Quantum programming with mixed states (extended abstract)

Paolo Zuliani [1]

Department of Computer Science
Princeton University
Princeton, NJ 08544, USA

Abstract

In this paper we offer a programming approach to quantum computation using mixed states. Mixed-state quantum systems generalise standard (pure) quantum systems by allowing the state of the system to be a probabilistic distribution of pure states. We build on previous work by Aharonov *et al.* and generalise their results from quantum circuits to probabilistic (and quantum) programs.

Key words: Quantum programming, mixed state, probabilistic computation, quantum circuit.

1 Introduction

Mixed-state systems are a generalisation of standard quantum systems for which the state is best described by a probability distribution over "pure" quantum states. Mixed state systems find application in the description of "real" quantum systems where, due to unavoidable causes (*e.g.* imperfections in our apparatuses or interactions with the environment), the exact state of the system cannot be specified. On the other hand, the standard model of quantum circuits assumes only pure states [6]. The difficulty in building a scalable quantum computer makes therefore even more important to have a model for quantum computation as close as possible to reality. A recent work by Aharonov *et al.* [1] extends the standard quantum circuit model by allowing mixed states.

The standard approach for dealing with mixed states is the so called density matrix formalism, and that has been used in Aharonov *et al.*'s work. In this paper we instead offer a programming approach based on qGCL, a programming language for quantum computation.

[1] Email: `pzuliani@cs.princeton.edu`

2 Quantum programming

We give here a short presentation of the features of qGCL (a full introduction can be found in [8]).

2.1 Quantum types

We define the type $\mathbb{B} \mathrel{\widehat{=}} \{0,1\}$, which we will treat as booleans or bits, depending on convenience. A classical register of size $n{:}\mathbb{N}$ is a vector of n booleans. The type of all registers of size n is then defined to be the set of boolean-valued functions on $\{0,1,\ldots,n-1\}$:

$$\mathbb{B}^n \mathrel{\widehat{=}} \{0,1,\ldots,n-1\} \longrightarrow \mathbb{B}\,.$$

The quantum analogue of $\mathbb{B}^n$ is the set of complex-valued functions on $\mathbb{B}^n$ whose squared modulus sum to 1:

$$q(\mathbb{B}^n) \mathrel{\widehat{=}} \{\chi{:}\mathbb{B}^n \longrightarrow \mathbb{C} \quad | \sum_{x:\mathbb{B}^n} |\chi(x)|^2 = 1\}\,.$$

An element of $q(\mathbb{B})$ is called a *qubit* and that of $q(\mathbb{B}^n)$ a *qureg*. Classical state is embedded in its quantum analogue by the Dirac delta function:

$$\begin{aligned} &\delta{:}\mathbb{B}^n \longrightarrow q(\mathbb{B}^n) \\ &\delta_x(y) \mathrel{\widehat{=}} (y = x)\,. \end{aligned}$$

The range of δ, $\{\delta_x \mid x{:}\mathbb{B}^n\}$, forms a *basis* for quantum states, that is:

$$\forall \chi{:}q(\mathbb{B}^n) \bullet \chi = \sum_{x:\mathbb{B}^n} \chi(x)\delta_x\,.$$

The Hilbert space $\mathbb{B}^n \longrightarrow \mathbb{C}$ (with the structure making it isomorphic to $\mathbb{C}^{2^n}$) is called the *enveloping space* of $q(\mathbb{B}^n)$. The usual scalar product becomes the application $\langle \cdot,\cdot\rangle{:}q(\mathbb{B}^n) \times q(\mathbb{B}^n) \to \mathbb{C}$ defined by:

$$\langle \psi, \phi\rangle \mathrel{\widehat{=}} \sum_{x:\mathbb{B}^n} \psi(x)^* \phi(x)$$

where * denotes complex conjugation. The *length* of ψ is defined $\|\psi\| \mathrel{\widehat{=}} \langle \psi,\psi\rangle^{\frac{1}{2}}$.

2.2 Quantum language qGCL

qGCL is an extension of pGCL [5], which in turn extends Dijkstra's guarded-command language with a probabilistic choice constructor in order to address probabilism. A guarded-command language program is a sequence of assignments, **skip** and **abort** manipulated by the standard constructors of sequential composition, conditional selection, repetition and nondeterministic choice [3]. A *quantum program* is a pGCL program invoking quantum procedures and the resulting language is called qGCL. Quantum procedures can be of three different kinds: *Initialisation* (or state preparation) followed by *Evolution* and finally by *Finalisation* (or observation).

Initialisation is a procedure which simply assigns to its qureg state the uniform square-convex combination of all standard states

$$\forall \chi{:}q(\mathbb{B}^n) \bullet \mathbf{In}(\chi) \mathrel{\widehat=} \left(\chi := \frac{1}{\sqrt{2^n}} \sum_{x:\mathbb{B}^n} \delta_x\right).$$

Quantum-mechanical systems evolve over time under the action of *unitary* transformations. *Evolution* thus consists of iteration of unitary transformations on quantum state. In qGCL unitary evolution may be introduced in two forms: explicit (unitary) transformations on quantum state and procedures. In this paper we shall use only the former, so for simplicity we do not describe the latter. Evolution of qureg χ under unitary operator U is described via the following assignment:

$$\chi := U(\chi).$$

The *no-cloning* theorem [9] forbids any assignment $\chi := U(\psi)$ if (syntactically) $\chi \neq \psi$.

The content of a qureg can be read (measured) through quantum procedure *Finalisation* and suitable *observables*. An observable is defined from a family of pairwise orthogonal subspaces which together span the enveloping space of the qureg being read. The axioms of quantum mechanics assert that the measurement "reduces" the qureg to lie in one of those subspaces with different probabilities. The result of the measurement is a number which uniquely identifies the "target" subspace.

Let $\mathcal{O}$ be an observable defined by the family of pairwise orthogonal subspaces $\{S_i \mid 0 \leqslant i < m\}$. In our notation we write $\mathbf{Fin}(\mathcal{O}, i, \chi)$ for the measurement of $\mathcal{O}$ on a quantum system described by state $\chi{:}q(\mathbb{B}^n)$, where i stores the result determining the subspace to which state χ is reduced. Finalisation is entirely defined using the probabilistic combinator of pGCL (see [8] for an unabridged treatment); in our notation we write:

$$\mathbf{Fin}\,(\mathcal{O}, i, \chi) \mathrel{\widehat=} \oplus \left[\left(i, \chi := j, \frac{P_{S_j}(\chi)}{\|P_{S_j}(\chi)\|}\right) \text{ @ } \langle \chi, P_{S_j}(\chi)\rangle \;\mid 0 \leqslant j < m \right]$$

where P_{S_j} is the projector onto subspace S_j.

In general, an observable is represented by a self-adjoint operator and the measurable values are exactly the eigenvalues of that operator. It is a generalisation, since by the well-known spectral theorem the eigenspaces of a self-adjoint operator are pairwise orthogonal and complete the enveloping space. That definition of **Fin** remains valid when an observable $\mathcal{O}$ is defined by a self-adjoint operator O.

The BNF syntax for qGCL is as follows:

$$\begin{aligned}
\langle \textit{qprogram} \rangle &::= \langle \textit{qstatement} \rangle \{ \, ; \, \langle \textit{qstatement} \rangle \} \\
\langle \textit{qstatement} \rangle &::= \chi := \langle \textit{unitary op} \rangle(\chi) \mid \\
&\quad\ \mathbf{Fin}(\langle \textit{identifier} \rangle, \langle \textit{identifier} \rangle, \langle \textit{identifier} \rangle) \mid \\
&\quad\ \mathbf{In}(\langle \textit{identifier} \rangle) \mid \\
&\quad\ \mathbf{skip} \mid x := e \mid \langle \textit{loop} \rangle \mid \langle \textit{conditional} \rangle \mid \\
&\quad\ \langle \textit{nondeterministic choice} \rangle \mid \\
&\quad\ \langle \textit{probabilistic choice} \rangle \mid \langle \textit{local block} \rangle \\
\chi &::= \langle \textit{identifier} \rangle \\
\langle \textit{loop} \rangle &::= \mathbf{while}\ \langle \textit{cond} \rangle\ \mathbf{do}\ \langle \textit{qstatement} \rangle\ \mathbf{od} \\
\langle \textit{cond} \rangle &::= \langle \textit{boolean expression} \rangle \\
\langle \textit{conditional} \rangle &::= \langle \textit{qstatement} \rangle \lhd \langle \textit{cond} \rangle \rhd \langle \textit{qstatement} \rangle \\
&\quad\ \text{executes the LHS if predicate } \langle \textit{cond} \rangle \text{ holds} \\
\langle \textit{nondeterministic choice} \rangle &::= \langle \textit{qstatement} \rangle \sqcap \langle \textit{qstatement} \rangle \\
\langle \textit{probabilistic choice} \rangle &::= \langle \textit{qstatement} \rangle \ {}_p\!\oplus \langle \textit{qstatement} \rangle \\
&\quad\ \text{executes (LHS,RHS) with probability } (p, 1-p) \\
\langle \textit{local block} \rangle &::= \mathbf{var}\ \bullet \langle \textit{qstatement} \rangle\ \mathbf{rav}
\end{aligned}$$

where $\langle \textit{unitary op} \rangle(\chi)$ is just some mathematical expression involving qureg χ - such expression should of course denote a unitary operator. qGCL supports procedures and specifications, which we omit here since we shall not use them.

Both probabilistic and nondeterministic choice may be written using a prefix notation, in case the branches are more than two. Let $[\ (P_j, r_j) \mid 0 \leqslant j < m\ ]$ be a finite indexed family of (program, number) pairs with $\sum_j r_j = 1$, then the probabilistic choice in which P_j is chosen with probability r_j is written in prefix form: $\oplus[\ P_j\ @\ r_j \mid 0 \leqslant j < m\ ]$. For nondeterministic choice the notation is similar.

3 Computing with mixed states

In this Section we compare and extend results of Aharonov *et al.* [1]. We begin by generalising their Theorem 1, which established that the quantum circuit model with mixed states is as efficient as the "standard" (*i.e.* unitary) quantum circuit model. We argue that such efficiency extends to any reversible and probabilistic program.

3.1 Equivalence of computing models

Aharonov *et al.* [1] proved that one could use quantum circuits with mixed states, paying only a polynomial slowdown. We generalise this result by means of the following theorem.

Theorem 3.1 *Probabilistic (terminating) programs can be efficiently simulated by reversible probabilistic programs.*

Proof It is well known that deterministic computations can be efficiently simulated by reversible machines [2]. In [10] we proved that any terminating probabilistic program can be replaced by an equivalent but reversible (probabilistic) program. In particular, one can reverse a binary probabilistic choice using a boolean and a conditional as a reverse statement, as shown in the following table:

statement S	reversible statement S_r	inverse statement S_i
$R\ {}_p\oplus\ S$	**push** b ⨾ $(R_r$ ⨾ **push** $T)\ {}_p\oplus\ (S_r$ ⨾ **push** $F)$	**pop** b ⨾ $(R_i \lhd b \rhd S_i)$⨾ **pop** b

where $v{:}D$ for some data type D and b is a boolean variable. □

To see that Theorem 3.1 generalises Aharonov *et al.*'s Theorem 1 we note that a quantum circuit with mixed states Q can be evidently implemented as a probabilistic program P_Q. Next, by virtue of Theorem 3.1, P_Q can be efficiently simulated by a reversible program, which could then be implemented as a unitary transformation.

It is worth seeing how one could actually simulate a quantum program with mixed states using just unitary evolution. In this case the problem is of course how to simulate a measurement unitarily. The standard approach to the problem uses the "superoperator" approach to Quantum Mechanics, in which the state is no longer a complex vector but rather a particular kind of complex matrix, the so-called *density matrix*. Then, admissible operations on a quantum system (including measurements) are postulated to be a special type of linear maps (also called superoperators) over matrices. In particular, any quantum operation is represented by some *completely positive* and *trace-preserving* superoperator. Finally, Stinespring-Kraus' decomposition theorem [4] establishes that any completely positive map is trace-preserving if and only if it is implemented by a unitary operator over a larger space. Such operator is called a *dilation* (or *unitary embedding*).

We now exemplify Stinespring-Kraus' theorem in the special case of a quantum measurement operator. Consider the measurement $\mathcal{O}$ represented by the family of orthogonal finite Hilbert spaces $\{\mathcal{H}_i \mid 0 \leq i < m\}$ decomposing the Hilbert space $\mathcal{H}$:

$$\mathcal{H} = \bigoplus_{0\leq i<m} \mathcal{H}_i$$

where $\oplus$ here denotes direct sum of subspaces. Such measurement is then described by the following dilation:

$$D{:}\mathcal{H} \to \mathcal{H} \otimes \mathcal{H}_E$$

$$D(v) \mathrel{\widehat{=}} \textstyle\bigoplus_{0\leq i<m} P_i(v) \otimes \delta_i$$

where P_i is the projector over $\mathcal{H}_i$ and $\mathcal{H}_E$ is a Hilbert space of dimension m. It can be shown that D is indeed unitary.

The Hilbert space $\mathcal{H}_E$ can be thought as the "environment" and in such case we have that any quantum system evolves unitarily together with its environment, leading eventually to a complicated entanglement. Therefore, we see that one of the main problems to the realisation of quantum computers, *i.e.* decoherence, is mathematically equivalent to entanglement between the computer and its environment. In the case of quantum computation we also observe that the environment can be used as a "pointer" to the state of the computation, as $\mathcal{H}_E$ may describe the status of some macroscopic apparatus returning visible measurements.

We now give an alternative, programming-oriented approach for unitary finalisation. It cannot fully "simulate" finalisation, but it seems to be adequate for all practical purposes. Suppose measurement $\mathcal{O}$ is non-degenerate; we recall that **Fin** is the probabilistic choice:

$$\mathbf{Fin}\,(\mathcal{O}, r, \chi) \mathrel{\widehat{=}} \oplus \left[\left(r, \chi := j, \frac{P_j(\chi)}{\|P_j(\chi)\|} \right) \ @ \ \langle \chi, P_j(\chi) \rangle \ \mid 0 \leqslant j < m \right]$$

where P_j is the projector onto subspace $\mathcal{H}_j$. In Theorem 3.1 we saw how to reverse (binary) probabilistic choice: the multiple choice used by Finalisation can be clearly handled by nested binary choices. Reversibility of assignments are addressed via stack operations: r is a standard variable and this does not pose any problem (the push operation can be implemented as a copy using the CNOT quantum transformation); χ is a qureg and the no-cloning theorem forbids copying of quregs. However, we show that Finalisation can be performed unitarily by using swap operations and an extra qureg. Without loss of generality we consider diagonal Finalisation, since basic results of linear algebra show that any observation can be unitarily reduced to a diagonal observation. It is possible to prove the following refinement:

$$\begin{aligned}
&\mathbf{Fin}(\Delta, r, \chi)\\
&\sqsubseteq\\
&\oplus\,[r := j \ @ \ |\chi(j)| \mid 0 \leqslant j < m\,] \,⨾\, \Box \left[r = i \rightarrow \psi, \chi := \delta_i, \tfrac{\chi(i)}{|\chi(i)|}\delta_i \mid 0 \leqslant i < m \right]
\end{aligned}$$

where $\psi{:}q(\mathbb{B}^m)$. The probabilistic choice over r can be reversed as discussed, and the conditional does not evidently pose problems. For χ we note that $\frac{\chi(i)}{|\chi(i)|}$ is a complex number of modulus 1 (also known as the *global phase*) and Quantum Mechanics' axioms consider χ and ψ as *physically equivalent* states, in the sense that no subsequent measurement is able to distinguish them. Therefore we can unitarily swap χ and ψ and let the computation going on over χ.

A similar argument cannot be applied in the case of degenerate observables. Suppose $\mathcal{O}$ is degenerate, then it is easy to show that:

$$\begin{array}{l}\mathbf{Fin}(\mathcal{O}, r, \chi)\\ =\\ \oplus\,[r := j \mathbin{@} \langle \chi, P_j(\chi)\rangle \mid 0 \leqslant j < m\,] \mathbin{;} \square \left[r = i \rightarrow \chi := \frac{P_j(\chi)}{\|P_j(\chi)\|} \ \mid 0 \leqslant i < m\right]\end{array}$$

and since the P_j's may project over l-dimensional subspaces ($l > 1$), we cannot substitute χ with a physically equivalent qureg. Also, a projector does not preserve traces, so Stinespring-Kraus' implies that we cannot unitarily implement each branch of the conditional.

We conclude by observing that one can always bring finalisation at the end of a computation: this is the so called principle of *deferred measurement* [6]. In qGCL it translates as the following lemma.

Lemma 3.2 (Principle of deferred measurement) *For $\chi{:}q(\mathbb{B}^n)$, $r{:}\mathbb{B}^n$, observable $\mathcal{O}$, and unitary operator $\mathcal{U}$ over χ, it holds:*

$$\begin{pmatrix}\mathbf{Fin}(\mathcal{O}, r, \chi)\mathbin{;}\\ \chi := U(\chi)\end{pmatrix} = \begin{pmatrix}\chi := U(\chi)\mathbin{;}\\ \mathbf{Fin}(\mathcal{O}', r, \chi)\end{pmatrix}$$

where $\mathcal{O}'$ is the observable corresponding to the self-adjoint operator UOU^{-1} (O corresponding to $\mathcal{O}$).

Proof Omitted.

Therefore one could in principle avoid irreversible computations until it is absolutely necessary, at the end of the computation (though it remains to be understood if this can be done also for iterating computations, *i.e.* programs using loops).

3.2 Probabilistic subroutines

In this Section we address Aharonov *et al.*'s [1] solution for the "subroutine problem" in quantum computation: in general, the function computed by a quantum circuit is a probabilistic one, therefore a problem arises when one wants to use such functions as subroutines in a bigger quantum circuit, since the standard theory of quantum circuits allows pure states only. Aharonov *et al.* first show how to formalise probabilistic function in the mixed-state model and then they show that such model is only polynomially faster than the standard quantum circuit model. In particular, their Theorem 2 establishes that any probabilistic function can be "simulated" by a standard quantum circuit using only a polynomially greater number of gates, with respect to the mixed-state quantum circuit implementation. Theorem 2 states that

$FQP^{FQP} = FQP$, where FQP is the set of probabilistic functions efficiently computable by quantum circuits.

A probabilistic function is defined as a function which outputs a number with probability depending on the input. More formally:

$$f{:}\mathbb{B}^m \to [0,1]^{\mathbb{B}^p}$$

$$f(i) \mathrel{\widehat{=}} j \quad \text{with probability } p_{i,j}\,, \quad \forall i{:}\mathrm{dom}(f) \bullet \textstyle\sum_j p_{i,j} = 1\ .$$

It can be shown that any such function can be represented as a probabilistic choice over a number of deterministic functions:

$$f = \oplus\ [\ d\ @\ w_d \mid d{:}(\mathbb{B}^m \to \mathbb{B}^p)\ ]$$

where $w_d \mathrel{\widehat{=}} \prod_i p_{i,d(i)}$ is of course the probability that (deterministic) function d gets applied. Aharonov *et al.* use this decomposition to define a *subroutine gate* that implements f as a mixed state in which the unitary version of all the deterministic functions d's are applied to the initial state with the induced probability w_d's. Next, they show that the subroutine gate can be efficiently implemented unitarily (the result mainly stems from the previous Theorem 1, of course).

We now consider the same problem in qGCL. The subroutine gate which implements function $f \mathrel{\widehat{=}} \oplus\ [\ f_d\ @\ w_d \mid 0 \le d < t\ ]$ is defined as:

$$G \mathrel{\widehat{=}} \oplus\ [\ \chi := U_d(\chi)\ @\ w_d \mid 0 \le d < t\ ]$$

where U_d is the unitary implementation of function f_d. We argue that in qGCL there is no "subroutine problem", *i.e.* probabilistic functions are naturally manipulated by the language. In fact, given the mixed state $\rho = \{(\psi_i, b_i) \mid 0 \le i < n\}$ it is easy to show that the evolution of ρ by G in qGCL is equivalent to that offered by the subroutine gate. In qGCL it can be proved that:

$$(\oplus\ [\ \chi := \psi_i\ @\ b_i \mid 0 \le i < n]\, ⨾\, G) = \\ \oplus\ [\ \chi := U_d(\psi_i)\ @\ b_i w_d \mid 0 \le i < n, 0 \le d < t\ ]$$

which is exactly the action over ρ of the subroutine gate implementing f.

With respect to the unitary implementation of G we can show the following refinement:

$$G \sqsubseteq (\oplus\ [r := d\ @\ w_d \mid 0 \leqslant d < t\ ]\ ⨾\ \square[r = d \to \chi := U_d(\chi) \mid 0 \leqslant d < t])$$

which means that G can be implemented (as intuition suggests) via a classical probabilistic choice and then a conditional. The probabilistic choice can be of course realised as a quantum computation. Without loss of generality we may suppose that $\exists k \mid 2^k = t$ and therefore with a qureg of size k we can simulate the probabilistic choice above as the tossing of k biased coins. The complexity of this method is parameterised by the number t of deterministic functions composing f, and an efficient implementation is possible, as the k quantum coins are independent and thus can be initialised and measured by an operator acting on the whole qureg. Therefore an equivalent of Theorem 2 holds for qGCL. Actually, Theorem 3.1 allows us to state that:

Theorem 3.3 *Probabilistic subroutines do not strengthen reversible computation, since they can be efficiently simulated by reversible programs.*

3.3 Error propagation

Finally, we set the background for studying error propagation in quantum programs with mixed states. Aharonov *et al.* [1] showed that in quantum circuit with mixed states, errors add linearly. Their Theorem 3 states that if a circuit using L gates, each with at most ϵ error, then the total error of the circuit is at most $O(L\epsilon)$. The result is proved within the superoperator approach, by defining an extension of the usual trace norm of operators.

Intuitively, a faulty gate F can be described in qGCL as:

$$F = (\chi := U_\epsilon(\chi) \,{}_\delta\!\oplus\, \chi := U(\chi))$$

where U_ϵ is the unitary "error" operator, and δ is the probability that U_ϵ is applied, instead of the correct operator U. The error of such a gate could then be just $\delta\epsilon$, where $\epsilon \mathrel{\widehat{=}} \sup_\chi \|U(\chi) - U_\epsilon(\chi)\|$. It seems that this model offers some flexibility over the single-parameter model of Aharonov *et al.*, since F can model the difference between the correct and the perturbed state, but also the probability of this happening. That might result useful when modelling real mixed-state systems (this is actually the model used in [7]). The U_ϵ being unitary is an assumption which turns out to be handy in calculations, but we recall that by the Stinespring-Kraus' theorem we can replace any quantum operation with a suitable unitary operator. This motivates such an assumption of the unitarity of U_ϵ.

However, one quickly realises that this error model is not adequate for calculations, because of the explosion of the probabilistic choice branches. We thus need a more compact formalism, and that is the task we start here.

We begin by defining mixed states, which can be thought as probability distributions over finite sequences of pure states.

Definition 3.4 *For a Hilbert space $\mathcal{H}$ we define*

$$mixed(\mathcal{H}) \mathrel{\widehat{=}} \Delta(iseq(\mathcal{H}))$$

where $iseq(\mathcal{H})$ denotes the finite injective sequences of elements of $\mathcal{H}$.

Any element of mixed($\mathcal{H}$) can be written as the matrix product σM, where σ is the row vector of probabilities, and M is the matrix of the amplitudes (each column corresponds to a vector state). We have the following lemma.

Lemma 3.5

$$\sigma|M|^2{:}\Delta(\mathcal{H})$$

$$\text{mixed}(\mathcal{H}_1 \bigoplus \mathcal{H}_2) = \text{mixed}(\mathcal{H}_1)\ cc\ \text{mixed}(\mathcal{H}_2)$$

$$\text{mixed}(\mathcal{H}_1 \bigotimes \mathcal{H}_2) = \text{mixed}(\mathcal{H}_1)\ \otimes\ \text{mixed}(\mathcal{H}_2)$$

where $|M|^2$ is the matrix of the moduli and cc means convex combination.

Proof Omitted.

The $\otimes$ tensor product of sequences used in the last lemma can be easily defined from the regular tensor product of vectors. We now define a unitary evolution for mixed states.

Definition 3.6 *For a unitary operator U over quregs in $\mathcal{H}$ we define $\mathcal{U}$ over mixed states:*

$$\mathcal{U}\text{: } \textit{mixed}(\mathcal{H}) \rightarrow\ \textit{mixed}(\mathcal{H})$$

$$\mathcal{U}(p_i, \chi_i; \sigma') \mathrel{\widehat{=}} p_i, U(\chi); \mathcal{U}(\sigma')$$

$$\mathcal{U}([\,]) \mathrel{\widehat{=}} ([\,])$$

where $[\,]$ denotes the empty sequence and σ' the tail of σ.

Lemma 3.7 *Let $\mathcal{H}$ be a Hilbert space and U a unitary operator over $\mathcal{H}$, then:*

- mixed$(\mathcal{H})$ *is a Hilbert space;*
- $\mathcal{U}$ *is unitary.*

Proof Omitted.

The faulty gate F can be generalised to work on mixed states.

Definition 3.8

$$\mathcal{F}{:}\textit{mixed}(\mathcal{H}) \rightarrow \textit{mixed}(\mathcal{H})$$

$$\mathcal{F}(p_i, \chi_i; \sigma') \mathrel{\widehat{=}} p_i\delta, U_\epsilon(\chi_i); p_i(1-\delta), U(\chi_i); \mathcal{F}(\sigma')$$

$$\mathcal{F}([\,]) \mathrel{\widehat{=}} ([\,])$$

4 Conclusions

We offered a programming approach for a model of quantum computation based on mixed states, and in doing so we obtained mild generalisations of previous work. As a future work we hope to use this formalisation to analyse the propagation of errors in a quantum computation involving mixed states.

We aim at proving bounds (and trade-offs, possibly) relating the probability of faulty behaviour and the discrepancy from expected behaviour.

Acknowledgement

This work has been supported by a *Marie Curie Outgoing International Fellowship* within the 6th Framework Programme of the European Commission.

References

[1] D. Aharonov, A. Kitaev, and N. Nisan. Quantum circuits with mixed states. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 20–30. ACM Press, 1998.

[2] Charles H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973.

[3] E. W. Dijkstra. Guarded commands, nondeterminacy and the formal derivation of programs. *CACM*, 18:453–457, 1975.

[4] Karl Kraus. *State, Effects, and Operations*, volume 190 of *Lecture Notes in Physics*. Springer-Verlag, 1983.

[5] Carroll Morgan and Annabelle McIver. *pGCL*: formal reasoning for random algorithms. *South African Computer Journal*, 22:14–27, 1999.

[6] Micheal A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000.

[7] Asher Peres. *Quantum Theory: Concepts and Methods*. Kluwer Academic Publishers, 1998.

[8] J. W. Sanders and P. Zuliani. Quantum programming. *Mathematics of Program Construction, Springer-Verlag LNCS*, 1837:80–99, 2000.

[9] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[10] Paolo Zuliani. Logical reversibility. *IBM Journal of Research and Development*, 45(6):807–818, 2001.