CPS and Simulation of Quantum Processes

Jerzy Karczmarczuk
University of Caen, France
karczma@info.unicaen.fr

Abstract

We continue the discussion of a proposed formerly functional formalism for simulation of quantum discrete systems. We underline the relevance of continuations, and in particular — of linear continuations for the simulation of quantum evolutive systems, notably of the "quantum circuits" supposed to constitute the building bricks of future quantum computers. Our simulation framework implements quantum states and operators through functional objects, so the relation between the chain of transformations of states and continuations is natural. The linearity is rather obvious as well: quantum states are never reused.

[This is a preliminary, **draft** version; a "stream of thought" rather than an article.
The problem is that I am not James Joyce. . .]

1 Introduction

Computer scientists continue their interest in the domain of *quantum computing* and its simulation — see e.g., [1, 2, 3] and many others — for several reasons, despite a rather slow (although steady) recent progress. Independently of the hope that the fabulous speed-up of some hard algorithms like those of Shor or Grover [4, 5], and of the possibility that one day quantum *programmable devices* will help to simulate other quantum systems [6, 7], etc., there are some other interesting points.

The epistemological status of *quantum information* and of the fact that obtaining this information destroys the underlying structure, is still not entirely clear, and some implementable constructions, even if they don't "solve the quantum mystery", at least help to shorten the gap between our formal understanding of quantum entities, and the sense of their models. Besides, the theory of algorithms realizable on quantum devices, and related matters are *per se* interesting study topics.

In [8] we attempted to implement in a purely functional language Haskell an elementary formalism permitting to "put into the computer" the typical quantum structures: states and operators, using functional objects. Our approach tends to be as anti-speculative as possible, and relies on established universal properties of quantum theory, we simply "jumped into" the category of Hilbert spaces in order to implement some standard quantum calculi, usually performed on paper. It turns out that within this framework, the chain of transformations of quantum states bears a striking resemblance with the continuation-passing protocol. While this is a preliminary work, and not everything is clear, this resemblance is obviously not accidental.

We will arrive at CPS in a natural, almost automatic way, using quite universal (and thus rather weak) properties of functions representing quantum entities. The main motivation to present this work is that we haven't seen this analogy in easily accessible literature, and our ambitions are rather modest. We don't plan to add anything really new to the theory of programming with continuations, but on the contrary, we hope

that this protocol turns out to be a promising approach in applications related to the modelling of quantum circuits.

2 Quantization, the Functional Way

The configuration (state) of a classical system is a set α of parameters: the position $\vec{p}$ of a particle, the excitation level n of an atom, the values $\downarrow$ or $\uparrow$ of a dichotomic variable q which represents some realization of a bit, etc. In the quantum world the **state is a normalized vector** in a metric (Hilbert) space, whose basis is usually parameterized by some classical configurations, so that we can call the basis vectors the "classically measurable quantum states", say, a "ket" $|\uparrow\rangle$ in the Dirac *bra-ket* notation.

But a *qubit* or any other may find itself in a superposition state $|\psi\rangle = g|\downarrow\rangle + h|\uparrow\rangle$ or $g|0\rangle + h|1\rangle$, provided everybody knows what is the physical meaning of indices '0' and '1', which in principle should not be confounded with any numerical values, but which are commonly used for obvious reasons. The scalar product squared: $|\langle 0|\psi\rangle|^2$ gives the probability that in this state the individual measurement yields '0' (e.g. $\uparrow$). The *bra* vector $\langle\psi|$ is the dual (co-vector) of $|\psi\rangle$.

A quantum system evolves[1] by acting upon them with linear unitary operators: $|\psi\rangle' = U|\psi\rangle$. The "evolution stops" when the observer projects the final state on some physically justified basis, and gets some answer. It may be an oracle-type answer (spin-up or down), or, after gathering some statistics by repeating the process with identically prepared initial states, it may take the form of some probabilistic measure.

Quantum states of independent (non-interacting) subsystems are tensor products of individual vectors: $|100\rangle \equiv |1\rangle \otimes |0\rangle \otimes |0\rangle$, and operators acting on them are also tensor products. For interacting systems states are usually superpositions of such tensor states; their separation into additive components may not be easy.

Computer (classical) realizations of quantum entities have many variants. In view of the simplicity of the evolution process, particularly inspiring seem the functional approaches, see [9, 10, 11]. The data types used to store quantum states may be algebraic compounds: lists, etc., and all approaches are legitimate in order to implement and run as efficiently as possible some "quantum programs". But we took another approach, whose advantage is that the vector structure of the state models arises *naturally*, that it is adaptable to any quantum systems, not only to qubits (or other discrete, finite sets of configurations), and moreover, it corresponds quite faithfully to models of quantum calculi used by physicists. States are functions.

A functor which takes a configuration α to a Hilbert space vector is the Haskell functional we call **`ket`**. If a "bit" is defined by the **`data QBit = B0 | B1`**, then **`ket B1`** is $|1\rangle$. The construction of **`ket`** is indirect, before its definition, and before unveiling its type, we shall construct first the co-vectors $\langle\alpha|$, which we call *axes*, equivalent (but not identical) to Dirac bras, which will come later, as linear functionals over kets. We use a variant of construction known sometimes as *Kolmogorov dilation theorem*. Suppose that the "classical sector" of the world is equipped by a particular measure, which discerns between identical and different configurations. For typical configurations we may define a kernel function called **`bracket`**

```
bracket a b = if a==b then 1 else 0
```

i.e., the Kronecker delta. (We won't need here more general definitions, but they are sometimes unavoidable.) The values 0 and 1 should be treated here — as it is usual in standard quantum mechanics —

[1] in the so called Schrödinger picture; in the alternative ***and equivalent*** Heisenberg picture, states remain static, but operators evolve: $\hat{A}' = U\hat{A}U^{+}$. All averages $\langle\phi|\hat{A}|\psi\rangle$ behave identically in both pictures.

to be complex numbers; we assume that they belong to the type **`Scalar`**. If we rename now **`axis = bracket`**, the form **`(axis alpha)`** is a unary function, and as such it constitutes naturally an element of a vector space with operations **`(*>), (<+>)`**:

```
axis a <+> axis b = \c -> axis a c + axis b c
h *> axis a = \c -> h*axis a c
```

(We shall omit the definition of Haskell class **`Vspace`** which specifies our overloaded operations.) In our framework **`axis alpha`** represents $\langle\alpha|$. In geometry typically co-vectors are functionals over vectors, but since the Hilbert space permits a natural definition of dual objects, we reverse the construction. Kets are functionals over axes:

```
ket alpha = \ax -> ax alpha
```

from which we assert

- Kets: (**`ket alpha`**), being unary functions are natural vectors. We can superpose them.
- Kets ***are linear functions***:

```
ket a (ax1 <+> ax2) = (ax1 <+> ax2) a = ax1 a + ax2 a
                    = ket a ax1 + ket a ax2
```

And finally, we notice that the definition of **`ket`** is equivalent to the standard Fischer-Plotkin [12] lifting functional of elementary values into the CPS domain... We may say that a state is a "quantum value" awaiting its final continuation (the measurement), and that this continuation is the appropriate axis which yields the final answer, the quantum probabilistic measure (the amplitude).

We promised to remain anti-speculative, but we cannot help expressing some conceptual observations. In several papers one may find the term "quantum values", and discussion about the information contents of those entities. While physicists tend to avoid these issues, since they do not convey much operational sense, in the domain of computations such discussions seem unavoidable, since the data structures used to model quantum states and observables should be analyzed from this viewpoint, if we want to understand the computational reversibility, the relation between the entropy and the decoherence, etc. Our framework provides a *minimalistic* answer to some of those questions. The information "hidden" inside is given by the explicit semantics of the functions **`(ket alpha)`** (elementary kets) and their linear superpositions. Our functional objects are the only *ontological* entities we have, they *are* fragments of the reality. Our model of the quantum world is functional[2], and we don't need to add any artificial constraints in order to prevent illicit "peeping inside quantum values" represented by classical datatypes. Moreover, functional objects being opaque are prevented from being trivially duplicated, which may be related to the non-cloning theorem.

2.1 Duality

[2]The question whether we think that the real quantum world is a functional entity must be left for long, evening discussions...

Filinski in his Master thesis [13] and elsewhere pointed out the duality between (lifted) values and continuations. This becomes particlarly trivial in the context of our linear geometry. Since $\langle\psi|\alpha\rangle = \langle\alpha|\psi\rangle^*$, we may immediately construct an axis dual to a ket **psi**:

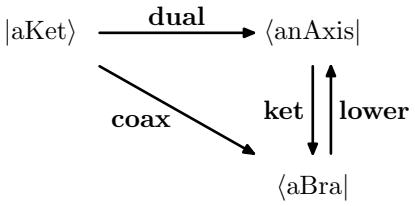

Figure 1: Duality conversions

```
dual psi = \a -> conjugate (psi (axis a))
```

Moreover, it is straightforward to *lift* kets to their dual space, the linear functionals which are bi-dual, thus isomorphic to axes, but they act on kets. It is this space which may be considered as the domain of Dirac bras (but we shall use them sparingly, axes and kets usually suffice). A bra is defined through

```
coax psi = \phi -> phi(dual psi)
```

or, even shorter, through an isomorphic lifting from axes: **br = ket ax**. Of course, **coax=ket . dual**. Bras can be lowered to axes by

```
lower br = \a -> br (ket a)          --    ⟨br|α⟩
```

and the diagram above is commutative. The proof that **ket** and **lower** are inverse is easy, but not entirely trivial, it holds for elementary axes and kets, and is generalized thanks to the vector structure of the framework. We have thus a full correspondence/duality between a quantum state and its measurement context. The usage of the functional **ket = flip id** for the lifting of axes into bras suggest that we should keep its type more polymorphic than restricted **ket :: Qdom->(Qdom->Scalar)->Scalar**, where **Qdom** is some configuration domain, e.g., **Qbit**. We might say that a co-vector (lifted to a bra) is the "quantum value", and a ket is its continuation.

Of course, one has to be quite cautionus with verbal analogies. In the language of categories and in the geometrical framework we have the term "duality" with related, (or rather subsumed) but not identical meaning. We shall see other similar cases, e.g. the term "linearity", which means something a bit different in geometry, and in the continuation theory, e.g., [14] (and in logic). To this collection we should add the term "adjoint".

3 Operators and Circuits

A quantum process in which the initial state $|\psi\rangle_0$ is sequentially acted upon by some operators $\hat{A}, \hat{B}, \hat{C}$, and results in a state $|\psi\rangle' = \hat{A}\hat{B}\hat{C}|\psi\rangle_0$, may be depicted as the following diagram: The time goes from

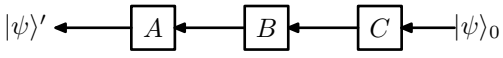

Figure 2: Chain of operators

right to the left. This is a convention contrary to that found in most other papers, but it corresponds better to the textual representation of the depicted process.

In practical cases, the states are compound. Disconnected lines represent tensor products of states, introduced in the next section. Separate boxes on product lines are tensor product operators. But in the

presence of interactions, the lines get connected. The Fig. 3 represents two typical elementary quantum gates, the "controlled not" and the Toffoli gate, and also their combination which realizes a 1-bit half-adder. Of course, much more complicated circuits are needed in order to implement some non-trivial computations,

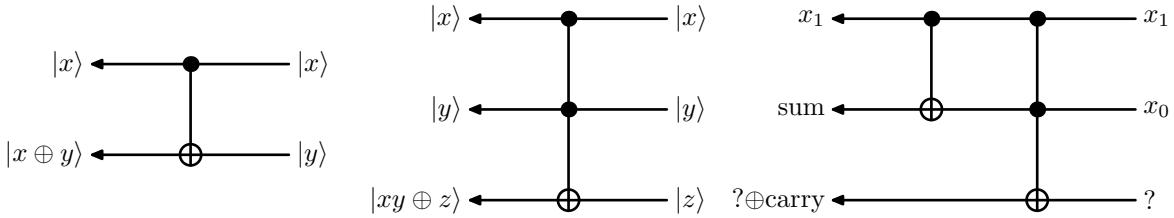

Figure 3: Controlled-not, Toffoli gate, and half-adder

it suffices to look the teleporting circuit on Fig. 4 [15], where L, R, etc. boxes are one-qubit operators which rotate, modifies the phase, or otherwise transform the appropriate state. For example,

$$S = \quad i|0\rangle\langle 0| - |1\rangle\langle 1|\,, \tag{1}$$

$$L = \quad R^{+} = (|1\rangle\langle 0| - |0\rangle\langle 1| + 1)/\sqrt{2}\,, \tag{2}$$

etc. It is not our purpose to discuss the details in this text. The functional representation of simple qubit operators can be found in [8]. The diagrammatic approach to the synthesis of programs is of course quite appealing, and the intuitive analogy between such diagrams, and the dataflow graphs representing electronic circuits etc., is a meaningful psychological factor. However, one should be careful: the lines *do not transport data*, at least not the classical data (but what are non-classical ones?...) What is interesting from the

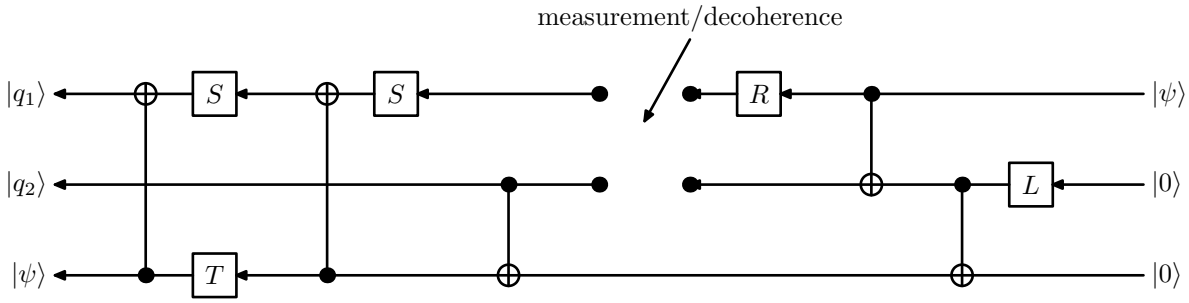

Figure 4: Brassard teleporting circuit

conceptual point of view, is the coexistence of two facts:

- The evolution is fully functional, moreover it is singly-threaded, 'linear' from the computational point of view. Every state vector is used exactly once. (This is not *only* the consequence of the algebraic linearity of operators; in presence of fan-outs (cloning) the computational linearity would break down.) If we want to speak about continuations, then they are linear.

- From the physical perspective the model shall represent a *changing, unique world*, not a sequence of state vectors which exist independently from each other.

One is tempted immediately to repeat the exclamation of Philip Wadler [16]. We don't want to confuse linear types and linear continuations, but conceptually they are both in the "used once" basket, [17], and for us it suffices.

Also, as Martin Odersky said in [18]: "... we see that linear types have so far been better at changing the world than at observing it". But as physicists, it is exactly what we need! As we know, in quantum physics we cannot gratuitously observe a state yielding a classical information about. Operators *must do* something to the state, and an observation terminates it (decoheres it, and entangles it with the observer, whenceupon the description of the system irremediably ceases to be complete). Finally, if quantum mechanics is considered to be complete, states cannot be discarded in the middle of computation. All that encourages us to think that linear continuations have strong potential to describe (model) typical quantum processes.

How can we *define* typical operators shown on the diagrams above? It seems difficult to create directly a model of an operator (like L or S above) which acts on kets: **`oper psi = ... (psi ...)`** , since **`psi`** will have to be applied to some other vector, it is the only thing which can be done with it.

If we begin with a classical function, say **`const B0`**, or **`cnot x = if x==B0 then B1 else B0`**, then the lifting of such a function **`cop`** to an operator acting on kets is

```
(lift cop) psi = \ax -> psi (\alpha -> ax (cop alpha))
```

or: **`lift = boost . boost`**, where **`boost = flip (.)`**. This is not a Plotkin-style functional, but the type pf the object surely is a typed "continued function", as in [19]. Of course, more general operators than those deducible from the classical poor subset are constructed – as the state vector themselves – through linear superpositions. For practical operations it is easier to exploite the algebraic linearity directly, and to begin the construction with operators acting on axes. Knowing that we have a natural mapping from vectors to co-vectors (from kets to axes), it is obvious that we have dually a converting functional for operators, but contravariant:

```
(lift1 aop) psi = \ax -> psi (aop ax)
```

In the domain of axes the, say, negation operation $|0\rangle\langle 1| + |1\rangle\langle 0|$ is implemented immediately:

```
anot ax = ax B0 *> axis B1 <+> ax B1 *> axis B0
```

It possesses the quality of Boolean negation, but it is a full-fledged linear operator. One not should forget (in general case; here it is trivial) that lifting of an operator to a dual space produces its algbraic adjoint.

In the following section we shall see the relation between operators and tensor products.

4 Tensor states and operators

Quantum structures begin to be interesting for computations when they describe compound systems (multi-qubit registers, etc.). As mentioned above, for two non-interacting susbsystems the global state is given by the tensor product [20], $|\phi\rangle|\psi\rangle$ (noted simply as $|\phi\rangle \otimes |\psi\rangle$, or $|\phi\psi\rangle$)which in functional spaces has a straightforward implementation; if **`psi = \ax -> psi ax`**, etc., then

```
phi <*> psi = \ax ay -> phi ax * psi ay
```

We are obliged to deal with n-ary functions, and so, with compound final continuations. The associativity is obvious, and the twist (argument flip) isomorphism as well. A more fundamental comment first. The tensor product of two states is not a Cartesian product, simply it is *not* a categorical product; there are no projections to individual components. Often, when presenting an introduction to quantum computing people say: if we have an interacting system, capable of producing an *entangled* state (not a tensor product), say, $|0\rangle|0\rangle + |1\rangle|1\rangle$, then the system becomes non-separable, we cannot extract the individual states, the decoherence of one component influences the second one, etc. All this is true.

But the issue is that *even without interactions* we cannot extract individual pure states from a tensor product! The structure of the ***model*** of quantum reality is such that states are global, despite fact that observation of one sub-system does not influence materially the other one (and that they can be observed separately). This is one of the biggest mysteries of the quantum conceptual picture, and it is deeper than the properties of entangled states, although it is the presence of entanglement which makes the story operationally non-trivial. Once more: A complete model (computer or other) of a compound quantum system cannot be split in independent fragments dealing separately with the subsystems, they possess *one*, shared global state. The duals of tensor states may be constructed through

```
(dual_2 psip)  = \alpha beta ->
   conjugate (psip (axis alpha) (axis beta))
```

This is a one object, a 2-axis. It cannot be directly considered as the continuation of a 2-ket, which needs 2 arguments. It can be practically used for calculating the state vector norm (in finitely-dimensional Hilbert spaces), but the connexion between duals and continuations should be re-analyzed...

Operators acting independently on subsets of "lines" are also products. Tensor products of operators acting on kets are constructed in a straightforward way from the adjoints acting on axes, by a multilinear lifting. For example,

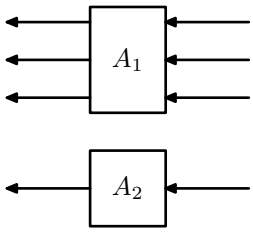

Figure 5: Compound operators

```
lift_2 ao1 ao2 psip  =
  \ax1 ax2 -> psip (ao1 ax1) (ao2 ax2)
```

which can be generalized to any number of arguments, and linearly combined into non-factorized operators. The paper [8] defines the appropriate constructor classes, permitting such tensor multiplications in the general case. The drawing at the right is an example of such a composite operator. From the adjoint perspective the construction **`lift_2 ao1 ao2`** might be seen as something which *produces* two outputs, implemented as an object which *consumes the consumer* of two inputs.

One doesn't need compound state vectors in order to see the relevance of tensor products. Already the single-ket operators may be considered as tensor products, constituted out of projectors $\hat{P} = |\alpha\rangle\langle\beta|$. The Dirac notation is handy on paper, such forms may act either on vectors or on co-vectors, and yield the appropriate scalar products, $\hat{P}|\psi\rangle = \langle\beta|\psi\rangle|\alpha\rangle$, and $\hat{P}\langle\phi| = \langle\phi|\alpha\rangle|\beta\rangle$ (a physicist would write: $\langle\phi|\hat{P}$; in Dirac notation the operators might act on their left (covector)argument, as adjoints), but the implementation must specialize the variant. If we want to construct an operator acting on axes from **`ketb`** representing $|\beta\rangle$, and **`axa`**: $\langle\alpha|$, then the construction is

```
ax_proj = \ax -> (ketb ax) *> axa
```

Of course, the dual variant is **`kt_proj=\psi -> (ket axa psi)*>ketb`**. In both cases they implement the well known equivalence in linear logic, restricted in this case to $A \multimap A \cong A^{\perp} \otimes A$, where A is either the type of axes or of kets. According to [21] this holds in *compact* autonomous categories, meaning here the finite dimension of our vector spaces. The tensor structure of our formalism needs further work.

5 Conclusions

From a certain distance, the dominant role of continuations in our model of quantum processes is utterly trivial, and may be traced back to Church. After all, if basic entities in the framework are functions, then — as in the lambda-calculus — the only way to *do something to them* is to *apply them*. Also, the linearity of operations in vector spaces and the computational linearity are strongly related, see [21].

Thus, most of the continuation properties "discovered" in our model can be formalized in categorical language, as we can see in [21]. We have avoided more formalization not because we think it is redundant, but we are simply not yet ready for it.

We exposed just a bridge which seems quite intuitive, but not very well developed for human reasons: the overlap between people interested in the theory of computations, and those who treat seriously the mathematical structures pertinent to the quantum theory, remains rather limited, although Girard's papers on Geometry of Interactions include examples dealing with Hilbert spaces, which would be immediately related to quanta by a reader oriented towards theoretical physics. Rare are attempts to marry both domains, see however [22] and other works of Vaughan Pratt who claims that linear logic and "quantum logic" are intimately related. All this needs further investigation.

From our personal perspective it is fascinating that formalisms developed in the crystal halls of logic, categories, and the theory of computation, may find some usage in the practice of quantum calculi, and we hope that one day they will influence the teaching of quantum mechanics as well. Of course, there is always a danger of 'philosophical abuse', and the confusion between the reality and the model, but, on the other hand, if we believe that the linear continuation semantics is something *natural* in our vision of quantum processes, we might begin to think differently about the information contents of quantum states.

6 Acknowledgements

I should not forget them.

References

[1] Michael A. Nielsen, Isaac L. Chuang,*Quantum Computation and Quantum Information*, Cambridge Univ. Press, (2000).

[2] Andrew Steane, *Quantum Computing*, Reports on Progress in Physics vol 61, pp 117-173, (1998). Online e-print: quant-ph/9708022.

[3] Julia Wallace, *Quantum Computer Simulation - A Review; ver. 2.0*, Univ. of Exeter tech. report, (1999), see also the site **`www.dcs.ex.ac.uk/~jwallace/simtable.html`**, (2002).

[4] Peter Shor, *Algorithms for quantum computation: discrete logarithms and factoring*, Proc. Symp. on Fundamentals of Computer Science, Los Alamitos, IEEE Press, (1994), pp. 124–134.

[5] L.K. Grover, *Quantum Mechanics Helps In Searching For a Needle in a Haystack*, Phys. Rev. Lett. **79**, (1997), p. 325. Also: L.K. Grover, *A fast quantum mechanical algorithm for database search*, Proc. 28th ACM Symp. on Theory of Computation, (1996), p. 212.

[6] Richard P. Feynman, *Simulating physics with computers*, Int. J. Theor. Phys. **21**, (1982), pp. 467–488.

[7] Christof Zalka, C. *Efficient simulation of quantum systems by quantum computers*. Proc. Roy. Soc. Lond., **A454**, (1998), pp. 313–322. Also: Fortschr. Phys. **46**, (1998), pp. 877–879.

[8] Jerzy Karczmarczuk, *Structure and Interpretation of Quantum Mechanics — a Functional Framework*, ACM SIGPLAN 2003 Haskell Workshop, Uppsala, August 2003, pp. 50–61.

[9] Shin-Cheng Mu, R. Bird, *Functional Quantum Programming*, 2nd Asian Workshop on Programming Languages and Systems, KAIST, Dajeaon, Korea, (2001).

[10] Amr Sabry, *Modeling Quantum Computing in Haskell*, ACM SIGPLAN 2003 Haskell Workshop, Uppsala, August 2003, pp. 39–49.

[11] Peter Selinger, *Towards a Quantum Programming Language*, To appear in Mathematical Structures in Computer Science, (2003).

[12] Gordon D. Plotkin, *Call by name, call by value and the λ-calculus*, Theor. Comp. Science **125**, (1975), pp. 125–159.

[13] Andrzej Filinski, *Declarative Continuations and Categorical Duality*, Master's thesis, Computer Science Department, University of Copenhagen (August 1989). DIKU Report 89/11.

[14] Andrzej Filinski, *Linear Continuations*, Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Albuquerque, (1992), pp. 27–38.

[15] C.H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres et W.K. Wootters, *Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels*, Phys. Rev. Lett., **70**:13, (1993), pp. 1895–1899.

[16] Philip Wadler, *Linear Types Can Change the World!*, IFIP TC Working Conference on Programming Concepts and Methods, Sea of Galilee, Israel, (1990), pp. 347–359.

[17] Josh Berdine, Peter O'Hearn, Uday Reddy, Hayo Thielecke, *Linear Continuation-Passing*, Higher-Order and Symbolic Computation, **15**, (2002), pp. 181–208.

[18] Martin Odersky, *Observers for Linear Types*, European Symposium on Programming, Rennes, France, (1992); Lecture Notes in Computer Science **582**, Springer-Verlag, pp. 390–407.

[19] Albert R. Meyer, Mitchell Wand, *Continuation semantics in Typed Lambda Calculi*, Logics of programs, Springer Lect. Notes in Comp. Sci. **193**, (1985), pp. 219–224.

[20] Diederik Aerts, Ingrid Daubechies, *Physical justification for using the tensor product to describe two quantum systems as one joint system*, Helvetica Physica Acta, **51**, (1978), pp. 661–675.

[21] Michael Barr, *$*$-Autonomous categories, and linear logic*, Mathematical Structures in Computer Science, **1** (1991), pp. 159–178.

[22] Vaughan Pratt, *Linear logic for generalized quantum mechanics*, Proc. Workshop on Physics and Computation (PhysComp), Dallas, (1992), pp. 166–180.