Quantum Science and Technology

PAPER

Weakly measured while loops: peeking at quantum states

Pablo Andrés-Martínez* and Chris Heunen
University of Edinburgh, United Kingdom
* Author to whom any correspondence should be addressed.
E-mail: p.andres-martinez@ed.ac.uk and chris.heunen@ed.ac.uk

Keywords: programming primitive, while loop, weak measurement

OPEN ACCESS

RECEIVED
4 October 2021

REVISED
19 December 2021

ACCEPTED FOR PUBLICATION
4 January 2022

PUBLISHED
11 February 2022

Abstract

A *while* loop tests a termination condition on every iteration. On a quantum computer, such measurements perturb the evolution of the algorithm. We define a while loop primitive using weak measurements, offering a trade-off between the perturbation caused and the amount of information gained per iteration. This trade-off is adjusted with a parameter set by the programmer. We provide sufficient conditions that let us determine, with arbitrarily high probability, a worst-case estimate of the number of iterations the loop will run for. As an example, we solve Grover's search problem using a while loop and prove the quadratic quantum speed-up is maintained.

1. Introduction

In quantum computer science, *while* loops do not hold centre stage. They are often used in a trivial way, for instance, when we say that a quantum algorithm is repeated until it succeeds. On the other hand, there are examples, such as Grover's algorithm, where a quantum operation is applied a fixed number of times, as in a *for* loop. But, if a subroutine's behavior is too costly (or even impossible) to predict, we require the use of a while loop to be sure it runs until the termination condition is met.

Measuring a quantum state perturbs it, so there is no way to test a while loop's termination condition without affecting the quantum computation. In this work, we propose a while loop primitive, which we call κ-while loop, where the detrimental effect of the measurement can be limited. The key component of a κ-while loop is the use of *weak measurements* to test the termination condition of the loop. Loosely speaking, tuning the parameter κ—known as the *measurement strength*—can reduce the amount of collapse, at the cost of gaining less information from each measurement. Under certain conditions, we may use κ-while loops to 'monitor' a property of the evolving system throughout the iterations, stopping only when success is certain.

We introduce two properties, *active guarantee* and *robustness*, that let us analyze the performance of κ-while loops. As proof of concept, we show that Grover's iteration satisfies these properties, thus allowing us to implement Grover's algorithm using a κ-while loop. We do not fix the number of iterations in advance, but knowledge of the database's size is still necessary to determine the value of κ.

The structure of the paper is as follows. In section 2 we discuss related work and define the notion of κ-measurements. Section 3 presents the main contribution: the definition of κ-while loops and the properties of *active guarantee* and *robustness*. In section 4, we apply our framework to Grover's search problem, providing an algorithm similar to the one previously proposed by Mizel [18]. We conclude with some discussion in section 5 and suggest possible applications of κ-while loops beyond quantum search.

2. Background

In this section we discuss the two approaches to control flow in quantum programming languages and introduce the concept of κ-measurement and its connection to weak measurements.

2.1. While loops in quantum computing

There are multiple examples of quantum programming languages in the literature (see reference [10] for a survey). We can classify these languages into two different paradigms: quantum control flow and classical control flow. In this section we summarise the most relevant aspects of each paradigm. For a more in-depth discussion, see reference [22].

Quantum control flow. QML [1] was the first quantum programming language exhibiting quantum control flow. In this programming language, an if-then-else statement

$$\text{if } b \text{ then } U \text{ else } W \text{ end}$$

corresponds to constructing a quantumly controlled gate. If the control qubit b is $|1\rangle_b$ then gate U is applied, and if it is $|0\rangle_b$ then gate W is applied. If b is in a superposition, both branches of the control flow coexist in superposition.

There are multiple problems with QML's original approach [3], which later works attempted to amend [3, 19, 24]. These works have tackled important issues of the language's semantics, but a persistent problem is the impossibility of implementing unbounded loops (and unbounded recursion). Each time the control flow forks, a fresh auxiliary qubit is required to act as the control condition, and this qubit gets entangled to the state of the computation[1]. Therefore, while loops—where the control flow forks on each iteration—would have their maximum number of iterations bounded by the size of the quantum memory; a similar problem would arise with unbounded recursion. Moreover, as pointed out by Linden and Popescu [16], a program with quantum control flow cannot support an unbounded loop where different terms of the superposition exit the loop at different times. More specifically, if the evolution is required to be unitary, terms that exited the loop at different times cannot interfere with each other, thus the control flow cannot be said to be fully quantum.

Classical control flow. These quantum programming languages follow the slogan 'quantum data, classical control' [20]. Consider a quantum register q that holds states from a Hilbert space $\mathcal{H}$ and let $\{M_0, M_1\}$ describe a projective measurement on $\mathcal{H}$. The statement

$$\text{while } M[q] = 0 \text{ do } U \text{ end}$$

corresponds to applying, on each iteration, unitary U on the state of register q followed by the projective measurement $\{M_0, M_1\}$ [23]. If the outcome of the measurement is 0, the loop keeps iterating; otherwise it halts and the next statement of the program is executed. There is no superposition of commands, as the loop either halts or continues iterating, hence the term classical control flow. This paradigm is simple to realise on physical devices—it is just a quantum chip controlled by a classical computer—and its semantics are better understood. The drawback of this approach is that applying a measurement on each iteration perturbs the state being observed, and thus alters the computation itself.

In this work we propose weakly measured while loops, an example of classical control flow where the effect of the collapse is kept below a threshold. Under certain conditions (see section 3.2), we prove that the performance of the quantum evolution is unaffected.

2.2. Weak measurements

Roughly speaking, a weak measurement is 'a measurement which gives very little information about the system on average, but also disturbs the state very little' [7]. The field of quantum feedback control uses weak measurements (often, continuous measurements) to monitor a state. The stream of measurement outcomes is used to control the strength of a Hamiltonian that corrects the system; see reference [26] for a survey. Our approach is inspired on these ideas, contextualised for their application to algorithm design. We restrict ourselves to the discrete-time setting and define a particular kind of parametrised measurement, the κ-measurement, that behaves as a weak measurement when κ is small.

Let $\mathcal{H}$ be a Hilbert space; we wish to apply a measurement to test whether a state $\psi \in \mathcal{H}$ satisfies certain property. Let B be an orthonormal basis of $\mathcal{H}$ such that the outcome of the measurement on each $b \in B$ is deterministic; we can characterise the property we wish to measure via a function $Q : B \to \{0, 1\}$ which determines whether $b \in B$ satisfies it $Q(b) = 1$ or not $Q(b) = 0$. We refer to Q as the *predicate* to be tested by the measurement: in computer science, a predicate is a function assigning to each element of a set a truth

[1] In certain situations, the state of the auxiliary qubit can be *uncomputed*, i.e. returned to its initial value by applying the inverse of part of the computation. However, this is only possible when none of the qubits used to compute the value of b are affected by U or W. Thus, auxiliary control qubits cannot be reused in general.

value. Assume the existence of a unitary $O_Q : \mathcal{H} \otimes \mathbb{C}^2 \to \mathcal{H} \otimes \mathbb{C}^2$ acting as the *oracle* of predicate Q:

$$O_Q|x, p\rangle = |x, p \oplus Q(x)\rangle. \tag{1}$$

Fix a value of parameter $\kappa \in [0, 1]$ and let $\mathcal{P} = span\{\bot, \top\}$ be an auxiliary space known as the *probe*. We define a unitary $E_{\kappa,Q}$ that rotates the state of the probe an amount according to κ only if the state in $\mathcal{H}$ satisfies Q:

$$E_{\kappa,Q} = (O_Q^\dagger \otimes I_\mathcal{P})\,(I_\mathcal{H} \otimes \Lambda(R_\kappa))\,(O_Q \otimes I_\mathcal{P}) \tag{2}$$

$$\Lambda(R_\kappa) = |0\rangle\langle 0| \otimes I_\mathcal{P} + |1\rangle\langle 1| \otimes R_\kappa \tag{3}$$

$$R_\kappa = \begin{pmatrix} \sqrt{1-\kappa} & \sqrt{\kappa} \\ \sqrt{\kappa} & -\sqrt{1-\kappa} \end{pmatrix}. \tag{4}$$

Notice that the auxiliary qubit where the oracle O_Q writes the result is restored to its initial state by $O_Q^\dagger$, so it may be reused. We initialise it to $|0\rangle$ and omit it in further discussions.

Remark 2.1. In (2) the only purpose of R_κ is to send $|\bot\rangle$ to $\alpha|\bot\rangle + \beta|\top\rangle$ so that $|\beta|^2 = \kappa$. The definition of R_κ provided is just one of infinitely many possible choices. In general, $Z(\theta)R_\kappa Z(\theta')$ for any θ and θ' is a valid choice for this unitary, where $Z(\theta)$ is a Z-rotation of angle θ.

Definition 2.2. A κ-measurement of predicate Q may be applied on any density matrix ρ by the following procedure:

- Apply the unitary $E_{\kappa,Q} : \mathcal{H} \otimes \mathcal{P} \to \mathcal{H} \otimes \mathcal{P}$ on the state $\rho \otimes |\bot\rangle\langle\bot|$,
- Apply a measurement on the probe space $\mathcal{P}$ determined by projectors

$$M_\mathcal{P} = \{I_\mathcal{H} \otimes |\bot\rangle\langle\bot|, I_\mathcal{H} \otimes |\top\rangle\langle\top|\}. \tag{5}$$

In essence, the state of the probe is entangled with the result of the oracle so that, when the probe is measured, we may learn something about the state in $\mathcal{H}$. The degree of the entanglement is determined by the parameter κ: for $\kappa = 1$ the entanglement is maximal, whereas for $\kappa = 0$ there is no entanglement at all.

Remark 2.3. After applying a measurement, we may describe the resulting state as a mixed state on the space $\mathcal{H} \otimes \mathcal{P}$. However, in doing so we would be omitting the information we have obtained from the classical outcome of the measurement: we in fact know what the state in $\mathcal{P}$ is. For our purposes, it is more elucidating to describe the two possible outcomes separately, providing both the resulting state in $\mathcal{H}$ when the measurement readout is $\top$ and that when the readout is $\bot$. Notice that, if the state in $\mathcal{H}$ prior to measurement were a pure state, both the outcome after we read $\bot$ or $\top$ will be pure states; this fact will be used in section 4.

If the outcome of the κ-measurement is $\top$, we are certain that the state left in space $\mathcal{H}$ satisfies predicate Q. On the other hand, outcome $\bot$ provides no definitive information about Q (except when $\kappa = 1$). For any state ρ, the probability of outcome $\top$ is

$$p_\top(\rho) = \kappa \cdot p_Q(\rho) \tag{6}$$

$$p_Q(\rho) = \mathrm{Tr}\left(O_Q(\rho \otimes |0\rangle\langle 0|)O_Q^\dagger\,(I_\mathcal{H} \otimes |1\rangle\langle 1|)\right), \tag{7}$$

where $p_Q(\rho)$ is the probability of predicate Q being satisfied by ρ. The smaller κ is, less likely it is that we read outcome $\top$. In exchange, the map corresponding to output $\bot$:

$$\mathcal{W}_\bot(\rho) = \frac{W_\bot(\rho \otimes |\bot\rangle\langle\bot|)W_\bot^\dagger}{1 - p_\top(\rho)} \tag{8}$$

$$W_\bot = (I_\mathcal{H} \otimes |\bot\rangle\langle\bot|)\,E_{\kappa,Q} \tag{9}$$

does *not* fully collapse the state to the subspace where Q is unsatisfiable.

3. A weakly measured while loop

This section contains the main contribution of our paper: we define κ-while loops and introduce the properties of $\mathcal{A}$-guarantee and robustness. In short, a κ-while loop is a classically controlled while loop where the test of the termination condition is realised by a κ-measurement.

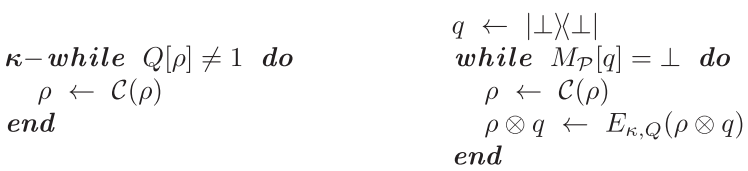

Figure 1. Left: the syntax we use to represent a κ-while loop; $\kappa \in [0,1]$ is a parameter set by the programmer and Q is the predicate to be measured. Right: the pseudocode that implements the κ-while loop on a programming language with classical control flow; $E_{\kappa,Q}$ and $M_{\mathcal{P}}$ are defined in section 2.2.

3.1. Motivation

Fix a predicate Q, and let $\mathcal{C}$ be a completely positive (CP) map acting on $\mathcal{H}$; we will refer to $\mathcal{C}$ as the *body* of the loop and write $D(\mathcal{H})$ for the set of density operators on $\mathcal{H}$.

Definition 3.1. Let $\sigma \in D(\mathcal{H})$ be an arbitrary state. Define a function $\mathcal{A}_{Q,\sigma} : \mathbb{N} \to \{0,1\}$ by

$$\mathcal{A}_{Q,\sigma}(n) \iff p_Q(\mathcal{C}^n(\sigma)) > \frac{1}{2}, \tag{10}$$

where p_Q is given in (7), and $\mathcal{C}^n(\sigma) = \overbrace{\mathcal{C}\,(\ldots(\mathcal{C}}^{n \text{ times}}(\sigma)))$. If $\mathcal{A}_{Q,\sigma}(n)$ is satisfied, we say n is an $\mathcal{A}$-iteration (read as *active iteration*).

If we somehow identified an $\mathcal{A}$-iteration m, we may use a simple approach to find a state satisfying Q: apply $\mathcal{C}^m(\sigma)$ and then perform a projective measurement; if the outcome does not satisfy Q, repeat the process from the initial state σ. Thanks to $\mathcal{A}_{Q,\sigma}(m)$ being satisfied, the probability of succeeding at some point within k restarts is $1 - 1/2^k$ so the probability of success quickly approaches 1 as k increases. This is an efficient approach whenever a small $\mathcal{A}$-iteration m is known. In fact, this is how the standard Grover's algorithm works, choosing an iteration m where p_Q is maximised and extremely close to 1.

However, this measure-restart strategy can only be implemented if we already know when $\mathcal{A}$-iterations will occur. The distribution of $\mathcal{A}$-iterations may become unpredictable as soon as some randomness is introduced in the body of the loop. For instance, consider the standard Grover's algorithm being implemented in a faulty machine where, without notice, the quantum memory may collapse to its initial state. If such collapse happens mid-computation, the evolution is effectively restarted, but the algorithm—oblivious to the collapse—continues for only the remaining fixed number of iterations. In contrast, a version of Grover's algorithm that uses a κ-while loop (such as the one we describe in section 4) will, by definition, keep iterating until it succeeds to find the target state. Although a contrived example, this illustrates how a while loop could provide reliability against unpredictable behavior. More realistic situations that would lead to an unpredictable distribution of $\mathcal{A}$-iterations could come from algorithms that, by design, apply mid-computation measurements or require interaction with the environment. Other situations where the use of κ-while loops may be advantageous are discussed in section 5.2.

In this section we present the concept of κ-while loops as an abstract quantum programming construct. In section 4 we present a version of Grover's algorithm that uses a κ-while loop and maintains the quadratic quantum speed-up. In the future, we hope to apply κ-while loops to practical problems where the distribution of $\mathcal{A}$-iterations cannot be predicted. To this end, we provide sufficient conditions that let us determine, for arbitrarily high probability, a worst-case estimate of the number of iterations the loop will run for. Remarkably, these conditions do not require us to know the precise distribution of $\mathcal{A}$-iterations, but only a guarantee of their proportion throughout the algorithm in the worst-case scenario.

3.2. The κ-while loop

In figure 1 we propose the syntax for the κ-while loop and its implementation on a quantum programming language with classical control flow. The syntax of the κ-while loop is meant to be read as 'repeat $\mathcal{C}$ while it is *not certain* that Q is satisfied'. On each iteration, the value of the state ρ will be updated to $\mathcal{C}(\rho)$, followed by a κ-measurement of predicate Q (see section 2.2). If the outcome of the κ-measurement is $\perp$, the loop keeps iterating; the state ρ becomes $\mathcal{W}_{\perp}(\rho)$, collapsing some of the quantum information. Otherwise, outcome $\top$ halts the loop and we succeed in obtaining a state ρ that satisfies Q.

Definition 3.2. A function $f : \mathbb{N} \to \mathbb{N}$ is a *guarantee* of $\mathcal{A}_{Q,\sigma}$ if:

$$\forall\, n \in \mathbb{N}, \quad \exists\, m \leqslant f(n) : n = |\{k \in \mathbb{N} | k < m,\, \mathcal{A}_{Q,\sigma}(k)\}|\,. \tag{11}$$

If such a function f exists, we are promised that there will be at least n active iterations within the first $f(n)$ applications of $\mathcal{C}$. In principle, this need not be a tight bound; for instance, we may know that, within the first thousand iterations, there will be at least ten $\mathcal{A}$-iterations. In that case, a valid guarantee of $\mathcal{A}_{Q,\sigma}$ would be $f(1) = f(2) = \cdots = f(10) = 1000$. This gives us little information about when any of these $\mathcal{A}$-iterations actually occur.

Definition 3.3. The evolution induced by a CP map $\mathcal{C}$ on a state σ is said to be ε-*robust* to κ-measurements of predicate Q if there is a function $g : \mathbb{N} \to \mathbb{N}$ such that:

$$\forall\ n \in \mathbb{N}, \quad \exists\ m \leqslant g(n) : |p_Q(\mathcal{C}^n(\sigma)) - p_Q((\mathcal{W}_\perp \mathcal{C})^m(\sigma))| \leqslant \varepsilon \tag{12}$$

for $\varepsilon \ll 1/2$. The function g is called a *witness* of the robustness.

Lemma 3.4. *If f is a guarantee of $\mathcal{A}_{Q,\sigma}$ and g witnesses that $\mathcal{C}$ is ε-robust, then $g \circ f$ is a guarantee of:*

$$\mathcal{A}'_{Q,\sigma}(n) \ \Leftrightarrow\ p_Q((\mathcal{W}_\perp \mathcal{C})^n(\sigma)) > \frac{1}{2} - \varepsilon. \tag{13}$$

Proof. After the first $f(n)$ applications of $\mathcal{C}$, there will be at least n $\mathcal{A}$-iterations. If we switch to weakly measured iterations $\mathcal{W}_\perp \mathcal{C}$, robustness tells us that within the first $g(f(n))$ iterations of the loop we will find those n active iterations again, but the probability p_Q might differ by ε, so it is at least $p_Q > 1/2 - \varepsilon$. Thus, for any $n \in \mathbb{N}$, there is $m \leqslant g(f(n))$ satisfying:

$$n = \left|\{k \in \mathbb{N} | k < m, \mathcal{A}'_{Q,\sigma}(k)\}\right|. \tag{14}$$

This concludes the proof. □

Lemma 3.4 guarantees that at least n of the first $g(f(n))$ iterations of the κ-while loop will be $\mathcal{A}'$-iterations. For each of these $\mathcal{A}'$-iterations, the probability of outcome $\top$ is at least $\kappa(1/2 - \varepsilon)$. Within N $\mathcal{A}'$-iterations the probability of loop termination is:

$$P_{\text{succ}} \geqslant 1 - \left(1 - \kappa(1/2 - \varepsilon)\right)^N. \tag{15}$$

If $x \in (0, 1)$, then $(1 - x)^{\frac{1}{x}} < \frac{1}{e}$. Therefore, if the algorithm is allowed to run for at least $N = \frac{2}{\kappa(1-2\varepsilon)}$ $\mathcal{A}'$-iterations, the probability of success is:

$$P_{\text{succ}} \geqslant 1 - \frac{1}{e} > \frac{1}{2}. \tag{16}$$

Therefore, with probability higher than $1/2$, our κ-while loop will halt within its first

$$T = g\left(f\left(\frac{2}{\kappa(1 - 2\varepsilon)}\right)\right) \tag{17}$$

iterations. More generally, for any $c \in \mathbb{N}$, the probability of success after cN $\mathcal{A}'$-iterations is greater than $1 - 1/e^c$. Therefore, with arbitrarily high probability, the loop halts successfully within $T_c = g(f(cN))$ iterations for small c.

Remark 3.5. Definition 3.2 can be weakened so that instead of f being a 'deterministic' guarantee of $\mathcal{A}_{Q,\sigma}$, we only impose that $f(n)$ satisfies (11) with probability higher than η. In such a case, after T_c iterations we can achieve a success probability arbitrarily close to η.

4. An example: Grover's algorithm

In Grover's search problem [12], we are given an unsorted set of elements B, about which we know no structure or heuristics, and a predicate $\chi : B \to \{0, 1\}$ that satisfies $\chi(\star) = 1$ for only one element $\star \in B$. We are tasked with finding this marked element $\star$. The standard Grover's algorithm defines an iteration operator G (using an oracle of χ) and applies it a fixed number of times K on an initial state $|\psi\rangle$. Afterward, a PVM on the basis B is applied, finding the marked element with high probability. In this section we discuss a different approach to Grover's search problem where a κ-while loop is used instead.

The algorithm's pseudocode is given in figure 2. When the κ-while loop halts, we are *certain* that the state is $|\star\rangle$. We do not fix the number of times G is applied; instead, we fix the measurement strength: κ. We will see that for $\kappa \approx |B|^{-1/2}$ the loop terminates within cK iterations with arbitrarily high probability, where c is a small constant and K is the number of times G is applied in the standard algorithm. Thus, the quadratic quantum speed-up of Grover's algorithm is preserved.

```
input: χ, |ψ⟩
output: q
begin
  q ← |ψ⟩
  κ−while χ[q] ≠ 1 do
    q ← G(q)
  end
end
```

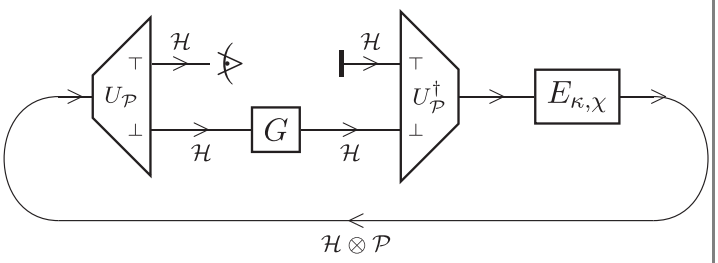

Figure 2. Left: pseudocode describing our algorithm. The while loop is controlled by a κ-measurement as described in section 3.2. Right: the algorithm's information flow; $E_{\kappa,\chi}$ is given in section 2.2, $U_{\mathcal{P}}$ applies the canonical isomorphism $\mathcal{H} \otimes \mathcal{P} \cong \mathcal{H} \oplus \mathcal{H}$, separating with respect to the orthogonal basis $\{|\bot\rangle, |\top\rangle\}$ of $\mathcal{P}$.

It is important to remark that, in the case of Grover's iterator G, we can easily predict when the active iterations occur (see section 3.1). Therefore, κ-while loops do not provide an algorithmic advantage on this problem. Instead, we present Grover's problem as a simple proof of concept of how while loops may be used in quantum algorithms without destroying the quantum speed-up. This approach to Grover's problem was first proposed by Mizel [18]. This section reformulates Mizel's results in the broader framework of section 3.2.

4.1. Standard Grover's algorithm

The standard Grover's algorithm acts on a quantum state in $\mathcal{H} = span\, B$, starting from the uniform superposition $|\psi\rangle$. Let $|\psi_1\rangle = |\star\rangle$ be the target state and

$$|\psi_0\rangle = \frac{1}{\sqrt{|B|-1}} \sum_{b \in B - \{\star\}} |b\rangle. \tag{18}$$

For any $a \in [0, 2\pi)$, define the state $|a\rangle$ as

$$|a\rangle = \cos a|\psi_0\rangle + \sin a|\psi_1\rangle. \tag{19}$$

Let $\alpha = \arcsin |\langle\star|\psi\rangle| \approx |B|^{-1/2}$. Notice that $|\alpha\rangle = |\psi\rangle$.

For any state $|\varphi\rangle \in \mathcal{H}$ we refer to its reflection operator as S_φ, given by:

$$S_\varphi = 2|\varphi\rangle\langle\varphi| - I_{\mathcal{H}}. \tag{20}$$

Grover's iteration is given by the operator

$$G = S_\psi S_{\psi_0}, \tag{21}$$

where S_{ψ_0} is implemented using a single call to the oracle of χ.

On a state $|a\rangle$, the action of Grover's iteration G is:

$$G|a\rangle = S_\psi S_{\psi_0}|a\rangle = S_\psi|-a\rangle = |a + 2\alpha\rangle. \tag{22}$$

In general, after k iterations:

$$G^k|\psi\rangle = \cos(\alpha + 2k\alpha)|\psi_0\rangle + \sin(\alpha + 2k\alpha)|\psi_1\rangle. \tag{23}$$

The standard Grover's algorithm applies G a total of $K = \lfloor(\pi\sqrt{|B|})/4\rfloor$ times on $|\psi\rangle$ so that the amplitude of $|\psi_1\rangle$ is maximised. Then, the state is measured on the basis B, finding the marked element $\star$ with high probability.

4.2. While loop approach

In this section we discuss the algorithm given in figure 2. To show that it retains the quantum speed-up, we provide a guarantee of $\mathcal{A}_\chi$ and prove that G is robust to κ-measurements of χ, as described in section 3.2. Our framework (section 3.2) is general enough to deal with mixed states and CP maps but, in this case, the body of the κ-while loop—the Grover operator G—is unitary and both the initial state $|\psi\rangle$ and the target $|\star\rangle$ are pure states. Therefore, as discussed in remark 2.3, all states involved in our analysis will be pure states.

Lemma 4.1. *The function* $f : \mathbb{N} \to \mathbb{N}$ *given by*

$$f(n) = 2n + K \tag{24}$$

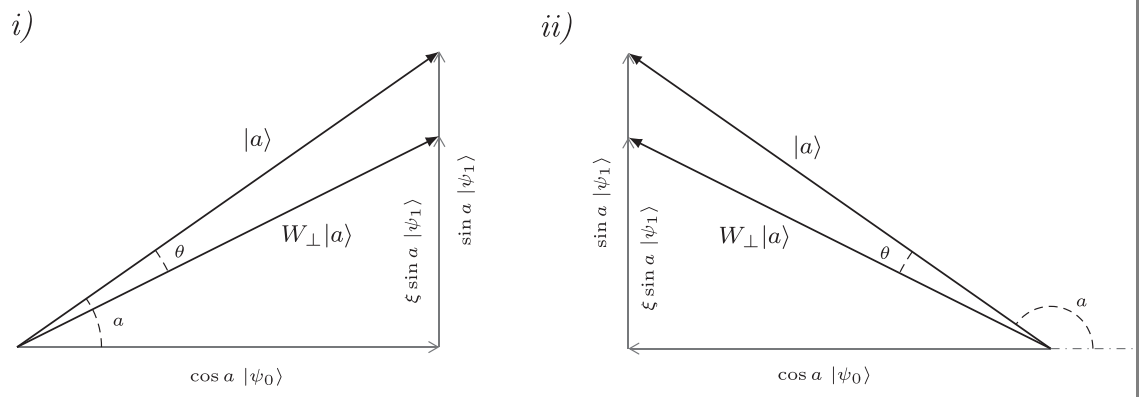

Figure 3. Geometric relation between the angle a before weak measurement and the offset $\theta(a,\kappa)$ after measuring outcome $\perp$. The construction is provided when (i) a is in the first quadrant and when (ii) a is in the second quadrant.

$$K = \left\lfloor \frac{\pi\sqrt{|B|}}{4} \right\rfloor \tag{25}$$

is a guarantee of $\mathcal{A}_\chi$.

Proof. By definition (19) of $|a\rangle$,

$$p_\chi(|a\rangle) = |\langle\star|a\rangle|^2 = \sin^2 a. \tag{26}$$

For two quarters of the circle, in particular whenever $a \in (\frac{\pi}{4}, \frac{3\pi}{4})$ or $a \in (\frac{5\pi}{4}, \frac{7\pi}{4})$, we have $p_\chi(|a\rangle) > \frac{1}{2}$. Moreover, $G|a\rangle = |a + 2\alpha\rangle$ according to (22), so the angle is increased a constant amount on each iteration. Thus, within $m \in \mathbb{N}$ iterations, approximately half of them are active. More precisely, we know that the number of $\mathcal{A}$-iterations is

$$n = \frac{m \pm K}{2}, \tag{27}$$

where the error margin K refers to the number of applications of G it takes to traverse a quarter of the circle, which is the longest interval of consecutive iterations not satisfying $\mathcal{A}_\chi$. Solving for m and taking the worst case scenario, we find that within $m = 2n + K$ iterations it is guaranteed that n are active. □

Consider applying a κ-measurement at iteration n: if the outcome is $\top$, the state becomes $|\star\rangle$, otherwise it suffers a collapse toward $|\psi_0\rangle$ determined by the projector $W_\perp$ from section 2.2.

$$\begin{aligned} W_\perp|a,\perp\rangle &= \cos a W_\perp|\psi_0,\perp\rangle + \sin a W_\perp|\psi_1,\perp\rangle \\ &= \cos a|\psi_0,\perp\rangle + \sin a(I_{\mathcal{H}} \otimes |\perp\rangle\langle\perp|)\, E_{\kappa,\chi}|\psi_1,\perp\rangle \\ &= \cos a|\psi_0,\perp\rangle + \sin a\sqrt{1-\kappa}\,|\psi_1,\perp\rangle. \end{aligned} \tag{28}$$

Thus, the output is still in a superposition between $|\psi_0\rangle$ and $|\psi_1\rangle$. For any $|a\rangle$ we may describe the action of $W_\perp$ on it as $W_\perp|a\rangle \propto |a - \theta(a,\kappa)\rangle$. Notice that $W_\perp$ returns an unnormalised state; the actual CPTP map $\mathcal{W}_\perp$ from (8) takes care of the normalization, so:

$$|a\rangle \overset{\mathcal{W}_\perp}{\longmapsto} |a - \theta(a,\kappa)\rangle. \tag{29}$$

Figure 3 shows a geometric construction of $\theta(a,\kappa)$ according to (28). We now find an upper bound of $\theta(a,\kappa)$; this will be used to prove robustness of G. To reduce clutter, we use the shorthand:

$$\xi = \sqrt{1-\kappa}. \tag{30}$$

Lemma 4.2. *For any* κ *and angle* a

$$|\theta(a,\kappa)| \leqslant \arcsin\left(\frac{1-\xi}{1+\xi}\right). \tag{31}$$

Proof. See appendix A. □

Corollary 4.3. *For any angle* a, $|\theta(a,\kappa)| \leqslant \arcsin\,\kappa$.

Proof. By definition, $\kappa \in [0,1]$. Using this fact, together with (30) and simple algebra, we can prove that

$$0 \leqslant \frac{1-\xi}{1+\xi} \leqslant \kappa. \tag{32}$$

Then, the corollary follows from lemma 4.2. □

In a κ-measurement of $\kappa \approx 1$ the collapse $\theta(a,\kappa)$ can be up to $\frac{\pi}{2}$, thus always sending the state back to $|\psi_0\rangle$, preventing a gradual evolution of the angle. In such a case, the algorithm would lose its quantum speed-up. Interestingly, corollary 4.3 shows we can keep $\theta(a,\kappa)$ small by bounding κ, so that the action of G overcomes the effect of the collapse.

Lemma 4.4. *The unitary map G is ε-robust to κ-measurements of χ for*

$$\kappa \leqslant \frac{1}{\sqrt{|B|}} \tag{33}$$

$$\varepsilon = \sin\, 3\alpha \tag{34}$$

which is witnessed by:

$$g(n) = 2n. \tag{35}$$

Proof. Let $\{a_n\}$ be the following sequence of angles:

$$\begin{aligned} a_{n+1} &= a_n + 2\alpha \\ a_0 &= \alpha. \end{aligned} \tag{36}$$

Then $G^n|\psi\rangle = |a_n\rangle$, where $|\psi\rangle$ is the initial state. Similarly, we define a sequence $\{b_n\}$ satisfying $(\mathcal{W}_\perp G)^n|\psi\rangle = |b_n\rangle$:

$$\begin{aligned} b_{n+1} &= b_n - \theta(b_n,\kappa) + 2\alpha \\ b_0 &= \alpha. \end{aligned} \tag{37}$$

Remember that $\sin\,\alpha = |B|^{-1/2}$, so the imposed bound on κ can be rephrased as $\kappa \leqslant \sin\alpha$. Thanks to corollary 4.3, this implies $|\theta(b_n,\kappa)| \leqslant \alpha$ on any iteration. Then $\alpha \leqslant b_{n+1} - b_n \leqslant 3\alpha$ for any $n \in \mathbb{N}$, whereas $a_{n+1} - a_n = 2\alpha$. Therefore

$$\exists\, m \leqslant 2n : b_m \leqslant a_n \leqslant b_{m+1} \tag{38}$$

and $a_n - b_m \leqslant 3\alpha$. Because $p_\chi(G^n|\psi\rangle) = \sin^2 a_n$ and $p_\chi((\mathcal{W}_\perp G)^m|\psi\rangle) = \sin^2 b_m$, it follows that

$$|p_\chi(G^n|\psi\rangle) - p_\chi((\mathcal{W}_\perp G)^m|\psi\rangle)| \;\leqslant\; \left|\sin^2 a_n - \sin^2(a_n \pm 3\alpha)\right|. \tag{39}$$

Using $\sin^2 x - \sin^2(x+y) = -\sin(y)\sin(2x+y)$ we reach the conclusion that, for any $n \in \mathbb{N}$

$$\exists\, m \leqslant 2n : |p_\chi(G^n|\psi\rangle) - p_\chi((\mathcal{W}_\perp G)^m|\psi\rangle)| \;\leqslant\; |\sin\, 3\alpha|. \tag{40}$$

Hence, G is ε-robust to κ-measurements for $\varepsilon = \sin 3\alpha$, with witness $g(n) = 2n$. □

From the general result of lemma 3.4 and the discussion following it, together with the assumption that α is small ($\sin 3\alpha \ll 1/2$), we conclude that the algorithm in figure 2 will succeed in finding the marked element $|\star\rangle$ within

$$T = g\left(f\left(\frac{2c}{\kappa(1-2\varepsilon)}\right)\right) \tag{41}$$

iterations, with arbitrarily high probability, for small $c \in \mathbb{N}$. Combining this with lemmas 4.1 and 4.4 we find that, for $\kappa \leqslant |B|^{-1/2}$, the number of iterations is within $\mathcal{O}\left(\sqrt{|B|}\right)$.

Remark 4.5. Our analysis yields an arguably large constant factor

$$T \approx (8c + \frac{\pi}{2})\sqrt{|B|}. \tag{42}$$

However, it is important to remark that this factor is an overestimation, due to the simplifications applied when finding f and g in this section. With these simplifications we intended to prioritise clarity of our proof strategy. Tighter bounds would yield a more accurate factor. In fact, figure 4 suggests the average number of iterations is approximately $2\sqrt{|B|}$.

Moreover, our choice of $\kappa = |B|^{-1/2}$ is not optimal. To prove lemma 4.4 we only needed that the collapse on each iteration was smaller than the increment of the angle due to G, i.e. for every angle a, $\theta(a,\kappa) < 2\alpha$. If we set $\kappa = 5|B|^{-1/2}$, and use the bound of $\theta(a,\kappa)$ given by lemma 4.2, we can verify that, for this instance of $|B| = 10^6$, $\theta(a,\kappa) < 1.3\alpha$, so the collapse is appropriately bounded. In this particular case, numerical analysis shows that the average number of iterations required is slightly smaller than $\frac{\pi}{4}\sqrt{|B|}$, which is the number of iterations the standard Grover's algorithm runs for. The optimal value of κ

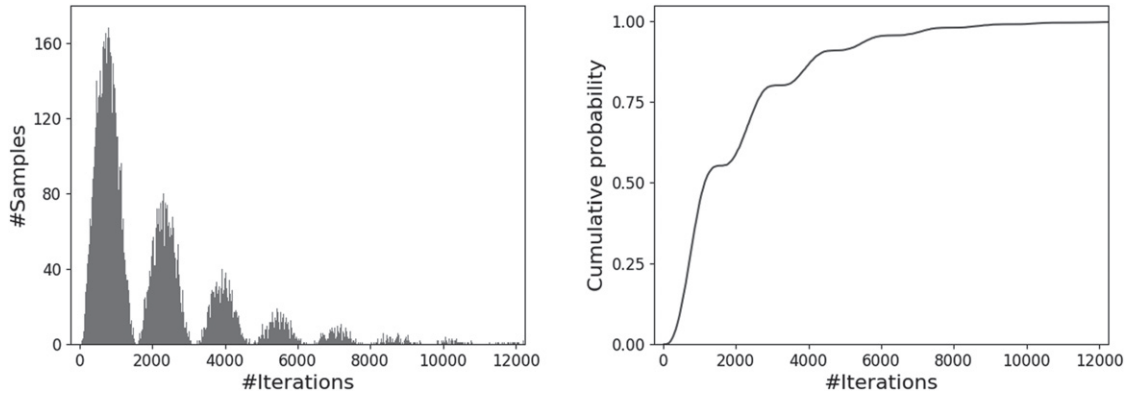

Figure 4. Distribution of the number of iterations before success for parameters $|B| = 10^6$ and $\kappa = |B|^{-1/2}$; left: histogram, right: cumulative probability distribution. Drawn from 10 000 samples, obtained by sampling the success probability $p_\top = \kappa \sin^2 b_n$ where b_n is given by (37). The median is approximately 1000 iterations, and mean 2000 iterations, i.e. approximately $\frac{1}{\kappa}$ and $\frac{2}{\kappa}$ respectively.

depends on the parameters of the problem (i.e. in this case, it may be different for different values of $|B|$). We conjecture that such an optimal value of κ exists for any instance of Grover's problem, so that our approach and the standard one coincide in their expected number of iterations.

The realisation that for a fine-tuned value of κ our algorithm would perform as well as the standard Grover's algorithm was first discussed in a recent paper [8]. The paper provides an automatic procedure, based on differentiable programming, to choose an optimal value of the parameters of a quantum program. It discusses our algorithm as an application of its approach and, for small values of $|B|$, finds an optimal κ satisfying the conjecture above.

Figure 4 helps us visualise the behavior of the algorithm: as the angle b_n monotonically increases throughout the iterations, the instantaneous probability of success $p_\top = \kappa \sin^2 b_n$ changes periodically. The peaks of the histogram correspond to intervals of $\mathcal{A}$-iterations where the probability of success approaches its maximum (that is, $p_\top \approx \kappa$); these alternate with troughs where the probability approaches 0.

5. Discussion

We have used the term *active iteration* to refer to the stages of the loop where halting is most likely. If the distribution of active iterations is not known, we cannot fix the number of iterations the loop should run for. In such a situation, it is natural to use a while loop, but these require testing a termination condition, perturbing the state. In this work, we have proposed κ-while loops, which let us keep the perturbation due to measurement below a threshold. We have introduced the properties of $\mathcal{A}$-guarantee and robustness, which let us identify the time complexity of algorithms using these κ-while loops.

We must emphasise that these κ-while loops can be realised in any quantum programming language with classical control flow, as indicated in figure 1. More precisely, κ-while loops are a particular instance of the while loops described in [23]. Hence, we expect it would be immediate to apply to κ-while loops any findings from the literature on classical control flow of quantum programs; for instance, verification of program correctness [9], study of loop termination [15, 23] and program semantics [20, 22]. Considering the control flow is classical, there is no superposition of different branches of the computation exiting the loop at different stages. However, the κ-while loop is not fully classical either, in the sense that some degree of superposition in the quantum data is maintained across iterations, which is enough to achieve quantum speed-up in certain situations.

The key component of the κ-while loop is a weak measurement parametrised by κ. Weak measurements are not new to quantum computer scientists, as they are at the heart of quantum feedback control [26]. Our κ-while loop can be seen as an example of quantum feedback control where the weak measurements are applied at discrete time steps. Discrete time feedback control has been used to protect a single qubit from decoherence [5, 11], while similar notions of weak measurement (over continuous time) have been proposed to monitor and drive complex evolutions [17]. However, weak measurements are rarely used in the design of quantum algorithms; Mizel's work [18] being the only case we are aware of. Defining the κ-while loop programming construct introduces weak measurements to algorithm designers and programming language experts in a language that is familiar to them.

5.1. Comparison to other versions of Grover's algorithm

We have shown that a κ-while loop may be used to implement Grover's algorithm; however, there is no algorithmic advantage with respect to the standard approach. The reason is that the distribution of $\mathcal{A}$-iterations throughout Grover's algorithm is easy to predict, and thus we can fix the number of iterations we should run it for in advance, instead of using a κ-while loop. Our approach can be easily extended to the setting of amplitude amplification [6].

In principle, our κ-while loop method (as well as Mizel's [18]) requires knowledge of the proportion γ of marked states in the database (in Grover's, $\gamma = 1/B$) so that $\kappa \approx \gamma^{-1/2}$ can be chosen appropriately. However, the tricks used in the standard algorithm to figure out γ can also be applied to our setting. Interestingly, whereas an under- or overestimation of γ causes the standard algorithm to behave unpredictably, our approach will always terminate in success. In particular, if we underestimate the proportion and set $\kappa \ll \gamma^{-1/2}$, the quadratic speed-up will be maintained, but we will run for more iterations than strictly necessary. On the other hand, if we overestimate it and set $\kappa \gg \gamma^{-1/2}$, there will be too much collapse per iteration, resulting in the loss of quantum speed-up.

The latter point was already argued by Mizel [18], who proposed a version of Grover's algorithm that is, in essence, the same as ours. Mizel presents the algorithm as a fixed point routine, where the state gradually converges to a marked state throughout the iterations. However, this fixed point behavior is an artifact of disregarding the outcome of the weak measurement in their analysis (see remark 2.3). In reality, the state keeps evolving at the same rate throughout the algorithm until a $\top$ measurement outcome occurs, when the state collapses onto the marked subspace. The field of quantum trajectories [7] focuses on understanding these kind of stochastic dynamics in the presence of weak measurements.

It is worth mentioning that algorithms for Grover's search using a fixed point approach do exist [13, 25]. In these, no measurement is applied during execution; instead, each iteration applies a unitary parametrised by a value that is gradually reduced throughout the algorithm. Intuitively, each iteration moves the state closer to a marked one, but each time the step is smaller to avoid overshooting. The drawback of this fixed point approach is that, unless we can implement a circuit where the unitary's parameter can be changed during runtime, each iteration requires a different circuit.

5.2. Applications beyond quantum search

We have presented a κ-while loop version of Grover's algorithm as a proof of concept. The next step in this project is to find practical uses of κ-while loops: quantum algorithms where the distribution of active iterations is unknown or it would be costly to predict. In section 3.1 we have suggested that choosing κ-while loops over for loops may make algorithms more reliable against unpredictable behavior in the body of the loop. It would be valuable to find an explicit example of a quantum algorithm that, under a realistic noise model, behaves more reliably when implemented using κ-while loops.

Another natural line of research is to identify practical problems where the distribution of $\mathcal{A}$-iterations is inherently unpredictable. The literature on quantum walks provides multiple results where quantum computers exhibit an advantage in the study of Markov processes [2, 21]. The unitary evolution of these quantum walks is defined in terms of the transition matrix of the Markov process, which is often only required to be symmetric and ergodic. Thus, the literature from this field provides us with a diverse set of examples of quantum algorithms whose evolution may be arbitrarily complex. We argue there may be Markov processes whose quantum walk can be proven to be robust to κ-measurements as in definition 3.3 while, at the same time, too unpredictable to let us estimate when the evolution reaches a desired state. The argument goes as follows: on one hand, the choice of κ will be determined by a lower bound of the rate at which the walker traverses the graph (so that κ-measurements are not so strong that they would prevent the walker from reaching certain regions of the graph); this rate may be estimated as the infimum of local rates, using notions such as conductance and effective resistances [4]. On the other hand, in order to predict when the walker reaches a particular state, we need to take into account the global behavior of the walk, which may be a more daunting task. We therefore find the field of [2, 4, 21] to be a particularly promising area where to look for applications of κ-while loops.

The notion of weak measurement used in our definition of κ-while loop is discrete. However, continuous measurements have been studied extensively [14] and the body of the while loop itself may be replaced by a continuous evolution given by a Hamiltonian, thus extending κ-while loops to the continuous-time setting. This may be valuable from an experimentalist perspective; for instance, it hints at implementations of algorithms (for instance, Grover's search) where the probe is continuously measured throughout the execution, thus simplifying control, e.g. we do not need to know when each iteration finishes and the next one starts.

The κ-while loop is a programming construct that offers a promising abstraction: a way to iteratively 'peek' at quantum states without completely collapsing them. The key challenge to its use in algorithms is

the need to strike a balance—by tuning parameter κ—so that the collapse is low enough, while the information gained is sufficient. The body of a κ-while loop may be any CP map, hence κ-while loops may be nested inside each other. As with classical while loops, termination implies the satisfaction of the predicate, which is a useful feature for the analysis of program correctness.

Acknowledgments

Pablo Andres-Martinez is supported by EPSRC Grant EP/L01503X/1 via the Centre for Doctoral Training in Pervasive Parallelism at the University of Edinburgh, School of Informatics. Chris Heunen is supported by EPSRC Fellowship EP/R044759/1.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Appendix A. A bound to the collapse

Lemma. *For any κ and angle a*

$$|\theta(a,\kappa)| \leqslant \arcsin\left(\frac{1-\xi}{1+\xi}\right).$$

Proof. Suppose $a \in [0, \frac{\pi}{2}]$ and use the properties of sines on the triangle from figure 3(i) to obtain:

$$\frac{\sin(a-\theta)}{\xi \sin a} = \frac{\sin(\pi/2 - a + \theta)}{\cos a}. \tag{43}$$

This simplifies as follows:

$$\begin{aligned} & \frac{\sin(a-\theta)}{\xi \sin a} = \frac{\cos(a-\theta)}{\cos a} \\ \Leftrightarrow \quad & \tan(a-\theta) = \xi \tan a \\ \Leftrightarrow \quad & \frac{\tan a - \tan\theta}{1 + \tan a \tan\theta} = \xi \tan a. \end{aligned} \tag{44}$$

Solving for θ gives:

$$\theta(a,\kappa) = \arctan\left(\frac{(1-\xi)\tan a}{1+\xi\tan^2 a}\right). \tag{45}$$

If $a \in \left[\frac{\pi}{2}, \pi\right]$ instead, a similar analysis yields:

$$\theta(a,\kappa) = -\arctan\left(\frac{(1-\xi)\tan a}{1+\xi\tan^2 a}\right) \tag{46}$$

which only differs from equation (45) in the sign. If a is in the third quadrant, then we obtain equation (45) and, if a is in the fourth quadrant, we get (46). These sign changes are convenient: the geometric analysis in figure 3 shows that in the first (and third) quadrant, the resulting angle is $a - |\theta|$, whereas in the second (and fourth) quadrant it is $a + |\theta|$. Thus the resulting angle is $a - \theta$ regardless of which quadrant a lies in.

Next, we determine the maximum value of $\theta(a,\kappa)$. To do so, fix κ and find the critical points of θ as a function on a. These happen where the derivative

$$\frac{\mathrm{d}\theta}{\mathrm{d}a} = 1 - \frac{\xi}{\cos^2 a + \xi^2 \sin^2 a} \tag{47}$$

vanishes. Critical points occur periodically (once in every quadrant), but all of them reach the same absolute value. The critical point within the first quadrant happens at:

$$a = \arccos\left(\sqrt{\frac{\xi}{\xi+1}}\right). \tag{48}$$

Applied to equation (45) this gives a tight upper bound:

$$|\theta(a,\kappa)| \leqslant \arctan\left(\frac{1-\xi}{2\sqrt{\xi}}\right). \tag{49}$$

Using the equality $\sin(\arctan(x)) = \frac{x}{\sqrt{x^2+1}}$, the fact that $\xi \in [0,1]$ and some basic algebra we reach the claim:

$$\sin|\theta(a,\kappa)| \leqslant \frac{1-\xi}{1+\xi}. \tag{50}$$

□

ORCID iDs

Pablo Andrés-Martínez https://orcid.org/0000-0003-4456-7052
Chris Heunen https://orcid.org/0000-0001-7393-2640

References

[1] Altenkirch T and Grattage J 2005 A functional quantum programming language *20th Annual IEEE Symp. on Logic in Computer Science (LICS'05)* (IEEE) pp 249–58
[2] Ambainis A, Gilyén A, Jeffery S and Kokainis M 2020 Quadratic speedup for finding marked vertices by quantum walks *Proc. 52nd Annual ACM SIGACT Symp. on Theory of Computing* pp 412–24
[3] Bădescu C and Panangaden P 2015 Quantum alternation: prospects and problems (arXiv:1511.01567)
[4] Belovs A 2013 Quantum walks and electric networks (arXiv:1302.3143)
[5] Brańczyk A M, Mendonça P E M F, Gilchrist A, Doherty A C and Bartlett S D 2007 Quantum control of a single qubit *Phys. Rev.* A **75** 012329
[6] Brassard G, Hoyer P, Mosca M and Tapp A 2002 Quantum amplitude amplification and estimation *Contemp. Math.* **305** 53–74
[7] Brun T A 2002 A simple model of quantum trajectories *Am. J. Phys.* **70** 719–37
[8] Fang W, Ying M and Wu X 2021 Differentiable quantum programming with unbounded loops (unpublished)
[9] Feng Y and Ying M 2020 Quantum Hoare logic with classical variables (arXiv:2008.06812)
[10] Gay S J 2006 Quantum programming languages: survey and bibliography *Math. Struct. Comput. Sci.* **16** 581
[11] Gillett G G *et al* 2010 Experimental feedback control of quantum systems using weak measurements *Phys. Rev. Lett.* **104** 080503
[12] Grover L K 1996 A fast quantum mechanical algorithm for database search *Proc. 28th Annual ACM Symp. on Theory of Computing* pp 212–9
[13] Grover L K 2005 Fixed-point quantum search *Phys. Rev. Lett.* **95** 150501
[14] Jacobs K and Steck D A 2006 A straightforward introduction to continuous quantum measurement *Contemp. Phys.* **47** 279–303
[15] Li Y and Ying M 2017 Algorithmic analysis of termination problems for quantum programs *POPL* vol 2
[16] Linden N and Popescu S 1998 The halting problem for quantum computers (arXiv:quant-ph/9806054)
[17] Lloyd S and Slotine J-J E 2000 Quantum feedback with weak measurements *Phys. Rev.* A **62** 012307
[18] Mizel A 2009 Critically damped quantum search *Phys. Rev. Lett.* **102** 150501
[19] Sabry A, Valiron B and Vizzotto J K 2018 From symmetric pattern-matching to quantum control *Int. Conf. on Foundations of Software Science and Computation Structures* (Springer) pp pp 348–64
[20] Selinger P 2004 Towards a quantum programming language *Math. Struct. Comput. Sci.* **14** 527–86
[21] Szegedy M 2004 Quantum speed-up of Markov chain based algorithms *45th Annual IEEE Symp. on Foundations of Computer Science* (IEEE) pp 32–41
[22] Ying M 2016 *Foundations of Quantum Programming* (San Mateo, CA: Morgan Kaufmann Publishers)
[23] Ying M and Feng Y 2010 Quantum loop programs *Acta Inf.* **47** 221–50
[24] Ying M, Yu N and Feng Y 2014 Alternation in quantum programming: from superposition of data to superposition of programs (arXiv:1402.5172)
[25] Yoder T J, Low G H and Chuang I L 2014 Fixed-point quantum search with an optimal number of queries *Phys. Rev. Lett.* **113** 210501
[26] Zhang J, Liu Y-X, Wu R-B, Jacobs K and Nori F 2017 Quantum feedback: theory, experiments, and applications *Phys. Rep.* **679** 1–60