Under consideration for publication in Math. Struct. in Comp. Science

Quantum Programming Languages
Survey and Bibliography

Simon J. Gay

Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK
Email: simon@dcs.gla.ac.uk

Received 13 October 2005; Revised 24 March 2006

The field of quantum programming languages is developing rapidly and there is a surprisingly large literature. Research in this area includes the design of programming languages for quantum computing, the application of established semantic and logical techniques to the foundations of quantum mechanics, and the design of compilers for quantum programming languages. This article justifies the study of quantum programming languages, presents the basics of quantum computing, surveys the literature in quantum programming languages, and indicates directions for future research.

1. Introduction

Feynman (1982) suggested that constructing computers based on the principles of quantum mechanics might enable the quantum systems of interest to physicists to be efficiently simulated, whereas this seemed to be very difficult with classical computers. Deutsch (1985) investigated the possible computational power of physically realizable computers, and formulated a quantum version of the Turing machine. Shor (1994) demonstrated that two important practical problems—factorizing integers and the "discrete logarithm" problem—could be solved efficiently by quantum computers. Grover (1996) showed that database search could also be made more efficient by the use of quantum computers. Since then, a substantial literature has developed in quantum algorithms and quantum complexity theory.

Another important theme in quantum computing has been the development of quantum cryptographic techniques, going back to the work of Bennett and Brassard (1984) which in turn built on work, not published until several years after its conception, by Wiesner (1983). There is an interesting interplay between quantum computing and quantum cryptography, in that while Shor's algorithm for integer factorization has the potential to undermine many current cryptosystems, quantum cryptographic systems can be proved secure against any form of attack, including attacks which make use of quantum computing.

Despite Deutsch's (1985) observation that "quantum computers raise interesting problems for the design of programming languages", computing scientists were slow to respond to this challenge. Although it now seems obvious that quantum information processing devices, like their classical counterparts, should be programmed in high-level, structured and well-defined languages, it was only in 1996 that work towards the design of such languages began to be published. There has been an explosion of publication since 2001.

The fact that physical implementations of quantum computers are still very limited, working with only a few qubits, and have not yet escaped from the physics laboratory, exposes the field of quantum programming languages to the criticism that it is pointless to study languages for programming non-existent hardware. This criticism is ill-founded, for several reasons.

Firstly, it overlooks the dramatic progress that has been made in the practical implementation of quantum cryptographic systems. Components for these systems are now commercially available, and it seems very likely that quantum cryptography will be an important technology long before quantum computers of useful size are constructed. Yet quantum cryptographic systems implement complex and delicate protocols, and will need to be programmed in systematic and principled ways; furthermore, the need to be confident of their security introduces a whole new dimension of programming language theory when notions of program equivalence are considered.

Secondly, the widespread use of programming languages that do not have a firm semantic foundation has caused huge problems for software engineering. Practical computing technologies have raced ahead of theoretical studies, and it is only recently that the latest developments in mainstream programming language design (for example, Java 1.5) have had the benefit of a theoretical basis which was thoroughly understood in advance. From this point of view, designing quantum programming languages before the hardware exists is, in some respects, an ideal situation.

Thirdly, and perhaps unexpectedly, the application of semantic, logical and especially category-theoretic techniques is providing a fascinating new perspective on quantum theory itself. Computing scientists are generating new insights into the foundations of quantum mechanics, which will be of value even if practical quantum computers are never built.

In this article I survey the present literature on quantum programming languages, and attempt a classification into major topics. Before doing so, I briefly summarise the basic concepts of quantum computing, in Section 2. I describe the classification criteria in Section 3, and discuss the papers themselves in Section 4. Finally, in Section 5, I indicate some directions for future research in this area. I have tried to be as inclusive as possible, and I have not made any judgement of the relative significance or quality of the papers surveyed. In particular, the relative amounts of text devoted to each paper do not indicate an assessment of their relative importance.

Selinger (2004a) published an earlier survey of quantum programming languages. His article uses a similar classification scheme and offers a different perspective on some of the issues in the field. A preliminary version of the present article appeared in *Bulletin of the EATCS*, June 2005. The bibliography is online at `www.dcs.gla.ac.uk/∼simon/quantum` and I intend to maintain it as a resource for the community.

2. Basics of Quantum Computing

In this section I will briefly introduce the aspects of quantum theory that form the basis of quantum computing, and describe an example, Deutsch's (1985) algorithm, which illustrates the potential power of quantum computers. More detailed presentations can be found in the books by Gruska (1999) and Nielsen & Chuang (2000). Rieffel and Polak (2000) give an account aimed at computer scientists, and Preskill's (1998) lecture notes are another valuable resource. Arrighi (2003) covers similar material to this section, with a more detailed discussion of the fundamentals of quantum theory.

A *quantum bit* or *qubit* is a physical system that has two basis states, conventionally written $|0\rangle$ and $|1\rangle$, corresponding to one-bit classical values. These could be, for example, spin states of a particle or polarization states of a photon, but we do not consider physical details. According to quantum theory, a general state of a quantum system is a *superposition* or linear combination of basis states. Concretely, a qubit has state $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$; states which differ only by a (complex) scalar factor with modulus 1 are indistinguishable. States can be represented by column vectors:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle.$$

Formally, a quantum state is a unit vector in a Hilbert space, i.e. a complex vector space equipped with an inner product satisfying certain axioms. In this section we will restrict attention to collections of qubits.

The basis $\{|0\rangle, |1\rangle\}$ is known as the *standard* basis. Other bases are sometimes of interest, especially the *diagonal* (or *dual*, or *Hadamard*) basis consisting of the vectors $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. For example, with respect to the diagonal basis, $|0\rangle$ is in a superposition of basis states:

$$|0\rangle = \frac{1}{\sqrt{2}}|+\rangle + \frac{1}{\sqrt{2}}|-\rangle.$$

Evolution of a closed quantum system can be described by a *unitary transformation.* If the state of a qubit is represented by a column vector then a unitary transformation U can be represented by a complex-valued matrix (u_{ij}) such that $U^{-1} = U^*$, where U^* is the conjugate-transpose of U (i.e. element ij of U^* is $\bar{u}_{ji}$). U acts by matrix multiplication:

$$\begin{pmatrix}\alpha'\\ \beta'\end{pmatrix} = \begin{pmatrix}u_{00} & u_{01}\\ u_{10} & u_{11}\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

A unitary transformation can also be defined by its effect on basis states, which is extended linearly to the whole space. For example, the *Hadamard* transformation is defined by

$$\begin{array}{lcl} |0\rangle & \mapsto & \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\ |1\rangle & \mapsto & \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle \end{array}$$

which corresponds to the matrix

$$\mathsf{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}.$$

The Hadamard transformation creates superpositions:

$$\mathsf{H}|0\rangle = |+\rangle \qquad \mathsf{H}|1\rangle = |-\rangle.$$

A key feature of quantum physics is the role of *measurement*. If a qubit is in the state $\alpha|0\rangle + \beta|1\rangle$ then measuring its value gives the result 0 with probability $|\alpha|^2$ (leaving it in state $|0\rangle$) and the result 1 with probability $|\beta|^2$ (leaving it in state $|1\rangle$). Protocols sometimes specify measurement with respect to a different basis, such as the diagonal basis; this can be expressed as a unitary change of basis, followed by a measurement with respect to the standard basis, followed by the inverse change of basis. Note that if a qubit is in state $|+\rangle$ then a measurement with respect to the standard basis gives result 0 (and state $|0\rangle$) with probability $\frac{1}{2}$, and result 1 (and state $|1\rangle$) with probability $\frac{1}{2}$. If a qubit is in state $|0\rangle$ then a measurement with respect to the diagonal basis gives result $+$ (and state $|+\rangle$) with probability $\frac{1}{2}$, and result $-$ (and state $|-\rangle$)) with probability $\frac{1}{2}$, because of the representation of $|0\rangle$ in the diagonal basis noted above.

To go beyond single-qubit systems, we consider tensor products of spaces (in contrast to the cartesian products used in classical systems). If spaces U and V have bases $\{u_i\}$ and $\{v_j\}$ then $U \otimes V$ has basis $\{u_i \otimes v_j\}$. In particular, a system consisting of n qubits has a 2^n-dimensional space whose standard basis is $|0\rangle \otimes \cdots \otimes |0\rangle, \ldots, |1\rangle \otimes \cdots \otimes |1\rangle$, which we write as $|00\ldots0\rangle \ldots |11\ldots1\rangle$. We can now consider measurements of single qubits or collective measurements of multiple qubits. For example, a 2-qubit system has basis $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ and a general state is $\alpha|00\rangle+\beta|01\rangle+\gamma|10\rangle+\delta|11\rangle$ with $|\alpha|^2+|\beta|^2+|\gamma|^2+|\delta|^2 = 1$. Measuring the first qubit gives result 0 with probability $|\alpha|^2 + |\beta|^2$ (leaving the system in state $\frac{1}{\sqrt{|\alpha|^2+|\beta|^2}}(\alpha|00\rangle+\beta|01\rangle)$) and result 1 with probability $|\gamma|^2+|\delta|^2$ (leaving the system in state $\frac{1}{\sqrt{|\gamma|^2+|\delta|^2}}(\gamma|10\rangle + \delta|11\rangle)$); in each case we renormalize the state by multiplying by a suitable scalar factor. Measuring both qubits simultaneously gives result 0 with probability $|\alpha|^2$ (leaving the system in state $|00\rangle$), result 1 with probability $|\beta|^2$ (leaving the system in state $|01\rangle$) and so on; note that the association of basis states $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ with results $0, 1, 2, 3$ is just a conventional choice. The power of quantum computing, in an algorithmic sense, results from calculating with superpositions of states; all the states in the superposition are transformed simultaneously (*quantum parallelism*) and the effect increases exponentially with the dimension of the state space. The challenge in quantum algorithm design is to make measurements which enable this parallelism to be exploited; in general this is very difficult.

Systems of two or more qubits can exhibit the phenomenon of *entanglement*, meaning that the states of the qubits are correlated. For example, consider a measurement of the first qubit of the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The result is 0 (and resulting state $|00\rangle$) with probability $\frac{1}{2}$, or 1 (and resulting state $|11\rangle$) with probability $\frac{1}{2}$. In either case a subsequent measurement of the second qubit gives a definite (non-probabilistic) result which is always the same as the result of the first measurement. This is true even if the entangled qubits are physically separated. Entanglement illustrates the key difference between the use of tensor product (in quantum systems) and cartesian product (in classical systems): an entangled state of two qubits is one which cannot be expressed as a tensor product of single-qubit states.

2.1. *Deutsch's Algorithm*

Deutsch's (1985) algorithm was the first demonstration that a quantum computer can solve a specific problem more efficiently than a classical computer. There are several variations of the algorithm; the version described here is based on the presentation by Nielsen and Chuang (2000).

Suppose that we have a black box which computes an unknown function $f : \{0,1\} \to \{0,1\}$. We want to know whether or not f is a constant function. Classically, it is obvious that we must evaluate $f(0)$ and $f(1)$ and compare the results. But we will see that a quantum computer can answer the question with only one evaluation of f.

Assume that we can ask for a quantum version of the black box; that is to say, a unitary transformation F which performs, in some sense, the same computation as f. Our first thought is to require

$$\begin{array}{rcl} F|0\rangle & = & |f(0)\rangle \\ F|1\rangle & = & |f(1)\rangle \end{array}$$

but this is not physically possible in general. F, as a unitary transformation, must be invertible, but f need not be invertible.

However, it is possible to construct a unitary transformation F on two qubits, such that

$$F|x\rangle|y\rangle \quad = \quad |x\rangle|y \oplus f(x)\rangle$$

where $\oplus$ is exclusive or. We therefore assume that this is the quantum version of the black box.

The trick now is to apply F to the state

$$|+\rangle|-\rangle = \frac{1}{2}(|0\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|0\rangle - |1\rangle|1\rangle)$$

but we need to do some calculation in order to be able to express the result in terms of the unknown function f.

From the definition of F we have

$$\begin{array}{rcl} F|x\rangle|0\rangle & = & |x\rangle|f(x)\rangle \\ F|x\rangle|1\rangle & = & |x\rangle|1 \oplus f(x)\rangle. \end{array}$$

Combining these equations to calculate $F|x\rangle|-\rangle$, we see that if $f(x) = 0$ then $F|x\rangle|-\rangle = \frac{1}{\sqrt{2}}|x\rangle(|0\rangle - |1\rangle)$, and if $f(x) = 1$ then $F|x\rangle|-\rangle = \frac{1}{\sqrt{2}}|x\rangle(|1\rangle - |0\rangle)$. Hence

$$\begin{array}{rcl} F|x\rangle|-\rangle & = & \dfrac{(-1)^{f(x)}}{\sqrt{2}}|x\rangle(|0\rangle - |1\rangle) \\ & = & (-1)^{f(x)}|x\rangle|-\rangle. \end{array}$$

We can now calculate

$$\begin{array}{rcl} F|+\rangle|-\rangle & = & \frac{1}{\sqrt{2}}(F|0\rangle|-\rangle + F|1\rangle|-\rangle) \\ & = & \left\{ \begin{array}{ll} \pm|+\rangle|-\rangle & \text{if } f(0) = f(1) \\ \pm|-\rangle|-\rangle & \text{if } f(0) \neq f(1) \end{array} \right. \end{array}$$

and the information about whether or not f is constant has been concentrated into the first qubit.

We can check that $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$, so applying H to the first qubit gives

$$\begin{array}{ll} \pm|0\rangle|-\rangle & \text{if } f(0) = f(1) \\ \pm|1\rangle|-\rangle & \text{if } f(0) \neq f(1) \end{array}$$

and finally a measurement of the first qubit reveals the desired information.

This algorithm embodies what seem to be the essential aspects of an efficient quantum algorithm: preparation of a superposed state, then application of unitary transformations in such a way as to take advantage of quantum parallelism and then concentrate the resulting global information into a single place, and finally an appropriate measurement.

Furthermore, we can clearly see the need for a formalized programming language syntax. The above description of Deutsch's algorithm combines equations (to be interpreted as transformations of the state) with narrative description, and mixes the definition of the algorithm with the proof of its correctness, in a way which could be very difficult to follow in more complex situations. Several of the papers surveyed in this article use Deutsch's algorithm to illustrate their programming languages.

3. Classification of Papers

I have used the following classification of the main theme of each paper.

1 Programming language design
- (a) imperative languages
- (b) functional languages and λ-calculi
- (c) other language paradigms

2 Semantics
- (a) applications of linear logic
- (b) categorical and domain-theoretic techniques
- (c) other semantic techniques

3 Compilation

The "semantics" classification refers to denotational techniques. Many of the papers whose primary emphasis is on language design also define semantics in an operational style. I have used the "semantics" classification for papers which do not define languages (for example, the semantic studies that focus on protocols) and for papers including language definitions but whose emphasis is on denotational semantics. Papers applying linear logic to the structural aspects of quantum computation are also included.

It was not always clear how to classify each paper: for example, there is naturally considerable overlap between functional programming and semantics, and between language design and compilation. It was also difficult to draw the boundary of the whole area. My primary concern was to concentrate on papers that discuss programming language design, semantics or compilation, or that apply logical and semantic techniques from computing science to quantum computation. I have omitted most papers on quantum logic (there is an extensive literature) except when a formal syntax has been defined in order to develop a proof system, or when the emphasis is on the application of linear logic to quantum computation.

I have not included papers describing systems for simulating quantum computation, except when there is significant emphasis on the design of the input language of the simulator. Most simulation systems are designed in one of two ways: either the input language is some notation for quantum circuits, or the simulator is packaged as an API for a standard programming language. These systems do not consider issues in the design of quantum programming languages, such as high-level quantum data structures or high-level quantum control structures. Glendinning (2005) is a useful catalogue of quantum simulation systems. When the quantum programming languages surveyed in the present article have been implemented via simulators, I have indicated this in the text.

4. The Papers

4.1. *Programming Language Design*

4.1.1. *Imperative Languages.* Deutsch (1985) defines quantum Turing machines (QTM) as the first model for general quantum computation, with the crucial property that superpositions of machine states are allowed, and defines a universal QTM. Earlier work by Benioff (1980) defines physical systems in which the laws of quantum mechanics would lead to the simulation of a classical Turing machine, but does not consider quantum computation; Albert (1983) defines a form of quantum automata, which computes with quantum states but is not programmable. Yao (1993) studies the properties of the universal QTM further, and Bernstein & Vazirani (1993; 1997) give a construction for an efficient universal QTM and discuss programming primitives. The concept of the QTM has been used as the basis for much work in complexity theory, which we do not consider here. Iriyama et al. (2004) define a generalized QTM (GQTM) in which states can evolve by non-unitary transformations, in order to model the computational capabilities of a *nonlinear chaos amplifier*. Special cases of the GQTM are the linear QTM and the unitary QTM. Perdrix and Jorrand define the measurement-based QTM (2004b; 2004c) as a model for measurement-based quantum computation, and the classically-controlled QTM (2004a) to capture the paradigm of quantum data and classical control.

Probably the earliest proposal for a formalized quantum programming language, as opposed to a notation for QTM definitions, is that of Knill (1996). He defines an imperative pseudocode suitable for implementation on a quantum random access machine (QRAM). The QRAM model is also a proposal of this paper; it is not defined formally, but consists of a register machine with the ability to perform quantum operations, including state preparation, unitary transformation and measurement, on quantum registers. Knill acknowledges that quantum pseudocode is not, in itself, precise enough to be an implementable quantum programming language; however, it is an important step beyond the use of *ad hoc* narrative descriptions of how quantum operators and measurements should be applied.

Another early piece of work towards a quantum programming language, with a simulation system, is that of Baker (1996). Although incomplete, this contains some significant ideas, especially the use of linear types (in this case, through the use of Concurrent Clean's uniqueness types in the implementation of the simulator) to control quantum state.

Over a period of several years, Ömer (1998, 2000, 2001, 2002, 2003) developed QCL, the first real quantum programming language, with a syntax inspired mainly by C. He has also implemented a simulator for the language. QCL contains a full classical programming language as a sublanguage, and provides a range of useful high-level quantum programming features such as memory management and automatic derivation of conditional versions of operators.

Bettelli et al. (2003) define a high-level language based on C++ and a collection of low-level primitive operators; the low-level primitives are based on the QRAM model but are intended to be architecture-independent as far as possible. They give a certain amount of detail of a compilation scheme, and have implemented their language in the form of a C++ library.

Sanders & Zuliani (2000) and Zuliani (2001) define the language qGCL, which is based on a guarded-command language. qGCL inherits from the guarded-command world a semantics in terms of either predicate transformers or relations, and a refinement calculus. Part of the emphasis of this work is on the derivation of quantum algorithms. Zuliani (2004, 2005b) has studied programming with non-determinism and mixed states in the context of qGCL.

Tafliovich (2004) defines a quantum programming language based on probabilistic predicative programming. The language itself is imperative, and a key aspect of the paradigm is a close connection between programs and specifications, with an emphasis on implementation by refinement.

4.1.2. *Functional Languages and Lambda-Calculi.* Maymin (1996) defines two extensions of the λ-calculus, with call-by-value big-step operational semantics. The first, a probabilistic λ-calculus (λ^p-calculus), incorporates distributions of terms, allowing functions to return randomized results. The second, a quantum λ-calculus (λ^q-calculus), goes further by allowing terms to be represented negatively in distributions; thus there is the possibility of destructive interference when distributions are combined. In neither case is it possible to associate numerical coefficients with the terms in a distribution, although it is possible for terms to be repeated. In the λ^q-calculus this means that it is impossible to express relative phases of anything other than 180 degrees. This seems like a significant limitation, although Maymin shows that λ^q-calculus is able to efficiently solve NP-complete problems, and in another paper (1997) argues that λ^q-calculus can efficiently simulate one-dimensional partitioned quantum cellular automata, which are equivalent to quantum Turing machines. It appears that λ^q-calculus may be strictly more expressive than physically realizable quantum computation.

Van Tonder (2004) defines a quantum λ-calculus, λ_q, with an operational semantics and an equational theory. It is a language for pure quantum computation: measurement is not considered. He analyzes the non-duplicability of quantum state in terms of linear logic and gives an inductive definition of well-formed λ_q terms which is, essentially, a simplified form of a linear type system. The paper includes some examples of quantum algorithms expressed in λ_q, and argues that λ_q is equivalent in computational power to quantum Turing machines. It also mentions a simulator for quantum algorithms, implemented in Scheme.

Valiron (2004a,b) and Selinger & Valiron (2005; 2006) define a higher-order functional programming language based on the idea of classical control and quantum data (following Selinger (2004c)). The language is based on call-by-value λ-calculus, and includes both classical and quantum data. The semantics is operational, although extending Selinger's (2004c) denotational semantics to the higher-order case is a goal, and there is a type system based on linear logic for control of quantum state. They prove type preservation and type safety properties, as expected for typed functional languages, and present a type inference algorithm.

Extending the work of Valiron (2004b), Perdrix (2005) defines a type system which reflects entanglement of parts of the quantum state. The idea is to enrich the type environment with an equivalence relation on quantum variables, representing an approximation to the entanglement relation.

Arrighi and Dowek (2004; 2005) define a linear-algebraic λ-calculus in which all functions are linear operators on vector spaces. Although modelling quantum computation is a major goal of their work, they develop their theory somewhat independently of the specifics of quantum mechanics in order to obtain a more general view of the meaning of linearity. A notable feature of this work is the identification of a correspondence between the pattern-matching notation familiar from functional programming and the linear-algebraic notation defining linear operators by their effect on basis vectors. The paper defines an operational semantics for the linear-algebraic λ-calculus and discusses the relationship between linearity in the senses of linear algebra and linear logic.

Altenkirch and Grattage (2005a; 2005b) define a first-order functional programming language, QML, in which control, as well as data, may be quantum. The semantics of QML is expressed in category-theoretic terms, which provides a basis for a denotational semantics in terms of superoperators (along the lines of Selinger's (2004c)) and a translation into quantum circuits. The type system of QML is based on linear logic, but focuses on the elimination of weakening (discarding quantum state) rather then contraction (duplication of quantum state); indeed, contraction is allowed but is interpreted as sharing rather than copying. Another goal of the type system is to control decoherence, and this is supported by introducing a judgement and derivation rules for orthogonality of state-spaces. In further work with Vizzotto and Sabry (Altenkirch et al. 2006) the authors develop a sound and complete equational theory for the pure (measurement-free) fragment of QML; this also yields a normalization algorithm for QML.

Danos et al. (2004) study the one-way model of quantum computation, and develop a notation for the key components of this model: entanglement, measurement, and local correction. This notation is based on patterns. The main contribution of the paper is to define a calculus of equations over patterns, which leads to an algorithm whereby any pattern can be transformed into a standard form consisting of entanglement followed by measurement followed by correction. Danos and Kashefi (2005a) describe sufficient conditions for measurement patterns to execute deterministically. Danos et al. (2005b) develop a similar calculus of two-qubit measurements, which is relevant to use of teleportation as a primitive for quantum computation.

Selinger's (2004c) work has been very influential in the development of functional languages for quantum computation. He defines a first-order functional language with a static type system, taking the approach of classical control and quantum data. The most significant aspect of this work is the denotational semantics. This uses the standard mechanism of complete partial orders and continuous functions, but in the setting of vector spaces and superoperators; a key feature is the treatment of partiality arising from non-terminating recursion or loops. Edalat (2004) has also studied the semantics of partial states in quantum computing.

Several papers investigate quantum programming within Haskell, following a tradition of developing domain-specific languages as collections of data types and functions in Haskell. Mu and Bird (2001) first describe a way of embedding non-deterministic programming in Haskell, using a monadic style. They extend this idea to give a representation of quantum programming, by defining a data type for quantum registers, although the coefficients in a superposition are restricted to being real numbers. However, they are able to express the Deutsch-Josza algorithm and Grover's search algorithm. Sabry (2003) extends this idea further by developing higher-level programming abstractions—the concepts of "virtual value" and "adaptor"—to support computation with entangled states. Vizzotto and da Rocha Costa (2005) extend the same ideas still further by considering concurrent quantum programming in the setting of Concurrent Haskell. This enables them to give examples involving quantum protocols, in which physical distribution is a key feature, rather than quantum algorithms. Karczmarczuk (2003) takes a more foundational approach to quantum programming within Haskell, by implementing the basic elements of quantum mechanics as data types and functions and constructing higher-level data structures from them. The work of Skibiński (2001) has similar goals.

4.1.3. *Other Language Paradigms.* Gay and Nagarajan (2005; 2006) define the process calculus CQP (Communicating Quantum Processes); Jorrand & Lalire (2004) and Lalire & Jorrand (2004) define QPAlg (Quantum Process Algebra). Both languages can describe systems combining classical and quantum computation and communication, and the goal of both lines of work is to support the formal specification and verification of quantum cryptographic protocols. Gay and Nagarajan's language CQP is equipped with a static type system which uses techniques from linear logic to express the constraint that each element of quantum state is physically owned by a unique component of a system. They prove type preservation and type safety properties for an operational semantics, and prove soundness and completeness of a typechecking algorithm. Lalire (2005, 2006) defines a probabilistic bisimulation for QPAlg, with the aim of formulating correctness properties of protocols in terms of process equivalence.

Papanikolaou (2004) outlines the definition of the language qSpec, which supports the definition of concurrent processes, with communication, in an imperative style inspired by the model-checking specification languages Promela and Probmela. He gives some definitions necessary to support a formal operational semantics, but does not define the semantics itself.

Mauerer (2005) defines the language cQPL, which is based on Selinger's (2004c) QPL with extensions for inter-process communication, allowing distributed systems and communication protocols to be programmed. The core syntax of QPL is extended to a familiar-looking concrete syntax, and there is a compiler and simulation system. A denotational semantics of cQPL is defined, based on that of QPL with extensions to record the location of qubits. The use of a type system to control qubits, in a similar way to the work of Gay and Nagarajan (2005; 2006), is also discussed.

Adão and Mateus (2005) define a process calculus for specifying and reasoning about quantum security protocols. The language is committed to the QRAM computational model with the addition of a cost model. They also define notions of observational equivalence and quantum computational indistinguishability, and demonstrate compositional reasoning about a quantum zero-knowledge protocol.

Danos et al. (2005a) extend the measurement calculus (Danos et al. 2004) to a language for modelling distributed systems. Individual agents compute by means of the measurement calculus, and communicate by exchanging values or qubits over process-calculus-style channels. D'Hondt (2005) studies the semantics of this language in more detail, and uses it as the basis for reasoning about the behaviour of systems; she also introduces a logic of knowledge in distributed quantum systems.

Di Pierro and Wiklicky (2001) extend the constraint programming paradigm to quantum computing. Starting with a probabilistic version of constraint logic programming, they define an alternative operational semantics in terms of quantum constraint systems.

Udrescu et al. (2004) use a hardware description language, of the kind standard in VLSI design, to describe quantum circuits.

4.2. *Semantics*

4.2.1. *Applications of Linear Logic.* A significant aspect of denotational semantic approaches to quantum computation is the use of structures based on linear logic—for example, categories with symmetric monoidal or compact closed structure or variations thereof. Linear type systems are also being widely used in quantum programming languages whose semantics is defined operationally.

Pratt (1992) was an early advocate of linear logic as the basis for a dynamic logic of quantum mechanics. He was motivated partly by the future prospect of quantum computing, but the paper itself discusses quantum mechanics more generally and is not specifically aimed at quantum computation. Inspired by Pratt's ideas, Wehr (1996) suggested that type theories based on linear logic would be appropriate for quantum programming languages, although he did not define a particular language. Girard (2004) has developed a quantum semantics of linear logic in terms of quantum coherent spaces.

4.2.2. *Categorical and Domain-Theoretic Techniques.* Abramsky (2004, 2005), Coecke (2004a; 2004b) and Abramsky & Coecke (2003; 2005; 2004) have developed a category-theoretic formulation of the axioms of quantum mechanics, in the setting of strongly compact closed categories with biproducts. A substantial application of their work is a study of information flow in quantum protocols, and especially the information flow enabled by the use of entangled states; they are able to prove correctness of a teleportation protocol within a categorical semantics. In related work, Duncan (2004) has constructed a new category-theoretic semantics of multiplicative linear logic within Abramsky and Coecke's framework, and Abramsky & Duncan (2006) have developed a categorical logic of quantum systems. Coecke (2005) studies the category-theoretic framework further, with the goal of simplifying its structures and axioms.

Blute et al. (2003) give another category-theoretic framework in which to understand the behaviour of quantum systems; they do not focus specifically on quantum computation. They discuss connections between their approach and linear logic, especially the use of the connectives of linear logic to describe entanglement.

Van Tonder's quantum λ-calculus (2004) has already been mentioned. He has also developed a type theory and denotational semantics for λ_q, formulated in category-theoretic terms (2003). The category-theoretic structure is that of a symmetric monoidal closed category, confirming the connection with linear logic.

Selinger's (2004c) language QPL has already been mentioned; its denotational semantics is one of the most important aspects. Selinger (2004d) also describes two ideas for a denotational semantics of a higher-order quantum programming language, and explains why neither of them is satisfactory. In more recent work (2005a) he studies Abramsky and Coecke's strongly compact closed categories (renaming them "dagger compact closed categories"), establishing closer connections between his previous (2004c) work and that of Abramsky and Coecke (2004).

Vizzotto et al. (2006) extend earlier work on modelling pure quantum computation with monads (discussed above in the functional programming section) (Mu and Bird 2001; Sabry 2003; Vizzotto and da Rocha Costa 2005) and show that by generalizing monads to arrows it is possible also to model measurement. Moreover, they achieve a general separation of quantum computation into a pure part and an effectful part. They express this scheme within Haskell, and give examples of the resulting programming style.

Kashefi (2003a,b) develops domain theory for quantum computation, obtaining a denotational semantics which has the same structure as a denotational semantics of classical probabilistic computation. She uses this semantics primarily to study computability.

4.2.3. *Other Semantic Techniques.* D'Hondt and Panangaden (2006) define predicate transformers and weakest preconditions for quantum computation, and establish a Stone duality between state-transformer semantics and weakest precondition semantics. This approach to semantics is language-independent, but as examples they define the semantics of Selinger's (2004c) language QPL and construct the predicate transformer corresponding to the essential operator in Grover's search algorithm.

Feng et al. (2005) consider a language similar to Selinger's (2004c) and prove equivalence of the superoperator semantics and a weakest precondition semantics similar to that of D'Hondt and Panangaden (2006).

Unruh (2005) defines a semantic model of quantum programs which produce classical output during execution. The execution of such a program is viewed as a measurement on the initial quantum state; the measurement outcome is the classical output produced during execution. The semantics is fully abstract with respect to a notion of distinguishability of quantum states.

Danos et al. (2005c) establish a new set of generators for unitary transformations, and show that these generators have simple implementations in the one-way model of quantum computation. This leads to a robust and efficient implementation of general unitary transformations in the one-way model. Danos and Kashefi (2005b) study a version of the one-way model in which only a limited range of measurements is allowed, and show that it is approximately universal with respect to unitary transformations.

Baltag and Smets (2006) define a quantum logic and its proof system. They give logical characterizations of entanglement and information-flow, using a teleportation protocol as an example. Brunet & Jorrand (2004) and Mateus & Sernadas (2006) also define quantum logics intended for reasoning about quantum systems in general as well as quantum programs.

Petersen and Oskin (2003) define an algebraic notation for expressing quantum algorithms, and establish some techniques for reasoning about quantum state, although the paper does not fully define a programming language syntax.

Grädel and Nowack (2003) use the formalism of abstract state machines to describe quantum algorithms, within the framework of classical control operating on quantum state.

4.3. *Compilation*

Svore et al. (2004, 2006) define a software architecture for a suite of tools which will transform a high-level language specification of a quantum algorithm into a low-level implementation targeted at a particular quantum computing technology. Their proposal is for compilation in two stages. The source language is a high-level language, which should of course support complex data and control structures. The intermediate language is a quantum assembly language, which should work at the level of qubits and unitary operators but should still be independent of any particular physical implementation scheme. The object language is a *quantum physical operations language*, which is specific to a particular physical system and may include, for example, explicit operations to bring individual qubits into physical proximity so that they can interact. An instance of this scheme has been developed, generating code for a fault-tolerant ion-trap architecture, which can then be simulated. Heller et al. (2002) have also defined the language Q-HSK, based on C, for programming simulations of quantum algorithms.

Aho and Svore (2003) develop algorithms for optimising the compilation of arbitrary unitary transformations into controlled single-qubit gates; such techniques are likely to be important in practical compiler systems for quantum programming.

Williams (2004) and Nagarajan et al. (2005) have developed a compiler from Selinger's (2004c) language QPL to an architecture which they call SQRAM (Sequential Quantum Random Access Machine); it is a hybrid classical-quantum architecture based on QRAM, and equipped with a specific instruction set. A key feature of the compiler is the implementation of an algorithm for decomposing arbitrary unitary operators into combinations of operators from a fixed universal set. A simulator for the SQRAM instruction set has also been developed.

Grattage and Altenkirch (2005) have developed a compiler from their language QML (2005a) into a representation of quantum circuits, using a categorical semantics as an intermediate form. The quantum circuit specifications can be simulated with a tool developed by the authors or with a third-party system. A characteristic of the QML language, which is reflected in the compiler, is the possibility of quantum control (superpositions of control flow paths) as well as quantum data.

Zuliani (2005a) defines a compiler from qGCL, the quantum guarded command language, to a simple quantum architecture. The compiler is presented in the normal-form style, in which compilation is expressed as a series of algebraic transformations from the source program to a directly executable form. This is a natural framework in which to study compiler correctness, which is the main emphasis of the work.

5. Future Directions

There is much scope for further research in all of the areas that I have surveyed in this article. In language design, the use of complex quantum data structures has not been fully explored, nor has the development of high-level quantum control structures. There has been relatively less emphasis on formal semantics for the imperative languages than for the functional languages, and it would be useful to redress the balance. Denotational semantics of higher-order quantum functional computation is still not solved. The area of compiling quantum programming languages has received relatively little attention. As languages increase in complexity and practical implementations become possible, the sophisticated techniques of classical compiler theory are likely to be very relevant, especially as quantum state may be a severely limited resource.

The physical implementation of quantum computers is an area of active research, and it is not yet clear which physical technology will be most successful. This has implications for programming language and compiler design: any given implementation technology may have a preferred set of operators and measurements that are easiest to implement. It may be necessary to develop a range of compilation techniques targetting different physical architectures, or a range of programming language features designed to exploit the preferred operations of different implementation schemes. Alternatively, it may be necessary to develop program transformation techniques that can shift from one implementation style to another, and in this case it would be desirable for the correctness of the transformations to be justifiable semantically.

6. Bibliography

Many papers have been published in the Archive (`www.arxiv.org`). Papers listed in the bibliography as "arXiv:quant-ph/0409065", for example, can be obtained through URLs of the form `www.arxiv.org/abs/quant-ph/0409065`. Bear in mind that the Archive is an unrefereed forum and papers published there should be regarded as technical reports.

Many of the papers exist in more than one version, often as an Archive paper and in some other form. I have included references to as many versions as I know about. Papers referred to as "manuscripts" were available from their authors' web pages at the time of writing.

References

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

Abramsky, S. (2005) Abstract scalars, loops and free traced and strongly compact closed categories. In *Proceedings of the First Conference on Algebra and Coalgebra in Computer Science*, number 3629 in Lecture Notes in Computer Science, pages 1–31. Springer.

Abramsky, S. and Coecke, B. (2003) Physical traces: Quantum vs. classical information processing. In *Proceedings of the 9th Conference on Category Theory and Computer Science (CTCS 2002)*, volume 69 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science. Also arXiv:cs.CG/0207057.

Abramsky, S. and Coecke, B. (2004) A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society. Also arXiv:quant-ph/0402130.

Abramsky, S. and Coecke, B. (2005) Abstract physical traces. *Theory and Applications of Categories* **14**(6):111–124.

Abramsky, S. and Duncan, R. (2006) A categorical quantum logic. *Mathematical Structures in Computer Science* **?**(?):?–? Preliminary version in Selinger (2004b).

Adão, P. and Mateus, P. (2005) A process algebra for reasoning about quantum security. In Selinger (2005b).

Aho, A. and Svore, K. (2003) Compiling quantum circuits using the palindrome transform. arXiv:quant-ph/0311008.

Albert, D. Z. (1983) On quantum-mechanical automata. *Physics Letters A* **98**(5–6):249–252.

Altenkirch, T. and Grattage, J. (2005a) A functional quantum programming language. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society. Also arXiv:quant-ph/0409065.

Altenkirch, T. and Grattage, J. (2005b) QML: Quantum data and control. Manuscript.

Altenkirch, T., Grattage, J., Vizzotto, J. K. and Sabry, A. (2006) An algebra of pure quantum programming. *Mathematical Structures in Computer Science* **?**(?):?–? Preliminary version in Selinger (2005b); also arXiv:quant-ph/0506012.

Arrighi, P. (2003) Quantum computation explained to my mother. *Bulletin of the EATCS* **80**. Also arXiv:quant-ph/0305045.

Arrighi, P. and Dowek, G. (2004) Operational semantics for formal tensorial calculus. In Selinger (2004b).

Arrighi, P. and Dowek, G. (2005) Linear-algebraic λ-calculus. arXiv:quant-ph/0501150.

Baker, G. (1996) Qgol: a system for simulating quantum computations: theory, implementation and insight. Honours thesis, Macquarie University; available as `www.ifost.org.au/~gregb/q-gol/QgolThesis.pdf`.

Baltag, A. and Smets, S. (2006) The logic of quantum programs. *Mathematical Structures in Computer Science* **?**(?):?–? Preliminary version in Selinger (2004b).

Benioff, P. (1980) The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics* **22**(5):563–591.

Bennett, C. H. and Brassard, G. (1984) Quantum Cryptography: Public-key Distribution and Coin Tossing. In *Proceedings of the IEEE International Conference on Computer Systems and Signal Processing*, pages 175–179.

Bernstein, E. and Vazirani, U. (1993) Quantum complexity theory. In *Proceedings of the ACM Symposium on Theory of Computing*. ACM Press.

Bernstein, E. and Vazirani, U. (1997) Quantum complexity theory. *SIAM Journal on Computing* **26**(5):1411–1473.

Bettelli, S., Calarco, T. and Serafini, L. (2003) Toward an architecture for quantum programming. *The European Physical Journal D* **25**:181–200.

Blute, R. F., Ivanov, I. T. and Panangaden, P. (2003) Discrete quantum causal dynamics. *International Journal of Theoretical Physics* **42**:2025–2041. Also arXiv:gr-qc/0109053.

Brunet, O. and Jorrand, P. (2004) Dynamic quantum logic for quantum programs. *International Journal of Quantum Information* **2**(1):45–54. Also arXiv:quant-ph/0311143.

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

Coecke, B. (2004b) Quantum information-flow, concretely, abstractly. In Selinger (2004b).

Coecke, B. (2005) De-linearizing linearity I: Projective quantum axiomatics from strong compact closure. In Selinger (2005b). Also arXiv:quant-ph/0506134.

Danos, V., D'Hondt, E., Kashefi, E. and Panangaden, P. (2005a) Distributed measurement-based quantum computation. In Selinger (2005b). Also arXiv:quant-ph/0506070.

Danos, V. and Kashefi, E. (2005a) Determinism in the one-way model. arXiv:quant-ph/0506062.

Danos, V. and Kashefi, E. (2005b) Pauli measurements are universal. In Selinger (2005b).

Danos, V., Kashefi, E. and Panangaden, P. (2004) The measurement calculus. arXiv:quant-ph/0412135.

Danos, V., Kashefi, E. and Panangaden, P. (2005b) 1-qubit versus 2-qubit measurement-based quantum computation. Manuscript.

Danos, V., Kashefi, E. and Panangaden, P. (2005c) Parsimonious and robust realisations of unitary maps in the one-way model. *Physical Review A* **72**(064301). Also arXiv:quant-ph/0411071.

Deutsch, D. (1985) Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A* **400**:97–117.

D'Hondt, E. (2005) *Distributed quantum computation: a measurement-based approach*. Ph.D. thesis, Vrije Universiteit Brussel.

D'Hondt, E. and Panangaden, P. (2006) Quantum weakest preconditions. *Mathematical Structures in Computer Science* **?**(?):?–? Preliminary version in Selinger (2004b); also arXiv:quant-ph/0501157.

Di Pierro, A. and Wiklicky, H. (2001) Quantum constraint programming. In *Proceedings of the APPSIA-GULP-PRODE Joint Conference on Declarative Programming*. Online proceedings at `http://www.di.uevora.pt/∼agp01/accepted.html`.

Duncan, R. (2004) Believe it or not, Bell states are a model of multiplicative linear logic. Technical Report PRG-RR-04-18, Programming Research Group, Oxford University Computing Laboratory.

Edalat, A. (2004) An extension of Gleason's theorem for quantum computation. *International Journal of Theoretical Physics* **43**(7–8):1827–1840. Also arXiv:quant-ph/0311070.

Feng, Y., Duan, R., Ji, Z. and Ying, M. (2005) Semantics of a purely quantum programming language. arXiv:cs.PL/0507043.

Feynman, R. P. (1982) Simulating physics with computers. *International Journal of Theoretical Physics* **21**(6–7):467–488.

Gay, S. J. and Nagarajan, R. (2005) Communicating quantum processes. In *Proceedings of the 32nd ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. ACM Press. Preliminary version in Selinger (2004b); also arXiv:quant-ph/0409052.

Gay, S. J. and Nagarajan, R. (2006) Typechecking communicating quantum processes. *Mathematical Structures in Computer Science* **?**(?):?–?

Girard, J.-Y. (2004) Between logic and quantic: a tract. In *Linear Logic in Computer Science*, number 316 in London Mathematical Society Lecture Notes, pages 346–381. Cambridge University Press.

Glendinning, I. (2005) Quantum programming languages and tools. Online catalogue at `www.vcpc.univie.ac.at/∼ian/hotlist/qc/programming.shtml`.

Grädel, E. and Nowack, A. (2003) Quantum computing and abstract state machines. In *Abstract State Machines — Advances in Theory and Applications*, number 2589 in Lecture Notes in Computer Science, pages 309–323. Springer.

Grattage, J. and Altenkirch, T. (2005) A compiler for a functional quantum programming language. Manuscript.

Grover, L. (1996) A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computation*, pages 212–219. ACM Press. Also arXiv:quant-ph/9605043.

Gruska, J. (1999) *Quantum Computing*. McGraw-Hill.

Heller, K., Svore, K. and Kamvar, M. (2002) Q-HSK: a quantum simulation language. Manuscript available as `www1.cs.columbia.edu/∼mkamvar/prev_work/cs4115/PLT_final_writeup.ps`.

Iriyama, S., Ohya, M. and Volovich, I. (2004) Generalized quantum Turing machine and its application to the SAT chaos algorithm. arXiv:quant-ph/0405191.

Jorrand, P. and Lalire, M. (2004) Toward a quantum process algebra. In *Proceedings of the 1st ACM Conference on Computing Frontiers*. ACM Press. Also arXiv:quant-ph/0312067.

Karczmarczuk, J. (2003) Structure and interpretation of quantum mechanics — a functional framework. In *Proceedings of the ACM SIGPLAN Workshop on Haskell*. ACM Press.

Kashefi, E. (2003a) *Complexity Analysis and Semantics for Quantum Computing*. Ph.D. thesis, Imperial College London.

Kashefi, E. (2003b) Quantum domain theory — definitions and applications. In *Proceedings of the International Conference on Computability and Complexity in Analysis*, number 302 – 8/2003 in Fernuniversität Hagen Informatik Berichte. Also arXiv:quant-ph/0306077.

Knill, E. (1996) Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory.

Lalire, M. (2005) A probabilistic branching bisimulation for quantum processes. arXiv:quant-ph/0508116.

Lalire, M. (2006) Relations among quantum processes: Bisimilarity and congruence. *Mathematical Structures in Computer Science* **?**(?):?–?

Lalire, M. and Jorrand, P. (2004) A process algebraic approach to concurrent and distributed computation: operational semantics. In Selinger (2004b). Also arXiv:quant-ph/0407005.

Mateus, P. and Sernadas, A. (2006) Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation* To appear. Also arXiv:math.LO/0503453.

Mauerer, W. (2005) *Semantics and Simulation of Communication in Quantum Computing*. Master's thesis, University Erlangen-Nuremberg.

Maymin, P. (1996) Extending the lambda calculus to express randomized and quantumized algorithms. arXiv:quant-ph/9612052.

Maymin, P. (1997) The lambda-q calculus can efficiently simulate quantum computers. arXiv:quant-ph/9702057.

Mu, S.-C. and Bird, R. (2001) Functional quantum programming. In *Proceedings of the 2nd Asian Workshop on Programming Languages and Systems*.

Nagarajan, R., Papanikolaou, N. and Williams, D. (2005) Simulating and compiling code for the sequential quantum random access machine. In Selinger (2005b).

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

Ömer, B. (1998) *A Procedural Formalism for Quantum Computing*. Master's thesis, Department of Theoretical Physics, Technical University of Vienna.

Ömer, B. (2000) *Quantum Programming in QCL*. Master's thesis, Institute of Information Systems, Technical University of Vienna.

Ömer, B. (2001) Procedural quantum programming. In *Proceedings of the AIP Conference on Computing Anticipatory Systems*. American Institute of Physics.

Ömer, B. (2002) Classical concepts in quantum programming. arXiv:quant-ph/0211100.

Ömer, B. (2003) *Structured Quantum Programming*. Ph.D. thesis, Technical University of Vienna.

Papanikolaou, N. K. (2004) *Techniques for Design and Validation of Quantum Protocols*. Master's thesis, University of Warwick.

Perdrix, S. (2005) Quantum patterns and types for entanglement and separability. In Selinger (2005b).

Perdrix, S. and Jorrand, P. (2004a) Classically-controlled quantum computation. arXiv:quant-ph/0407008.

Perdrix, S. and Jorrand, P. (2004b) Measurement-based quantum Turing machines and questions of universalities. arXiv:quant-ph/0402156.

Perdrix, S. and Jorrand, P. (2004c) Measurement-based quantum Turing machines and their universality. arXiv:quant-ph/0404146.

Petersen, A. and Oskin, M. (2003) A new algebraic foundation for quantum programming languages. In *Proceedings of the 2nd Workshop on Non-Silicon Computing*.

Pratt, V. (1992) Linear logic for generalized quantum mechanics. In *Proceedings of the IEEE Workshop on Physics and Computation*.

Preskill, J. (1998) Lecture notes for physics 219: Quantum information and computation. Available from `www.theory.caltech.edu/people/preskill/ph219`.

Rieffel, E. G. and Polak, W. (2000) An introduction to quantum computing for non-physicists. *ACM Computing Surveys* **32**(3):300–335. Also arXiv:quant-ph/9809016.

Sabry, A. (2003) Modelling quantum computing in Haskell. In *Proceedings of the ACM SIGPLAN Workshop on Haskell*. ACM Press.

Sanders, J. W. and Zuliani, P. (2000) Quantum programming. In *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*. Springer.

Selinger, P. (2004a) A brief survey of quantum programming languages. In *Proceedings of the 7th International Symposium on Functional and Logic Programming*, volume 2998 of *Lecture Notes in Computer Science*. Springer.

Selinger, P., editor (2004b) *Proceedings of the 2nd International Workshop on Quantum Programming Languages*, number 33 in TUCS General Publications. Turku Centre for Computer Science.

Selinger, P. (2004c) Towards a quantum programming language. *Mathematical Structures in Computer Science* **14**(4):527–586.

Selinger, P. (2004d) Towards a semantics for higher-order quantum computation. In Selinger (2004b).

Selinger, P. (2005a) Dagger compact closed categories and completely positive maps. In Selinger (2005b).

Selinger, P., editor (2005b) *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science. Elsevier Science.

Selinger, P. and Valiron, B. (2005) A lambda calculus for quantum computation with classical control. In *Proceedings of the 7th International Conference on Typed Lambda Calculi and Applications (TLCA)*, volume 3461 of *Lecture Notes in Computer Science*. Springer. Also arXiv:cs.LO/0404056.

Selinger, P. and Valiron, B. (2006) A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science* **?**(?):?–?

Shor, P. W. (1994) Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 124–134. IEEE Press.

Skibiński, J. (2001) Haskell simulator of quantum computer. Available from `web.archive.org/web/20010630025035/www.numeric-quest.com/haskell`.

Svore, K., Aho, A., Cross, A., Chuang, I. and Markov, I. (2006) A layered software architecture for quantum computing design tools. *Computer* **39**(1):74–83.

Svore, K., Cross, A., Aho, A., Chuang, I. and Markov, I. (2004) Toward a software architecture for quantum computing design tools. In Selinger (2004b).

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

Udrescu, M., Prodan, L. and Vlăduţiu, M. (2004) Using HDLs for describing quantum circuits: a framework for efficient quantum algorithm simulation. In *Proceedings of the 1st ACM Conference on Computing Frontiers*. ACM Press.

Unruh, D. (2005) Quantum programs with classical output streams. In Selinger (2005b).

Valiron, B. (2004a) *A functional programming language for quantum computation with classical control*. Master's thesis, University of Ottawa.

Valiron, B. (2004b) Quantum typing. In Selinger (2004b).

van Tonder, A. (2003) Quantum computation, categorical semantics and linear logic. arXiv:quant-ph/0312174.

van Tonder, A. (2004) A lambda calculus for quantum computation. *SIAM Journal on Computing* **33**(5):1109–1135. Also arXiv:quant-ph/0307150.

Vizzotto, J. K., Altenkirch, T. and Sabry, A. (2006) Structuring quantum effects: Superoperators as arrows. *Mathematical Structures in Computer Science* **?**(?):?–? Also arXiv:quant-ph/0501151.

Vizzotto, J. K. and da Rocha Costa, A. C. (2005) Concurrent quantum programming in Haskell. In VII Congresso Brasileiro de Redes Neurais, Sessão de Computação Quântica.

Wehr, M. (1996) Quantum computing: a new paradigm and its type theory. Lecture given at the Quantum Computing Seminar, Lehrstuhl Prof. Beth, Universität Karlsruhe.

Wiesner, S. (1983) Conjugate coding. *SIGACT News* **15**(1):78–88. Original manuscript written circa 1970.

Williams, D. (2004) *Quantum Computer Architecture, Assembly Language and Compilation*. Master's thesis, University of Warwick.

Yao, A. C. (1993) Quantum circuit complexity. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 352–361. IEEE Press.

Zuliani, P. (2001) *Quantum Programming*. DPhil thesis, University of Oxford.

Zuliani, P. (2004) Non-deterministic quantum programming. In Selinger (2004b).

Zuliani, P. (2005a) Compiling quantum programs. *Acta Informatica* **41**(7–8):435–474.

Zuliani, P. (2005b) Quantum programming with mixed states. In Selinger (2005b).