities.

Thus then if $A = F(B)$ then the subobject lattice of A is isomorphic to the subobject lattice of B since F is a categorical equivalence (arrows with codomain $\langle A_1, \ldots, A_n \rangle$ in $\mathbf{C}^{\oplus}$ corresponds to arrows with codomain $A_1 \oplus \ldots \oplus A_n$ in $\mathbf{C}$, and vice-versa.) Thus $\mathbf{C}^{\oplus}$ has orthocomplemented subobject lattices iff $\mathbf{C}$ does. Similarly isomorphism preserves pullbacks, and thus $\mathbf{C}^{\oplus}$ has pullbacks iff $\mathbf{C}$ does. Hence for our above logical semantics to make sense we require that the category $\mathbf{C}$ itself has pullbacks and orthocomplemented subobject lattices, and this is something that we have already investigated.

We make one final comment. We have hence seen that we can model meanings of formula as subobjects of our space X, and in the special case of **FdHilb** this is subspaces, as mentioned. In our concrete exposition of the logical semantics above, we were primarily interested in subsets of $S \times Q^q$ — indeed a subspace of this did not make sense since it is only the latter structure that is a Hilbert space. However, in the **FdHilb** special case of our abstract semantics the classical component S has been made into a Hilbert space, and so then we can talk about subspaces of the compound system $S \otimes F(Q^q)$, i.e. subspaces of $k^m.F(Q^q)$, linking with classical quantum logic [8] which the direct concrete case does not.

5.4 An alternative formulation

In fact there is no reason for the logical semantics to be internal. Indeed, since we have "concretisation" via. elements $I \to X$ we can define a predicate on A to be a subset of $\mathbf{C}(I, A)$. Note then we automatically have an orthocomplement structure (in fact a Boolean algebra) on $\mathcal{P}(\mathbf{C}(I, A))$ as the usual powerset ordering. Note then predicates relate to subsets rather than subspaces, which violate intuitions of traditional quantum logic. The former approach using subobjects could be compared to the denotational semantics (where we *internalise* the meaning of a program by the arrow in the category) while this latter is more like the operational semantics (using rules that can involve external operations on arrows.)

Here we define the meaning of formulae as subsets of $\mathbf{CPM(C)}^{\oplus}(I, X)$. The meaning of the logical formulae in terms of their subcomponents should be clear — the interesting cases are the atomic formula case and the modal operator. Clearly then as in the concrete case $[\![\langle C\rangle\phi]\!] = \{s : \mathcal{D}[\![C]\!].s \in [\![\phi]\!]\}$. As mentioned above the atomic formula "variable = value" holds for a probability distribution if it is known to hold *with certainty* in that distribution. Thus, $[\![v = x]\!]$ holds precisely for arrows $I \to X = S \otimes F(H)$ of the form $q_{(v,x)} \otimes \phi$ for some ϕ. We would not be limited by this though, as we could have arbitrary predicates on distributions as we would not have the linearity constraint as in the internal case — predicates can be represented by general subsets rather than subspaces.

Note that the subobject approach above corresponds more to the denotational semantics — internalising our ideas with the use of subobjects — and this latter approach corresponds to a more operational semantics, using the hom-set spaces themselves to represent our elements; and once again this is very much (too much in this case?) like the concrete treatment (and clearly this concrete treatment will make sense in any category; we do not require pullbacks etc which proved too strong in one of the cases above.)

6 Conclusions

To recap, we have firstly expanded the treatment of the simple imperative quantum programming language in [3] giving full proofs of correspondance results between the operational semantics and denotational semantics, as well as considering the program logics in more depth and providing a few basic results regarding their interaction with the other semantics; and a worked example. We have then used the categorical quantum notions of strongly compact closed categories with biproducts, together with cpo-enrichment of the superoperator category, to express the semantics of this language at this abstract level; of which the concrete semantics can be seen as a special case. Finally we have investigated how we might also raise the logical semantics to the abstract level using ideas of categorical logic (and in doing this found that this was not a direct abstraction of our concrete logic semantics, but related more so to traditional quantum logic.)

Looking back on the dissertation, the author feels that it has indeed been a reasonably successful endeavour. The expansion of the talk [3] was a successful and enlightening one, in particular a) proving the result for the `while` case using approximations etc. and b) performing the whole framework in a more unfamiliar probabilistic setting, leading to some quite subtle points. At the abstraction stage, we represent the quantum part using the categorical axiomatics of quantum mechanics; and working with these has indeed been pleasant — partially due to the diagrammatic 2D notion of working with compact closed categories developed by Samson Abramsky and Bob Coecke. For the classical side of the abstract case, the very natural route following [6],[24] and [23] was to use biproducts; and indeed the free biproduct completion. Doing such actually kind of reduces the abstract case to the concrete case, since what we assume effectively amounts to classical functions. However, this has still been interesting, in particular formulating how to interact between this classical notion and the abstract quantum axiomatics. From the point of view of the abstract semantics (including the logical semantics,) one of the interesting issues corresponds to *internalisation* — to represent a construct as an arrow in the category (via e.g. closure) that construct must satisfy the categorical axiomatics, which in particular, for example, requires that all arrows be linear; and thus non-linear concepts generally require some trick to be internalised. This is not the case, for example, in categories such as **Set**.

6.1 Further Directions

One potential further direction is to consider how we might implement our operational semantics. Clearly this can be done for the concrete operational semantics easily in a very "real" way — assuming we have a quantum computer QRAM machine we can simply run the reduction as above, and it will all run smoothly (assuming of course idealised hardware etc, as we have throughout.) Otherwise we can simulate the quantum state (using internal representations of complex numbers, modulo finite precision ideas etc) and for measurements use e.g. a pseudorandom generator to perform the correct action at the required probability. We can also create our simulator of the abstract operational semantics at a *categorical* level. Thus, we could have a virtual machine that takes in a representation of the category **C** as an input and then "runs" the operational semantics of a given program by accessing operations within **C**. We can then provide example categories **C** such as **FdHilb** that we can plug into our virtual machine that would then run as in the concrete case. Thus, our machine will be abstractly organised in the same way as our abstract semantics. Ideas for representing categories computationally in this manner come from [21], which fundamentally exploits the observation that many theorems in category theory have constructive proofs (e.g. existence of initial algebras of ω-continuous functors [2], a direct generalisation of our domain theory fixed point theorem mentioned above [4].) As a base type, for example, we could have `type (a,o) cat = { dom : a → o ; cod : a → o ; comp : a → a → a ; id : o → a }`.

As mentioned in the original talk [3] we could make the language slightly more structured, introducing types etc, making the language more functional. It is these typed languages that fit into ideas of category theory more easily, and would perhaps give a different flavour to our language and its semantics. Fundamentally, though, these semantics will still be the same: using the cpo-enrichment for recursion, biproducts and CPM construction to represent the spaces in general etc; it's just effectively the values of k, m and the number of qubits could be varied. Of course, once this has been done, we could extend our ideas to other programming language ideas: the main issue that our framework above does *not* allow is that of infinite (classical) datatypes since we represent our space by a finite biproduct. However, perhaps this will be solvable by using the *infinite biproduct completion* of a monoid-enriched category, should such an animal exist. Indeed, this implies that the finite classical state space was perhaps not so necessary after all. Fitting in other ideas from general programming language theory [25] could be automatically obtained by the monad ideas mentioned above.

As in a language in [4] we could include arbitrary primitive commands in our language (as well as our primitive Boolean and arithmetic expressions.) We could have classical basic commands (with primitive denotations consisting of classical arrows $S \to S$), quantum commands (with denotations $H \to H$) or just general commands that alter the probability distribution ($X \to X$), although in this case we would lose the ideas of the operational semantics and thus correspondance theorems.

The categories that we use to model our programs require stronger structure than the basic SCCCBs, as has been mentioned (such as cpo-enrichment and its interaction with the other features, as well as the logical semantic requirements.) We have already investigated to some degree which of the logical semantic requirements follow automatically from the SCCCB structure; investigating independence of all of our requirements on the category could be a further area of study to investigate.

Our language clearly deals with a single, independent quantum system. A further area of research is the delicate interaction between quantum systems and ideas of concurrency; and seeing if we can model these ideas into our language. Also, from a practical point of view, we have of course assumed in the above very idealised hardware and have assumed away realistically necessary ideas such as error-correction. Seeing how these ideas fit into our framework could also be another potential direction.

We could give a detailed presentation of combining our abstract semantics for a program describing the quantum teleportation protocol with the reasoning in [6]. We would make further assumptions on the category as [6] does (e.g. a teleportation basis) and show that in this case the semantics of our language gives the arrow representing teleportation in [6], which we know to be correct by reasoning in [6].

Finally, we could continue investigation of subobjects, pullbacks and our logical semantics. Combining a category with biproducts and the lattice structure on subobjects effectively gives us an Abelian category, and adding in the strong compact closure further leads us towards categories that have been studied before. From a more theoretical point of view, investigating this interaction could also be a direction of interest. In particular, we have noted that while **Rel** does not support pullbacks (which we need for the abstract semantics of our modal operator) it is the category of relations over a topos, and we could try to find some level of abstraction satisfied by this that allows us to define our modal operator that is also satisfied by **FdHilb**.

7 Acknowledgements

First and foremost I'd like to thank my supervisor Samson Abramsky for his invaluable help and support with this dissertation and the background areas to it; and in particular for odd comments that have often led to new insights. I'd also like to thank my general supervisor Luke Ong for his guidance throughout the year, as well as Michael Collins for very useful email correspondance. Also thanks are owed to the many Oxford course lecturers, tutors and fellow students over the past four years who have helped to make very interesting subjects even more of a delight to explore.

This dissertation was produced in LaTeX using Paul Taylor's commutative diagrams package and the PIC preprocessor language for diagrams.

Finally, my MSc in MFoCS is gratefully supported by an EPSRC grant, and I thank Michael Collins and the MFoCS committee for helping to make this possible.

8 References

[1] S. Abramsky, *Axiomatics of Cloning and Deleting* lecture slides.

[2] S. Abramsky, *Categories, Proofs and Programs* Oxford University lectures notes.

[3] S. Abramsky, *A Cook's tour of a simple quantum programming language* lecture slides.

[4] S. Abramsky, *Domain Theory* Oxford University lecture notes.

[5] S. Abramsky and B. Coecke, Abstract Physical Traces, in *Theory and Applications of Categories*, vol 14, 111–124, 2005.

[6] S. Abramsky and B. Coecke, *A categorical semantics of quantum protocols*, in: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, LICS 2004,* IEEE Computer Society Press, 2004, pp. 415–425.

[7] A. Baltag and S. Smets: *Complete Axiomatizations for Quantum Actions*, International Journal of Theoretical Physics, 44(12): p.2267–2282, 2005.

[8] G. Birkhoff and J. von Neumann. (1936) The logic of quantum mechanics. *Annals of Mathematics* 37, 823-843.

[9] B. Coecke *De-linearizing Linearity: Projective Quantum Axiomatics From Strong Compact Closure.* Electronic Notes in Theoretical Computer Science 170: 49–72 (2007).

[10] B. Coecke, Kindergarten Quantum Mechanics. Lecture notes, 2005.

[11] B. Coecke, *Quantum Computer Science* Oxford Univeristy lecture notes.

[12] P. Freyd and A. Scedrov, *Categories, Allegories.* North-Holland 1990.

[13] J. Gibbons, Calculating Functional Programs. *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction* 2000: 149–202.

[14] R. Goldblatt, *Topoi, the Categorical Analysis of Logic*, Studies in Logic volume 98.

[15] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and Reasoning about Systems*, Cambridge University Press 2000.

[16] A. Joyal and R. Street, *The geometry of tensor calculus I*, Advances in Mathematics 88 (1991), pp. 55–112, as referred to in [23].

[17] G. M. Kelly and M. L. Laplaza. Coherence for compact closed categories. *Journal of Pure and Applied Algebra* 19, 193–213, 1980.

[18] B. Pierce, *Basic Category Theory for Computer Science*, Foundations of Computing series, MIT press.

[19] Poigne, A. *Basic Category Theory.* In: *Handbook of Logic in Computer Science* (Ed. S.Abramsky, D.Gabbai, T.Maibaum), Oxford University Press, 1992, p.413–634.

[20] H. Priestly, *Algebraic and Relational Methods in Propositional Logics* Oxford University lecture notes.

[21] D. Rhydeheard and R. Burstull, *Computational Category Theory.* Prentice Hall 1988.

[22] P. Selinger. A brief survey of quantum programming languages. In: Proceedings of the 7th International Symposium on Functional and Logic Programming. *Springer-Verlag Lecture Notes in Computer Science* 2998.

[23] P. Selinger. Dagger compact closed categories and completely positive maps (extended abstract) *Electronic Notes in Theoretical Computer Science* 170: 139–163 (2007).

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

[25] M. Spivey, *Programming Languages* Oxford Univeristy lecture notes (Hilary Term 2005).