On Counterfactual Computation

Paolo Zuliani
Department of Computer Science
Princeton University
Princeton, NJ 08544, USA
pzuliani@cs.princeton.edu

Abstract

In this paper we pursue two targets. First, showing that counterfactual computation can be rigorously formalised as a quantum computation. Second, presenting a new counterfactual protocol which improve previous protocols. Counterfactual computation makes use of quantum mechanics' peculiarities to infer the outcome of a quantum computation without running that computation. In this paper, we first cast the definition of counterfactual protocol in the quantum programming language qGCL, thereby showing that counterfactual computation is an example of quantum computation. Next, we formalise in qGCL a probabilistic extension of counterfactual protocol for decision problems (whose result is either 0 or 1). If p_G^r denotes for protocol G the probability of obtaining result r "for free" (*i.e.* without running the quantum computer), then we show that for any probabilistic protocol $p_G^0 + p_G^1 \leq 1$ (as for non-probabilistic protocols). Finally, we present a probabilistic protocol K which satisfies $p_K^0 + p_K^1 = 1$, thus being optimal. Furthermore, the result is attained with a single insertion of the quantum computer, while it has been shown that a non-probabilistic protocol would obtain the result only in the limit (*i.e.* with an infinite number of insertions).

1 Introduction

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 [4, 5] 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 [1] 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 [8] 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 [4] and then further formalised by Mitchison and Jozsa [5].

The aim of this paper is twofold:

- to show how a programming language, qGCL [7], can be used for rigorously describing and reasoning about counterfactual computation, thereby embedding it in a more general framework, which in particular includes classical, probabilistic and quantum computation;

- to present a new example of counterfactual computation, along with its proof of correctness in qGCL. In particular, we consider a probabilistic extension of counterfactual computation.

We assume the reader has some knowledge of the basics of quantum computing.

2 Quantum programming

We give here a short presentation of the features of qGCL (a full introduction can be found in [7]). qGCL has been used to describe and reason about all known quantum algorithms and to derive the Deutsch-Jozsa algorithm from its specification. The problem of compiling qGCL code has been studied in [9].

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:

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

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 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 pGCL [6], which in turn extends Dijkstra's guarded-command language with a probabilistic choice constructor in order to address probabilism. The BNF syntax for qGCL is as follows:

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

where for brevity we omit the formal definitions of $\langle identifier\rangle$ and $\langle boolean\ expression\rangle$; $\langle unitary\ op\rangle(\chi)$ is just some mathematical expression involving qureg χ - such expression should of course denote a unitary operator.

Probabilistic 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: $\oplus[\ p_j\ @\ r_j \mid 0 \leqslant j < m\ ]$.

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. Evolution of qureg χ under unitary operator U is described via the assignment $\chi := U(\chi)$. The well-known *no-cloning* theorem 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. 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 $\chi{:}q(\mathbb{B}^n)$, 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 [7]

for an unabridged treatment); in our notation we write:

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

where P_{S_j} is the projector onto subspace S_j. We denote by Δ the observable spanned by the computational basis, also known as *diagonal* measurement.

Semantics for pGCL (and in turn for qGCL) can be given either relationally [3] or in terms of expectation transformers [6]. We shall use the latter, due to its simplicity in calculations. 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 *maximum guaranteed probability* that p holds after the execution of P.

In the Appendix we briefly review expectation-transformer semantics and some associated programming laws used in this paper.

3 Counterfactual computation

3.1 An example

We begin by giving the simple example of counterfactual computation introduced by Jozsa [4]. Suppose we are given a decision problem (*i.e.* a problem with a binary solution, "yes" or "no") and a quantum computer Q 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, but we assume that Q works reversibly, so that at the end of the computation the output will be placed in the output register as bit-wise XOR. We map "off/on" and "no/yes" to δ_0/δ_1, so for example δ_{00} means switch at "off" and result "no". The functioning of the computer is thus:

$$Q(\delta_{ij}) \mathrel{\widehat{=}} \delta_{i(j \oplus r)}$$

where $\oplus$ is bit-wise XOR and r is the result of the problem. We suppose that the computation takes at most a finite time T.

From Table 1 we see that for $r = 0$ the computer behaves as the identity transform, *i.e.* "do nothing", while for $r = 1$ it behaves as the well known CNOT transform over the switch and output qubits. By assuming that switch and output are encoded by qureg χ, we can readily model Q by program QC:

$$QC(\chi) \mathrel{\widehat{=}} (QC_0(\chi) \,\square\, QC_1(\chi))$$
$$QC_0(\chi) \mathrel{\widehat{=}} \mathbf{skip}\ , \quad QC_1(\chi) \mathrel{\widehat{=}} \chi := CNOT(\chi)$$

thus representing our ignorance about the inner working of the computer and the result of the decision problem. The goal is to start with the switch "off" and, after at least a time T, to determine which operation **skip** or CNOT has been performed, without setting the switch to "on".

Consider the following program N:

$$N \mathrel{\widehat{=}} [\ \chi := \delta_{00} \fatsemi \chi := H \otimes \mathbb{1}(\chi) \fatsemi QC(\chi) \fatsemi \chi := H \otimes \mathbb{1}(\chi)\ ]$$

$r=0$	$r=1$	
$\delta_{00} \to \delta_{00}$	$\delta_{00} \to \delta_{00}$	switch "off"
$\delta_{01} \to \delta_{01}$	$\delta_{01} \to \delta_{01}$	switch "off"
$\delta_{10} \to \delta_{10}$	$\delta_{10} \to \delta_{11}$	switch "on"
$\delta_{11} \to \delta_{11}$	$\delta_{11} \to \delta_{10}$	switch "on"

Table 1: Functioning of the quantum computer Q

where $\chi{:}q(\mathbb{B}^2)$, $\mathbb{1}$ is the identity transform over qubits, and H is the single-qubit Hadamard transform defined as:

$$H{:}q(\mathbb{B}) \to q(\mathbb{B})$$
$$H(\delta_x) \mathrel{\widehat{=}} \frac{1}{\sqrt{2}}(\delta_0 + (-1)^x \delta_1)\ .$$

We show that if the result of the problem is 1, then N can probabilistically infer it with probability $\frac{1}{4}$, without running the computer. We reason on program N:

N
$=$ law A-2, definition of H
$\chi := \frac{1}{\sqrt{2}}(\delta_{00} + \delta_{10}) \fatsemi QC(\chi) \fatsemi \chi := H \otimes \mathbb{1}(\chi)$
$=$ definition of QC and law A-3
$[\chi := \mathrm{CNOT}(\frac{1}{\sqrt{2}}(\delta_{00} + \delta_{10}))] \,\square\, [\chi := \frac{1}{\sqrt{2}}(\delta_{00} + \delta_{10}) \fatsemi \mathbf{skip}] \fatsemi \chi := H \otimes \mathbb{1}(\chi)$
$=$ definition of CNOT and **skip** identity
$[\chi := \frac{1}{\sqrt{2}}(\delta_{00} + \delta_{11})] \,\square\, [\chi := \frac{1}{\sqrt{2}}(\delta_{00} + \delta_{10})] \fatsemi \chi := H \otimes \mathbb{1}(\chi)$
$=$ laws S-3, A-3
$[\chi := H \otimes \mathbb{1}(\frac{1}{\sqrt{2}}(\delta_{00} + \delta_{11}))] \,\square\, [\chi := H \otimes \mathbb{1}(\frac{1}{\sqrt{2}}(\delta_{00} + \delta_{10}))]$
$=$ definition of H
$[\chi := \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{10} - \delta_{11})] \,\square\, [\chi := \delta_{00}]$

Suppose we now measure χ in the standard basis: if $r = 0$ (*i.e.* RHS of the nondeterministic choice) we always measure 00 and, because with probability $\frac{1}{4}$ we may measure 00 when $r = 1$, we cannot reliably infer the result of the computation. Suppose now $r = 1$: with probability $\frac{1}{4}$ we measure 10 and we know for sure that 1 is the result of the problem. Furthermore, the computer has not run, because if it had the output register (initially set to 0) should display 1. Therefore, if $r = 1$, with probability $\frac{1}{4}$ we learn the result of the problem without running the computer! The output 10 is thus a *counterfactual outcome*. Finally, with probability $\frac{1}{4}$ each we measure 01 and 11, thereby learning that $r = 1$, but the computer has run (the output register has changed).

3.2 Formal definition

We now code in qGCL the definition of *protocol* given by Mitchison and Jozsa [5]. For a datatype D we denote by $seq(D)$ the datatype of finite sequences of elements of type D.

Definition 3.1. A **protocol** G is a terminating program of the following type:

$$G \mathrel{\widehat{=}} \mathbf{var}\ \chi{:}q(\mathbb{B}^n), o{:}seq(\mathbb{B}^n), \psi{:}q(\mathbb{B}^p), s{:}\mathbb{B}^p \bullet\ body \fatsemi \mathbf{Fin}(\Delta, s, \psi)\ \mathbf{rav}$$

where:

- *body*, according to Mitchison and Jozsa [5], 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' QC, where the state of two selected qubits is swapped into the switch and output registers of the computer."*
- o returns the list of outcomes of the measurements of steps of type (b).

In order to formally describe what we mean by saying that the computer has not run, we procede as follows. After each insertion of QC we project the state over two orthogonal subspaces, the "off" and "on" subspaces, by entangling it with a qubit from ψ. That transformation is defined as:

$$E{:}q(\mathbb{B}^n) \to q(\mathbb{B}^{n+1})$$
$$E(v) \mathrel{\widehat{=}} (P_{\delta_0} \otimes \mathbb{1})v \otimes \delta_0 + (P_{\delta_1} \otimes \mathbb{1})v \otimes \delta_1.$$

We note that we need as many qubits as the number of insertions of the computer. The action of E is thus to create two coherent superpositions of the state vector, one 'living' in the off subspace, the other living in the on subspace. By measuring ψ at the end of the computation we can recognise a computation which has always taken place in the (desirable) off subspace: in that case ψ would reduce to δ_{0^p} (equivalently, s is the p-bit string 0, an "all-off" string). Our method is equivalent to the graphical *history* approach of Mitchison and Jozsa [5]: o and s collectively denote a history. The advantage of our approach is that it embeds all the necessary concepts in a single, general-purpose programming formalism.

We are now ready to formalise the definition of counterfactual computation in qGCL.

Definition 3.2. Given a protocol G, a sequence $m{:}seq(\mathbb{B}^n)$ is a *counterfactual outcome* of type $r{:}\mathbb{B}$ if the following two conditions hold:

$$(1) \quad \forall c{:}\mathbb{B}^m \bullet wp.G_r.[\ o = m, s = c\ ] = 0 \quad \text{iff} \quad c \neq 0$$
$$(2) \quad wp.G_{1-r}.[\ \textstyle\sum_{\chi_i:M} \chi_i = 0\ ] = 1$$

where M is the set of state vectors for which $o = m$, and G_r denotes protocol G when $QC = QC_r$, *i.e.* only operation QC_r is performed by the computer.

Condition (1) states that if QC_r is used in the protocol, then m is seen iff the (only) computation leading to it has always stayed in the off subspace. Therefore we can infer the result (r) of the problem for "free", since the switch of the quantum computer was always found at off. Condition (2) states that when QC_{1-r} is used, then all the computations leading to m annihilate themselves, by means of the so-called *destructive interference*. That implies $wp.G_{1-r}.[o = m] = 0$, *i.e.* m never occurs (the converse needs not to hold) and we are sure that the result of the problem is r.

3.3 Limits on counterfactual computation

In this section we show how qGCL can be effective in reasoning about counterfactual computation. We begin by noting that a sequence of measurement outcomes cannot be a counterfactual outcome of type 0 and 1 at the same time. That is, if we define $CF_G(t)$ as the set of counterfactual outcomes of type t for protocol G, then the sets $CF_G(0)$ and $CF_G(1)$ are disjoint.

Theorem 3.1. *For any protocol G we have that $CF_G(0) \cap CF_G(1) = \emptyset$.*

Proof. A contradiction arises between condition (2) for $m{:}CF_G(0)$ and condition (1) for $m{:}CF_G(1)$. □

Let p^0_G and p^1_G denote the probability of learning "for free" outcomes 0 and 1 respectively, in a protocol G; in our notation we write:

$$\forall r{:}\mathbb{B} \bullet p^r_G \mathrel{\widehat{=}} \sum_{m:seq(\mathbb{B}^n)} wp.G_r.[\ o = m, s = 0\ ].$$

The sum is well defined as G always terminates. One may wish to design a protocol for which both p^0_G and p^1_G are greater than 0 and perhaps $p^0_G + p^1_G > 1$. Mitchison and Jozsa [5] stopped further conjectures, showing that for any protocol the two probabilities sum to at most 1. We replay here their result in our formalism.

Theorem 3.2 (Mitchison and Jozsa)**.** *For any protocol G we have:*

$$p^0_G + p^1_G \leqslant 1\,.$$

Proof. We reason:

$p^0_G + p^1_G$

$=$ definition of p^r_G

$\sum_{t:\mathbb{B}} \sum_{m:seq(\mathbb{B}^n)} wp.G_r.[\ o = m, s = 0\ ]$

$=$ logic and definition of $CF_G(\cdot)$

$\sum_{m:CF_G(0)} wp.G_0.[\ o = m, s = 0\ ] + \sum_{l:CF_G(1)} wp.G_1.[\ o = l, s = 0\ ]$

$=$ in the off subspace $wp.G_0 = wp.G_1$

$\sum_{m:CF_G(0)} wp.G_0.[\ o = m, s = 0\ ] + \sum_{l:CF_G(1)} wp.G_0.[\ o = l, s = 0\ ]$

$=$ Theorem 3.1

$\sum_{m:CF_G(0)} wp.G_0.[\ o = m, s = 0\ ]$

$\leqslant$ logic

$\sum_{m:seq(\mathbb{B}^n)} wp.G_0.[\ o = m, s = 0\ ]$

$=$ wp-semantics and 1 top element

$wp.G_0.[\ s = 0\ ] \leqslant 1$

□

Mitchison and Jozsa also derived complexity constraints between p^r_G and the number of times the computer is inserted. They showed that for any protocol G such that $p^0_G + p^1_G = 1 - \epsilon$, the number of insertions of the computer must necessarily tend to infinity as ϵ tends to 0.

4 Probabilistic extension

In this section we formalise in qGCL a probabilistic extension of counterfactual computation proposed by Mitchison and Jozsa [5]. In particular, we consider the case in which we allow a relaxation of condition (1) of definition 3.2, while (2) is carried over intact.

Definition 4.1. Given a protocol G, a sequence $m{:}seq(\mathbb{B}^n)$ is an *approximate counterfactual outcome* of type $r{:}\mathbb{B}$ if the following two conditions hold:

$$(1') \quad wp.G_r.[\ o = m, s \neq 0\ ] < \epsilon$$
$$(2') \quad wp.G_{1-r}.[\ \textstyle\sum_{\chi_i:M} \chi_i = 0\ ] = 1$$

where ϵ is a small real in the $(0, 1]$ interval.

Condition $(1')$ implies that, when using QC_r in the protocol, m may arise from a computation which does not lie in the off subspace, *i.e.* the computer has run throughtout the protocol. However, the probability of such an event is bounded by the small number ϵ. It is of course expected that the computation of the off subspace leading to m has probability greater than ϵ. Together, the two conditions ensure that when we see m the answer to the decision problem is r (because of $(2')$), and with high probability the computer has not run.

Probabilistic protocols (*i.e.* protocols which feature approximate counterfactual outcomes) face some of the limitations of non-probabilistic ones.

Theorem 4.1. *For any probabilistic protocol G we must have $p^0_G + p^1_G \leq 1$.*

Proof. The proof of Theorem 3.2 still applies, as condition $(2')$ is what ensures that an outcome can be only be measured under either G_r or G_{1-r}. □

However, probabilistic protocols do not require an infinite number of insertions of the computer in order to reach the limit 1. The next example shows that a single insertion suffices.

4.1 Probabilistic protocol

We describe a probabilistic protocol which can infer the answer to the decision problem with certainty, but requires a run of the quantum computer with probability $\frac{1}{2}$. 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:

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

We now measure χ using observable $\{\mathbb{C}v_0, \mathbb{C}v_1, (\mathbb{C}v_0 \oplus \mathbb{C}v_1)^\perp\}$ where $\mathbb{C}v_i$ is the unidimensional spaces spanned by v_i and $\perp$ denotes orthogonality. Since $v_0 \perp v_1$, we are thus able to learn r with certainty and we do 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 $\frac{1}{2}$.

In qGCL the protocol is coded as follows:

$$\begin{aligned} K \mathrel{\widehat{=}} \textbf{var}\ & \chi{:}q(\mathbb{B}^2), \psi{:}q(\mathbb{B}), o{:}\{0,1,2\}, s{:}\mathbb{B} \bullet \\ & \mathbf{In}(\chi)\,; \\ & \chi := T_{\delta_{11}}(\chi)\,; \\ & QC(\chi)\,; \\ & \chi, \psi := E(\chi)\,; \\ & \mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi)\,; \\ & \mathbf{Fin}(\mathcal{S}, s, \chi \otimes \psi) \\ & \textbf{rav} \end{aligned}$$

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

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

- observable $\mathcal{V} \mathrel{\widehat{=}} \{V_0, V_1, V_2\}$ has $V_i \mathrel{\widehat{=}} \mathbb{C}E(v_i)$ for $i{:}\mathbb{B}$, and $V_2 \mathrel{\widehat{=}} (V_0 \oplus V_1)^\perp$
- observable $\mathcal{S} \mathrel{\widehat{=}} \{S_0, S_1\}$ has $S_i \mathrel{\widehat{=}} \mathbb{C}^2 \otimes \mathbb{C}\delta_i$

Proposition 4.2. *Outcome $m{:}\mathbb{B}$ is an approximate counterfactual outcome of type m for protocol K. In particular, we have:*

$$\text{(a)} \quad wp.K_m.[o = m, s \neq 0] = \tfrac{1}{2}$$
$$\text{(b)} \quad wp.K_{1-m}.[\ \textstyle\sum_{\chi_i:M} \chi_i = 0\ ] = 1$$

Proof. We reason directly on K:

K

$=$ definition of **In**

$\chi := \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{10} + \delta_{11})$⨟
$\chi := T_{\delta_{11}}(\chi)$⨟
$QC(\chi)$⨟
$\chi, \psi := E(\chi)$⨟
$\mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi)$⨟
$\mathbf{Fin}(\mathcal{S}, s, \chi \otimes \psi)$

$=$ definition of T_f and law A-2

$\chi := v_0$⨟
$QC(\chi)$⨟
$\chi, \psi := E(\chi)$⨟
$\mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi)$⨟
$\mathbf{Fin}(\mathcal{S}, s, \chi \otimes \psi)$

$=$ definition of QC and laws A-3, S-3

$$\left(\begin{array}{l}\chi := v_0 \fatsemi \\ \chi := \mathrm{CNOT}(\chi) \fatsemi \chi, \psi := E(\chi)\end{array}\right) \Box \left(\begin{array}{l}\chi := v_0 \fatsemi \\ \mathbf{skip} \fatsemi \chi, \psi := E(\chi)\end{array}\right) \fatsemi$$
$\mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi)$⨟
$\mathbf{Fin}(\mathcal{S}, s, \chi \otimes \psi)$

$=$ law A-2, definition of CNOT and **skip** identity

$(\chi, \psi := E(v_1)) \Box (\chi, \psi := E(v_0))$⨟
$\mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi)$⨟
$\mathbf{Fin}(\mathcal{S}, s, \chi \otimes \psi)$

$=$ law S-2

$$\left(\begin{array}{l}\chi, \psi := E(v_1) \fatsemi \\ \mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi) \fatsemi\end{array}\right) \Box \left(\begin{array}{l}\chi, \psi := E(v_0) \fatsemi \\ \mathbf{Fin}(\mathcal{V}, o, \chi \otimes \psi) \fatsemi\end{array}\right) \fatsemi$$
$\mathbf{Fin}(\mathcal{S}, s, \chi \otimes \psi)$

$=$ definition of **Fin**

$$\oplus \left[\left(\begin{array}{l} \chi, \psi := E(v_1)\fatsemi \\ o, \chi, \psi := j, \frac{P_{V_j}(\chi\otimes\psi)}{\|P_{V_j}(\chi\otimes\psi)\|} \end{array} \right) @\langle \chi\otimes\psi, P_{V_j}(\chi\otimes\psi)\rangle \mid j{:}\{0,1,2\} \right] \square$$
$$\oplus \left[\left(\begin{array}{l} \chi, \psi := E(v_0)\fatsemi \\ o, \chi, \psi := k, \frac{P_{V_k}(\chi\otimes\psi)}{\|P_{V_k}(\chi\otimes\psi)\|} \end{array} \right) @\langle \chi\otimes\psi, P_{V_k}(\chi\otimes\psi)\rangle \mid k{:}\{0,1,2\} \right] \fatsemi$$
$$\mathbf{Fin}(\mathcal{S}, s, \chi\otimes\psi)$$

$=$ law A-2 and linear algebra

$$(o, \chi, \psi := 1, E(v_1)) \,\square\, (o, \chi, \psi := 0, E(v_0))\fatsemi$$
$$\mathbf{Fin}(\mathcal{S}, s, \chi\otimes\psi)$$

$=$ law S-2 and definition of **Fin** and E

$$\oplus \left[\left(\begin{array}{l} o, \chi, \psi := 1, \frac{1}{2}(\delta_{000}+\delta_{010}+\delta_{111}-\delta_{101})\fatsemi \\ s, \chi, \psi := j, \frac{P_{S_j}(\chi\otimes\psi)}{\|P_{S_j}(\chi\otimes\psi)\|} \end{array} \right) @\langle \chi\otimes\psi, P_{S_j}(\chi\otimes\psi)\rangle \mid j{:}\mathbb{B} \right] \square$$
$$\oplus \left[\left(\begin{array}{l} o, \chi, \psi := 0, \frac{1}{2}(\delta_{000}+\delta_{010}+\delta_{101}-\delta_{111})\fatsemi \\ s, \chi, \psi := k, \frac{P_{S_k}(\chi\otimes\psi)}{\|P_{S_k}(\chi\otimes\psi)\|} \end{array} \right) @\langle \chi\otimes\psi, P_{S_k}(\chi\otimes\psi)\rangle \mid k{:}\mathbb{B} \right]$$

$=$ law A-2 and linear algebra

$$[(o, s, \chi, \psi := 1, 0, \tfrac{1}{\sqrt{2}}(\delta_{000}+\delta_{010})) \,{}_{\frac{1}{2}}\!\oplus\, (o, s, \chi, \psi := 1, 1, \tfrac{1}{\sqrt{2}}(\delta_{111}-\delta_{101}))] \,\square$$
$$[(o, s, \chi, \psi := 0, 0, \tfrac{1}{\sqrt{2}}(\delta_{000}+\delta_{010})) \,{}_{\frac{1}{2}}\!\oplus\, (o, s, \chi, \psi := 0, 1, \tfrac{1}{\sqrt{2}}(\delta_{101}-\delta_{111}))]$$

By inspection we can easily see that claims (a) and (b) are fully satisfied (K_1 is the LHS of the nondeterministic choice, while K_0 is the RHS). □

We observe that protocol K thus exhibits two counterfactual outcomes: 0 and 1. Mitchison and Jozsa also exhibited a protocol with two counterfactual outcomes which fully satisfies definition 3.2, and for which $p_0 = p_1 = 0.172$, thereby giving $p_0 + p_1 = 0.344$. However, the main advantage of protocol K is that it reaches the probability bound 1 with a single insertion of the computer, while a standard protocol reaches 1 only in the limit.

Proposition 4.3. *Protocol K is optimal, that is $p^0_K + p^1_K = 1$.*

Proof. It follows by Proposition 4.2 and Theorem 4.1. □

We note that K is also optimal with respect to the number of insertions of the computer, since at least one insertion is required by any protocol. In Table 2 we provide a summary of the features of standard and probabilistic counterfactual computation.

We conclude by providing another example of probabilistic protocol, which stems from protocol K. Of course it cannot improve protocol K, but it provides another example of protocol (and of quantum computation, in the end). We first recall the "inversion about the mean" transform introduced firstly by Grover in his search algorithm [2]. It is the unitary transform $M(\cdot)$ defined as:

$$M{:}q(\mathbb{B}^n) \to q(\mathbb{B}^n)$$
$$(M\chi)(x) \mathrel{\widehat{=}} 2\left(\tfrac{1}{2^n}\textstyle\sum_{y:\mathbb{B}^n}\chi(y)\right) - \chi(x).$$

Together with transform T_f, they form Grover's algorithm core iteration.

Our new protocol works like protocol K, except that before the last (diagonal) finalisation we first unitarily transform the switch and the output register in such a way "to drive" the

	G standard protocol	G probabilistic protocol
$p^0_G + p^1_G$	≤ 1 (Mitchison and Jozsa [5])	≤ 1 (Theorem 4.1; protocol K attains exactly 1)
Number N of insertions of the computer	$N \to \infty$ when $p^0_G + p^1_G \to 1$ (Mitchison and Jozsa [5])	N can be as low as 1 (our protocol K)

Table 2: Standard vs. probabilistic protocols

switch to the off state. In qGCL it is coded as follows, where for simplicity we did not include the qubit ψ for distingushing between the off and on subspaces:

$$
\begin{aligned}
O \mathrel{\widehat{=}} \mathbf{var}\ & \chi{:}q(\mathbb{B}^2),\ o, s{:}\mathbb{B}^2 \bullet \\
& \mathbf{In}(\chi)\fatsemi \\
& \chi := T_{\delta_{11}}(\chi)\fatsemi \\
& QC(\chi)\fatsemi \\
& \mathbf{Fin}(\mathcal{W}, o, \chi)\fatsemi \\
& (\chi := T_{\delta_{10}}(\chi)) \lhd o \rhd (\chi := T_{\delta_{11}}(\chi))\fatsemi \\
& \chi := T_{\delta_{00}}(\chi)\fatsemi \\
& \chi := M(\chi)\fatsemi \\
& \mathbf{Fin}(\Delta, s, \chi) \\
& \mathbf{rav}
\end{aligned}
$$

where $\mathcal{W}$ is the observable $\{\mathbb{C}v_0, \mathbb{C}v_1, (\mathbb{C}v_0 \oplus \mathbb{C}v_1)^\perp\}$. We argue that outcome $m{:}\mathbb{B}$ is an approximate counterfactual outcome of type m, and that $p^0_O = p^1_O = \frac{1}{2}$. We give here an informal proof of our claim.

We know from the functioning of K that, assuming QC_r has been executed, the state after the measurement of $\mathcal{W}$ is:

$$\chi = \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{1r} - \delta_{1\neg r}).$$

The following conditional thus transforms the state to:

$$\chi = \frac{1}{2}(\delta_{00} + \delta_{01} + \delta_{10} + \delta_{11}).$$

The next instruction flips the sign of the amplitude for basis state δ_{00}:

$$\chi = \frac{1}{2}(-\delta_{00} + \delta_{01} + \delta_{10} + \delta_{11}).$$

The execution of M on χ gives $\chi = \delta_{00}$. This means that we always end up in the initial "off" state δ_{00}. Furthermore, we recorded the outcome of the measurement of $\mathcal{W}$, which identifies with certainty the result of the decision problem. We note that although O always ends in δ_{00}, there are computations which return the outcome m, but do not belong to the off subspace. Those computations have been annihilated by the functioning of the algorithm (by embedding state χ via transformation E one can easily calculate them). Therefore, we cannot claim that the result is always for free, and that motivates condition (1) of definition 3.1 requiring that only the computation in the off subspace must have non-zero probability.

5 Conclusions

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. In this paper we showed how it is possible to formalise counterfactual computation in a framework provided by the quantum programming language qGCL, thereby showing that counterfactual computation is just an example of quantum computation. The main benefit of this approach is the possibility of exploiting the well-established body of programming laws which comes with qGCL. Next, we proposed a probabilistic extension of the original definition of counterfactual computation and we casted it into qGCL. We showed that probabilistic counterfactual protocols share some of the limitations of non-probabilistic protocols. In particular, for any probabilistic protocol G we must have $p^0_G + p^1_G \leq 1$. We presented a probabilistic protocol K for which $p^0_K + p^1_K = 1$, thus being optimal. Furthermore, our protocol K requires a single insertion of the quantum computer, while any non-probabilistic protocol would reach the upper bound 1 only with an infinite number of insertions. Therefore, the probabilistic relaxation of counterfactual computation helps only with respect to the complexity of the protocol.

6 Acknowledgements

This work was mainly carried out while visiting the Oxford University Computing Laboratory, with the support of *Consiglio Nazionale delle Ricerche* (Italy). The author is currently supported by a *Marie Curie Outgoing International Fellowship* within the 6^{th} Framework Programme of the European Commission. The author would like to thank Jeff Sanders for his support.

References

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

[2] L. K. Grover. A fast quantum mechanical algorithm for database search. In *STOC '96: Proceedings of the 28th Annual Symposium on the Theory of Computing*, pages 212–219, 1996.

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

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

[5] G. Mitchison and R. Jozsa. Counterfactual computation. *Proceedings of the Royal Society of London* A, 457:1175–1193, 2001.

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

[7] J. W. Sanders and P. Zuliani. Quantum programming. In *MPC '00: Mathematics of Program Construction, Springer LNCS*, volume 1837, pages 80–99, 2000.

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

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

A pGCL semantics

In this section we briefly review expectation-transformer semantics for pGCL [6], qGCL's parent language. Furthermore, we list some associated programming laws used in this work.

Definition. The *state* x of a program P is the array of global variables used during the computation. That is

$$x \mathrel{\widehat{=}} (v_1, \ldots, v_n) : T_1 \times T_2 \times \ldots \times T_n.$$

The Cartesian product $T_1 \times T_2 \times \ldots \times T_n$ of all the data types used is called the *state space* of program P.

An *expectation* is a $[0,1]$-valued function on a state space X and may be thought of as a "probabilistic predicate". The set $\mathcal{Q}$ of all expectations is defined:

$$\mathcal{Q} \mathrel{\widehat{=}} X \to [0,1].$$

Expectations can be ordered using the standard pointwise functional ordering for which we shall use the symbol $\Rrightarrow$, and $p \Rrightarrow q$ means "p everywhere no more than q". Standard predicates are easily embedded in $\mathcal{Q}$ by identifying *true* with expectation $\mathbf{1}$ and *false* with $\mathbf{0}$. For a standard predicate p we shall write $[p]$ for its embedding.

The pair $(\mathcal{Q}, \Rrightarrow)$ forms a complete lattice, with greatest element the constant expectation $\mathbf{1}$ and least element the constant expectation $\mathbf{0}$. For $i, j{:}\mathcal{Q}$ we shall write $i \equiv j$ iff $i \Rrightarrow j$ and $j \Rrightarrow i$ (or $i \Lleftarrow j$). The set $\mathcal{J}$ of all expectation transformers is defined:

$$\mathcal{J} \mathrel{\widehat{=}} \mathcal{Q} \to \mathcal{Q}.$$

Not every expectation transformer corresponds to a computation: only the *sublinear* ones do [6].

The following table gives the expectation-transformer semantics for some pGCL commands (we shall retain the wp prefix of predicate-transformer calculus for convenience):

$wp.\mathbf{abort}.q$	$\mathrel{\widehat{=}}$	$\mathbf{0}$
$wp.\mathbf{skip}.q$	$\mathrel{\widehat{=}}$	q
$wp.(x := E).q$	$\mathrel{\widehat{=}}$	$q[x\backslash E]$
$wp.(R \fatsemi S).q$	$\mathrel{\widehat{=}}$	$wp.R.(wp.S.q)$
$wp.(R \lhd cond \rhd S).q$	$\mathrel{\widehat{=}}$	$[cond] * (wp.R.q) + [\neg cond] * (wp.S.q)$
$wp.(R \,\Box\, S).q$	$\mathrel{\widehat{=}}$	$(wp.R.q) \sqcap (wp.S.q)$
$wp.(R \,{}_p\!\oplus S).q$	$\mathrel{\widehat{=}}$	$p * (wp.R.q) + (1-p) * (wp.S.q)$

where $q{:}\mathcal{Q}$, $x{:}X$, $p{:}[0,1]$ and $cond$ is an arbitrary predicate over state space; $q[x\backslash E]$ denotes the expectation obtained after replacing all free occurrences of x in q by expression E; $\sqcap$ denotes the greatest lower bound. Recursion is treated in general using the existence of fixed points in $\mathcal{J}$.

We now list a few algebraic programming laws which we used in the paper; the semantic models adopted and proofs can be found in [3, 6]. In the following laws we use the term e to indicate an expression whose type is determined by the context.

Law (Id "**skip** identity"). $(P \fatsemi \mathbf{skip}) = (\mathbf{skip} \fatsemi P) = P$

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 (S-3). $(P \,\Box\, Q) \fatsemi R = (P \fatsemi R) \,\Box\, (Q \fatsemi R)$

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

Law (A-2). $(x := e \fatsemi x := f) = (x := f[e\backslash x])$

Law (A-3). $(x := e \fatsemi P \,\Box\, Q) = (x := e \fatsemi P) \,\Box\, (x := e \fatsemi Q)$

Since standard conditional is a particular case of probabilistic choice, laws S-2 and A-1 hold for that, too.