Universality of quantum Turing machines with deterministic control

P. Mateus A. Sernadas A. Souto

Dep. Matemática, Instituto Superior Técnico, Universidade de Lisboa, Portugal
and
SQIG, Instituto de Telecomunicações, Lisboa, Portugal
{pmat,acs,a.souto}@math.ist.utl.pt

January 29, 2014

Abstract

A simple notion of quantum Turing machine with deterministic, classical control is proposed and shown to be powerful enough to compute any unitary transformation which is computable by a finitely generated quantum circuit. An efficient universal machine with the s-m-n property is presented. The BQP class is recovered. A robust notion of plain Kolmogorov complexity of quantum states is proposed and compared with those previously reported in the literature.

Keywords: quantum computation, computational complexity, quantum Kolmogorov complexity.

1 Introduction

The original notion of quantum Turing machine was proposed by Deutsch in [6] and its main properties established in [2]. However, most of the work in quantum computation has been carried out using quantum circuits, since the seminal papers by Deutsch–Jozsa [7], Shor [18] and Grover [9]. In fact, the original notion of Turing machine raised some difficulties with the halting condition. This issue was addressed by several authors. For instance, in [20, 16] notions of quantum Turing machine are proposed using probabilistic, classical control and, thus, completely avoiding the issue. Herein, we propose yet another notion of quantum Turing machine using deterministic, classical control.

Such a deterministic-control quantum Turing machine, in short dcq Turing machine, is a classical Turing machine enriched with a quantum tape on which the machine can apply unitary quantum operations but cannot make measurements. Like in quantum circuits, our approach follows the idea that one can carry out quantum computations by following a classical procedure that in some steps may involve quantum manipulations but no measurements which are delayed to the end. The control and the halting condition of dcq Turing machines are, thus, completely classical and deterministic. Once the computation halts the contents of the quantum tape can be accessed by a suitable measurement. During the computation the contents of the quantum tape do not affect the behavior of the machine.

After the detailed presentation in Section 2 of dcq Turing machines and how they can be used for computing unitary quantum operators and deciding classical problems, we proceed to establish the key results that are required for accepting dcq Turing machines as model of quantum computation.

In Section 3 we show that, for a given n, every quantum circuit on n qubits can be efficiently emulated by a dcq Turing machine. This fibered emulation of quantum circuits is improved in Section 5 where it is shown that every computable family of quantum circuits can be efficiently emulated by a dcq Turing machine which allows the recovery of BQP. The uniform emulation of quantum circuits relies on the techniques used in Section 4 for establishing the existence of a universal dcq Turing machine that can efficiently emulate any dcq Turing machine – the counterpart for dcq Turing machines of the Hennie–Stearns theorem [10]. In fact, universality is obtained as a corollary of a stronger result – the counterpart for dcq Turing machines of Kleene's s-m-n theorem. The latter is a polynomial translatability result which plays an important role in Section 6 for defining a robust notion of plain quantum Kolmogorov complexity using universal dcq Turing machines.

With these results in hand, dcq Turing machines appear as a viable model of quantum computation with some obvious advantages over quantum circuits for the development of quantum computability and complexity which are discussed in Section 7.

2 Deterministic control of quantum computations

By a *deterministic-control quantum Turing machine* (in short, *dcq Turing machine*) we mean a variant of a binary Turing machine with two tapes, one classical and the other with quantum contents, which are infinite in both directions. Depending only on the state of the classical finite control automaton and the symbol being read by the classical head, the quantum head acts upon the quantum tape, a symbol can be written by the classical head, both heads can be moved independently of each other and the state of the control automaton can be changed.

A computation ends if and when the control automaton reaches the halting state ($\mathsf{q_h}$). Notice that the contents of the quantum tape do not affect the computation flow, hence the deterministic control and, so, the deterministic halting criterion. In particular, the contents of the quantum tape do not influence at all if and when the computation ends.

The quantum head can act upon one or two consecutive qubits in the quantum tape. In the former case, it can apply any of the following operators to the qubit under the head: identity (Id), Hadamard (H), phase (S) and π over 8 ($\pi/8$). In the latter case, the head acts on the qubit under it and the one immediately to the right by applying swap (Sw) or control-not ($\mathsf{c\text{-}Not}$) with the control qubit being the qubit under the head.

Initially, the control automaton is in the starting state ($\mathsf{q_s}$), the classical tape is filled with blanks (that is, with $\square$'s) outside the finite input sequence x of bits, the classical head is positioned over the rightmost blank before the input bits, the quantum tape contains three independent sequences of qubits – an infinite sequence of $|0\rangle$'s followed by the finite input sequence $|\psi\rangle$ of possibly entangled qubits followed by an infinite sequence of $|0\rangle$'s, and the quantum head is positioned over the rightmost $|0\rangle$ before the input qubits. In this situation, we say that the machine starts with input $(x, |\psi\rangle)$.

The control automaton is defined by the partial function

$$\delta : Q \times \mathbb{A} \rightharpoonup \mathbb{U} \times \mathbb{D} \times \mathbb{A} \times \mathbb{D} \times Q$$

where: Q is the finite set of control states containing at least the two distinct states $\mathsf{q_s}$ and $\mathsf{q_h}$ mentioned above; $\mathbb{A}$ is the alphabet composed of 0, 1 and $\square$; $\mathbb{U}$ is the set $\{\mathsf{Id}, \mathsf{H}, \mathsf{S}, \pi/8, \mathsf{Sw}, \mathsf{c\text{-}Not}\}$ of primitive unitary operators that can be applied to the quantum tape; and $\mathbb{D}$ is the set $\{\mathsf{L}, \mathsf{N}, \mathsf{R}\}$ of possible head displacements – one position to the left, none, and one position to the right.

For the sake of a simple halting criterion, we assume that $(\mathsf{q_h}, a) \notin \operatorname{dom} \delta$ for every $a \in \mathbb{A}$ and $(q, a) \in \operatorname{dom} \delta$ for every $a \in \mathbb{A}$ and $q \neq \mathsf{q_h}$. Thus, as envisaged, the computation carried out by the machine does not terminate if and only if the halting state $\mathsf{q_h}$ is not reached.

The machine evolves according to δ as expected:

$$\delta(q, a) = (U, d, a', d', q')$$

imposes that if the machine is at state q and reads a on the classical tape, then the machine applies the unitary operator U to the quantum tape, displaces the quantum head according to d, writes symbol a' on the classical tape, displaces the classical head according to d', and changes its control state to q'.

In short, by a dcq Turing machine we understand a pair (Q, δ) where Q and δ are as above.

Concerning computations, the following terminology becomes handy. The machine is said *to start from* $(x, |\psi\rangle)$ or to receive *input* $(x, |\psi\rangle)$ if: (i) the initial content of the classical tape is x surrounded by blanks and the classical head is positioned in the rightmost blank before the classical input x; (ii) the initial content of the quantum tape is $|\psi\rangle$ surrounded by $|0\rangle$'s and the quantum head is positioned in the rightmost $|0\rangle$ before the quantum input $|\psi\rangle$. Observe that the qubits containing the quantum input are not entangled with the other qubits of the quantum tape. When the quantum tape is completely filled with $|0\rangle$'s we say that the quantum input is $|\varepsilon\rangle$.

Furthermore, the machine is said *to halt at* $(y, |\varphi\rangle)$ or to produce *output* $(y, |\varphi\rangle)$ if the computation terminates and: (i) the final content of the classical tape is y surrounded by blanks and the classical head is positioned in the rightmost blank before the classical output y; (ii) the final content of the quantum tape is $|\varphi\rangle$ surrounded by $|0\rangle$'s and the quantum head is positioned in the rightmost $|0\rangle$ before the quantum output $|\varphi\rangle$. In this situation we may write

$$M(x, |\psi\rangle) = (y, |\varphi\rangle).$$

Clearly, the qubits containing the quantum output are not entangled with the other qubits of the quantum tape.

For each $n \in \mathbb{N}^+$, denote by H^n the Hilbert space of dimension 2^n. A unitary operator

$$U : \mathsf{H}^n \to \mathsf{H}^n$$

is said to be *dcq computable* if there is a dcq Turing machine (Q, δ) that, for every unit vector $|\psi\rangle \in \mathsf{H}^n$, when starting from $(\varepsilon, |\psi\rangle)$ produces the quantum output $U|\psi\rangle$. Note that the final content of the classical tape is immaterial.

A (classical) problem

$$X \subseteq \{0,1\}^*$$

is said to be *dcq decidable* if there is a dcq Turing machine (Q, δ) that, for every $x \in \{0,1\}^*$, when starting from $(x, |\varepsilon\rangle)$ produces a quantum output $|\varphi\rangle$ such that:

$$\begin{cases} \mathsf{Prob}\,(\mathsf{Proj}_1|\varphi\rangle = 1) > 2/3 & \text{if } x \in X \\ \mathsf{Prob}\,(\mathsf{Proj}_1|\varphi\rangle = 0) > 2/3 & \text{if } x \notin X. \end{cases}$$

Moreover, problem X is said to be (time) *dcq bounded error quantum polynomial*, in short in dcBQP, if there are polynomial $\xi \mapsto P(\xi)$ and a dcq Turing machine deciding X that, for each x, produces the output within $P(|x|)$ steps.

As expected, we shall establish in due course that the quantum computation concepts above coincide with those previously introduced in the literature using quantum circuits.

It is straightforward to see that dcq decidability coincides with the classical notion. It is enough to take into account that the dcq Turing machines can be emulated by classical Turing machines using a classical representation of the contents of the quantum tape that might be reached from $(x, |\varepsilon\rangle)$.

In the sequel we also need the following notion that capitalizes on the fact that dcq Turing machines can work like classical machines by ignoring the quantum tape. A function

$$f : \{0,1\}^* \rightharpoonup \{0,1\}^*$$

is said to be *classically dcq computable* if there is a dcq Turing machine (Q, δ) that, for every $x \in \{0,1\}^*$, when starting from input $(x, |\psi\rangle)$ produces the classical output $f(x)$ if $x \in \operatorname{dom} f$ and fails to halt with a meaningful classical output if $x \notin \operatorname{dom} f$.

Again, it is straightforward to show that classical dcq computability coincides with classical computability.

3 Emulating quantum circuits

Our goal now is to prove that every unitary operator $U : \mathsf{H}^n \to \mathsf{H}^n$ that can be computed by a quantum circuit is also dcq computable. To this end, we need some preliminary concepts and results concerning quantum circuits.

Recall that, as shown in [4], for any unitary operator $U : \mathsf{H}^n \to \mathsf{H}^n$ and $e \in \mathbb{R}^+$, there is a quantum circuit C built only with gates in

$$\{\mathsf{H}, \mathsf{S}, \pi/8, \text{c-Not}\}$$

that approximates U up to e:

$$\max_{\||\psi\rangle\|=1} \|(U - C)|\psi\rangle\| < e.$$

Accordingly, one says that $U : \mathsf{H}^n \to \mathsf{H}^n$ is *quantum-circuit computable* (in short, *qc computable*) if there is a quantum circuit C built only with gates in $\{\mathsf{H}, \mathsf{S}, \pi/8, \mathsf{c\text{-}Not}\}$ such that $U|\psi\rangle = C|\psi\rangle$ for each $|\psi\rangle \in \mathsf{H}^n$. The result above amounts to saying that the set of unitary operators which are computable by such finitely generated quantum circutis is dense in the set of unitary operators.

Observe that, *a fortiori*, any unitary operator can be approximated by a quantum circuit using gates in $\mathbb{U}$. We included Id and Sw in the set $\mathbb{U}$ of primitive operators only as a matter of convenience. For instance, Id coincides with $\mathsf{H} \circ \mathsf{H}$.

By a *seesaw quantum circuit* we mean a circuit only using gates in $\mathbb{U}$ and such that:

(i) Two-qubit gates are applied only to adjacent qubits.

(ii) The control qubit of each $\mathsf{c\text{-}Not}$ gate is the topmost one.

(iii) The first gate and the last gate are applied to the top qubit.

(iv) The topmost qubit of each subsequent gate is the topmost qubit of the previous gate or adjacent to it.

As the next lemma shows, it is enough to work with seesaw quantum circuits, incurring only in a polynomial penalty on the number of gates. Recall that $|C|$ denotes the number of gates of quantum circuit C.

Lemma 1 There is a polynomial $(\xi_1, \xi_2) \mapsto P(\xi_1, \xi_2)$ such that, for any quantum circuit C on n qubits with gates in $\mathbb{U}$, there is a seesaw quantum circuit C' with at most $P(n, |C|)$ gates that computes the same unitary operator.

Proof: Let C be a quantum circuit operating in n qubits and assume that $C = U_{|C|} \circ \cdots \circ U_1$ with each $U_i \in \mathbb{U}$.

Observe that, since in a seesaw circuit the first and last gate are applied to the first qubit, the composition of two seesaw circuits is still a seesaw circuit. Thus, we start by showing that each gate of the circuit can be emulated by a seesaw circuit incurring in a polynomial cost on the number of qubits. The proof proceeds by case analysis.

1. A one-qubit gate U applied to qubit j can be emulated by the following seesaw circuit with at most $2n - 1$ gates:

$$\begin{array}{ll} \mathsf{Id}[1] \circ \cdots \circ \mathsf{Id}[j-1] \circ & \text{(get back to the top qubit)} \\ U[j] \circ & \text{(apply the gate)} \\ \mathsf{Id}[j-1] \circ \cdots \circ \mathsf{Id}[1]. & \text{(reach the target qubit)} \end{array}$$

2. The gate Sw applied to qubits j and l with $j < l$ can be emulated by the following seesaw circuit with at most $2n - 3$ gates:

   $\mathsf{Id}[1] \circ \cdots \circ \mathsf{Id}[j-1] \circ$ (get back to the top qubit)

   $\mathsf{Sw}[j, j+1] \circ \cdots \circ \mathsf{Sw}[l-2, l-1] \circ$ (move the second target qubit to position j pushing down the intermediate qubits)

   $\mathsf{Sw}[l-1, l] \circ$ (swap the target qubits)

   $\mathsf{Sw}[l-2, l-1] \circ \cdots \circ \mathsf{Sw}[j, j+1] \circ$ (move the first target qubit to position $l-1$ pushing up the intermediate qubits)

   $\mathsf{Id}[j-1] \circ \cdots \circ \mathsf{Id}[1]$ (reach the first target qubit).

3. The gate c-Not applied to qubit l controlled by qubit j with $j < l$ can be emulated by the following seesaw circuit with at most $2n - 3$ gates:

   $\mathsf{Id}[1] \circ \cdots \circ \mathsf{Id}[j-1] \circ$ (get back to the top qubit)

   $\mathsf{Sw}[j, j+1] \circ \cdots \circ \mathsf{Sw}[l-2, l-1] \circ$ (move the control qubit back to position j pushing down the intermediate qubits)

   $\mathsf{c\text{-}Not}[l-1, l] \circ$ (apply c-Not as required)

   $\mathsf{Sw}[l-2, l-1] \circ \cdots \circ \mathsf{Sw}[j, j+1] \circ$ (move the control qubit to position $l-1$ pushing up the intermediate qubits)

   $\mathsf{Id}[j-1] \circ \cdots \circ \mathsf{Id}[1]$ (reach the control qubit).

4. The gate c-Not applied to qubit j controlled by qubit l with $j < l$ can

be emulated by the following seesaw circuit with at most $2n-1$ gates:

$\mathsf{Id}[1] \circ \cdots \circ \mathsf{Id}[j-1] \circ$	(get back to the top qubit)
$\mathsf{Sw}[j, j+1] \circ \cdots \circ \mathsf{Sw}[l-2, l-1] \circ$	(move the target qubit back to position j pushing down the intermediate qubits)
$\mathsf{Sw}[l-1, l] \circ$	(swap the control qubit with the target qubit)
$\mathsf{c\text{-}Not}[l-1, l] \circ$	(apply c-Not as required)
$\mathsf{Sw}[l-1, l] \circ$	(swap the target qubit with the control qubit)
$\mathsf{Sw}[l-2, l-1] \circ \cdots \circ \mathsf{Sw}[j, j+1] \circ$	(move the target qubit to position $l-1$ pushing up the intermediate qubits)
$\mathsf{Id}[j-1] \circ \cdots \circ \mathsf{Id}[1]$	(reach the target qubit).

To the simulate the circuit C we compose sequentially the seesaw circuits that emulate each gate of C. Therefore, such composite seesaw circuit has at most $|C|(2n-1)$ gates and emulates C. QED

It pays off to introduce the notion of seesaw circuits since it is quite easy to emulate them by dcq Turing machines.

Lemma 2 There is a polynomial $\xi \mapsto P(\xi)$ such that, for any seesaw quantum circuit C on n qubits with gates in $\mathbb{U}$, there is a dcq Turing machine such that, for every unit vector $|\psi\rangle \in \mathsf{H}^n$, the computation starting from $(\varepsilon, |\psi\rangle)$ produces the quantum output $C|\psi\rangle$ in at most $P(|C|)$ steps.

Proof: Let $C = U_{|C|} \circ \cdots \circ U_1$ be a seesaw circuit acting on n qubits. Consider the dcq Turing machine (Q, δ) where $Q = \{\mathsf{q_s}, q_1, \ldots, q_k, \mathsf{q_h}\}$ and δ is as follows:

$$\delta(\mathsf{q_s}, a) = (\mathsf{Id}, \mathsf{R}, a, \mathsf{N}, q_1)$$
$$\delta(q_i, a) = (U_i, d_i, a, \mathsf{N}, q_{i+1}) \text{ for each } i = 1, \ldots, |C|-1$$
$$\delta(q_{|C|}, a) = (U_{|C|}, \mathsf{L}, a, \mathsf{N}, \mathsf{q_h}),$$

for each $a \in \{\square, 0, 1\}$, with

$$d_i = \begin{cases} \mathsf{L} & \text{if the topmost qubit acted upon by } U_{i+1} \text{ is above} \\ & \quad \text{the topmost qubit acted upon by } U_i \\ \mathsf{N} & \text{if the topmost qubit acted upon by } U_{i+1} \text{ is} \\ & \quad \text{the topmost qubit acted upon by } U_i \\ \mathsf{R} & \text{if the topmost qubit acted upon by } U_{i+1} \text{ is below} \\ & \quad \text{the topmost qubit acted upon by } U_i \end{cases}$$

for each $i = 1, \ldots, |C| - 1$. Clearly, this machine emulates seesaw circuit C and, for each $|\psi\rangle \in \mathsf{H}^n$, produces the required quantum output $C|\psi\rangle$ in precisely $|C| + 2$ steps. QED

The following result is an immediate consequence of the two lemmas above.

Theorem 3 (Fibered emulation of quantum circuits)
There is a polynomial $(\xi_1, \xi_2) \mapsto P(\xi_1, \xi_2)$ such that, for any quantum circuit C on n qubits with gates in $\mathbb{U}$, there is a dcq Turing machine such that, for every unit vector $|\psi\rangle \in \mathsf{H}^n$, the computation starting from $(\varepsilon, |\psi\rangle)$ produces the quantum output $C|\psi\rangle$ in at most $P(n, |C|)$ steps.

In fact, it is possible to establish a stronger result showing that a single dcq Turing machine is enough for emulating a uniform family of quantum circuits. We establish the stronger result only in Section 5 because its proof relies on techniques used in Section 4 for building universal machines. Observe that uniform emulation is necessary for verifying that problems decidable by quantum circuits are decidable by dcq Turing machines, a fact that is also established in Section 5.

With Theorem 3 in hand, we are ready to establish the equivalence between the two models of quantum computation only with respect to unitary transformations.

Theorem 4 For each $n \in \mathbb{N}^+$, a unitary transformation $U : \mathsf{H}^n \to \mathsf{H}^n$ is qc computable if and only if it is dcq computable.

Proof:
($\Rightarrow$) If U is qc computable then, by definition, there exists a quantum circuit with gates in $\{\mathsf{H}, \mathsf{S}, \pi/8, \mathsf{c\text{-}Not}\}$ (and, so, *a fortiori* with gates in $\mathbb{U}$) that computes it. Hence, by Theorem 3, U is computed by a dcq Turing machine.

($\Leftarrow$) Assume that there exists a dcq Turing machine that, for every $|\psi\rangle \in \mathsf{H}^n$, starting on input $(\varepsilon, |\psi\rangle)$ produces the output $U(|\psi\rangle)$ in a finite number of steps. Observe that the sequence of unitary operators in $\mathbb{U}$ applied by any dcq Turing machine to qubits in the quantum tape does not depend on the $|\psi\rangle$ at hand. Clearly, the finite sequence of operators applied by the machine computing U defines a (seesaw) quantum circuit that computes U. QED

4 Universal machines

The existence of universal dcq Turing machines is proved below as a corollary of the following result – the counterpart for dcq Turing machines of the *s-m-n* theorem. Both are established with polynomial bounds as required for the subsequent development of complexity theory.

Theorem 5 (Polynomial translatability)
There is a dcq Turing machine T such that, for any dcq Turing machine $M = (Q, \delta)$, there is a map

$$s : \{0,1\}^* \to \{0,1\}^*$$

which is classically dcq computable in linear time and fulfills the following conditions:

$$\begin{cases} \forall\, p, x \in \{0,1\}^*, |\psi\rangle \in \mathsf{H}^n, n \in \mathbb{N}^+ & M(p\Box x, |\psi\rangle) = T(s(p)\Box x, |\psi\rangle) \\ \exists\, c \in \mathbb{N} \;\; \forall\, p \in \{0,1\}^* & |s(p)| \leq |p| + c. \end{cases}$$

Moreover, there is a polynomial $(\xi_1, \xi_2, \xi_3) \mapsto P(\xi_1, \xi_2, \xi_3)$ such that if M starting from $(p\Box x, |\psi\rangle)$ produces the output in k steps then T produces the same output in at most

$$P(|p| + |x|, |Q|, k)$$

steps when starting from $(s(p)\Box x, |\psi\rangle)$.

Proof: Without any of loss of generality assume that

$$Q = \{q_0, q_1, \ldots, q_\nu, q_{\nu+1}\}$$

with $\mathsf{q_s} = q_0$ and $\mathsf{q_h} = q_{\nu+1}$. Hence, $|Q| = \nu + 2$. Consider the map

$$s = p \mapsto \underline{\delta}\, 111\, p : \{0,1\}^* \to \{0,1\}^*$$

where $\underline{\delta}$ encodes δ as follows.

Encoding of δ

$\underline{\delta(q_0,0)}\,\underline{\delta(q_0,1)}\,\underline{\delta(q_0,\square)}\ldots\underline{\delta(q_i,0)}\,\underline{\delta(q_i,1)}\,\underline{\delta(q_i,\square)}\ldots\underline{\delta(q_\nu,0)}\,\underline{\delta(q_\nu,1)}\,\underline{\delta(q_\nu,\square)}$

with each

$$\underline{\delta(q,a)} = \underline{U}\,\underline{d}\,\underline{a'}\,\underline{d'}\,\underline{q'} \in \{0,1\}^*$$

where

$$\underline{U} = \begin{cases} 000 & \text{if} \quad U = \mathsf{Id} \\ 001 & \text{if} \quad U = \mathsf{H} \\ 010 & \text{if} \quad U = \mathsf{S} \\ 011 & \text{if} \quad U = \pi/\mathsf{8} \\ 100 & \text{if} \quad U = \mathsf{Sw} \\ 101 & \text{if} \quad U = \mathsf{c\text{-}Not} \end{cases}$$

$$\underline{d} = \begin{cases} 00 & \text{if} \quad d = \mathsf{L} \\ 01 & \text{if} \quad d = \mathsf{N} \\ 11 & \text{if} \quad d = \mathsf{R} \end{cases}$$

$$\underline{a'} = \begin{cases} 00 & \text{if} \quad a' = 0 \\ 11 & \text{if} \quad a' = 1 \\ 10 & \text{if} \quad a' = \square \end{cases}$$

$$\underline{d'} = \begin{cases} 00 & \text{if} \quad d' = \mathsf{L} \\ 01 & \text{if} \quad d' = \mathsf{N} \\ 11 & \text{if} \quad d' = \mathsf{R} \end{cases}$$

$$\underline{q'} = 1^{j+1}00$$

assuming that $\delta(q,a) = (U,d,a',d',q')$ and $q' = q_j$.

Notice that one can identify in $s(p)$ the end of the encoding of $\underline{\delta}$ since each $\underline{\delta(q,a)}$ starts with $\underline{U}$ and the sequence 111 does not encode any gate. Clearly, as defined, s can be dcq computed in linear time and fulfills the conditions in the statement of the theorem by taking $c = |\underline{\delta}| + 3$.

Encoding of configurations

It is necessary to encode in the classical tape of T the current classical configuration of M (composed of the current contents of the classical tape, the current position of the classical head and the current state of the control automaton). There is no need to encode the quantum configuration of M

since in a dcq Turing machine it does not affect its transitions. In due course, when explaining how M computations are emulated by T computations we shall see how quantum configurations of T are made to follow those of M. The following notation becomes handy for describing classical configurations of dcq Turing machines.

We write

$$w \; \overset{q}{\overset{\triangledown}{a}} \; w'$$

for stating that the machine in hand is at state q, its classical head is over a tape cell containing symbol a, with the finite sequence w of symbols to the left of the head, with the finite sequence w' of symbols to the right of the head, and with the rest of the classical tape filled with blanks.

Before we describe how a classical configuration of M is encoded in T we need to introduce some notation. Recall that a symbol $a \in \{0, 1, \Box\}$ is encoded as

$$\underline{a} = \begin{cases} 00 & \text{if} \quad a = 0 \\ 11 & \text{if} \quad a = 1 \\ 10 & \text{if} \quad a = \Box. \end{cases}$$

We denote by $\underline{a}_1$ and $\underline{a}_2$ the first and second bit of $\underline{a}$, respectively. The reverse encoding of a is $\underline{\underline{a}} = \underline{a}_2\underline{a}_1$. Given a string $w = w_1 \dots w_m \in \{0, 1, \Box\}^*$, we denote its encoding by $\underline{w} = \underline{w_1} \dots \underline{w_m}$ and its reverse encoding by $\underline{\underline{w}} = \underline{\underline{w_m}} \dots \underline{\underline{w_1}}$.

The classical configuration

$$w \; \overset{q_i}{\overset{\triangledown}{a}} \; w'$$

of M should be encoded as the following classical configuration of T

$$\underline{w} \; \Box \; \underbrace{\underbrace{1 \dots 1}_{\nu+i-1} \; \Box \; \underbrace{1 \dots 1}_{i+1}}_{q_i} \; \overset{q'}{\overset{\triangledown}{\Box}} \; \underline{\delta} \; 111 \; \underline{\underline{w'}} \; \underline{a}_2\underline{a}_1$$

where q' is a state of T representing the stage where the machine is able to start emulating a transition of M. As we shall see later, whenever a transition of M has just been emulated by T and the resulting state is not the halting state of M, T is at state q'.

Initial classical configuration

The initial classical configuration of M is

$$\overset{\overset{q_0}{\triangledown}}{\Box}\, p \,\Box\, x$$

and, moreover, the initial classical configuration of T is

$$\overset{\overset{q_0'}{\triangledown}}{\Box}\, \underbrace{\underline{\delta}\ 111\ p}_{s(p)} \,\Box\, x,$$

where q_0' is the initial state of T. The objective of this stage is to change the initial classical configuration of T to the encoding of the initial configuration of M, as described before, that is:

$$\underbrace{1\ldots 1 \,\Box\, 1}_{\text{encoding of } q_0}\ \overset{\overset{q'}{\triangledown}}{\Box}\ \underline{\delta}\ 111\ \underline{\underline{\Box p \Box x}}.$$

Writing the encoding of q_0 can be done straightforwardly in $O(k)$ steps. It remains to describe how to encode $\Box p \Box x$ in reverse order within $O((|p| + |x|)^2)$ steps, keeping $\underline{\delta}$ 111 unchanged:

1. Encoding of x Recall that $x = x_1 \ldots x_m$ has no blanks, and therefore $\underline{x} = x_1 x_1 \ldots x_m x_m$. The idea is to shift $x_2 \ldots x_m$ to the right, duplicate x_1 in the vacated cell and then iterate this process to $x_2 \ldots x_m$. First, the head moves on top of x_2 and copies the contents of $x_2 \ldots x_m$ one cell to the right, leaving the original cell of x_2 with a blank. Then the head moves back to x_1 and copies its contents to the cell on its right. The process is iterated for $x_2 \ldots x_m$ until the last symbol of x is reached. Since shifting to the right the contents of m cells, leaving the first one blank, can be done with a linear number of steps in m, this operation takes a quadratic number of steps on the size of x.

2. Encoding the $\Box$ **in** $p\Box x$ First, the encoding of x is shifted one cell to the right and then, the head is moved back to the top of the first two blanks separating p and $\underline{x}$. Finally, the head replaces the two blanks by 10 and it is parked in the 1. Note that this can be done with a linear number of steps on the size of x, and moreover, the encoding of $\Box x$ has no blanks.

3. Encoding of p Let l be the size of p. The encoding of p is similar to the encoding of x. First the encoding of $\square x$ is shifted three cells to the right. Then the head of the machine is moved to the beginning of p. Notice that the machine can identify it as the first cell on the right of $\underline{\delta}\ 111$. Next, p is shifted one cell to the right (which leaves two blanks before $\square x$) and the head of the machine is moved to the cell containing p_l. The machine copies p_l to the two cell immediately on its right and writes $\square$ in the original cell. After these steps, the content of the classical tape is:

$$\underline{\delta}\ 111\ \square\ p_1 \ldots p_{l-1}\ \overset{\overset{q'}{\triangledown}}{\square}\ \underline{p_l \square x}.$$

Next $\underline{p_l \square x}$ is shifted one cell to the right and the process of writing the encoding of p_l in the tape is repeated for p_{l-1}, p_{l-2}, ... until p_1. The end of this construction is reached whenever the symbol $\square$ is placed after $\underline{\delta}\ 111$ is read. Finally, $\underline{p \ \square \ x}$ is shifted two cells to the left.

4. Reversing the encoding of $p\square x$ Assume that $\underline{p\square x} = y_1 \ldots y_m$ where $m = |\underline{p}| + |\underline{x}| + 2$ is the contents of the cells containing the encoding of $p\square x$. The objective is to replace this contents by $y_m \ldots y_1$. First, the cell containing y_1 is replaced by a blank and y_1 is copied to the right cell of y_m. Second, the sequence $y_2 \ldots y_m$ is shifted one cell to the left. This process is repeated with $y_2 \ldots y_m$ in such a way that y_2 is copied to the left of y_1 and until the contents of the tape is $\square y_m \ldots y_1$. Finally, the blank symbol is removed when $y_m \ldots y_1$ is shifted one cell to the left. Observe that the operations leading to $\square y_m \ldots y_1$, take $O(m^2)$ steps. Moreover, the final shift is linear, and so the overall stage takes a quadratic number of steps.

5. Placing the reverse encoding of a blank at the right end The head is moved to the right until the first blank is found. Then, the head writes a 0 and moves one cell to the right, where it writes a 1. Finally, the head is moved to left until the first blank is found.

It is straightforward to check that the overall cost of these operations is quadratic on $|p| + |x|$ and that the five stages above require just a constant number of states in T (that is, the number of states does not depend on p and x).

Transition emulation

We describe the steps needed to emulate in T one step by M. Assume that the transition to be emulated is $\delta(q_i, a) = (U, d, a', d', q_j)$ and that T is at the following classical configuration:

$$\underline{w}\ \underbrace{\underbrace{\underline{1\cdots1}}_{\nu-i-1}\ \square\ \underbrace{\underline{1\cdots1}}_{i+1}}_{\text{encoding of } q_i}\ \overset{q'}{\overset{\triangledown}{\square}}\ \underline{\delta}\ 111\ \underline{\underline{w'}}\ \underline{a_2}\ \underline{a_1}.$$

The objective is to set T at the following classical configuration

$$\underline{w''}\ \underbrace{\underbrace{\underline{1\cdots1}}_{\nu-j-1}\ \square\ \underbrace{\underline{1\cdots1}}_{j+1}}_{\text{encoding of } q_j}\ \overset{q'}{\overset{\triangledown}{\square}}\ \underline{\delta}\ 111\ \underline{\underline{w'''}}.$$

where, depending on the move of the classical head, three cases may occur:

- if $d' = N$ then $w = w''$ and $w''' = a'w$;
- if $d' = L$ then $w'' = w_1 \ldots w_{|w|-1}$ and $w''' = w_{|w|}a'w'$;
- if $d' = R$ then $w'' = wa'$ and $w''' = w'_2 \ldots w'_{|w'|}$.

Next, we detail how T performs the emulation of $\delta(q_i, a)$.

1. Identifying the value a The head of the classical tape of T is moved to $\underline{a}_1$ which is the rightmost cell that is not blank. The head reads the contents of that cell and the contents of the cell on its left, which has $\underline{a}_2$, and goes to a different state of T depending on the value a. The cost of this operation is linear in the the number of states of M and on the space used by M.

2. Parking the head at the encoding of $\delta(q_i, a)$ in $\underline{\delta}$. First the head is moved to the cell containing the rightmost 1 of the encoding of q_i. Notice that, such encoding has at least one 1 to the right of the blank. Since the head starts from position $\underline{a}_2$, such 1 is on the left to the first blank that the head finds while reading the classical tape from right to the left. So, this operation is at most linear in the size of the space used by M. Recall that δ is encoded as

$$\underline{\delta} = \underline{\delta(q_0,0)\delta(q_0,1)\delta(q_0,\Box)}\ldots\underline{\delta(q_i,0)\delta(q_i,1)\delta(q_i,\Box)}\ldots\underline{\delta(q_\nu,0)\delta(q_\nu,1)\delta(q_\nu,\Box)}.$$

Moreover, each $\underline{\delta(q_i,a)}$ ends with 00 and starts with nine cells corresponding to $\underline{U}\cdot\underline{d}\cdot\underline{a'}\cdot\underline{d'}$ and a sequence of 1's, encoding the resulting state of that transition. This stage consists in a loop with progress variable, say r, starting from $r=1$ until $r=i+1$. The goal of the loop is to replace the r rightmost 1's of encoding of q_i by 0's while the 00, at the end of $\underline{\delta(q_{r-1},\Box)}$, are replaced by $\Box\Box$. The end of the loop $r=i+1$, is detected when a blank symbol is read in the encoding of q_i. For each value of r we keep only a pair of $\Box\Box$ in $\underline{\delta}$: those at the end of $\underline{\delta(q_{r-1},\Box)}$. When $r=i+1$, the encoding of $\delta(q_i,\Box)$ is marked in δ with $\Box\Box$, and so, it remains to park the head in the first cell of the encoding of $\delta(q_i,a)$. This movement can be achieved taking into account the symbol a read in the previous stage.

Observe that all the operations performed in this stage depend linearly on the space used by M (on the right of its classical head) and quadratically on the number of states of M.

3. Identifying and applying U Using the first three cells of $\underline{\delta(q_i,a)}$, the machine T identifies the unitary transformation and applies it to its own quantum tape.

4. Performing the d-move of the quantum head Using the fourth and fifth cells of $\underline{\delta(q_i,a)}$, T identifies the movement of the quantum head and operates accordingly on its own quantum head.

5. Identifying and writting a' Using the sixth and seventh cells of $\underline{\delta(q_i,a)}$, T identifies the encoding of the symbol a' to be written under $\underline{a_2}\underline{a_1}$. The encoding of a' in $\underline{\delta(q_i,a)}$ is marked with two blanks and $\underline{a'}$ is copied in reversed order to $\underline{a_2}\underline{a_1}$, which are the two rightmost non-blank cells. After completing the last operation, the head returns to the original position and restores $\underline{a'}$ in $\underline{\delta(q_i,a)}$.

Notice that the operations of this stage can be done in a linear number of steps on the space used by M the input and linearly in the number of states.

6. Performing the d'-move of the classical head The ninth and tenth cells of $\underline{\delta(q_i,a)}$ store the movement of the classical head. If $d'=N$ nothing has to be done. W.l.o.g. assume $d' = R$. First, the encoding of d' is marked with two blanks. Then the rightmost non-blank cells have to be copied (in reverse order) to the left of the leftmost non-blank cells. Clearly the rightmost non-blank cells have to be replaced by two blanks if $|w'| > 0$, and have to be replaced by 01 (the reverse encoding of a blank) if $|w'| = 0$. Mutatis mutandis if $d' = L$.

This stage can be done in a linear number of steps on the space in the classical tape used by M.

7. Updating the emulated state to q_j Assume that q_j is not the halting state and recall that q_j is encoded as $1^{j+1}00$ at the rightmost part of $\underline{\delta(q_i, a)}$. The idea is to update the emulated state q_i to q_j by replacing each 1 in $1^{j+1}00$ at $\underline{\delta(q_i, a)}$ by a $\square$ while updating the cells used to encode the current state of $\overline{M}$. Given Stage 2, the cells used to encode q_i contain $1^{\nu-i-1}\square 0^{i+1}$. If $j \leq i$ then we replace $j+1$ rightmost 0's by 1's and then place a $\square$ left to them. If $j > i$, then the $i+1$ rightmost 0's are replaced by 1's and after, the blank has to be carried to the left while being replaced by a 1, until there are $(j+1)$ 1's. This process ends when all 1's in $1^{j+1}00$ have been replaced by blanks. After the cells encoding the emulated state are updated, the encoding of q_j in $\underline{\delta(q_i, a)}$ is restored, by replacing the blanks by 1's.

This stage does not depend on the input of M, but only quadratically in the number of states of M.

If q_j is the halting state, we have to restore the contents of the classical tape to $wa'w'$, with the head positioned over a'. This corresponds to inverting the process used to prepare the initial configuration, erasing the encoding of q_i and δ. Such stage can be done in a number of steps quadratic to the space used by M and linearly in the number of states of M.

Efficiency of the emulation

Since the space used by M is bounded by k, the overall emulation is polynomial (in fact quadratic) on $|p| + |x|$, ν and k. QED

A machine T fulfilling the conditions of Theorem 5 is said to enjoy the *s-m-n* property. Any such machine is universal as shown in the next result.

Theorem 6 (Polynomial universality)
Let T be a dcq Turing machine enjoying the *s-m-n* property. Then, for any

dcq Turing machine $M = (Q, \delta)$, there is $p \in \{0,1\}^*$ such that

$$M(x, |\psi\rangle) = T(p\Box x, |\psi\rangle) \quad \forall\, x \in \{0,1\}^*, |\psi\rangle \in \mathsf{H}^n, n \in \mathbb{N}^+.$$

Moreover, there is a polynomial $(\xi_1, \xi_2, \xi_3) \mapsto P(\xi_1, \xi_2, \xi_3)$ such that if M, when starting from $(x, |\psi\rangle)$, produces the output in k steps then T produces the same output in at most

$$P(|x|, |Q|, k)$$

steps when starting from $(p\Box x, |\psi\rangle)$.

Proof: Consider M' such that $M'(\varepsilon\Box x, |\psi\rangle) = M(x, |\psi\rangle)$. By applying Theorem 5 to M' and choosing $p = s(\varepsilon)$ the result follows. QED

5 Recovering the BQP class

Using the approach adopted in the proof of Theorem 5, we are ready to establish the uniform variant of Theorem 3 and to prove that problems decidable by quantum circuits are also decidable by dcq Turing machines. Both results are obtained with polynomial bounds, as required for recovering the BQP class using dcq Turing machines.

Recall that a set $X \subseteq \{0,1\}^*$ is said to be *quantum-circuit decidable* (in short, *qc decidable*) if there is a computable sequence of quantum circuits

$$k \mapsto C_k$$

such that, for each k, we have:

1. The circuit C_k operates in H^{n_k} with $n_k \geq k$.
2. The following holds for each x with $k \geq |x|$. The circuit C_k when presented with input $|x\rangle$ (padded to the top with $|0\rangle$'s as needed) produces some output $|\varphi\rangle \in \mathsf{H}^{n_k}$ such that

$$\begin{cases} \mathsf{Prob}\,(\mathsf{Proj}_1|\varphi\rangle = 1) > 2/3 & \text{if } \ x \in X \\ \mathsf{Prob}\,(\mathsf{Proj}_1|\varphi\rangle = 0) > 2/3 & \text{if } \ x \notin X. \end{cases}$$

Theorem 7 (Uniform emulation of quantum circuits)
There is a polynomial $(\xi_1, \xi_2) \mapsto P(\xi_1, \xi_2)$ such that, for each computable sequence of quantum circuits $k \mapsto C_k$, there are a dcq Turing machine T and a classically dcq computable map $g : \{0,1\}^* \to \{0,1\}^*$ such that, for every $n \in \mathbb{N}^+$ and $|\psi\rangle \in \mathsf{H}^{n_k}$, the computation starting from $(g(1^k), |\psi\rangle)$ produces quantum output $C_k|\psi\rangle$ in at most $P(k, |C_k|)$ steps.

Proof: Assume some (efficient) description of quantum circuits in $\{0,1\}^*$. The computable sequence of quantum circuits $k \mapsto C_k$ is a computable map $c : \{0,1\}^* \to \{0,1\}^*$ such that $c(1^k)$ is the description of C_k. Given the constructive proof of Lemma 1, there is a polynomial-time map $w : \{0,1\}^* \to \{0,1\}^*$ that receives a description of a circuit C and outputs the description of a seesaw circuit equivalent to C. By the proof of Lemma 2, there is a polynomial-time map $t : \{0,1\}^* \to \{0,1\}^*$ that receives a description of a seesaw circuit C and outputs the encoding of a Turing machine that emulates it. We consider that the encoding of a machine $M = (Q, \delta)$ is

$$1^{|Q|-1}0\underline{\delta}111$$

where $\underline{\delta}$ is the encoding of δ described in the proof of Theorem 5.

Consider $g = t \circ w \circ c$. Observe that, thanks to proof of Lemma 1 and Lemma 2, there is a polynomial $(\xi_1, \xi_2) \mapsto P'(\xi_1, \xi_2)$ such that the number of steps the Turing machine encoded by $t(w(c(1^k)))$ takes to emulate C_k is $P'(k, |C_k|)$.

It remains to present a Turing machine T that emulates, with polynomial overhead, the one encoded in $g(1^k)$ with no classical input. To this end, we adapt the Turing machine described in the proof of Theorem 5. As we shall see, we only need to modify the *initial classical configuration*.

Initial classical configuration The initial classical configuration of the machine M encoded by $g(1^k)$ is

$$\overset{\overset{q_0}{\triangledown}}{\square}$$

and, moreover, the initial classical configuration of T is

$$\overset{\overset{q_0'}{\triangledown}}{\square}\ \underbrace{1\ldots1}_{|Q|-1}\ 0\ \underline{\delta}\ 111.$$

The objective of this stage is to change the initial classical configuration of T to the encoding of the initial configuration of M, as described in the proof of Theorem 5, that is:

$$\underbrace{1\ldots1\,\square\,1}_{\text{encoding of } q_0}\ \overset{\overset{q'}{\triangledown}}{\square}\ \underline{\delta}\ 111\ \underline{\underline{\square}}.$$

This is achieved by replacing the first blank by a 1, and then moving the head to the right until a 0 is found. When a 0 is found, it is replaced by a blank and then the head moves two cells to the left, where it writes a blank. Next, the head moves to the right until it passes the rightmost 111, and writes the encoding of $\underline{\underline{\sqcup}}$. Finally, the head moves to the first blank found to the left, and changes the states of T to q'. Recall that q' is the state of T representing the stage where T is able to start emulating a transition of M.

Transition emulation Precisely the same as that in the proof of Theorem 5.

Observe that T emulates the machine encoded by $g(1^k)$ with polynomial overhead, and, moreover, the machine encoded by $g(1^k)$ requires $P'(k, |C_k|)$ steps. Thus, there is a polynomial $(\xi_1, \xi_2) \mapsto P(\xi_1, \xi_2)$ such that the computation of T starting from input $(g(1^k), |\psi\rangle)$ with $|\psi\rangle$ in H^{n_k} produces quantum output $C_k|\psi\rangle$ in at most $P(k, |C_k|)$ steps. QED

With the uniform emulation of quantum circuits by dcq Turing machines in hand, it is straightforward to show that dcq decidability is entailed by qc decidability. In fact, the two concepts coincide. The converse capitalizes on the fact that quantum circuit decidability coincides with classical decidability.

Theorem 8 A set is qc decidable if and only if it is dcq decidable.

Proof:

($\Rightarrow$) Assume that X is qc decidable. The goal is to build a dcq Turing machine T that, upon receiving $(x, |\varepsilon\rangle)$, produces quantum output $|\varphi\rangle$ such that

$$\begin{cases} \mathsf{Prob}\,(\mathsf{Proj}_1|\varphi\rangle = 1) > 2/3 & \text{if } \; x \in X \\ \mathsf{Prob}\,(\mathsf{Proj}_1|\varphi\rangle = 0) > 2/3 & \text{if } \; x \notin X. \end{cases}$$

Since X is qc decidable, there is a computable map $c : \{0,1\}^* \to \{0,1\}^*$ such that $c(1^k)$ is the description of the quantum circuit C_k operating in n_k qubits. Consider the map $g = t \circ w \circ c$ presented in the proof of Theorem 7. The Turing machine T works as follows. First, it writes $x\Box|x|$ in the classical tape. Next, it writes $|x\rangle$ in the quantum tape using n_k qubits (eventually, padded to the top with $|0\rangle$'s as needed). Next, it emulates $g(|x|)$ by mimicking the Turing machine presented in the proof of Theorem 7. This dcq Turing machine will decide precisely the same set as that of the uniform family of quantum circuits.

($\Leftarrow$) The sketch of the proof is as follows. Start by noticing that qc decidability coincides with Turing decidability [13]. Thus, quantum circuits accessing an oracle for a computable map only decide Turing decidable problems. Let (Q, δ) be a dcq Turing machine that decides X. Then, (Q, δ) starting from $(x, |\varepsilon\rangle)$, produces a quantum output $|\varphi\rangle$ for which the outcome of the measurement of its first qubit is used to decide whether $x \in X$. For each $k \in \mathbb{N}$, let o_k be the highest dimension of a quantum output $|\varphi\rangle$ produced by (Q, δ) for some input $(x, |\varepsilon\rangle)$ with $|x| \leq k$. Observe that (Q, δ), starting from $(x, |\varepsilon\rangle)$, applies to the quantum tape a sequence $(U_1(x) \dots, U_{m_x}(x))$ of primitive unitary operators, where each $U_i(x)$ is applied to the j_i-th qubit of the quantum tape with $j_i \in \{1, \dots, o_k\}$. By definition of a dqc Turing machine, if the primitive gate is binary, only adjacent qubits are modified. In particular, if U_i is a Sw gate, we assume that the j_i qubit is the topmost one; if U_i is a $\mathsf{c\text{-}Not}$ gate, the topmost qubit is the control qubit, and we assume that j_i is the index of that control qubit.

Let $G(x)$ be some encoding of the sequence

$$((U_1(x), j_1), \dots, (U_{m_x}(x), j_{m_x}))$$

that unambiguously define the sequence of applications such that

$$G(x) = G(U_1(x), j_1) \dots G(U_{m_x}(x), j_{m_x}).$$

Clearly, $x \mapsto G(x)$ is decidable, and moreover, $|G(U_i(x), j_i)| \in O(\log(o_k))$. Furthermore, let $n_k = k + o_k + \max_{|x| \leq k} |G(x)|$.

We are now able to describe the uniform family of quantum circuits $k \mapsto C_k$ that emulate (Q, δ) such that each C_k operates in H^{n_k} and it can access an oracle for $x \mapsto G(x)$. The circuit starts with input $|0\rangle^{\otimes n_k - k}|x\rangle$ and then, calls the oracle, resulting in the state $|0\rangle^{\otimes o_k}|G(x)\rangle|x\rangle$. Finally a universal circuit reads $G(x)$ and applies sequentially the primitive unitary operators $U_i(x)$ to the j_i-th topmost qubit. Since the size of $G(x)$ is upper-bounded for $|x| \leq k$, such universal circuit exists and we sketch it briefly. Observe that such circuit can be built so that each pair $(U_i(x), j_i)$ of the sequence is emulated with a linear number of gates in n_k. Indeed, this emulation can be achieved by making a sequence of control operations over the $O(\log(o_k))$ qubits that encode $(U_i(x), j_i)$, where each control operation takes into account one of the six primitive quantum gates applied to one of the o_k topmost qubits. Upon applying the previously presented universal circuit, the n_k qubits of the circuit are in state $|\varphi\rangle|G(x)\rangle|x\rangle$. So, by measuring the first qubit we obtain precisely the same distribution as the output by (Q, δ) for input $(x, |\varepsilon\rangle)$. QED

Taking into account that the dcq Turing machine presented in $(\Rightarrow)$ of the proof above emulates the given uniform family of quantum circuits with a polynomial-time overhead, it is immediate to show that $\mathsf{BQP} \subset \mathsf{dcBQP}$. The converse relies on the fact that the class BQP is closed for the consultation of oracles for polynomial-time maps.

Theorem 9 $\mathsf{dcBQP} = \mathsf{BQP}$

Proof:

$(\supset)$ To obtain that $\mathsf{dcBQP} \supset \mathsf{BQP}$ it is enough to note that the Turing machine presented in $(\Rightarrow)$ of the proof of Theorem 8 emulates a uniform family of quantum circuits with polynomial-time overhead.

$(\subset)$ To show that $\mathsf{dcBQP} \subset \mathsf{BQP}$ we use the fact that $\mathsf{BQP}^{\mathsf{P}} = \mathsf{BQP}$ as proved in [2]. Let (Q, δ) be a dcq Turing machine that decides X in polynomial time, then the map $x \mapsto G(x)$ used in $(\Leftarrow)$ of the proof of Theorem 8 for (Q, δ) can be computed in polynomial time. The uniform family of circuits $k \mapsto C_k$ emulating (Q, δ) is precisely the same as that in $(\Leftarrow)$ of the proof of Theorem 8, using an oracle for $x \mapsto G(x)$. We only need to observe that the size of $G(x)$ is polynomial in k and so, the universal quantum circuit has a polynomial number of gates in k, since each primitive quantum gate applied by (Q, δ) can be emulated in this universal quantum circuit with a linear number of gates in n_k. Since $x \mapsto G(x)$ is a polynomial-time map, $X \in \mathsf{BQP}^{\mathsf{P}} = \mathsf{BQP}$. QED

6 Quantum Kolmogorov complexity

Kolmogorov complexity aims at measuring the quantity of information contained in an object. When the object is classical, i.e. a bit string, one can measure that quantity via the length of the shortest program that computes the string in a given universal Turing machine. In consequence of Kleene's *s-m-n* theorem, this classical notion of Kolmogorov complexity is robust in the sense that it does depend (up to a constant) on the adopted universal Turing machine. For a recent survey see [1].

When the object is a quantum state, it is not yet clear how one should proceed. Several definitions of quantum Kolmogorov complexity have been reported in the literature with different approaches and motivations.

In [19], the Kolmogorov complexity of a pure quantum state $|\varphi\rangle$ is calculated as the minimum value of the sum of (i) the length of a classical program computing (in the chosen quantum Turing machine) an approximation $|\psi\rangle$ of $|\varphi\rangle$, and (ii) the negative log-fidelity of the approximation of $|\psi\rangle$ to $|\varphi\rangle$.

In [3, 12], the Kolmogorov complexity of a qubit string is defined as the length of the shortest quantum bit string that given to a given quantum universal Turing machine produces the qubit string with high fidelity. This notion has a similar flavor to Vitányi's approach, but, rather than considering only classical programs, it considers the possibility of quantum ones.

In [8], using the concept of universal semi-density matrix μ, a generalization of the notion of universal semimeasure, two variants of Kolmogorov complexity of a pure quantum state $|\varphi\rangle$ are defined ($\underline{H}(|\varphi\rangle) = -\langle\varphi|\log\mu|\varphi\rangle$ and $\overline{H}(|\varphi\rangle) = -\log\langle\varphi|\mu|\varphi\rangle$), compared and carried over to density matrices.

Later, in [11], the Kolmogorov complexity of a pure quantum state $|\varphi\rangle$ is defined as the length of the shortest description of a quantum circuit capable of producing such state (or an approximation of it).

Herein, we propose to define *plain quantum Kolmogorov complexity* using a dcq Turing machine T enjoying the *s-m-n* property. Recall that there are dcq Turing machines enjoying the *s-m-n* property, thanks to Theorem 5. More concretely,

$$\mathbf{C}(y, |\varphi\rangle/x, |\psi\rangle) = \min\{|p| : T(p\Box x, |\psi\rangle) = (y, |\varphi\rangle)\}$$

is defined to be the *conditional plain descriptive complexity* (for T) of the pair $(y, |\varphi\rangle)$ given the pair $(x, |\psi\rangle)$. In particular,

$$\mathbf{C}(|\varphi\rangle) = \mathbf{C}(\varepsilon, |\varphi\rangle/\varepsilon, |\varepsilon\rangle) = \min\{|p| : T(p\Box\varepsilon, |\varepsilon\rangle) = (\varepsilon, |\varphi\rangle)\}$$

is the *unconditional plain descriptive complexity* (for T) of $|\varphi\rangle$. Other particular cases can be of interest.

Our notion of quantum Kolmogorov complexity is robust – it does not depend on the choice of T as long as T enjoys the *s-m-n* property.

Indeed, given two dcq Turing machines T, T' enjoying the *s-m-n* property, let s be a map that translates T'-programs to T-programs and $c \in \mathbb{N}$ be such that $|s(p)| \leq |p| + c$ for every $p \in \{0,1\}^*$. Clearly, by Theorem 5, such

s and such c exist since T is assumed to enjoy the s-m-n property. Then,

$$\begin{array}{rcl}\mathbf{C}'(y,|\varphi\rangle/x,|\psi\rangle)+c & = & \min\{|p| : T'(p\Box x,|\psi\rangle)=(y,|\varphi\rangle)\}+c \\ & = & \min\{|p| : T(s(p)\Box x,|\psi\rangle)=(y,|\varphi\rangle)\}+c \\ & \geq & \min\{|s(p)| : T(s(p)\Box x,|\psi\rangle)=(y,|\varphi\rangle)\} \\ & \geq & \min\{|p| : T(p\Box x,|\psi\rangle)=(y,|\varphi\rangle)\} \\ & = & \mathbf{C}(y,|\varphi\rangle/x,|\psi\rangle).\end{array}$$

By a symmetric argument, since T' is also assumed to enjoy the s-m-n property, there is $c' \in \mathbb{N}$ such that:

$$\mathbf{C}(y,|\varphi\rangle/x,|\psi\rangle)+c' \geq \mathbf{C}'(y,|\varphi\rangle/x,|\psi\rangle).$$

That is, $\mathbf{C}(y,|\varphi\rangle/x,|\psi\rangle)$ and $\mathbf{C}'(y,|\varphi\rangle/x,|\psi\rangle)$ are identical up to a constant.

Our definition of quantum Kolmogorov complexity is plain since it is not prefix free. However, it is straightforward to introduce the notion of prefix-free dcq Turing machine and establish the existence of prefix-free dcq Turing machines enjoying the s-m-n property, for instance, by adding a write-only quantum tape to the classical prefix-free Turing machine used in [17]. Using such machines for defining Kolmogorov complexity will lead to a prefix-free notion.

For a comparison between plain and prefix-free classical Kolmogorov complexities see, for example, [1]. In the quantum case, the definition proposed in [19] is prefix free since it is based on a prefix-free variant of the Deutsch machine, while the definition in [3] is plain. The latter paper contains a preliminary comparison between the two approaches by capitalizing on results presented in [8].

Our notion of unconditional plain Kolmogorov complexity of a pure quantum state $|\varphi\rangle$ is very close to the one proposed in [11] which uses quantum circuits. Indeed, recall from Section 3 that dcq Turing machines work like seesaw circuits. Furthermore, there is $c \in \mathbb{N}$ such that for every seesaw circuit there is a dcq Turing machine that emulates it such that the length of the description of the machine coincides, up to c, with the length of the description of the circuit. Recall also that there is $d \in \mathbb{N}$ such that for every arbitrary quantum circuit C there is an equivalent seesaw circuit C' such that the length of the description of C' coincides, up to d, with the length of the description of C. Therefore,

$$\exists\, e \in \mathbb{N} \;\; \forall\, |\varphi\rangle \qquad \mathbf{C}(|\varphi\rangle) \leq \mathbf{C}_{\mathsf{MB}}(|\varphi\rangle)+e$$

where $\mathbf{C}_{\mathsf{MB}}(|\varphi\rangle)$ is the Kolomogorov complexity of $|\varphi\rangle$ according to [11]. Indeed, just take $e = c + d$. It is easy to see that the converse bounding

$$\exists\, e \in \mathbb{N}\ \ \forall\, |\varphi\rangle \quad \mathbf{C}_{\mathsf{MB}}(|\varphi\rangle) \leq \mathbf{C}(|\varphi\rangle) + e$$

also holds provided that $\mathbb{U}$ is adopted in the definition of $\mathbf{C}_{\mathsf{MB}}$. Observe that in [11] there is no robustness analysis concerning different choices of the set of primitive gates and no definition of conditional Kolmorogov complexity is given.

7 Outlook

By endowing a classical Turing machine with a quantum tape on which the machine can apply unitary operators but cannot make measurements, we were able to avoid the difficulties of the halting condition of the original quantum Turing machine, while keeping its advantages over circuits which follow from the existence of efficient universal machines. In particular, we established that deterministic-control quantum (dcq) Turing machines can efficiently emulate finitely generated quantum circuits, paving the way for recovering, for instance, the BQP complexity class. We also proved an efficient counterpart of the *s-m-n* theorem for dcq Turing machines and obtained as an immediate corollary the existence of efficient universal dcq Turing machines. The *s-m-n* result allowed the definition of a robust notion of quantum Kolmogorov complexity.

Notwithstanding the well known equivalence between quantum circuits and Deutsch machines [14, 15], the results we established in this paper show that a quantum transition table is not required for achieving similar equivalence results, at least in the Monte Carlo scenario.

The power of dcq Turing machines (that do not have access to the contents of the quantum tape during the computation) comes as no surprise since it is well known that, in a quantum computation, measurements can be delayed until the end, thanks to Schmidt's decomposition theorem.

Most of the work in quantum computation has been developed using quantum circuits. The availability of a viable Turing model of quantum computation – like the one proposed herein – opens the door to shifting the attention towards problems which are better addressed using machines.

For instance, one should look at interactive dcq Turing machines for applications in information security concerning universal composability [5]. In another direction, one should use dcq Turing machines for developing the theory and applications of prefix-free quantum Kolomogorov complexity, possibly with bounded resources (including space), and comparing these notions with physical measures of the complexity of quantum states, like fidelity [21]. In due course, one should extend our approach to dcq machines acting on density operators instead of pure quantum states. It seems also interesting to investigate the power of dcq Turing machines for computing Las Vegas algorithms, and to study space complexity classes obtained with dcq Turing machines and compare them with those obtained by Watrous in [20] using quantum Turing machines with probabilistic control.

Acknowledgments

This work was partially supported, under the PQDR (Probabilistic, Quantum and Differential Reasoning) initiative of SQIG at IT, by FCT and EU FEDER, namely via the FCT PEst-OE/EEI/LA0008/2013 project, as well as by the European Union's Seventh Framework Programme for Research (FP7), namely through project LANDAUER (GA 318287). The third author also acknowledges the FCT postdoc grant SFRH/BPD/76231/2011.

References

[1] G. Barmpalias. Algorithmic randomness and measures of complexity. *The Bulletin of Symbolic Logic*, 19(3):318–350, 2013.

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

[3] A. Berthiaume, W. Dam, and S. Laplante. Quantum Kolmogorov complexity. *Journal of Computer and System Sciences*, pages 201–221, 2001.

[4] P. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan. On universal and fault-tolerant quantum computing. In *Proceedings of FOCS*, pages 486–494. Society Press, 1999.

[5] R. Canetti, I. Damgård, S. Dziembowski, Y. Ishai, and T. Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Proceedings of EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 262–279, Berlin Heidelberg, 2001. Springer.

[6] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[7] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Royal Society of London Proceedings Series A*, 439:553–558, 1992.

[8] P. Gács. Quantum algorithmic entropy. *Journal of Physics A: Mathematical and General*, 34(35):6859, 2001.

[9] L. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of STOC*, pages 212–219. ACM, 1996.

[10] F. Hennie and R. Stearns. Two-tape simulation of multitape Turing machines. *Journal of ACM*, 13(4):533–546, 1966.

[11] C. Mora and H. Briegel. Algorithmic complexity and entanglement of quantum states. *Physical Review Letters*, 95:200503, 2005.

[12] M. Müller. Strongly universal quantum Turing machines and invariance of Kolmogorov complexity. *IEEE Transactions on Information Theory*, 54(2):763–780, 2008.

[13] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[14] H. Nishimura and M. Ozawa. Computational complexity of uniform quantum circuit families and quantum turing machines. *Theoretical Computer Science*, 276(1-2):147–181, 2002.

[15] H. Nishimura and M. Ozawa. Perfect computational equivalence between quantum turing machines and finitely generated uniform quantum circuit families. *Quantum Information Processing*, 8(1):13–24, 2009.

[16] S. Perdrix and P. Jorrand. Classically-controlled quantum computation. *Electronic Notes in Theoretical Computer Science*, 135(3):119–128, 2006.

[17] G. Rozenberg and A. Salomaa. *The Secret Number. An Exposition of Chaitin's Theory*, pages 175–215. World Scientific Publications Company, Singapore, 2007.

[18] P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41:303–332, 1999.

[19] P. Vitányi. Quantum Kolmogorov complexity based on classical descriptions. *IEEE Transactions on Information Theory*, 47(6):2464–2479, 2001.

[20] J. Watrous. On the complexity of simulating space-bounded quantum computations. *Computational Complexity*, 12(1/2):48–84, 2004.

[21] P. Zanardi and N. Paunkovic. Ground state overlap and quantum phase transitions. *Physical Review E*, 74(-):031123–031123, 2006.