Proc. QPL 2004, pp. 179–195

Non-deterministic quantum programming

Paolo Zuliani*

Abstract

In standard computation, non-determinism is used for specifying programs' behaviour, without having to specify details of implementation. In quantum computation, non-determinism is either meant to be "classical" probabilism or it is not considered at all, since quantum computation is the physical theory of computation and thus it does not deal with non-implementable features. In this work we will instead show that non-determinism may be useful also in quantum computation. In particular, we consider non-determinism embedded in a programming language for quantum computation, the quantum Guarded-Command Language (qGCL), and use that for describing and reasoning about counterfactual computation and mixed-state systems.

1 Introduction

In standard computation, non-determinism provides a way for specifying and reasoning about programs. In particular, non-determinism is used for specifying programs' behaviour, without having to specify details of implementation. Such details may be made more precise (refined) at a later stage, though program correctness is preserved throughout reasoning. In quantum computation, non-determinism is either meant to be "classical" probabilism or it is not considered at all, since quantum computation is the physical theory of computation and thus it does not deal with non-implementable features. In this work we will instead argue that non-determinism may be useful also in quantum computation.

We consider non-determinism embedded in a programming language for quantum computation, the quantum Guarded-Command Language, qGCL [9]. qGCL has a rigorous semantics and an associated refinement calculus, and it has been successfully used to describe and reason about all known quantum algorithms and also to derive one of them (the Deutsch-Jozsa algorithm) from its specification. In this work, we use qGCL equipped with a non-deterministic choice construct to model and reason about counterfactual computation and quantum mixed-state systems.

Counterfactuality is the capacity to infer propositions about an event by the sole fact that the event might have occurred, it is not required the event to actually take place. Counterfactual computation [6] makes use of quantum mechanics' peculiarities to infer the outcome of a computation without running that computation. In particular, it is possible to devise methods for probabilistically inferring the outcome of a computation without

*Facoltà di Scienze e Tecnologie Informatiche, Libera Università di Bolzano, piazza Domenicani 3, 39100 Bolzano, Italy, pzuliani@unibz.it

actually running the computation: the mere fact that the quantum computer implementing that computation might have run is sufficient. We illustrate an example of counterfactual computation and show that it can be rigorously formalised and reasoned about by means of qGCL.

Mixed-state systems [3] 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. We show that qGCL can model such systems by proving that the treatment of mixed states in qGCL is consistent with the corresponding quantum formalism, the density matrix formalism.

The qGCL treatment of counterfactual computation and mixed-state systems thus provide an example of use of programming languages for describing and analysing quantum computation in the broadest sense.

2 Quantum programming

We give here a short presentation of the features of qGCL (a full introduction can be found in [9]). We assume some familiarity with the basics of quantum computation [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 given by transform q and it 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* (called the *standard 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 z^* is the complex conjugate of $z{:}\mathbb{C}$. 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 the *probabilistic* Guarded-Command Language, pGCL [7], which in turn extends Dijkstra's Guarded-Command Language GCL [1] with probabilism. A Guarded-Command Language program is a sequence of assignments and **skip** manipulated by the standard constructors of sequential composition, conditional selection, repetition and nondeterministic choice [1]. Assignments is in the form $x := e$, where x is a vector of program variables and e a vector of expressions whose evaluations always terminate with a single value. pGCL denotes the Guarded-Command Language extended with the binary constructor ${}_p\oplus$ for $p{:}[0,1]$, in order to deal with probabilism. Other pGCL basic statements and constructors which will be used in this paper are:

- **var** • **rav**, variable declaration and local block;
- sequential composition, $R \fatsemi S$;
- iteration, $\mathbf{while}\ cond\ \mathbf{do}\ S\ \mathbf{od}$;
- conditional, $R \lhd cond \rhd S$, executes R if $cond$ is true and S otherwise;
- probabilistic choice, $R\ {}_p\oplus\ S$, which executes R with probability p and S with probability $1 - p$;
- non-deterministic choice, $R \,\Box\, S$, that executes either R or S, the choice of which is unknown at the current level of abstraction (it is also known as "demonic" choice);
- procedure declaration, $\mathbf{proc}\ P(param) \mathrel{\widehat{=}} body$, where $body$ is a pGCL statement and $param$ is the parameter list, which may be empty. Parameters can be declared as **value**, **result** or **value result**: a **value** parameter is read-only, a **result** parameter is write-only and a **value result** parameter can be read and written. Procedure P is invoked by simply writing its name and filling the parameter list according to P's declaration.

For the probabilistic combinator ${}_p\oplus$ we allow p to be an expression whose evaluation returns a real in $[0,1]$. Probabilistic choice may be written using a prefix notation, in case the branches are more than two. Let $[\ (P_j, r_j) \bullet 0 \leqslant j < m\ ]$ be a finite indexed family of (program, number) pairs with $\sum_{0 \leqslant j < m} r_j = 1$, then the probabilistic choice in which P_j is chosen with probability r_j is written in prefix form

$$\oplus[\ P_j \ @\ r_j \bullet 0 \leqslant j < m\ ]$$

(whose advantage is to avoid the normalising factors required by nested infix form).

Semantics for pGCL can be given either relationally [4] or in terms of expectation transformers [7]. Expectation-transformer semantics is a probabilistic extension of the

predicate-transformer one. In predicate-transformer semantics a transformer maps post-conditions to their weakest pre-conditions. Analogously, an expectation transformer represents a computation by mapping post-expectations to their greatest pre-expectations. We shall retain the wp prefix notation of predicate-transformer calculus for convenience and we denote the greatest pre-expectation of post-expectation q on program P by $wp.P.q$. For a standard predicate p we denote by $[p]$ its embedding into expectation transformers: the greatest pre-expectation $wp.P.[p]$ is then the *minimum guaranteed probability* that p holds after the execution of P.

pGCL enjoys a refinement calculus, which derives from the semantics above; when we say that program Q refines program P, written $P \sqsubseteq Q$, we mean:

$$P \sqsubseteq Q \mathrel{\widehat{=}} \forall q{:}\mathcal{Q} \bullet wp.P.q \Rrightarrow wp.Q.q \ .$$

Intuitively, $P \sqsubseteq Q$ means that Q is at least at deterministic as P. When $P \sqsupseteq Q$ and $P \sqsubseteq Q$ then P and Q are equal programs and we write $P = Q$.

In pGCL (demonic) non-determinism is expressed semantically as the combination of all possible probabilistic resolutions:

$$wp.(P \,\Box\, Q) = \sqcap\{wp.(P \,{}_r\!\oplus\, Q) \bullet r{:}[0,1]\}.$$

Thus a non-deterministic choice between two programs is refined by any probabilistic choice between them:

$$\forall r{:}[0,1] \bullet P \,\Box\, Q \sqsubseteq P \,{}_r\!\oplus\, Q.$$

2.2.1 Quantum programs

A *quantum program* is a pGCL program invoking quantum procedures and the resulting language is called qGCL. Quantum procedures can be of three 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

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

Evolution models the evolution of quantum systems and consists of iteration of unitary transformations on quantum state. A unitary operator U is invertible and preserves scalar products or, equivalently:

$$\forall \chi, \psi{:}q(\mathbb{B}^n) \bullet \langle U(\chi), U(\psi)\rangle = \langle \chi, \psi\rangle.$$

In qGCL evolution is modelled via assignment: for example, $\chi := U(\chi)$ models the evolution of qureg χ by means of unitary transform U.

The content of a qureg can be read (measured) through quantum procedure *Finalisation* and suitable *observables*. Let $\mathcal{O}$ be an observable defined by the family of pairwise orthogonal subspaces $\{S_j \mid 0 \leqslant j < m\}$. In our notation we write $\mathbf{Fin}[\mathcal{O}](i, \chi)$ for the

measurement of $\mathcal{O}$ on a quantum system described by state $\chi{:}q(\mathbb{B}^n)$. After the measurement, variable i stores the index of the subspace to which the state is reduced and χ stores that state. Finalisation is entirely defined using the probabilistic combinator of pGCL:

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

where P_{S_j} is the projector onto subspace S_j. In the case of the one-dimensional space $\mathbb{C}\psi \mathrel{\widehat{=}} \{\alpha\psi \bullet \alpha{:}\mathbb{C}\}$ spanned by $\psi{:}q(\mathbb{B}^n)$ the projector is defined by:

$$\forall \chi{:}q(\mathbb{B}^n) \bullet P_\psi(\chi) \mathrel{\widehat{=}} \langle \psi, \chi \rangle \psi \ .$$

We also note that if $\|P_{S_j}(\chi)\| = 0$ then $\langle \chi, P_{S_j}(\chi) \rangle = 0$ and therefore by laws P-1 and P-2 (see Appendix) the definition holds in that case, too.

The definition of **Fin** remains also valid when an observable $\mathcal{O}$ is defined by a self-adjoint operator O. In that case the projector for the j-th eigenspace of O is written $P^j_{\mathcal{O}}$.

3 Counterfactuality

Counterfactuality is the fact that the sole possibility for an event to occur allows one to gain some information about that event, even though it did not actually occur. Counterfactual computation [5, 6] uses peculiar features of quantum mechanics to infer counterfactual statements about the result of a computation. In particular, it is possible to devise methods for probabilistically inferring the outcome of a computation without actually running the computation: the mere fact that the quantum computer implementing that computation might have run is sufficient.

One of the first examples of counterfactuality was given by Elitzur and Vaidman [2] with the so-called *interaction-free* measurements. That technique allows determining the presence of an object by means of a test particle, possibly with no "interaction" occurring between the object and the test particle. A potential application of this technique is the acquisition of the image of an object without any light or other radiation interacting with the object (see [10] for example).

If one replaces the object with a quantum computer implementing some computation C and the test particle with the computer's "switch", it is then possible to know the outcome of computation C without the computer ever being turned on. This application of quantum mechanics is known as counterfactual computation and it was firstly introduced by Jozsa [5] and then further formalised by Mitchison and Jozsa [6].

4 Counterfactual computation

4.1 Introduction

We begin by recalling the terminology and concepts introduced by Mitchison and Jozsa [6]. Suppose we are given a decision problem (*i.e.* a problem with a binary solution, "yes" or "no") and a quantum computer QC with an "on-off" switch programmed to solve that

problem when the switch is set to "on". Therefore we need a qubit to represent the switch and another qubit for the result of the computation. The computer might need an extra qureg to use during its functioning, thus we need in total two qubits and a qureg, whose size depends on the problem being solved.

When the switch is set to "on" the computer carries out one of the two unitary operations QC_0 or QC_1, depending on the answer of the problem; we suppose the computation takes at most T time units. By mapping "off" to 0 and "on" to 1 we see that QC_0 is just the identity transform over the switch and output qubits, while QC_1 is Feynman's CNOT transformation on the same two qubits. We recall the definition of CNOT on standard bits:

$$\forall c, x{:}\mathbb{B} \bullet CNOT(c, x) \mathrel{\widehat{=}} (c, \neg xc + \neg cx) \ .$$

That definition is easily lifted to qubits via δ.

The goal is to start with the switch "off" and, after at least a time T, to determine which operation QC_0 or QC_1 has been performed, without setting the switch to "on". We assume we are unable to access the extra qureg.

In qGCL the computer QC can be modelled as:

$$QC \mathrel{\widehat{=}} CNOT \,\square\, \mathbf{skip}$$

thus naturally representing our ignorance about the inner working of the computer and the result of the decision problem. However, another equivalent modellisation of QC will be more useful for us. By semantics arguments it is possible to show that:

$$QC = \left(\begin{array}{l} \mathbf{var}\ t{:}\mathbb{B}\bullet \\ \quad CNOT \lhd t \rhd \mathbf{skip} \\ \mathbf{rav} \end{array} \right)$$

That is, non-deterministic choice can be seen as a conditional over an uninitialised boolean variable.

We now code in qGCL the definition of *protocol* given by Mitchison and Jozsa [6]. For a datatype T we denote by $seq(T)$ the datatype of the sequences of elements of type T. The empty sequence is denoted by $\langle\,\rangle$; concatenation is denoted by $+$.

Definition 4.1. A **protocol** G is a terminating procedure with the following signature:

$$\mathbf{proc}\ G\ (\mathbf{value}\ t{:}\mathbb{B},\ \mathbf{result}\ o{:}seq(\mathbb{B}^n),\ \mathbf{result}\ s{:}seq(\mathbb{B})) \mathrel{\widehat{=}} body$$

where:

- $body$, according to Mitchison and Jozsa [6], is "*a sequence of steps where each step is one of the following:*
  - *(a) A unitary operation (not involving the computer) on a finite number of specified qubits.*
  - *(b) A measurement on a finite number of specified qubits.*
  - *(c) An 'insertion of the computer'* (either QC_0 or QC_1)*, where the state of two selected qubits is swapped into the switch and output registers of the computer, a time T is allowed to elapse and finally the states are swapped back out into the two selected qubits".*

- t specifies the computation to be performed by the quantum computer QC (*i.e.* either QC_0 or QC_1).
- o returns the list of outcomes of the measurements of steps of type (b).
- s returns the list of "switch observation" outcomes resulting from the measurement of the switch qubit after each insertion of the computer (steps of type (c)).

The two lists o and s collectively denote a *history* [6]. We are now ready to formalise the definition of counterfactual computation in our approach.

Definition 4.2. A sequence of measurement outcomes $m{:}seq(\mathbb{B}^n)$ is a *counterfactual outcome* of type $t{:}\mathbb{B}$ if there exists a protocol G satisfying the following two conditions:

$$1) \quad \forall c{:}seq(\mathbb{B}) \bullet wp.G(1-t,o,s).[o = m \wedge s = c] \equiv \mathbf{0}$$
$$2) \quad \forall c{:}seq(\mathbb{B}) \bullet c \neq 0^* \quad \text{iff} \quad wp.G(t,o,s).[o = m \wedge s = c] \equiv \mathbf{0},$$

where 0^* denotes an all-zero sequence.

Condition 1 states that when QC_{1-t} is used in the protocol, m is seen with probability zero, *i.e.* it never occurs. Condition 2 states that if QC_t is used then for any switch sequence but an all-off one, m is never seen. The only way to have m as outcome sequence is when QC_t is used and the switch sequence contains only "off"s. Therefore we may say that we can infer the result (t) of the computation for "free", since the switch of the quantum computer was always found at "off".

4.2 Example

We code here in qGCL the following example developed by Jozsa [5]. We first explain it in an informal way, in order to put the idea forward.

It starts with the switch and output registers set to state δ_{00}. They are then "rotated" to state $(\cos\theta \cdot \delta_{00} + \sin\theta \cdot \delta_{10})$, where $\theta \mathrel{\widehat{=}} \frac{\pi}{2N}$ for some positive integer N. The quantum computer is then inserted, thereby giving the state $(\cos\theta \cdot \delta_{00} + \sin\theta \cdot \delta_{1t})$, depending on which QC_t is used. We now measure the output register (right-hand qubit) and we have to distinguish two cases: $t = 0$ and $t = 1$. If $t = 0$ then the measurement does not affect the state and we repeat the preceding steps, from the rotation of the qubits on. If $t = 1$ the measurement reduces the state to δ_{00} with probability $\cos^2\theta$ and to δ_{11} with probability $\sin^2\theta$. In the former case the protocol repeats the preceding steps; in the latter case we learn that the result of the computation is 1 and the computer has run, thereby halting the protocol.

After N iterations, if $t = 0$ the state will have been "rotated" to δ_{10} with probability 1. If $t = 1$, the state will be δ_{00} with probability $\cos^{2N}\theta$. In this case we learn that the answer to the decision problem is 1 without running the computer, since the switch has always been seen at "off"! The probability $\cos^{2N}\theta$ approaches 1 as N grows. To summarise:

- if the answer to the problem is 0 we learn that for certainty, but the computer has run;

- if the answer is 1 then with high probability we learn that for "free", *i.e.* without running the computer.

In qGCL the protocol is formalised by procedure S:

$$
\begin{array}{l}
\textbf{proc}\ S\ (\textbf{value}\ t{:}\mathbb{B},\ \textbf{result}\ o{:}seq(\mathbb{B}),\ \textbf{result}\ s{:}\mathbb{B}) \mathrel{\widehat{=}} \\
\quad \textbf{var}\ \chi{:}q(\mathbb{B}^2),\ r{:}\mathbb{B}\,,i{:}\{0,\ldots,N\} \bullet \\
\qquad r,i,o := 0,0,\langle\rangle ; \\
\qquad \chi := \delta_{00} ; \\
\qquad \textbf{while}\ (i < N \wedge \neg r)\ \textbf{do} \\
\qquad\quad \chi := (H_\theta \otimes \mathbb{1})(\chi) ; \\
\qquad\quad QC(t,\chi) ; \\
\qquad\quad \textbf{Fin}[\mathcal{O}](r,\chi) ; \\
\qquad\quad (\textbf{skip}) \lhd r \rhd (i,o := i+1, o+r) \\
\qquad \textbf{od} \\
\qquad \textbf{Fin}[\mathcal{S}](s,\chi) ; \\
\qquad o := o + s \\
\quad \textbf{rav}
\end{array}
$$

where:

χ is the qureg of size 2 representing the switch qubit and the output qubit;

N is a positive integer;

H_θ is the (unitary) "rotation" transformation defined as ($\theta = \frac{\pi}{2N}$):

$$
\begin{array}{l}
H_\theta{:}q(\mathbb{B}) \to q(\mathbb{B}) \\
H_\theta(\chi)(x) \mathrel{\widehat{=}} (1-x)(\chi(0)\cos\theta - \chi(1)\sin\theta) + x(\chi(0)\sin\theta + \chi(1)\cos\theta)
\end{array}
$$

$\mathbb{1}$ is the identity transform of appropriate size;

QC is the quantum computer (see Section 4.1):

$$QC(t,\chi) \mathrel{\widehat{=}} (\chi := CNOT(\chi) \lhd t \rhd \textbf{skip});$$

observable $\mathcal{O}$ measures the output qubit ($\oplus$ denotes direct sum of subspaces):

$$\mathcal{O} \mathrel{\widehat{=}} [(\mathbb{C}\delta_0 \oplus \mathbb{C}\delta_1) \otimes \mathbb{C}\delta_0,\ (\mathbb{C}\delta_0 \oplus \mathbb{C}\delta_1) \otimes \mathbb{C}\delta_1].$$

observable $\mathcal{S}$ measures the switch qubit:

$$\mathcal{S} \mathrel{\widehat{=}} [\mathbb{C}\delta_0 \otimes (\mathbb{C}\delta_0 \oplus \mathbb{C}\delta_1),\ \mathbb{C}\delta_1 \otimes (\mathbb{C}\delta_0 \oplus \mathbb{C}\delta_1)];$$

We have omitted the extra qureg for the quantum computer, as it is not useful to our aims; we recall that δ_{00} is just shorthand for $\delta_0 \otimes \delta_0$.

The sequence $m \mathrel{\widehat{=}} 0^{N+1}$, *i.e.* the $(N+1)$-long sequence of zeroes, is a counterfactual computation of type 1. In particular m satisfies the conditions:

$$\begin{aligned} &1) \quad \forall c{:}\mathbb{B} \bullet wp.S(0,o,s).[o = 0^{N+1} \wedge s = c] \equiv \mathbf{0} \\ &2) \quad c = 1 \quad \text{iff} \quad wp.S(1,o,s).[o = 0^{N+1} \wedge s = c] \equiv \mathbf{0}. \end{aligned}$$

Condition 1 expresses the fact that 0^{N+1} is never seen if $t = 0$. Condition 2 states that 0^{N+1} is seen if and only if the switch is at off.

Furthermore we have that :

$$3) \quad wp.S(1,o,s).[o = 0^{N+1} \wedge s = 0] \equiv \cos^{2N}\theta,$$

and in this case the computer has not run ($s = 0$), yet we know that the answer to our problem is 1. Therefore we have a probability $\cos^{2N}\theta$ of learning the result for "free", *i.e.* without running the computer. By increasing N we can make this probability as close to 1 as we wish.

Complete proof for the three claims above can be found in [11].

4.3 Probabilistic extension

In this section we formalise in qGCL the probabilistic extension of counterfactual computation proposed by Mitchison and Jozsa [6]. In particular, they considered the case in which we allow a relaxation of conditions 1 and 2 of the definition of counterfactual outcome (definition 4.2). We thus have the following definition.

Definition 4.3. A sequence of measurement outcomes $m{:}seq(\mathbb{B}^n)$ is an *approximate counterfactual outcome* of type $t{:}\mathbb{B}$ if there exists a protocol G satisfying the following two conditions:

$$\begin{aligned} &1') \quad \forall c{:}seq(\mathbb{B}) \bullet wp.G(1-t,o,s).[o = m \wedge s = c] < \epsilon \\ &2') \quad \textstyle\sum_{c:seq(\mathbb{B}),c\neq 0^*} wp.G(t,o,s).[o = m \wedge s = c] < \epsilon \end{aligned}$$

where 0^* denotes any all-zero ("all-off") sequence and ϵ is a small real in the $(0,1]$ interval.

Condition $2'$ means that m has little probability to be seen when the computer has run (because $c \neq 0^*$). Similarly, condition $1'$ means that m has little probability to be seen when the "wrong computer" QC_{1-t} is used. Together, the two conditions ensure that when we see m then with high probability the answer to the decision problem is t, and with high probability the quantum computer QC has not run.

4.4 Probabilistic protocol

We now give an example of probabilistic protocol which can infer the answer to the decision problem with certainty, but requires a run of the quantum computer with probability 50%. We first draft the functioning of the protocol in words, then we code it in qGCL, and

we finally prove its correctness. Again, for simplicity we write the state of the switch and of the output qubits as a single qureg.

We start with the switch and output register in the equally-weighted superposition of standard states, that is $\chi = \frac{1}{2}\sum_{i:\mathbb{B}^2} \delta_i$. Then we perform phase inversion on state δ_{11}, thus giving $\chi = \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{10} - \delta_{11})$. We apply the quantum computer:

$$\chi = \begin{cases} v_0 \mathrel{\widehat{=}} \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{10} - \delta_{11}) & \text{if } t = 0 \\ v_1 \mathrel{\widehat{=}} \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{11} - \delta_{10}) & \text{if } t = 1 \end{cases}$$

We now measure χ using observable $\mathcal{V}$ ($\perp$ denotes the orthogonal subspace)

$$\begin{aligned} \mathcal{V} &\mathrel{\widehat{=}} [V_0, V_1, (V_0 \oplus V_1)^{\perp}] \\ V_0 &\mathrel{\widehat{=}} \mathbb{C}v_0\ , \quad V_1 \mathrel{\widehat{=}} \mathbb{C}v_1\ . \end{aligned}$$

Since $v_0 \perp v_1$, we are thus able to learn t with certainty and we dot not perturb state χ. A subsequent measurement of the switch qubit reduces χ to its "off" subspace (*i.e.* switch set to 0) with probability 50%.

In qGCL the protocol is coded as follows:

$$\begin{array}{l} \mathbf{proc}\ K\ (\mathbf{value}\ t{:}\mathbb{B},\ \mathbf{result}\ o{:}\mathbb{B},\ \mathbf{result}\ s{:}\mathbb{B}) \mathrel{\widehat{=}} \\ \quad \mathbf{var}\ \chi{:}q(\mathbb{B}^2) \bullet \\ \qquad \mathbf{In}(\chi)\fatsemi \\ \qquad \chi := T_{\delta_{11}}(\chi)\fatsemi \\ \qquad QC(t,\chi)\fatsemi \\ \qquad \mathbf{Fin}[\mathcal{V}](o,\chi)\fatsemi \\ \qquad \mathbf{Fin}[\mathcal{S}](s,\chi)\fatsemi \\ \quad \mathbf{rav} \end{array}$$

where:

for function $f{:}\mathbb{B}^n \to \mathbb{B}$ between registers, unitary transformation T_f between quregs inverts χ (pointwise) about 0 if f holds and otherwise leaves it unchanged

$$\begin{aligned} &T_f{:}q(\mathbb{B}^n) \to q(\mathbb{B}^n) \\ &(T_f\chi)(x) \mathrel{\widehat{=}} (-1)^{f(x)}\chi(x) \end{aligned}$$

observable $\mathcal{S} \mathrel{\widehat{=}} [S_0, S_1]$ measures the switch qubit:

$$S_0 \mathrel{\widehat{=}} \mathbb{C}\delta_0 \otimes (\mathbb{C}\delta_0 \oplus \mathbb{C}\delta_1), \quad S_1 \mathrel{\widehat{=}} \mathbb{C}\delta_1 \otimes (\mathbb{C}\delta_0 \oplus \mathbb{C}\delta_1)\,.$$

It can be easily shown that for $m{:}\mathbb{B}$ we have:

a) $wp.K(m,o,s).[o = m, s = 0] = \mathbf{\frac{1}{2}}$
b) $wp.K(1-m,o,s).[o = m] = \mathbf{0}$.

We reason directly on procedure K's body:

$$
\begin{array}{ll}
& K's\ body \\
= & \text{definition of } \mathbf{In} \\
& \chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}+\delta_{10}+\delta_{11}) \fatsemi \\
& \chi := T_{\delta_{11}}(\chi) \fatsemi \\
& QC(t,\chi) \fatsemi \\
& \mathbf{Fin}[\mathcal{V}](o,\chi) \fatsemi \\
& \mathbf{Fin}[\mathcal{S}](s,\chi) \\
= & \text{definition of } T_f \text{ and law A-2} \\
& \chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}+\delta_{10}-\delta_{11}) \fatsemi \\
& QC(t,\chi) \fatsemi \\
& \mathbf{Fin}[\mathcal{V}](o,\chi) \fatsemi \\
& \mathbf{Fin}[\mathcal{S}](s,\chi) \\
= & \text{definition of } QC \text{ and law A-1} \\
& \left(\begin{array}{l} \chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}+\delta_{10}-\delta_{11}) \fatsemi \\ \chi := CNOT(\chi) \end{array}\right) \lhd t \rhd \left(\begin{array}{l} \chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}+\delta_{10}-\delta_{11}) \fatsemi \\ \mathbf{skip} \end{array}\right) \fatsemi \\
& \mathbf{Fin}[\mathcal{V}](o,\chi) \fatsemi \\
& \mathbf{Fin}[\mathcal{S}](s,\chi) \\
= & \text{law A-2, definition of } CNOT \text{ and } \mathbf{skip} \text{ identity} \\
& (\chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}-\delta_{10}+\delta_{11})) \lhd t \rhd (\chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}+\delta_{10}-\delta_{11})) \fatsemi \\
& \mathbf{Fin}[\mathcal{V}](o,\chi) \fatsemi \\
& \mathbf{Fin}[\mathcal{S}](s,\chi) \\
= & \text{law S-2} \\
& \left(\begin{array}{l} \chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}-\delta_{10}+\delta_{11}) \fatsemi \\ \mathbf{Fin}[\mathcal{V}](o,\chi) \end{array}\right) \lhd t \rhd \left(\begin{array}{l} \chi := \tfrac{1}{2}(\delta_{00}+\delta_{01}+\delta_{10}-\delta_{11}) \fatsemi \\ \mathbf{Fin}[\mathcal{V}](o,\chi) \end{array}\right) \fatsemi \\
& \mathbf{Fin}[\mathcal{S}](s,\chi) \\
= & \text{introduce } v_0, v_1 \text{ and definition of } \mathbf{Fin} \\
& \oplus \left[\left(\begin{array}{l} \chi := v_1 \fatsemi \\ o,\chi := j, \frac{P_{\mathcal{V}_j}(\chi)}{\|P_{\mathcal{V}_j}(\chi)\|} \end{array}\right) @\langle \chi, P_{\mathcal{V}_j}(\chi)\rangle \mid j{:}\{0,1,2\}\right] \lhd t \rhd \\
& \quad \oplus \left[\left(\begin{array}{l} \chi := v_0 \fatsemi \\ o,\chi := k, \frac{P_{\mathcal{V}_k}(\chi)}{\|P_{\mathcal{V}_k}(\chi)\|} \end{array}\right) @\langle \chi, P_{\mathcal{V}_k}(\chi)\rangle \mid k{:}\{0,1,2\}\right] \fatsemi \\
& \mathbf{Fin}[\mathcal{S}](s,\chi) \\
= & \text{law A-2 and linear algebra}
\end{array}
$$

$$
\begin{array}{l}
(o,\chi := 1, v_1) \lhd t \rhd (o,\chi := 0, v_0)^{\circ}_{?} \\
\mathbf{Fin}[\mathcal{S}](s,\chi) \\
= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{law S-2 and definition of } \mathbf{Fin} \\
\oplus \left[\left(\begin{array}{l} o,\chi := 1, v_1^{\circ}_{?} \\ s,\chi := j, \frac{P_{\mathcal{S}_j}(\chi)}{\|P_{\mathcal{S}_j}(\chi)\|} \end{array} \right) @\langle \chi, P_{\mathcal{S}_j}(\chi)\rangle \mid j{:}\mathbb{B} \right] \lhd t \rhd \\
\quad \oplus \left[\left(\begin{array}{l} o,\chi := 0, v_0^{\circ}_{?} \\ s,\chi := k, \frac{P_{\mathcal{S}_k}(\chi)}{\|P_{\mathcal{S}_k}(\chi)\|} \end{array} \right) @\langle \chi, P_{\mathcal{S}_k}(\chi)\rangle \mid k{:}\mathbb{B} \right] \\
= \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{law A-2 and linear algebra} \\
[(o,s,\chi := 1,0, \frac{1}{\sqrt{2}}(\delta_{00}+\delta_{01}))\ {}_{\frac{1}{2}}\!\oplus (o,s,\chi := 1,1, \frac{1}{\sqrt{2}}(\delta_{11}-\delta_{10}))] \\
\quad \lhd t \rhd \\
[(o,s,\chi := 0,0, \frac{1}{\sqrt{2}}(\delta_{00}+\delta_{01}))\ {}_{\frac{1}{2}}\!\oplus (o,s,\chi := 0,1, \frac{1}{\sqrt{2}}(\delta_{10}-\delta_{11}))]
\end{array}
$$

By inspection we can easily see that conditions a) and b) are fully satisfied. Therefore, outcome $m{:}\mathbb{B}$ is an approximate counterfactual outcome of type m for protocol K, which thus exhibits two counterfactual outcomes (*i.e.* 0 and 1). We also note that via condition b) K enjoys a stronger property than condition $2'$ of definition 4.3: that reflects the fact that K can establish the correct answer of the problem with certainty. In some sense, our protocol lies between standard counterfactual protocols, defined by 4.2, and approximate counterfactual protocols.

Mitchison and Jozsa [6] also exhibited a protocol with two counterfactual outcomes which fully satisfies condition 1 and 2 of definition 4.2. However, their protocol is somehow less efficient than ours, as for that $p_0 = p_1 = 0.172$, thereby giving $p_0 + p_1 = 0.344$. Our protocol K clearly gives $p_0 + p_1 = 1$.

5 Mixed States and Density Matrices

The notion of mixed state arises from the observation that in fact we might not know the exact state of a quantum system. Because of imperfections in our apparatuses or unavoidable interactions with the environment, the best we can do is to express the state of the system as a probability distribution over quantum states. The *mixed state* $\rho \mathrel{\widehat{=}} \{(\psi_i, w_i) \mid 0 \leqslant i < n\}$ represents the state of a quantum system where the probability of the system to be in quantum state ψ_i is w_i (of course we must have $\sum_{0\leqslant i<n} w_i = 1$).

In vector-space Quantum Mechanics, states are represented by normalised vectors in some Hilbert space. In the same approach, mixed states are instead represented by a particular class of operators over Hilbert spaces, called *density matrices*. The density matrix corresponding to mixed state ρ is the operator $\hat{\rho}$ defined:

$$\hat{\rho} \mathrel{\widehat{=}} \sum_{0\leqslant i<n} w_i P_{\psi_i}.$$

Another approach is to specify no state at all and just saying that the system is in "some" state. In computing that equals to declare a variable: we define the state of an

object to be of some type (*i.e.* the range of possible values), though we do not know the actual state (*i.e.* the actual content of the variable) and an assignment is needed before using that variable. In qGCL we would write $\mathbf{var}\chi{:}q(\mathbb{B}) \bullet \mathbf{rav}$, which can be proved to be equal to the non-deterministic choice over all possible qubits. Again, we model our ignorance using non-determinism.

By laws D-2 and D-3 it is possible to prove that:

$$(\mathbf{var}\ \chi{:}q(\mathbb{B}) \bullet\ \mathbf{rav}) \sqsubseteq \left(\begin{array}{l} \mathbf{var}\ \chi{:}q(\mathbb{B})\bullet \\ \quad \oplus[\ \chi := \psi_i \mathbin{@} w_i \mid 0 \leqslant i < n\ ] \\ \mathbf{rav} \end{array}\right)$$

Therefore, the mixed state ρ above is simply represented by:

$$\oplus[\ \chi := \psi_i \mathbin{@} w_i \mid 0 \leqslant i < n\ ].$$

This is consistent with the density matrix formalism, as we show in the next sections. We observe that the state after finalisation of a qureg is indeed a mixed state.

5.1 Observation of mixed states

In this section we show that qGCL fully satisfies Quantum Mechanics' axioms for the observation of a mixed state.

We now consider the state reduction (observation) of a density matrix caused by an observable $\mathcal{O}$ with spectrum $\{0, \ldots, m-1\}$. We first recall that quantum state ψ is reduced to state ψ^j_{red} defined as:

$$\psi \to \psi^j_{red} \mathrel{\widehat{=}} \frac{P_j\psi}{\|P_j\psi\|} \qquad 0 \leqslant j < m.$$

The action of $\mathcal{O}$ on the density matrix $\hat{\rho} \mathrel{\widehat{=}} (\sum_{0\leqslant i<n} w_i P_{\psi_i})$ reduces it to $\hat{\rho}_{red}$:

$$\hat{\rho} \to \hat{\rho}_{red} \mathrel{\widehat{=}} \sum_{0\leqslant j<m} P^j \hat{\rho} P^j.$$

It is a matter of simple linear algebra to show that:

$$\hat{\rho}_{red} = \sum_{0\leqslant i<n,\, 0\leqslant j<m} w_i \langle \psi_i, P_j\psi_i\rangle P_{{\psi_i}^j_{red}} \tag{1}$$

that is, $\hat{\rho}$ is the mixed state composed by states ${\psi_i}^j_{red}$ with probabilities $w_i\langle\psi_i, P_j\psi_i\rangle$, where ${\psi_i}^j_{red}$ is the projection of ψ_i by means of projector P_j.

We now compute the observation of a mixed state in qGCL:

$\oplus[\chi := \psi_i \mathbin{@} \alpha_i \mid 0 \leqslant i < n] \fatsemi \mathbf{Fin}[\mathcal{O}](r, \chi)$

=

$\oplus[(\chi := \psi_i \fatsemi \mathbf{Fin}[\mathcal{O}](r, \chi)) \mathbin{@} \alpha_i \mid 0 \leqslant i < n]$

= Definition of **Fin**

$$\oplus \left[\begin{array}{l} \Big(\chi := \psi_i \fatsemi \oplus \Big[\Big(r, \chi := j, \frac{P_{S_j}(\chi)}{\|P_{S_j}(\chi)\|}\Big) \ @ \langle \chi, P_{S_j}(\chi)\rangle \ \mid 0 \leqslant j < m \Big]\Big) \ @ \\ \qquad \alpha_i \mid 0 \leqslant i < n \end{array} \right]$$

$=$ programming law

$$\oplus \left[\begin{array}{l} \oplus \Big[\Big(\chi := \psi_i \fatsemi r, \chi := j, \frac{P_{S_j}(\chi)}{\|P_{S_j}(\chi)\|}\Big) \ @ \ \langle \psi_i, P_{S_j}(\psi_i)\rangle \ \mid 0 \leqslant j < m \Big] @ \\ \qquad \alpha_i \mid 0 \leqslant i < n \end{array} \right]$$

$=$ programming law

$$\oplus \left[\begin{array}{l} \oplus \Big[\Big(r, \chi := j, \frac{P_{S_j}(\psi_i)}{\|P_{S_j}(\psi_i)\|}\Big) \ @ \ \langle \psi_i, P_{S_j}(\psi_i)\rangle \ \mid 0 \leqslant j < m \Big] @ \\ \qquad \alpha_i \mid 0 \leqslant i < n \end{array} \right]$$

$=$ programming law

$$\oplus \left[\Big(r, \chi := j, \frac{P_{S_j}(\psi_i)}{\|P_{S_j}(\psi_i)\|}\Big) \ @ \ \alpha_i \langle \psi_i, P_{S_j}(\psi_i)\rangle \ \mid \begin{array}{l} 0 \leqslant j < m \\ 0 \leqslant i < n \end{array} \right]$$

$=$ introduce A

A

From program A we see that the final state after observation is the mixed state:

$$\oplus \left[\Big(\chi := \frac{P_{S_j}(\psi_i)}{\|P_{S_j}(\psi_i)\|}\Big) \ @ \ \alpha_i \langle \psi_i, P_{S_j}(\psi_i)\rangle \ \mid \begin{array}{l} 0 \leqslant j < m \\ 0 \leqslant i < n \end{array} \right]$$

which is consistent with the density matrix formalism.

Quantum mechanics postulates that the probability of observing a particular outcome $k{:}\{0, \ldots, m-1\}$ is:

$$\text{Prob}(\mathcal{O} = k; \rho) \mathrel{\widehat{=}} \sum_{0 \leqslant i < n} w_i \langle \psi_i, P_k \psi_i \rangle$$

with the assumption that the classical probabilities w_i's and the quantum-mechanical probabilities are independent. We now calculate in qGCL the probability of observing a particular outcome $k{:}\{0, \ldots, m-1\}$. First we abstract program A by deleting the initialisation of χ:

$$A' \mathrel{\widehat{=}} \oplus \left[(r := j) \ @ \ \alpha_i \langle \psi_i, P_{S_j}(\psi_i)\rangle \ \mid \begin{array}{l} 0 \leqslant j < m \\ 0 \leqslant i < n \end{array} \right]$$

then we calculate $wp.A'.[r = k]$, the probability that the observation of $\mathcal{O}$ returns k:

$wp.A'.[r = k]$

$=$ semantics of probabilistic assignment

$\sum_{0 \leqslant j < m, 0 \leqslant i < n} \alpha_i \langle \psi_i, P_{S_j}(\psi_i)\rangle \cdot wp.(r := j).[r = k]$

$=$ semantics of assignment

$\sum_{0 \leqslant j < m, 0 \leqslant i < n} \alpha_i \langle \psi_i, P_{S_j}(\psi_i)\rangle \cdot [j = k]$

$=$ logic

$\sum_{0 \leqslant i < n} \alpha_i \langle \psi_i, P_{S_k}(\psi_i) \rangle = \text{Prob}(\mathcal{O} = k; \rho)$

5.2 Evolution of mixed states

The evolution of a density matrix $\hat{\rho} \mathrel{\widehat{=}} (\sum_{0 \leqslant i < n} w_i P_{\psi_i})$ under unitary operator U is defined to be:

$$\hat{\rho} \to \hat{\rho}' \mathrel{\widehat{=}} U \hat{\rho} U^\dagger$$

where $U^\dagger$ is the conjugate transpose of U. It can be easily shown that $\hat{\rho}'$ is the mixed state $\{(U\psi_i, w_i) \mid 0 \leqslant i < n\}$, that is:

$$\hat{\rho}' = \sum_{0 \leqslant i < n} w_i P_{U\psi_i}. \tag{2}$$

We now consider the unitary evolution of a mixed state using qGCL:

$\oplus[\, \chi := \psi_i \mathbin{@} \alpha_i \mid 0 \leqslant i < n \,] \fatsemi \chi := U(\chi)$

=

$\oplus[\, (\chi := \psi_i \fatsemi \chi := U(\chi)) \mathbin{@} \alpha_i \mid 0 \leqslant i < n \,]$

=

$\oplus[\, \chi := U(\psi_i) \mathbin{@} \alpha_i \mid 0 \leqslant i < n \,]$

We have obtained the mixed state expressed by equation 2.

6 Conclusions

We showed that a high-level, unfeasible construct, non-deterministic choice, may be used to formalise and reason about quantum computation, a theory which focuses only on implementable computing. In particular, we used a non-deterministic choice constructor coupled with the quantum programming language qGCL to formalise counterfactual computation and mixed-state quantum systems.

Counterfactual computation allows a seemingly paradoxical effect: to infer the result of a computation without running it. This remarkable fact can be achieved by means of peculiar properties of quantum mechanics. We used qGCL to formalise and reason about examples of counterfactual computation.

Mixed states and density matrices are a general formalism for dealing with probabilistic quantum systems, *i.e.* quantum systems for which a probabilistic distribution is the best description of the state of the system. We showed that qGCL can cope with mixed state systems by showing that density matrices and qGCL treat mixed states in an equivalent way. Moreover, no new construct is introduced into the language, thereby keeping it at the minimum.

7 Acknowledgements

This work was done when visiting the Oxford University Computing Laboratory, with the support of *Consiglio Nazionale delle Ricerche* (Italy). The author would like to thank Jeff Sanders for his support.

A Programming laws

We list a few algebraic laws which hold for pGCL programs; the semantic models adopted and proofs can be found in [4, 7].

Law (P-1). $P \;{}_1\!\oplus\; Q = P$

Law (P-2). $P \;{}_r\!\oplus\; Q = Q \;{}_{1-r}\!\oplus\; P$

Law (S-2). $(P \;{}_r\!\oplus\; Q) \fatsemi R = (P \fatsemi R) \;{}_r\!\oplus\; (Q \fatsemi R)$

Law (A-1). $(x := e) \fatsemi (P \;{}_r\!\oplus\; Q) = (x := e \fatsemi P) \;{}_{r[x\backslash e]}\!\oplus\; (x := e \fatsemi Q)$

In law A-1 $r[x\backslash e]$ denotes the expression obtained replacing all free occurrences of x in r by e. Since standard conditional is a particular case of probabilistic choice, laws S-2 and A-1 hold for that, too.

In the next two laws D is any data type; in law D-2 variable x must not appear in expression p.

Law (D-2). $\mathbf{var}\, x{:}D \bullet (P \;{}_p\!\oplus\; Q) = (\mathbf{var}\, x{:}D \bullet P) \;{}_p\!\oplus\; (\mathbf{var}\, x{:}D \bullet Q)$

Law (D-3). $\mathbf{var}\, x{:}D \sqsubseteq (\mathbf{var}\, x{:}D \bullet x := e)$

In law D-3 e is any expression of type D, meaning that initialising a variable will make a program more deterministic.

References

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

[2] Avshalom C. Elitzur and Lev Vaidman. Quantum mechanical interaction-free measurements. *Foundations of Physics*, 32(7):987–997, 1993.

[3] Chris J. Isham. *Lectures on quantum theory*. Imperial College Press, 1997.

[4] H. Jifeng, A. McIver, and K. Seidel. Probabilistic models for the guarded command language. *Science of Computer Programming*, 28:171–192, 1997.

[5] Richard Jozsa. Quantum effects in algorithms. *QCQC '98 Springer-Verlag LNCS*, 1509:103–112, 1999.

[6] Graeme Mitchison and Richard Jozsa. Counterfactual computation. *Proceedings of the Royal Society of London* A, 457:1175–1193, 2001.

[7] C. Morgan, A. McIver, and K. Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, May 1996.

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

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

[10] G.A. White, F.R. Mitchell, O. Nairz, and P. Kwiat. Interaction-free imaging. *Physical Review A*, 58:605–613, 1998.

[11] Paolo Zuliani. *Quantum Programming*. PhD thesis, Oxford University Computing Laboratory, 2001. Available at http://www.comlab.ox.ac.uk.