International Journal of Quantum Information
© World Scientific Publishing Company

ABSTRACT QUANTUM COMPUTING MACHINES AND QUANTUM COMPUTATIONAL LOGICS

MARIA LUISA DALLA CHIARA
Dipartimento di Lettere e Filosofia,
Università di Firenze,
Via Bolognese 52, I-50139 Firenze, Italy.
dallachiara@unifi.it

ROBERTO GIUNTINI, GIUSEPPE SERGIOLI
Dipartimento di Pedagogia, Psicologia, Filosofia,
Università di Cagliari,
Via Is Mirrionis 1, I-09123 Cagliari, Italy.
giuntini@unica.it
giuseppe.sergioli@gmail.com

ROBERTO LEPORINI
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione,
Università di Bergamo,
viale Marconi 5, I-24044 Dalmine (BG), Italy.
roberto.leporini@unibg.it

Received Day Month Year
Revised Day Month Year

Classical and quantum parallelism are deeply different, although it is sometimes claimed that *quantum Turing machines* are nothing but special examples of *classical probabilistic machines* [10]. We introduce the concepts of *deterministic state machine*, *classical probabilistic state machine* and *quantum state machine*. On this basis we discuss the question: to what extent can quantum state machines be simulated by classical probabilistic state machines? Each state machine is devoted to a single task determined by its program. Real computers, however, behave differently, being able to solve different kinds of problems. This capacity can be modeled, in the quantum case, by the mathematical notion of *abstract quantum computing machine*, whose different programs determine different quantum state machines. The computations of abstract quantum computing machines can be linguistically described by the formulas of a particular form of quantum logic, termed *quantum computational logic.*

Keywords: Quantum computation; quantum computational logics.

1. Introduction

The abstract mathematical model for quantum computers has been often represented in terms of the notion of *quantum Turing machine*, the quantum counterpart of the classical notion of *Turing machine*. But what exactly are quantum Turing machines? So far, the literature has not provided a rigorous "institutional" concept of *quantum Turing machine*. Some definitions seem to be based on a kind of "imitation" of the classical definition of *Turing machine*, by referring to a *tape* (where the symbols are written) and to a *moving head* (which changes its position on the tape).[a] These concepts, however, seem to be hardly applicable to physical quantum computers. Both in the classical and in the quantum case, it is expedient to consider a more abstract concept: the notion of *state machine*, which neglects both tapes and moving heads. Every finite computational task realized in different computational models proposed in the literature can be simulated by a state machine.[b] In order to compare classical and quantum computational parallelism, we will analyze the concepts of (classical) *deterministic state machine*, (classical) *probabilistic state machine* and *quantum state machine*. On this basis we will discuss the question: to what extent can quantum state machines be simulated by probabilistic state machines?

Each state machine is devoted to a single task determined by its program. Real computers, however, behave differently, being able to solve different kinds of problems, which may be chosen by computer-users. In the quantum case, such concrete computation-situations can be modeled by the mathematical notion of *abstract quantum computing machine*, whose different programs determine different quantum state machines. We will see how quantum computations can be linguistically described by the formulas of a particular form of quantum logic, termed *quantum computational logic.*

2. Classical deterministic and probabilistic machines

We will first introduce the notion of *deterministic state machine*. On this basis, *probabilistic state machines* will be represented as stochastic variants of deterministic machines that are able to calculate different outputs with different probability-values.

Definition 1. *Deterministic state machine.*
A *deterministic state machine* is an abstract system **M** based on the following elements:

1. A finite set $\mathcal{S}$ of *internal states*, which contains an *initial state* s_{in} and includes a set of *halting states* $\mathcal{S}_{halt} = \{s_{halt_j} \,|\, j \in J\}$.[c]
2. A finite alphabet, which can be identified with the set $\{0, 1\}$ of the two classical bits. Any *register* represented by a bit-sequence $w = (x_1, \ldots, x_n)$ is a *word* (of length n). Any pair (s, w) consisting of an internal state s and of a word w

[a]See, for instance, Fouché et al.[8].
[b]See, for instance, Savage [11] and Gudder [9].
[c]As expected, at the beginning of a computation the machine is an initial state; while, at the end of the computation, the machine *stops* assuming a halting state.

represents a possible *configuration* of **M**, which is interpreted as follows: **M** is in the internal state s and w is the word written on an ideal tape.

3. A set of words that represent possible *word-inputs* for **M**.
4. A *program*, which is identified with a finite sequence $(R_0, \dots, R_t)$ of rules. Each R_i is a partial function that transforms configurations into configurations. We may have: $R_i = R_j$ with $i \neq j$. The number i, corresponding to the rule R_i, represents the i-th step of the program. The following conditions are required:

   4.1 The rule R_0 is defined for any configuration (s_0, w_0), where s_0 is the initial state s_{in} and w_0 is a possible word-input. We have: $R_0 : (s_0, w_0) \mapsto (s_1, w_1)$, where s_1 is different from the initial state and from all halting states (if $t \neq 0$).

   4.2 For any i $(0 < i < t)$, $R_i : (s_i, w_i) \mapsto (s_{i+1}, w_{i+1})$, where s_{i+1} is different from all $s_i, \dots, s_0$ and from all halting states.

   4.3 $R_t : (s_t, w_t) \mapsto (s_{t+1}, w_{t+1})$,
   where s_{t+1} is a halting state.

   Each configuration (s_{i+1}, w_{i+1}) represents the *output* for the step i and the *input* for the step $i + 1$.

Apparently, each deterministic state machine is devoted to a single task that is determined by its program. The concept of *computation* of a deterministic state machine can be then defined as follows.

Definition 2. *Computation of a deterministic state machine.*
A *computation* of a deterministic state machine **M** is a finite sequence of configurations $((s_0, w_0), \dots, (s_{t+1}, w_{t+1}))$, where:

1. w_0 is a possible word-input of **M**.
2. $s_0, \dots, s_{t+1}$ are different internal states of **M** such that: $s_0 = s_{in}$ and s_{t+1} is a halting state.
3. For any i $(0 \leq i \leq t)$, $(s_{i+1}, w_{i+1}) = R_i((s_i, w_i))$, where R_i is the i-th rule of the program.

The configurations (s_0, w_0) and (s_{t+1}, w_{t+1}) represent, respectively, the *input* and the *output* of the computation; while the words w_0 and w_{t+1} represent, respectively, the *word-input* and the *word-output* of the computation.

Let us now turn to the concept of *probabilistic state machine*. The only difference between deterministic and probabilistic state machines concerns the program, which may be stochastic in the case of a probabilistic state machine (**PM**). In such a case, instead of a sequence of rules, we will have a sequence $(Seq_0, \dots, Seq_t)$ of sequences of rules such that: $Seq_0 = (R_{0_1}, \dots, R_{0_r}), \dots, Seq_t = (R_{t_1}, \dots, R_{t_l})$. Each rule R_{i_j} (occurring in the sequence Seq_i) is associated to a probability-value p_{i_j} such that: $\sum_j p_{i_j} = 1$. From an intuitive point of view, p_{i_j} represents the probability that the rule R_{i_j} be applied at the i-th step. A deterministic state machine is, of course, a special case of a probabilistic state machine characterized by the following property: each sequence Seq_i consists of a single rule R_i.

Any probabilistic state machine naturally gives rise to a graph-structure for any choice of an input-configuration $conf_0 = (s_0, w_0)$. As an example, consider the following simple case: a probabilistic state machine **PM** whose program consists of two sequences, each consisting of two rules: $Seq_0 = (R_{0_1}, R_{0_2}), Seq_1 = (R_{1_1}, R_{1_2})$. The graph associated to **PM** for the configuration $conf_0$ is illustrated by Figure 1.

How do probabilistic machines compute? In order to define the concept of *computation* of a probabilistic machine, let us first introduce the notions of *program-path* and of *computation-path* of a given probabilistic machine.

Definition 3. *Program-path and computation-path.*
Let **PM** be a probabilistic state machine with program $(Seq_0, \ldots, Seq_t)$.

- A *program-path* of **PM** is a sequence $\mathcal{P} = (R_{0_h}, \ldots, R_{i_j}, \ldots, R_{t_k})$, consisting of t rules, where each R_{i_j} is a rule from Seq_i.
- For any choice of an input (s_0, w_0), any program-path $\mathcal{P}$ determines a sequence of configurations $\mathcal{CP} = ((s_0, w_0), \ldots, (s_i, w_i), \ldots, (s_{t+1}, w_{t+1}))$, where $(s_{i+1}, w_{i+1}) = R_{i_j}(s_i, w_i)$ and R_{i_j} is the i-th element of $\mathcal{P}$. This sequence is called the *computation-path* of **PM** determined by the program-path $\mathcal{P}$ and by the input (s_0, w_0).
  The configuration (s_{t+1}, w_{t+1}) represents the output of $\mathcal{CP}$.

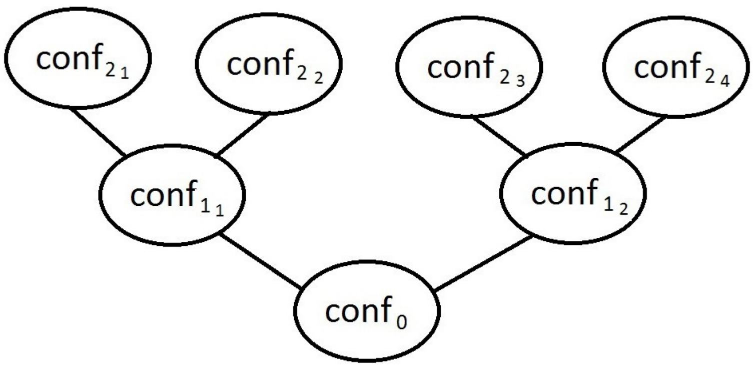

Figure 1. The graph of **PM**.

Any program-path $\mathcal{P} = (R_{0_h}, \ldots, R_{i_j}, \ldots, R_{t_k})$ has a well determined probability-value $p(\mathcal{P})$, which is defined as follows (in terms of the probability-values of its rules): $p(\mathcal{P}) := p_{0_h} \cdot \ldots \cdot p_{i_j} \cdot \ldots \cdot p_{t_k}$. As expected, the probability-value of a program-path $\mathcal{P}$ naturally determines the probability-values of all corresponding computation-paths. It is sufficient to put: $p(\mathcal{CP}) := p(\mathcal{P})$. Consider now the set $\mathbf{P_{PM}}$ of all program-paths and the set $\mathbf{CP_{PM}}$ of all computation-paths of a

probabilistic machine **PM**. One can easily show that:
$\sum_i \{p(\mathcal{P}_i) | \mathcal{P}_i \in \mathbf{P_{PM}}\} = \sum_i \{p(\mathcal{CP}_i) | \mathcal{CP}_i \in \mathbf{CP_{PM}}\} = 1$.

On this basis the concept of *computation* of a probabilistic state machine can be defined as follows.

Definition 4. *Computation of a probabilistic state machine.*
A *computation* of a probabilistic state machine **PM** with input (s_0, w_0) is the system of all computation-paths of **PM** with input (s_0, w_0).

Unlike the case of deterministic state machines, a computation of a probabilistic state machine does not yield a unique output. For any choice of a configuration-input (s_0, w_0), the computation-output is a system of possible configuration-outputs (s^i_{t+1}, w^i_{t+1}), where each (s^i_{t+1}, w^i_{t+1}) corresponds to a computation-path $\mathcal{CP}_i$. As expected, each (s^i_{t+1}, w^i_{t+1}) has a well determined probability-value that is defined as follows:
$p((s^i_{t+1}, w^i_{t+1})) := \sum_i \left\{ p(\mathcal{CP}_i) | \text{the configuration-output of } \mathcal{CP}_i \text{ is } (s^i_{t+1}, w^i_{t+1}) \right\}$.
One can easily show that the sum of the probability-values of all configuration-outputs of any machine **PM** is 1.

3. Quantum state machines

Before introducing the notion of *quantum state machine* it is expedient to recall some basic concepts used in quantum computation. Any piece of quantum information is mathematically represented as a density operator ρ living in a Hilbert space $\mathcal{H}^{(n)} := \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_{n-times}$ (where $n \geq 1$ and $\otimes$ is the tensor product). A *quregister* is a pure state, represented as a unit-vector $|\psi\rangle$ of a space $\mathcal{H}^{(n)}$ or, equivalently, as the corresponding density operator $P_{|\psi\rangle}$ (the projection-operator that projects over the closed subspace determined by $|\psi\rangle$). A *qubit* (or *qubit-state*) is a quregister of the space $\mathbb{C}^2$. A *register* (which is a particular quregister representing a *certain* piece of information) is an element $|x_1, \ldots, x_n\rangle$ of the canonical orthonormal basis of a space $\mathcal{H}^{(n)}$ (where $x_i \in \{0, 1\}$); a *bit* is a register of $\mathbb{C}^2$.[d] We will denote by $\mathfrak{D}(\mathcal{H}^{(n)})$ the set of all density operators of $\mathcal{H}^{(n)}$.

Quantum information is processed by (quantum logical) gates: unitary operators that transform quregisters in a reversible way. In the following we will use three gates that have a special computational and logical interest: the *negation*, the *Toffoli-gate* and the *Hadamard-gate*.

Definition 5. *The negation.*
For any $n \geq 1$, the *negation* is the linear operator $\mathtt{NOT}^{(n)}$ defined on $\mathcal{H}^{(n)}$ such that, for every element $|x_1, \ldots, x_n\rangle$ of the canonical basis:

[d]Notice that the choice of the two vectors that represent, in this framework, the two classical bits depends on the choice of the basis for the space $\mathbb{C}^2$. As happens in classical information theory, there are only two bits, while the set of all qubits is non-denumerable.

$\mathtt{NOT}^{(n)}|x_1,\ldots,x_n\rangle = |x_1,\ldots,x_{n-1}\rangle \otimes |1-x_n\rangle.$

In particular, we obtain: $\mathtt{NOT}^{(1)}|0\rangle = |1\rangle$; $\mathtt{NOT}^{(1)}|1\rangle = |0\rangle$, according to the classical truth-table of negation.

Definition 6. *The Toffoli-gate.*
For any $m,n,p \geq 1$, the *Toffoli-gate* is the linear operator $\mathtt{T}^{(m,n,p)}$ defined on $\mathcal{H}^{(m+n+p)}$ such that, for every element $|x_1,\ldots,x_m\rangle \otimes |y_1,\ldots,y_n\rangle \otimes |z_1,\ldots,z_p\rangle$ of the canonical basis:

$$\begin{aligned}&\mathtt{T}^{(m,n,p)}|x_1,\ldots,x_m,y_1,\ldots,y_n,z_1,\ldots,z_p\rangle\\ &\qquad = |x_1,\ldots,x_{m-1},y_1,\ldots,y_{n-1},z_1,\ldots,z_{p-1}\rangle \otimes |x_m,x_n,(x_m\cdot y_n \widehat{+} z_p)\rangle,\end{aligned}$$

where $\widehat{+}$ represents the addition modulo 2.

Definition 7. *The Hadamard-gate.*
For any $n \geq 1$, the *Hadamard-gate* is the linear operator $\sqrt{\mathtt{I}}^{(n)}$ defined on $\mathcal{H}^{(n)}$ such that, for every element $|x_1,\ldots,x_n\rangle$ of the canonical basis:

$$\sqrt{\mathtt{I}}^{(n)}|x_1,\ldots,x_n\rangle = |x_1,\ldots,x_{n-1}\rangle \otimes \frac{1}{\sqrt{2}}\left((-1)^{x_n}|x_n\rangle + |1-x_n\rangle\right).$$

In particular we obtain: $\sqrt{\mathtt{I}}^{(1)}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$; $\sqrt{\mathtt{I}}^{(1)}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

The Hadamard-gate represents a “genuine quantum gate” that can create superpositions, starting from register-inputs. At the same time, the negation and the Toffoli-gate (which always transform registers into registers) can be regarded as “semi-classical gates”: reversible versions of the classical Boolean functions. The Toffoli-gate has a special logical interest, also because it allows us to define a *reversible conjunction* $\mathtt{AND}^{(m,n)}$ for all quregisters $|\psi\rangle$ of $\mathcal{H}^{(m+n)}$:

$$\mathtt{AND}^{(m,n)}|\psi\rangle := \mathtt{T}^{(m,n,1)}(|\psi\rangle \otimes |0\rangle)$$

(where the bit $|0\rangle$ plays the role of an *ancilla*).

All gates can be canonically extended to density operators. Let G be any gate defined on $\mathcal{H}^{(n)}$. The corresponding *density-operator gate* ${}^{\mathfrak{D}}G$ (also called *unitary quantum operation*) is defined as follows for any $\rho \in \mathfrak{D}(\mathcal{H}^{(n)})$: ${}^{\mathfrak{D}}G\rho = G\,\rho\,G^\dagger$ (where $G^\dagger$ is the adjoint of G).

We will now introduce the concept of *quantum state machine*, which can be intuitively regarded as a kind of quantum superposition of “many” classical deterministic state machines. For the sake of simplicity, we will consider here quantum state machines whose possible inputs and outputs are represented by pure states. A generalization to the case of density operators can be obtained in a natural way.

Definition 8. *Quantum state machine.*
A *quantum state machine* is an abstract system $\mathbf{QM}$ associated to a Hilbert space $\mathcal{H}^{\mathbf{QM}} = \mathcal{H}^H \otimes \mathcal{H}^S \otimes \mathcal{H}^W$, whose unit-vectors $|\psi\rangle$ represent possible pure states of a quantum system that could physically implement the computations of the state machine. The following conditions are required:

1. $\mathcal{H}^H$ (which represents the halting-space) is the space $\mathcal{H}^{(1)}(= \mathbb{C}^2)$, where the two elements of the canonical basis $(\{|0\rangle_H, |1\rangle_H\})$ correspond to the states "the machine does not halt" and "the machine halts", respectively.
2. $\mathcal{H}^{\mathcal{S}}$ (which represents the internal-state space) is associated to a finite set $\mathcal{S}$ of classical internal states. We require that $\mathcal{H}^{\mathcal{S}} = \mathcal{H}^{(m)}$, where 2^m is the cardinal number of $\mathcal{S}$. Accordingly, the set $\mathcal{S}$ can be one-to-one associated to a basis of $\mathcal{H}^{\mathcal{S}}$.
3. $\mathcal{H}^W$ (which represents the word-space) is identified with a Hilbert space $\mathcal{H}^{(n)}$ (for a given $n \geq 1$). The number n determines the length of the registers $|x_1, \ldots, x_n\rangle$ that may occur in a computation. Shorter registers $|x_1, \ldots, x_h\rangle$ (with $h < n$) can be represented in the space $\mathcal{H}^{(n)}$ by means of convenient *ancillary bits.*
   Let $B^{\mathbf{QM}}$ be a basis of $\mathcal{H}^{\mathbf{QM}}$, whose elements are unit-vectors having the following form: $|\varphi_i\rangle = |h_i\rangle|s_i\rangle|x_{i_1}, \ldots, x_{i_n}\rangle$, where $|h_i\rangle$ belongs to the basis of $\mathcal{H}^H$, while $|s_i\rangle$ belongs to the basis of $\mathcal{H}^S$. Any unit-vector $|\psi\rangle$ of $\mathcal{H}^{\mathbf{QM}}$ that is a superposition of basis-elements $|\varphi_i\rangle$ represents a possible *computational state* of **QM**. The expected interpretation of a computational state $|\psi\rangle = \sum_i c_i|h_i\rangle|s_i\rangle|x_{i_1}, \ldots, x_{i_n}\rangle$ is the following:
   - the machine in state $|\psi\rangle$ might be in the halting state $|h_i\rangle$ with probability $|c_i|^2$;
   - the machine in state $|\psi\rangle$ might correspond to the classical configuration $(s_i, (x_{i_1}, \ldots, x_{i_n}))$ with probability $|c_i|^2$.
     Hence, the state $|\psi\rangle$ describes a kind of *quantum co-existence* of different classical deterministic configurations.[e]
4. The set of *possible inputs* of **QM** is identified with the set of all computational states that have the following form: $|\psi\rangle = \sum_i c_i|0_H\rangle|s_{in}\rangle|x_{i_1}, \ldots, x_{i_n}\rangle$.
5. Like a deterministic state machine, a quantum state machine **QM** is characterized by a *program.* In the quantum case, a program is identified with a sequence $(U_0, \ldots, U_t)$ of unitary operators of $\mathcal{H}^{\mathbf{QM}}$, where we may have: $U_i = U_j$ with $i \neq j$. The following conditions are required:
   a) for any possible input $|\psi_0\rangle$, $U_0(|\psi_0\rangle) = |\psi_1\rangle$ is a superposition of basis-elements having the following form: $|h_i^1\rangle|s_i^1\rangle|x_{i_1}^1, \ldots, x_{i_n}^1\rangle$, where all s_i^1 are different from s_{in} and $|h_i^1\rangle = |0_H\rangle$, if $t \neq 0$.
   b) For any j $(0 < j < t)$, $U_j(|\psi_j\rangle) = |\psi_{j+1}\rangle$ is a superposition of basis-elements having the following form: $|0_H\rangle|s_i^{j+1}\rangle|x_{i_1}^{j+1}, \ldots, x_{i_n}^{j+1}\rangle$.
   c) $U_t(|\psi_t\rangle) = |\psi_{t+1}\rangle$ is a superposition of basis-elements having the following form: $|1_H\rangle|s_{halt_j}\rangle|x_{i_1}^{t+1}, \ldots, x_{i_n}^{t+1}\rangle$.

[e]As is well known, quantum superpositions have been often described as a kind of co-existence of alternative properties (which may appear strange and puzzling from a classical point of view). In the philosophical debate about quantum theory one has often discussed the question that concerns the "epistemic" or the "ontic" nature of quantum probabilities. The arguments developed in this article seem to confirm an interpretation that is close to an "ontic choice".

The concept of *computation* of a quantum state machine can be now defined in a natural way.

Definition 9. *Computation of a quantum state machine.*
Let **QM** be a quantum state machine, whose program is the operator-sequence $(U_0, \ldots, U_t)$ and let $|\psi_0\rangle$ be a possible input of **QM**. A *computation* of **QM** with input $|\psi_0\rangle$ is a sequence $\mathcal{QC} = (|\psi_0\rangle, \ldots, |\psi_{t+1}\rangle)$ of computational states such that: $|\psi_{i+1}\rangle = U_i(|\psi_i\rangle)$, for any i $(0 \leq i \leq t)$. The vector $|\psi_{t+1}\rangle$ represents the output of the computation, while the density operator $Red^3(|\psi_{t+1}\rangle)$ (the reduced state of $|\psi_{t+1}\rangle$ with respect to the third subsystem) represents the word-output of the computation.

Consider now a quantum state machine whose program is $(U_0, \ldots, U_t)$. Each U_i naturally determines a corresponding word-operator U_i^W, defined on the word-space $\mathcal{H}^W$. Generally, it is not guaranteed that all word-operators are unitary. But it is convenient to refer to quantum state machines that satisfy this condition. In this way, any quantum state machine (whose word-space is $\mathcal{H}^{(n)}$) determines a *quantum circuit*, consisting of a sequence $(U_0^W, \ldots, U_t^W)$ of gates, where n represents the *width*, while $t+1$ represents the *depth* of the circuit. Viceversa, we can assume that any circuit $(U_0^W, \ldots, U_t^W)$ gives rise to a quantum state machine, whose halting states and whose internal states are supposed to be chosen in a conventional way.

To what extent can quantum state machines be simulated by classical probabilistic state machines? We will discuss this question by referring to a celebrated quantum experiment, based on the *Mach-Zehnder interferometer* (represented by Figure 2).

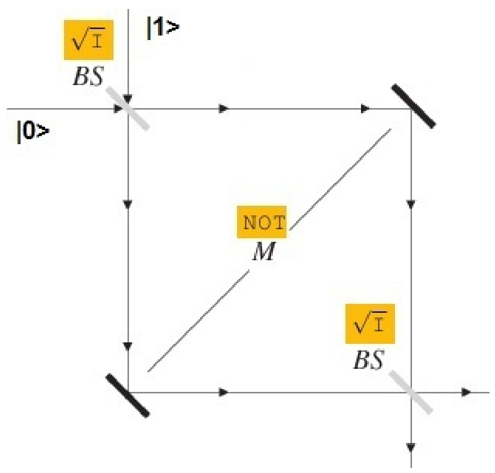

Figure 2. The Mach-Zehnder interferometer.

The physical situation can be sketched as follows. Consider a photon-beam (possibly consisting of a single photon) and assume that $|0\rangle$ describes the state of photons moving along the x direction, while $|1\rangle$ describes the state of photons moving along the y direction. All photons go through a first *beam splitter* that "splits" them giving rise to the following effect: within the box each photon follows a path corresponding either to the x-direction or to the y-direction with probability $\frac{1}{2}$. Soon after, on both paths, all photons are reflected by a *mirror* that inverts their direction. Finally, the photons pass through a second beam splitter that determines the output-state. Suppose that all photons entering into the interferometer-box are moving in the x-direction. According to a "classical way of thinking" we would expect that the photons detected at the end of the process will move either along the x-direction or along the y-direction with probability $\frac{1}{2}$. The result of the experiment is, instead, completely different: the Mach-Zehnder interferometer *always* transforms the input-state $|0\rangle$ into the output-state $|0\rangle$; while the input-state $|1\rangle$ is transformed into $|1\rangle$. From a mathematical point of view, such a "surprising" result can be explained by using, in an essential way, the concept of superposition. The apparatuses (used in the Mach-Zehnder experiment) can be mathematically represented by two gates. A beam splitter can be regarded as a physical implementation of the *Hadamard-gate* $\sqrt{\mathtt{I}}^{(1)}$, which transforms the two classical bits $|0\rangle$ and $|1\rangle$ into two (different) genuine superpositions. As a consequence, inside the Mach-Zehnder box, a photon (which is in state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$) turns out to satisfy at the same time two alternative properties: the property of moving along the x-direction and the property of moving along the y-direction. We have here a characteristic quantum parallelism: a single photon "seems to go along" two different paths at the same time! The second apparatus of the Mach-Zehnder interferometer (the mirror), can be regarded as a physical implementation of the gate $\mathtt{NOT}^{(1)}$. Accordingly, the *Mach-Zehnder circuit* can be identified with the following sequence of three gates (all defined on the space $\mathbb{C}^2$):

$$(\sqrt{\mathtt{I}}^{(1)}, \mathtt{NOT}^{(1)}, \sqrt{\mathtt{I}}^{(1)}).$$

Let us now apply the Mach-Zehnder circuit to the input $|0\rangle$. We obtain:
$\sqrt{\mathtt{I}} : |0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$; $\mathtt{NOT} : \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$;
$\sqrt{\mathtt{I}} : \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \mapsto |0\rangle$.
We can see, in this way, how the Mach-Zehnder circuit transforms the input-state $|0\rangle$ into the output-state $|0\rangle$. In a similar way, the input-state $|1\rangle$ is transformed into the output-state $|1\rangle$.

Is there any natural "classical counterpart" for the Hadamard-gate? A natural candidate might be a particular example of a probabilistic state machine that we can conventionally call *the classical probabilistic* `NOT`*-state machine* ($\mathbf{PM}^{\mathtt{NOT}}$). Such machine can be defined as follows:

- The set of possible word-inputs of $\mathbf{PM}^{\mathtt{NOT}}$ is the set of words $\{(0), (1)\}$.
- The program of $\mathbf{PM}^{\mathtt{NOT}}$ consists of the following sequence of rules:

$$Seq_0 = (R_{0_1}, R_{0_2}),$$

where:
$R_{0_1} : (s_{in}, (x)) \mapsto (s_{halt_j}, (x))$ and $p(R_{0_1}) = \frac{1}{2}$;
$R_{0_2} : (s_{in}, (x)) \mapsto (s_{halt_j}, (1-x))$ and $p(R_{0_2}) = \frac{1}{2}$.

Consider, for instance, the input $(s_{in}, (0))$. The output will be the following set:

$$\left\{(s_{halt_j}, (0)), (s_{halt_j}, (1))\right\}.$$

On this basis, a "classical probabilistic Mach-Zehnder state machine" would determine (for the word-input (0)) the word-graph illustrated by Figure 3.

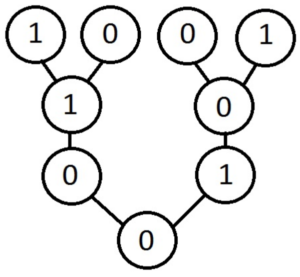

Figure 3. A word-graph for a "classical probabilistic Mach-Zehnder state machine".

Such a machine turns out to compute both the words (0) and (1) with probability $\frac{1}{2}$. Interestingly enough, this is the same probabilistic result that is obtained in the quantum case, when one performs a measurement inside the interferometer-box.

The arguments we have developed seem to confirm the following conjecture: the characteristic superposition-patterns, that may occur during a quantum computation (when no measurement is performed during the computation-process), cannot be generally represented by probabilistic state machines. Quantum parallelism (based on superpositions) and classical parallelism are deeply different. This conclusion seems to be in agreement with a position defended by Feynman in his pioneering article [7]:

Can a quantum system be probabilistically simulated by a classical (probabilistic, I'd assume) universal computer? In other words, a computer which will give the same probabilities as the quantum system does. If you take the computer to be the classical kind I've described so far (not the quantum kind described in the last section) and there're are no changes in any laws, and there's no hocus-pocus, the answer is certainly, No! This is called the hidden-variable problem: it is impossible to represent the results of quantum mechanics with a classical universal device.

4. Quantum circuits and quantum computational logics

Quantum circuits can be linguistically described in the framework of a particular form of quantum logic, termed *quantum computational logic* (**QCL**).[f] Let us briefly recall the basic features of this logic, whose formulas are supposed to denote pieces of quantum information (density operators living in some Hilbert space $\mathcal{H}^{(n)}$), while the logical connectives correspond to some particular gates. Accordingly, the language $\mathcal{L}$ of **QCL** contains atomic formulas ($\mathbf{q}, \mathbf{q}_1, \mathbf{q}_2, \ldots$) including two privileged formulas $\mathbf{t}$ (the *Truth*) and $\mathbf{f}$ (the *Falsity*) that denote, respectively, the density operators $P_{|1\rangle}$ and $P_{|0\rangle}$ (which correspond to the bits $|1\rangle$ and $|0\rangle$). The connectives of $\mathcal{L}$ are at least the following : the negation $\neg$ (corresponding to the gate $\mathtt{NOT}^{(n)}$), the ternary Toffoli-connective T (corresponding to the gate $\mathtt{T}^{(m,n,p)}$), the square root of identity $\sqrt{id}$ (corresponding to the gate $\sqrt{\mathtt{I}}^{(n)}$). Hence, any formula will have one of the following forms: $\mathbf{q}$, $\neg\alpha$, $\sqrt{id}\alpha$, $\mathsf{T}(\alpha, \beta, \gamma)$. Recalling the definition of $\mathtt{AND}^{(m,n)}$, a binary conjunction $\wedge$ can be defined in terms of the Toffoli-connective: $\alpha \wedge \beta := \mathsf{T}(\alpha, \beta, \mathbf{f})$ (where $\mathbf{f}$ plays the role of a *syntactical ancilla*).

By *atomic complexity* of a formula α we mean the number $At(\alpha)$ of occurrences of atomic subformulas in α. For instance, the atomic complexity of the (contradictory) formula $\alpha = \mathbf{q} \wedge \neg\mathbf{q} = \mathsf{T}(\mathbf{q}, \neg\mathbf{q}, \mathbf{f})$ is 3. The number $At(\alpha)$ plays an important semantic role, since it determines the *semantic space* $\mathcal{H}^{\alpha} = \mathcal{H}^{(At(\alpha))}$, where any density operator representing a possible *informational meaning* of α shall live. We have, for instance, $\mathcal{H}^{\mathsf{T}(\mathbf{q},\neg\mathbf{q},\mathbf{f})} = \mathcal{H}^{(3)}$. Any formula α can be naturally decomposed into its parts giving rise to a special configuration, called the *syntactical tree* of α ($STree^{\alpha}$). Roughly, $STree^{\alpha}$ can be represented as a sequence of *levels* consisting of subformulas of α. The *bottom-level* is (α), while all other levels are obtained by dropping, step by step, all connectives occurring in α. Hence, the *top-level* is the sequence of atomic formulas occurring in α. For instance, the syntactical tree of the formula $\alpha = \mathsf{T}(\mathbf{q}, \neg\mathbf{q}, \mathbf{f})$ is the following sequence of *levels*:

$$(Level_3^{\alpha} = (\mathbf{q}, \mathbf{q}, \mathbf{f}),\ Level_2^{\alpha} = (\mathbf{q}, \neg\mathbf{q}, \mathbf{f}),\ Level_1^{\alpha} = (\mathsf{T}(\mathbf{q}, \neg\mathbf{q}, \mathbf{f})).$$

For any α, $STree^{\alpha}$ uniquely determines the *gate-tree* of α: a sequence of gates all defined on the space $\mathcal{H}^{\alpha}$. As an example, consider again the formula $\alpha = \mathsf{T}(\mathbf{q}, \neg\mathbf{q}, \mathbf{f})$. In the syntactical tree of α the second level has been obtained (from the third level) by repeating the first occurrence of $\mathbf{q}$, by negating the second occurrence of $\mathbf{q}$ and by repeating $\mathbf{f}$; while the first level has been obtained (from the second level) by applying the Toffoli-connective. Accordingly, the gate-tree of α can be naturally identified with the following gate-sequence:

$$({}^{\mathfrak{D}}\mathtt{I}^{(1)} \otimes {}^{\mathfrak{D}}\mathtt{NOT}^{(1)} \otimes {}^{\mathfrak{D}}\mathtt{I}^{(1)},\ {}^{\mathfrak{D}}\mathtt{T}^{(1,1,1)}).$$

This procedure can be naturally generalized to any α.

[f] See Dalla Chiara et al. [2], [4], [5].

We consider here a *holistic* version of the quantum computational semantics[g], based on the notion of *holistic model*: a special map `Hol` that assigns to each level of the syntactical tree of any formula α a *global informational meaning*, represented by a density operator living in the semantic space of α. This global meaning determines the *contextual meanings* of all subformulas occurring in the syntactical tree of α. Suppose that $Level_i^\alpha = (\beta_{i_1}, \ldots, \beta_{i_r})$. It is natural to describe $\texttt{Hol}(Level_i^\alpha)$ as a possible state of a composite quantum system consisting of r subsystems. Hence, the *contextual meaning* of the occurrence β_{i_j} (in $STree^\alpha$) can be identified with the reduced state of $\texttt{Hol}(Level_i^\alpha)$ with respect to the j-th subsystem. On this basis, a *holistic model* of the language $\mathcal{L}$ can be defined as a map `Hol` that satisfies the following conditions for any formula α:

1) $\texttt{Hol}(Level_i^\alpha) \in \mathfrak{D}(\mathcal{H}^\alpha)$.
2) `Hol` assigns the same contextual meaning to different occurrences of one and the same subformula of α (in $STree^\alpha$).
3) The contextual meanings of the true formula **t** and of the false formula **f** are the density operators $P_{|1\rangle}$ and $P_{|0\rangle}$, respectively.
4) `Hol` preserves the logical form of α by interpreting the connectives of α as the corresponding gates.

Finally, the meaning assigned by a model `Hol` to a formula α is identified with the density operator that `Hol` assigns to the bottom-level of the syntactical tree of α.

Any formula α of the language $\mathcal{L}$ can be regarded as a synthetic logical description of a quantum circuit $\mathcal{C}^\alpha$, determined by the gate-tree of α. For instance, the Mach-Zehnder circuit $(\sqrt{\texttt{I}}^{(1)}, \texttt{NOT}^{(1)}, \sqrt{\texttt{I}}^{(1)})$ turns out to be described by the formula $\sqrt{id}\,\neg\sqrt{id}\,\mathbf{q}$. Given a formula α, any model `Hol` fixes a possible input and the corresponding output for the circuit $\mathcal{C}^\alpha$. The input is represented by the density operator ρ_{in} that `Hol` assigns to the top-level of the syntactical tree of α, while the output is the density operator ρ_{out} associated by `Hol` to the bottom-level of the same tree. Accordingly, we can write: $\mathcal{C}^\alpha(\rho_{in}) = \rho_{out}$. Due to the correspondence between quantum circuits and quantum state machines, it turns out that any formula α of the quantum computational language can be associated to a particular quantum state machine $\mathbf{QM}^\alpha$ (whose halting states and whose internal states are supposed to be chosen in a conventional way).

5. Abstract quantum computing machines

State machines represent *rigid* systems: each machine has a definite program, devoted to a single task. Real computers, however, behave differently, being able to solve different kinds of problems (which can be chosen by computer-users). We will now investigate a "more liberal" concept of machine that will be called *abstract*

[g]See [4,5].

quantum computing machine. The intuitive idea can be sketched as follows. Consider a finite gate-system $\mathfrak{G} = (G_1^{(n_1)}, \ldots, G_1^{(n_t)})$, where each $G_i^{(n_i)}$ is defined on a word-space $\mathcal{H}^{(n_i)}$.[h] The system $\mathfrak{G}$ determines an infinite set of *derived gates* that can be obtained as appropriate combinations of elements of $\mathfrak{G}$, by using gate-tensor products and gate-compositions. An interesting example is represented by the gate system

$$\mathfrak{G}^* = (\mathtt{I}^{(1)}, \mathtt{NOT}^{(1)}, \sqrt{\mathtt{I}}^{(1)}, \mathtt{T}^{(1,1,1)}),$$

where $\mathtt{I}^{(1)}$ (the identity-gate), $\mathtt{NOT}^{(1)}$ (the negation-gate), $\sqrt{\mathtt{I}}^{(1)}$ (the Hadamard-gate) are defined on the space $\mathcal{H}^{(1)}$ $(= \mathbb{C}^2)$, while $\mathtt{T}^{(1,1,1)}$ (the Toffoli-gate) is defined on the space $\mathcal{H}^{(3)}$.[i] Notice that for any $n, m, p \geq 1$, the gates $\mathtt{NOT}^{(n)}$, $\sqrt{\mathtt{I}}^{(n)}$, $\mathtt{T}^{(m,n,p)}$ can be represented as derived gates of the system $\mathfrak{G}^*$. We have, for instance, $\mathtt{NOT}^{(n)} = \underbrace{\mathtt{I}^{(1)} \otimes \ldots \otimes \mathtt{I}^{(1)}}_{(n-1)-times} \otimes \mathtt{NOT}^{(1)}$.

Any gate-system $\mathfrak{G}$ gives rise to an infinite family $\mathfrak{C}^{\mathfrak{G}}$ of circuits

$$^{\mathfrak{G}}\mathcal{C}_j^{(n)} = (^{\mathfrak{G}}G_1^{(n)}, \ldots, {}^{\mathfrak{G}}G_t^{(n)}),$$

where each $^{\mathfrak{G}}G_i^{(n)}$ is a derived gate of $\mathfrak{G}$, defined on the space $\mathcal{H}^{(n)}$. For instance, the Mach-Zehnder circuit $(\sqrt{\mathtt{I}}^{(1)}, \mathtt{NOT}^{(1)}, \sqrt{\mathtt{I}}^{(1)})$ represents an example of a circuit that belongs to $\mathfrak{C}^{\mathfrak{G}^*}$ (the circuit-family determined by the gate-system $\mathfrak{G}^*$).

On this basis, it seems reasonable to assume that any choice of a finite gate-system $\mathfrak{G}$ determines an *abstract quantum computing machine* $\mathbf{AbQCM}^{\mathfrak{G}}$ whose programs correspond to the circuits that belong to the family $\mathfrak{C}^{\mathfrak{G}}$. Since any circuit can be associated to a particular quantum state machine, any abstract quantum computing machine can be also regarded as an infinite family of quantum state machines, corresponding to different programs that the abstract machine can perform. Accordingly, any circuit $^{\mathfrak{G}}\mathcal{C} \in \mathfrak{C}^{\mathfrak{G}}$, applied to an appropriate input, represents a *computation* of the abstract machine $\mathbf{AbQCM}^{\mathfrak{G}}$. We can say that $\mathbf{AbQCM}^{\mathfrak{G}}$ *computes* the output $|\psi\rangle_{out}$ for the input $|\psi\rangle_{in}$ iff there is a circuit $^{\mathfrak{G}}\mathcal{C} \in \mathfrak{C}^{\mathfrak{G}}$ such that $^{\mathfrak{G}}\mathcal{C}|\psi\rangle_{in} = |\psi\rangle_{out}$.

An interesting question concerns the possibility of a *universal abstract quantum computing machine*, that can play the role of the universal Turing machine in classical computation. This question has a negative answer: one can prove that no abstract quantum computing machine can be *perfecly universal.* This is a consequence of the following theorem.

[h]For the sake of simplicity, we are considering here gates G that are unitary operators (of a space $\mathcal{H}^{(n)}$). Of course, the procedure can be easily generalized to the case of density-operator gates $^{\mathfrak{D}}G$.

[i]The system $\mathfrak{G}^*$ is, in fact, redundant. For, the identity operator can be defined in terms of the Hadamard-gate, while the negation-gate can be defined in terms of the Toffoli-gate. However, dealing with a primitive identity and with a primitive negation (defined on the space $\mathcal{H}^{(1)}$) turns out to be more convenient from a computational and from a logical point of view.

Theorem 1. For any space $\mathcal{H}^{(n)}$ there is no finite system of gates (defined on $\mathcal{H}^{(n)}$) such that any gate G of $\mathcal{H}^{(n)}$ can be represented as a finite combination of elements of the system.

Proof. Straightforward by cardinality-reasons (the hypothesis that any gate G of $\mathcal{H}^{(n)}$ can be represented as a finite combination of elements of a finite gate-system would imply that the set of all possible gates of $\mathcal{H}^{(n)}$ is denumerable). □

In spite of this negative result, one can usefully have recourse to a notion of *approximately universal gate system*, which is justified by the following theorem (proved by Shi [12] and Aharonov [1]).[j]

Theorem 2. For any gate G of $\mathcal{H}^{(n)}$ and for any choice of a non-negative real number ϵ there is a finite sequence of gates $(G_1, \dots, G_u)$ (of $\mathcal{H}^{(n)}$) such that:

- $(G_1, \dots, G_u)$ is a circuit belonging to the family $\mathfrak{C}^{\mathfrak{G}^*}$.
- For any vector $|\psi\rangle$ of $\mathcal{H}^{(n)}$, $\|G|\psi\rangle - G_1 \dots G_u|\psi\rangle\| \leq \epsilon$.

Thus, the family $\mathfrak{C}^{\mathfrak{G}^*}$ has the capacity of approximating with arbitrary accuracy any possible gate. On this basis, the machine $\mathbf{AbQCM}^{\mathfrak{G}^*}$ can be reasonably represented as an *approximately universal abstract quantum computing machine.* Notice that all circuits in the family $\mathfrak{C}^{\mathfrak{G}^*}$ (hence all programs of $\mathbf{AbQCM}^{\mathfrak{G}^*}$) can be syntactically represented by means of formulas expressed in the language $\mathcal{L}$ of quantum computational logic (whose connectives are: the negation $\neg$, the Hadamard-connective $\sqrt{id}$ and the Toffoli-connective T). Both $\mathfrak{C}^{\mathfrak{G}^*}$ and the set of all formulas of $\mathcal{L}$ are denumerable sets. At the same time, the set of all possible inputs and outputs of quantum computations is, obviously, non-denumerable. Unlike classical computations, quantum computations cannot be faithfully represented in a purely syntactical way (in the framework of a denumerable language). One of the basic tasks of quantum computational semantics is creating a link between the (denumerable) world of circuits and the (non-denumerable) world of possible inputs and outputs of quantum computations.

What can be said about the computational power of $\mathbf{AbQCM}^{\mathfrak{G}^*}$? One can easily realize that $\mathbf{AbQCM}^{\mathfrak{G}^*}$ is able to compute in an *exact* way all recursive numerical functions. For, any sequence of natural numbers can be represented as a register and any computation of a recursive function (applied to a register-input) can be represented as an appropriate combination of the "Boolean" gates (the negation and the Toffoli-gate). Thus, the machine $\mathbf{AbQCM}^{\mathfrak{G}^*}$ is able to compute whatever the universal Turing machine is able to compute. Is $\mathbf{AbQCM}^{\mathfrak{G}^*}$ able to compute anything else in the domain of natural numbers? In other words, are there any $\mathfrak{G}^*$-circuits (where the Hadamard-gate plays an essential role) that can approximately compute (with arbitrary precision) some non-recursive functions? A

[j]See also [6] and [3].

discussion about this hard problem (which is obviously connected with the validity of the *Church-Turing thesis*) goes beyond the limits of our article.

6. Acknowledgment

R.Giuntini and G.Sergioli gratefully acknowledge the support of the Sardinia Region within the Project "Modeling the uncertainty: quantum theory and imaging processing", LR 7/8/2007. RAS CRP-59872.

References

1. D. Aharonov, A simple proof that Toffoli and Hadamard are quantum universal, quant-ph/0301040.
2. M.L. Dalla Chiara, R. Giuntini, and R. Leporini, Logics from quantum computation, *Int. J. Theor. Phys.* **3** (2005), 293–337.
3. M.L. Dalla Chiara, H. Freytes, R. Giuntini, A. Ledda, and G. Sergioli, The algebraic structure of an approximately universal system of quantum computational gates, *Found. Phys.* **39 (6)** (2009), 559–572.
4. M.L. Dalla Chiara, R. Giuntini, A. Ledda, R. Leporini, and G. Sergioli, Entanglement as a semantic resource, *Found. Phys.* **40** (2010), 1494–1518.
5. M.L. Dalla Chiara, R. Giuntini, R. Leporini, and G. Sergioli, A first-order epistemic quantum computational semantics with relativistic-like epistemic effects, *Fuzzy Sets and Systems* (2015) doi:10.1016/j.fss.2015.09.002.
6. D. Deutsch, A. Barenco, and A. Ekert, Universality in quantum computation, quant-ph/9505018v1.
7. R.P. Feynman, Simulating physics with computers, *Int. J. Theor. Phys.* **21** (1982) 467–488.
8. W. Fouché, J. Heidema, G. Jones, and P.H. Potgieter, Deutsch's universal quantum Turing machine (Revisited), quant-ph/0701108v1.
9. S. Gudder, Quantum automata: An overview, *Int. J. Theor. Phys.* **38 (9)** (1999) 2261–2282.
10. R. Penrose, *Shadows of the Mind* (Oxford University Press, 1994).
11. J.E. Savage, *Models of computation: exploring the power of computing* (Addison Wesley, 1998).
12. Y. Shi, Both Toffoli and Controlled-Not need little help to do universal quantum computation, quant-ph/0205115.