Paraconsistent Machines and their Relation to Quantum Computing

JUAN C. AGUDELO, *Ph.D. Program in Philosophy, area of Logic, IFCH and Group for Applied and Theoretical Logic–CLE, State University of Campinas–UNICAMP, Brazil. Logic and Computation Research Group, Eafit University, Colombia.*
E-mail: juca.agudelo@gmail.com

WALTER CARNIELLI, *IFCH and Group for Applied and Theoretical Logic–CLE, State University of Campinas–UNICAMP, Brazil. Security and Quantum Information Group–IT, Lisbon, Portugal.*
E-mail: walter.carnielli@gmail.com

Abstract

We describe a method to axiomatize computations in deterministic Turing machines (TMs). When applied to computations in non-deterministic TMs, this method may produce contradictory (and therefore trivial) theories, considering classical logic as the underlying logic. By substituting in such theories the underlying logic by a paraconsistent logic we define a new computation model, the *paraconsistent Turing machine*. This model allows a partial simulation of superposed states of quantum computing. Such a feature allows the definition of paraconsistent algorithms which solve (with some restrictions) the well-known Deutsch's and Deutsch-Jozsa problems. This first model of computation, however, does not adequately represent the notions of *entangled states* and *relative phase*, which are key features in quantum computing. In this way, a more sharpened model of paraconsistent TMs is defined, which better approaches quantum computing features. Finally, we define complexity classes for such models, and establish some relationships with classical complexity classes.

Keywords: Paraconsistent Turing machines, paraconsistent computation, quantum computation, Deutsch's problem, Deutsch-Jozsa problem, entangled states.

1 Introduction

The undecidability of first-order logic (FOL) was first proved by Alonzo Church in [11] and an alternative proof with the same result was presented by Alan Turing in [28]. In his article, Turing defined an abstract model of automatic machines, now known as *Turing machines* (TMs), and demonstrated that there are unsolvable problems for that class of machines. By axiomatizing machine computations in first-order theories, he could then prove that the decidability of FOL would imply the solution of established unsolvable problems. Consequently, by *reductio ad absurdum*, FOL is shown to be undecidable. Turing's proof was simplified by Richard Büchi in [8], and a more recent and clear version of this proof is presented by George Boolos and Richard Jeffrey in [4, Chapter 10].

By following [4] and adding new axioms, we define a method to obtain adequate theories for computations in *deterministic* TMs (DTMs) (Section 2), which verify a formal notion of representation of TM computation (here introduced), therefore enhancing the standard way in which TMs are expressed by means of classical FOL.

Next, we will show that by using the same axiomatization method for *non-deterministic* TMs (NDTMs), we obtain (in some cases) contradictory theories, and therefore trivial theories in view

Vol. 20 No. 2, © The Author, 2009. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oxfordjournals.org
Published online 10 November 2009 doi:10.1093/logcom/exp072

of the underlying logic. At this point, we have two options in sight to avoid triviality: (i) The first option, consisting in the classical move of restricting theories in a way that contradictions could not be derived (just representing computations in NDTMs); (ii) the second option, consisting in substituting the underlying logic by a paraconsistent logic, supporting contradictions and providing a way to define new models of computation through the interpretation of the theories. The first option is sterile: it is incapable of producing a new model of computation. In this article, we follow the second option, initially defining in Section 3 a model of *paraconsistent* TMs (ParTMs), using the paraconsistent logic $LFI1^*$ (see [10]). We then show that ParTMs allow a partial simulation of 'superposed states', an important feature of quantum computing (Section 3.1). By using this property, and taking advantage of 'conditions of inconsistency' added to the instructions, we show that the quantum solution of Deutsch's and Deutsch-Jozsa problems can be simulated in ParTMs (Section 3.1.1). However, as explained in Section 3.1.2, ParTMs are not adequate to simulate *entangled states* and *relative phases*, which are key features in quantum computing. Thus, still in Section 3.1.2, we define a paraconsistent logic with a *non-separable* conjunction and, with this logic, a new model of paraconsistent TMs is defined, which we call *entangled paraconsistent* TMs (EParTMs). In EParTMs, uniform entangled states can be successfully simulated. Moreover, we describe how the notion of relative phase (see [23, p. 193]) can be introduced in this model of computation.[1]

The *paraconsistent computability theory* has already been mentioned in [26, p. 196] as an 'emerging field of research'. In that paper, *dialethic machines* (or *D-machines*) are summarily described as TMs acting under a dialethic logic (a kind of paraconsistent logic) in the presence of a certain contradiction, but no concrete definition of any paraconsistent computation model is presented. A D-machine of 'type 1' is described as a machine that, when encounters a contradiction (i.e. 'for some statement A, both A and $\sim A$ appear in its output or among its inputs'), it 'can proceed with its computation satisfactorily'. However, it is not clearly specified how A and $\sim A$ could appear in the inputs and outputs of the machines,[2] neither how the D-machine proceed when a contradiction is encountered. Moreover, the authors sketch which they call 'a central idea of paraconsistent computability theory', consisting in claiming that 'such TMs [D-machines of type 1] may be employed to compute diagonal functions that are classically regarded as uncomputable'. Their arguments are obscure in view of the lack of a clear definition of the adopted computational model. D-machines of 'type 2' are defined as 'machines whose *meta* logic is dialethic: for such a machine, M, one of whose states is x, "M is in x" and '$\sim$(M is in x)' may both be the case'. The authors propose two questions to the paraconsistent community: 'it is an open question whether D-machines of type 2 compute classically uncomputable functions and, if so, which.', and a second question asks 'whether or not the halting theorem holds within a paraconsistent framework'. In terms of Sylvan and Copeland [26], the ParTMs and EParTMs proposed here would correspond to paraconsistent machines of type 2, but we prove (see Section 3.2) that such models of computation *do not* break the Church-Turing thesis. In this way, we provide a negative reply (restricted to our proposed models) for the first question formulated in [26], showing that *hypercomputation* (i.e. the computation of non-Turing computable functions) is not an inherent property of paraconsistent models of computation. As a consequence, and taking into account that classical TMs are special cases of ParTMs and EParTMs, we also provide a negative reply to the

[1]ParTMs were first presented in [3] and relations with quantum computing were presented in [1], but here we obtain some improvements and introduce the model of EParTMs, which represents a better approach to quantum computing.

[2]Several questions can be posed to the sloppy presentation of [26]: Does 'statement A' in a D-machine correspond to a input/output symbol, or does it correspond to a sequence of symbols? How is $\sim A$ represented in a D-machine? It just corresponds to another symbol, supposed to be contradictory to A, or are inputs and outputs signaled by a kind of positive and negative signs? A and $\sim A$ needs to appear in the same cell to be considered a contradiction? How could they appear together?

second open question: the halting problem *does* hold for ParTMs and EParTMs, and therefore its failure is any underlying feature of paraconsistent computation models.

It is convenient to emphasize that although the paraconsistent models of computation we present here do not extend the class of computable functions, they provide a new framework to study quantum computing and parallel computation in general, and also provide a way to view computational complexity as logic relative. Definitions of computational complexity classes for ParTMs and EParTMs, in addition to interesting relations with existing classical complexity classes, are presented in Section 3.2.

The paraconsistent approach to quantum computing here presented is just one way to describe the role of quantum features in the process of computation by means of non-classical logics and to investigate computational complexity from a logical viewpoint; in [2] another way to define a model of computation based upon another paraconsistent logic is proposed, also related with quantum computing. The relationship between these two different definitions of paraconsistent computation is a task that needs to be addressed in future work.

2 Axiomatization of TM computations

As mentioned above, a method to define first-order theories for TM computations has already been introduced in [28]. Although this is a well-known construction, in view of the important role of this method in our definition of paraconsistent TMs we will describe it in detail, following [4, Chapter 10]. This method will be extended to deal with a formal notion of representation (here introduced) of TM computation.

Considering $Q=\{q_1,\dots,q_n\}$ as a finite set of states and $\Sigma=\{s_1,\dots,s_m\}$ as a finite set of read/write symbols, we will suppose that TM instructions are defined by quadruples of one of the following types (with the usual interpretations, where R means a movement to the right, and L means a movement to the left):

$$q_i s_j s_k q_l, \tag{I}$$

$$q_i s_j R q_l, \tag{II}$$

$$q_i s_j L q_l. \tag{III}$$

By convention, we will enumerate the instants of time and the cells of the tape by integer numbers, and we will consider that machine computations begin at time 0, with a symbol sequence on the tape (the *input* of the computation), and with the machine in state q_1 scanning the symbol on cell 0. s_1 will be assumed to be an *empty* symbol.

In order to represent the computation of a TM $\mathcal{M}$ with input α (hereafter $\mathcal{M}(\alpha)$), we initially define the first-order theory $\Delta_{FOL}(\mathcal{M}(\alpha))$ over the first-order language $\mathcal{L}=\{Q_1,\dots,Q_n,S_1,\dots,S_m,<,',0\}$,[3] where symbols Q_i, S_j and $<$ are binary predicate symbols, $'$ is a unary function symbol and 0 is a constant symbol. In the intended interpretation $\mathcal{I}$ of the sentences in $\Delta_{FOL}(\mathcal{M}(\alpha))$, variables are interpreted as integer numbers, and symbols in $\mathcal{L}$ are interpreted in the following way:

- $Q_i(t,x)$ indicates that $\mathcal{M}(\alpha)$ is in state q_i, at time t, scanning the cell x;
- $S_j(t,x)$ indicates that $\mathcal{M}(\alpha)$ contains the symbol s_j, at time t, on cell x;

[3]The subscript *FOL* on Δ aims to emphasize the fact that we are considering the *classical* FOL as the underlying logic of the theory, i.e. $\Delta_{FOL}\vdash A$ means $\Delta\vdash_{FOL} A$. A different subscript will indicate that another (non-classical) FOL is being taken into consideration.

- $<(x,y)$ indicates that x is less than y, in the standard order of integer numbers;
- $'(x)$ indicates the successor of x;
- 0 indicates the number 0.

To simplify notation, we will use $x<y$ instead of $<(x,y)$ and x' instead of $'(x)$. The theory $\Delta_{FOL}(\mathcal{M}(\alpha))$ consists of the following axioms:

- Axioms establishing the properties of $'$ and $<$:

$$\forall z \exists x (z = x'), \tag{A1}$$

$$\forall z \forall x \forall y (((z = x') \wedge (z = y')) \rightarrow (x = y)), \tag{A2}$$

$$\forall x \forall y \forall z (((x < y) \wedge (y < z)) \rightarrow (x < z)), \tag{A3}$$

$$\forall x (x < x'), \tag{A4}$$

$$\forall x \forall y ((x < y) \rightarrow (x \neq y)). \tag{A5}$$

- An axiom for each instruction i_j of $\mathcal{M}$. The axiom is defined depending, respectively, on the instruction type (I), (II) or (III) as:

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x) \wedge S_k(t',x) \wedge \forall y \Big((y \neq x) \rightarrow \Big(\bigwedge_{i=1}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Big) \Bigg), \tag{Ai_j (I)}$$

$$\forall t \forall x \Bigg(\Big(Q_i(t,x) \wedge S_j(t,x) \Big) \rightarrow \Big(Q_l(t',x') \wedge \forall y \Big(\bigwedge_{i=1}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Bigg), \tag{Ai_j (II)}$$

$$\forall t \forall x \Bigg(\Big(Q_i(t,x') \wedge S_j(t,x') \Big) \rightarrow \Big(Q_l(t',x) \wedge \forall y \Big(\bigwedge_{i=1}^{m} \big(S_i(t,y) \rightarrow S_i(t',y) \big) \Big) \Big) \Bigg). \tag{Ai_j (III)}$$

- An axiom to specify the initial configuration of the machine. Considering the input $\alpha = s_{i_0} s_{i_1} \ldots s_{i_{p-1}}$, where p represents the length of α, this axiom is defined by:

$$Q_1(0,0) \wedge \left(\bigwedge_{j=0}^{p-1} S_{i_j}(0,0^j) \right) \wedge \forall y \left(\left(\bigwedge_{j=0}^{p-1} y \neq 0^j \right) \rightarrow S_1(0,y) \right), \tag{Aα}$$

  where 0^j means j iterations of the successor ($'$) function to constant 0.

In [4], a sentence H is defined to represent the halting of the computation, and it is thus proved that $\Delta_{FOL}(\mathcal{M}(\alpha)) \vdash H$ iff the machine $\mathcal{M}$ with input α halts. In this way, the decidability of FOL implies the solution for the *halting problem*, a well-known unsolvable problem; this proves (by *reductio ad absurdum*) the undecidability of FOL. For Boolos and Jeffrey's aims, $\Delta_{FOL}(\mathcal{M}(\alpha))$ theories are strong enough, but our purpose here is to attain a precise logical representation of TM computations. Therefore, we will formally define the notion of representability of a TM computation and show that new axioms must be added to $\Delta_{FOL}(\mathcal{M}(\alpha))$ theories. Our definition of the representation of a TM computation (Definition 3) is founded upon the definitions of the representation of functions and relations (Definitions 1 and 2) in theories introduced by Alfred Tarski in collaboration with Andrzej Mostowski and Raphael M. Robinson in [27].

Definition 1
Let f be a function of arity k, Δ an arbitrary theory and $\varphi(x_1,\dots,x_k,x)$ a wff (with $k+1$ free variables) in Δ. The function f is *represented* by φ in Δ if $f(m_1,\dots,m_k)=n$ implies (bars are used to denote numerals):

(1) $\Delta\vdash\varphi(\bar{m}_1,\dots,\bar{m}_k,\bar{n})$,
(2) if $n\neq p$ then $\Delta\vdash\neg\varphi(\bar{m}_1,\dots,\bar{m}_k,\bar{p})$ and
(3) $\Delta\vdash\varphi(\bar{m}_1,\dots,\bar{m}_k,\bar{q})\rightarrow\bar{q}=\bar{n}$.

Definition 2
Let R be a relation of arity k, Δ an arbitrary theory and $\varphi(x_1,\dots,x_k)$ a wff (with k free variables) in Δ. The relation R is *represented* by φ in Δ if:

(1) $(m_1,\dots,m_k)\in R$ implies $\Delta\vdash\varphi(\bar{m}_1,\dots,\bar{m}_k)$ and
(2) $(m_1,\dots,m_k)\notin R$ implies $\Delta\vdash\neg\varphi(\bar{m}_1,\dots,\bar{m}_k)$.

Definition 3
Let $\mathcal{M}$ be a TM, α the input for $\mathcal{M}$ and $\mu(\mathcal{M}(\alpha))=\langle\mathbb{Z},Q_1^\mu,Q_2^\mu,\dots,Q_n^\mu,S_1^\mu,S_1^\mu,\dots,S_m^\mu,<^\mu,'^\mu,0^\mu\rangle$ the structure determined by the intended interpretation $\mathcal{I}$.[4]. A theory Δ, in the language $\mathcal{L}=\{Q_1,Q_2,\dots,Q_n,S_0,S_1,\dots,S_{m-1},<,',0\}$, *represents the computation of* $\mathcal{M}(\alpha)$ if:

(1) $<^\mu$ is represented by $\varphi(x,y):=x<y$ in Δ,
(2) $'^\mu$ is represented by $\varphi(x,y):=x'=y$ in Δ,
(3) Q_i^μ $(i=1,\dots,n)$ are represented by $\varphi(x,y):=Q_i(x,y)$ in Δ, and
(4) S_j^μ $(j=1,\dots,m)$ are represented by $\varphi(x,y):=S_j(x,y)$ in Δ.

Theorem 4
Let $\mathcal{M}$ be a TM and α the input for $\mathcal{M}$. The theory $\Delta_{FOL}(\mathcal{M}(\alpha))$ cannot represent the computation of $\mathcal{M}(\alpha)$.

Proof. We show that condition 2 of Definition 2 cannot be satisfied for relations Q_i and S_j: indeed, when $\mathcal{M}(\alpha)$ is in state q_i, at time t and position x, it is not in any other state q_j $(i\neq j)$; in this case, we have that $\Delta_{FOL}(\mathcal{M}(\alpha))\vdash Q_i(\bar{t},\bar{x})$ (by the proof in [4, Chapter 10]), but on the other hand we have that $\Delta_{FOL}(\mathcal{M}(\alpha))\nvdash\neg Q_j(\bar{t},\bar{x})$, because a non-standard TM with the same instructions of $\mathcal{M}$ (but allowing multiple simultaneous states: starting the computation in two-different simultaneous states, for example) also validates all axioms in $\Delta_{FOL}(\mathcal{M}(\alpha))$. A similar situation occurs with relations S_j. We can also define other non-standard TMs which allow different symbols and states, on different positions of the tape, at times before the beginning or after the end of the computation, in such a way that the machine validates all axioms in $\Delta_{FOL}(\mathcal{M}(\alpha))$. ■

Theorem 4 shows that it is necessary to expand the theories $\Delta_{FOL}(\mathcal{M}(\alpha))$ in order to disallow non-standard interpretations and to grant representation of computations in accordance with Definition 3. We thus define the notion of an *intrinsic theory of the computation of* $\mathcal{M}(\alpha)$ as the theory

[4] $\mathbb{Z}$ represents the integers, the relations Q_i^μ express couples of instants of time and positions for states q_i in the computation of $\mathcal{M}(\alpha)$, relations S_j^μ express couples of instants of time and positions for symbols s_j in the computation of $\mathcal{M}(\alpha)$, $<^\mu$ is the standard strict order on $\mathbb{Z}$, $'^\mu$ is the successor function on $\mathbb{Z}$ and 0^μ is the integer 0.

$\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$ by specifying which new axioms have to be added to $\Delta_{FOL}(\mathcal{M}(\alpha))$ theories, so that these extended theories are able to represent their respective TM computations (Theorem 5). For the specification of such axioms, we will suppose that before the beginning of any computation and after the end of any computation (if the computation halts), the machine is in none of its states and no symbols (not even the empty symbol) occurs anywhere in its tape. New axioms are defined as follows:

- An axiom to define the situation of $\mathcal{M}(\alpha)$ before the beginning of the computation:

$$\forall t \forall x \left((t<0) \rightarrow \left(\left(\bigwedge_{i=1}^{n} \neg Q_i(t,x) \right) \wedge \left(\bigwedge_{j=1}^{m} \neg S_j(t,x) \right) \right) \right). \tag{At0}$$

- An axiom to define the situation of $\mathcal{M}(\alpha)$ after the end of the computation (if the computation halts):

$$\forall t \forall x \left(\neg \left(\bigvee_{q_i s_j \in I} \left(Q_i(t,x) \wedge S_j(t,x) \right) \right) \rightarrow \right.$$
$$\left. \forall u \forall y \left(t<u \rightarrow \left(\left(\bigwedge_{i=1}^{n} \neg Q_i(u,y) \right) \wedge \left(\bigwedge_{j=1}^{m} \neg S_j(u,y) \right) \right) \right) \right), \tag{Ath}$$

  where subscript $q_i s_j \in I$ means that, in the disjunction, only combinations of $q_i s_j$ coincident with the first two symbols of some instruction of $\mathcal{M}$ are taken into account.
- An axiom for any state symbol q_i of $\mathcal{M}$ establishing the uniqueness of any state and any position in a given instant of time:

$$\forall t \forall x \left(Q_i(t,x) \rightarrow \left(\left(\bigwedge_{j \neq i} \neg Q_j(t,x) \right) \wedge \forall y \left(y \neq x \rightarrow \bigwedge_{i=1}^{n} \neg Q_i(t,y) \right) \right) \right). \tag{Aq_i}$$

- An axiom for any read/write symbol s_i of $\mathcal{M}$ establishing the uniqueness of any symbol in a given instant of time and position:

$$\forall t \forall x \left(S_i(t,x) \rightarrow \bigwedge_{i \neq j} \neg S_j(t,x) \right). \tag{As_j}$$

THEOREM 5
Let $\mathcal{M}$ be a TM and α the input for $\mathcal{M}$. Then, the intrinsic theory $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$ represents the computation of $\mathcal{M}(\alpha)$.

PROOF. Representation for the relation $<^{\mu}$ and for the function $'^{\mu}$ is easy to proof. Representation for relations Q_i^{μ} and S_j^{μ} follows from the proof in [4, Chapter 10] and direct applications of the new axioms in the intrinsic theory $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$. ■

The definitions and theorems above consider only DTMs (i.e. TMs with no pairs of instructions with the same two initial symbols); the next theorem establishes that the method of axiomatization defined above, when used to NDTMs, produces contradictory theories (in some cases).

Theorem 6
Let $\mathcal{M}$ be a NDTM and α an input for $\mathcal{M}$. If $\mathcal{M}(\alpha)$ reaches an ambiguous configuration (i.e. a configuration where multiple instructions can be executed), then its intrinsic theory $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$ is contradictory.

Proof. By the proof in [4, Chapter 10], it is deduced a formula that expresses the ambiguous configuration. Then, by using theorems corresponding to the possible instructions that can be executed in the ambiguous configuration, there are deduced formulas expressing multiplicity of states, positions or symbols in some cell of the tape. Thus, by using axiom ($\mathrm{A}q_i$) or ($\mathrm{A}s_j$), a contradiction is deduced. ■

In [24, p. 48], Odifreddi, in his definition of a TM, establishes a condition of 'consistency' for the machine disallowing the existence of 'contradictory' instructions (i.e. instructions with the same two initial symbols), which corresponds to the notion of DTM. Thus, NDTMs are those that do not accomplish the condition of consistency. Theorem 6 shows that Odifreddi's idea of consistency in TMs coincides with the consistency of the intrinsic theories $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$.

Note that contradictions in intrinsic theories $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$ arise by the multiple use of axioms ($\mathrm{A}i_{\mathrm{j}}$ (I)), ($\mathrm{A}i_{\mathrm{j}}$ (II)) and ($\mathrm{A}i_{\mathrm{j}}$ (III)) for the same instance of t in combination with the use of axioms ($\mathrm{A}q_i$) or ($\mathrm{A}s_j$). The multiple use of axioms ($\mathrm{A}i_{\mathrm{j}}$ (I)), ($\mathrm{A}i_{\mathrm{j}}$ (II)) and ($\mathrm{A}i_{\mathrm{j}}$ (III)) for the same instance of t indicates the simultaneous execution of multiple instructions, which can derive (at time $t+1$) multiplicity of symbols in the cell of the tape where the instruction is executed, or multiplicity of states and positions, while axioms ($\mathrm{A}q_i$) and ($\mathrm{A}s_j$) establish the uniqueness of such elements. Intrinsic theories $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$ can be easily adapted to deal with the idea that only one instruction is chosen to be executed when the machine reaches an ambiguous configuration, obtaining adequate theories for NDTMs computations. However, this is not our focus in this article. We are interested in generalizing the notion of TM by using a paraconsistent logic, as this is a fecund way of approaching quantum computing and computational complexity from a logical viewpoint.

3 Paraconsistent TMs

There are many paraconsistent logics. They are proposed from different philosophical perspectives but share the common feature of being logics which support contradictions without falling into deductive trivialization. Although in the definition of ParTMs we could, in principle, depart from any first-order paraconsistent logic, we will use the logic $LFI1^*$ (see [9]) because it possesses an already established proof theory and a first-order three-valued semantics, as well as properties that allows natural interpretations of consequences of $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$ theories[5] as 'paraconsistent computations', and also allows the addition of conditions to control the execution of instructions involving multiplicity of symbols and states.[6] $LFI1^*$ is the first-order extension of $LFI1$, which by its turn extends positive classical logic by introducing connectives of consistency $\circ$ and inconsistency $\bullet$. In particular, in $LFI1^*$ inconsistency is identified with contradiction by means of the equivalence $\bullet A \leftrightarrow (A \wedge \neg A)$. The connective $\circ$ is a powerful operator which permits to distinguish, in principle, between consistency and non-contradiction by taking the notion of consistency as primitive in the object language, and offers a unified approach to almost all paraconsistent logics (see [10] for details and for a more elaborate discussion).

[5]Intrinsic theories $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$ are obtained by substituting the underlying logic of $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$ theories by $LFI1^*$.
[6]It is worth to remark that the choice of another paraconsistent logic, with other features, can lead to different notions of ParTMs, as is the case in Section 3.1.2.

The system $LFI1^*$ is a member of the class of the Logics of Formal Inconsistency (LFIs) and there are translations from classical and paraconsistent FOLs into $LFI1^*$ and back. This means that, despite its status as a subsystem of classical logic, $LFI1^*$ can codify any classical reasoning.

For intrinsic theories $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$, the proof in [4, Chapter 10] continues to hold, because $LFI1^*$ is an extension of positive classical logic. Thus, as described above, the use of multiple axioms describing instructions for the same instance of t indicates simultaneous execution of the instructions, which gives place to multiplicity of symbols in the cell and multiplicity of states and positions. Such a multiplicity, in conjunction with axioms ($\mathrm{A}q_i$) and ($\mathrm{A}s_j$), entails contradictions which are identified in $LFI1^*$ with inconsistencies. Thus, inconsistency in $\Delta^{\star}_{LFI1^*}(\mathcal{M}(\alpha))$ theories characterize multiplicity.

By taking advantage of the robustness of $LFI1^*$ in the presence of inconsistencies and their interpretation as multiplicity, we can supply the ParTMs with conditions of inconsistency in the two initial symbols of instructions in order to control the process of computation. $q_i^{\bullet}$ will indicate that the instruction will only be executed in configurations where the machine is in multiple states or multiple positions, and $s_j^{\bullet}$ will indicate that the instruction will only be executed in cells with multiple symbols. These conditions correspond to put the connective $\bullet$, respectively, in front of the predicate Q_i or S_j in the antecedent of the axioms related to the instructions. These apparently innocuous conditions are essential for taking advantage of the parallelism provided by ParTMs and EParTMs. As will be argued below, inconsistency conditions on the instructions seem to be a more powerful mechanism than quantum interference, which is the instrument provided by quantum computation taking advantage of quantum parallelism.

Note that axioms ($\mathrm{A}i_{\mathrm{j}}$ (I)), ($\mathrm{A}i_{\mathrm{j}}$ (II)) and ($\mathrm{A}i_{\mathrm{j}}$ (III)) not only express the action of instructions but also specify the preservation of symbols unmodified by the instructions. Thus, in ParTMs, we have to take into account that any instruction is executed in a specific position on the tape, carrying symbols from cells not modified by the instruction to the next instant of time; this is completed independently of the execution of other instructions.

A ParTM is then defined as:

Definition 7
A *ParTM* is a NDTM such that:

- When the machine reaches an ambiguous configuration, it *simultaneously* executes all possible instructions, which can produce multiplicity on states, positions and symbols in some cells of the tape;
- Each instruction is executed in the position corresponding to the respective state; symbols in cells unmodified by the instructions are carried to the next instant of time;
- *Inconsistency* (or *multiplicity*) conditions are allowed on the first two symbols of the instructions (as described above); and
- The machine stops when there are no instructions to be executed; at this stage some cells of the tape can contain multiple symbols, any choice of them represents a result of the computation.

The next example illustrates how a ParTM performs computations:

Example 8
Let $\mathcal{M}$ be a ParTM with instructions: $i_1 : q_1 0 0 q_2$, $i_2 : q_1 0 1 q_2$, $i_3 : q_2 0 R q_3$, $i_4 : q_2 1 R q_3$, $i_5 : q_3 \emptyset 1 q_4$, $i_6 : q_4 0 0 q_5$, $i_7 : q_4 1 0 q_5$, $i_8 : q_4 1^{\bullet} * q_5$, $i_9 : q_5 * 1 q_5$. Figure 1 schematizes the computation of $\mathcal{M}$, beginning in position 0, state q_1 and reading the symbol 0 (with symbol $\emptyset$ in all other cells of the tape). Instructions to be executed in each instant of time t are written within parentheses (note that instruction

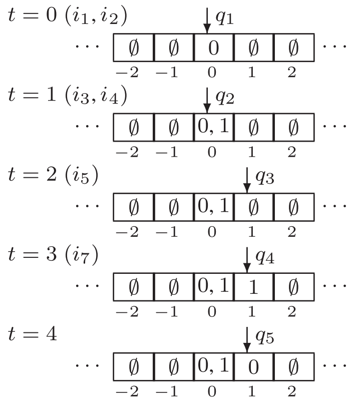

FIGURE 1. Example of computation in a ParTM.

i_8 is not executed at time $t = 3$ because of the inconsistency condition on the scanned symbol). $\mathcal{M}$ will be useful in the paraconsistent solution of Deutsch's and Deutsch-Jozsa problems (Section 3.1.1).

3.1 Simulating quantum computation through paraconsistent TMs

In the Neumann–Dirac formulation, quantum mechanics is synthesized in four postulates (cf. [23, Section 2.2]): The first postulate establishes that states of isolated physical systems are represented by unit vectors in a Hilbert space (known as the *space state* of the system); the second postulate claims that the evolution of closed quantum systems is described by unitary transformations in the Hilbert space; the third postulate deals with observations of physical properties of the system by relating physical properties with Hermitian operators (called *observables*) and establishing that, when a measurement is performed, an eigenvalue of the observable is obtained (with a certain probability depending upon the state of the system) and the system collapses to its respective eigenstate; finally, the fourth postulate establishes the tensor product of the state spaces of the component systems as being the state space of the compound system, allowing us to represent the state of a compound system as the tensor product of the state of its subsystems (when the states of the subsystems are known).

The best known models of quantum computation, namely *quantum Turing machines* (QTMs) and *quantum circuits* (QCs), are direct generalizations of TMs and boolean circuits, respectively, using the laws of quantum mechanics.

By taking into account the postulates of quantum mechanics briefly described above, a QTM (introduced in [16]) is defined by considering elements of a TM (state, position and symbols on the tape) as being observables of a quantum system. The configuration of a QTM is thus represented by a unit vector in a Hilbert space and the evolution is described by a unitary operator (with some restrictions in order to satisfy the requirement that the machine operates by 'finite means', see [16, p. 7] and [25]). Since the configuration of a QTM is described by a unit vector, it is generally a linear superposition of basis states (called a *superposed state*), where the basis states represent classical TM configurations. In quantum mechanics, superposed states can be interpreted as the coexistence of multiple states, thus a QTM configuration can be interpreted as the simultaneous existence of multiple classical TM configurations. The linearity of the operator describing the evolution of a QMT allows us to think in the parallel execution of the instructions over the different states (possibly an exponential number) present in the superposition. Unfortunately, to know the result of the computation, we have to perform a measurement of the system and, by the third postulate of quantum mechanics, we can obtain only one classical TM configuration in a probabilistic way, in effect, irredeemably losing all other configurations. The art of 'quantum programming' consists in taking advantage of the intrinsic parallelism of the model, by using quantum interference[7] to increase amplitudes of desired states before the measurement of the system, with the aim to solve problems more efficiently than in the classical case.

As shown in [25], the evolution of a QTM can be equivalently specified by a *local transition function*[8] $\delta : Q \times \Sigma \times \{\Sigma \cup \{R,L\}\} \times Q \to \mathbb{C}$, which validates some conditions related to the unitary of state vectors and the reversibility of operators. In this definition, the transition $\delta(q_i, s_j, Op, q_l) = c$ can be interpreted as the following action of the QTM: if the machine is in state q_i reading the symbol s_j, it follows with the probability amplitude c that the machine will perform the operation Op (which can be either to write a symbol or to move on the tape) and reaches the state q_l. The amplitude c cannot be interpreted as the probability of performing the respective transition, as with probabilistic TMs. Indeed, QTMs do not choose only one transition to be executed; they can perform multiple transitions simultaneously in a single instant of time in a superposed configuration. Moreover, configurations resulting from different transitions can interfere constructively or destructively (if they represent the same classical configuration), respectively, increasing or decreasing the amplitude of the configuration in the superposition.

By taking into account that each choice function on the elements of a ParTM, in an instant of time t, gives a classical TM configuration; a configuration of a ParTM can be viewed as a *uniform*[9] superposition of classical TM configurations. This way, ParTMs seem to be similar to QTMs: we could see ParTMs as QTMs without amplitudes (which allows us only to represent uniform superpositions). However, in ParTMs, actions performed by different instructions mix indiscriminately, and thus all combinations of the singular elements in a ParTM configuration are taken into account, which makes it impossible to represent entangled states by only considering the multiplicity of elements as superposed states (this point is discussed in Section 3.1.2). Another difference between the ParTMs and QTMs models is that 'superposed states' in the former model do not supply a notion of relative phase (corresponding to signs of basis states in uniform superpositions), an important feature of quantum superpositions required for quantum interference. Such a feature, as mentioned before, is the key mechanism for taking advantage of quantum parallelism. However, inconsistency conditions on the instructions of ParTMs allows us to take advantage of 'paraconsistent parallelism', and this seems to be a more powerful property than quantum interference (this point is fully discussed in Section 3.1.2). In spite of the differences between ParTMs and QTMs, ParTMs are able to simulate important features of quantum computing; in particular, they can simulate uniform non-entangled superposed quantum states and solve the Deutsch and Deutsch-Jozsa problems preserving the efficiency of the quantum algorithms, but with certain restrictions (see Section 3.1.1). In Section 3.1.2, we define another model of ParTMs, based on a paraconsistent logic endowed with a 'non-separable' conjunction, which enables the simulation of uniform entangled states and

[7]Quantum interference is expressed by the addition of amplitudes corresponding to equal basis states in a superposed state. When signs of amplitudes are the same, their sum obtains a greater amplitude; in this case, we say that the interference is *constructive*. Otherwise, amplitudes subtract and we say that the interference is *destructive*. Quantum interference occurs in the evolution of one quantum state to another.

[8]Some changes were made in the definition of δ to deal with the quadruple notation for instructions we are using here.

[9]A superposed state is said to be *uniform* if all states in the superposition, with amplitude different of 0, have amplitudes with the same magnitude.

represents a better approach for the model of QTMs. We also show that a notion of ‘relative phase’ can be introduced in this new model of computation.

3.1.1 Paraconsistent solutions for Deutsch and Deutsch-Jozsa problems

Given an arbitrary function $f:\{0,1\}\to\{0,1\}$ and an ‘oracle’ (or black box) that computes f, Deutsch’s problem consists in defining a procedure to determine if f is *constant* ($f(0)=f(1)$) or *balanced* ($f(0)\neq f(1)$) allowing only one query to the oracle. Classically, the procedure seems to require two queries to the oracle in order to compute $f(0)$ and $f(1)$, plus a further step for the comparison; but by taking advantage of the quantum laws the problem can be solved in a more efficient way, by executing just a single query. A probabilistic quantum solution to Deutsch’s problem was first proposed in [16] and a deterministic quantum algorithm was given in [12]. The deterministic solution is usually formulated in the QCs formalism, so we briefly describe this model of computation before presenting the quantum algorithm.

The model of QCs (introduced in [17]) is defined by generalizing the boolean circuit model in accordance with the postulates of quantum mechanics: the classical unit of information, the *bit*, is generalized as the *quantum bit* (or *qubit*), which is mathematically represented by a unit vector in a two-dimensional Hilbert space; classical logic gates are replaced by unitary operators; registers of qubits are represented by tensor products and measurements (following conditions of the third postulate above) are accomplished at the end of the circuits in order to obtain the output of the computation.[10] Under this definition, the QC depicted in Figure 2 represents a deterministic solution to Deutsch’s problem.

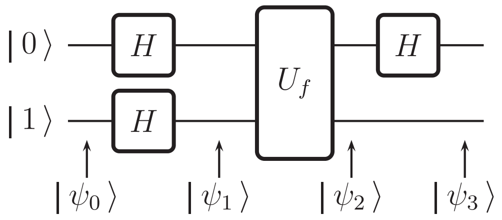

FIGURE 2. QC to solve Deutsch’s problem.

In the figure, squares labeled by H represent *Hadamard* gates. A Hadamard gate is a quantum gate which performs the following transformations ($|\cdot\rangle$ representing a vector in Dirac’s notation):

$$\begin{aligned} H\!:\!|0\rangle &\mapsto \frac{1}{\sqrt{2}}\big(|0\rangle+|1\rangle\big)\\ |1\rangle &\mapsto \frac{1}{\sqrt{2}}\big(|0\rangle-|1\rangle\big). \end{aligned} \tag{1}$$

The rectangle labeled by U_f represents the quantum oracle that performs the operation $U_f(|x,y\rangle)=|x,y\oplus f(x)\rangle$, where $|x,y\rangle$ represents the tensor product $|x\rangle\otimes|y\rangle$ and $\oplus$ represents the addition module 2. Vectors $|\psi_i\rangle$ are depicted to explain, step by step, the process of computation:

(1) At the beginning of the computation, the input register takes the value $|\psi_0\rangle=|0,1\rangle$;

[10]For a detailed introduction to QCs see [23].

(2) After performing the two first Hadamard gates, the following superposition is obtained:

$$|\psi_1\rangle = H|0\rangle \otimes H|1\rangle = \frac{1}{2}\big((|0\rangle + |1\rangle)\otimes(|0\rangle - |1\rangle)\big); \tag{2}$$

(3) By applying the operation U_f, one obtains:

$$\begin{aligned}|\psi_2\rangle &= U_f\left(\frac{1}{2}\big(|0,0\rangle - |0,1\rangle + |1,0\rangle - |1,1\rangle\big)\right) \\ &= \frac{1}{2}\big(|0,0\oplus f(0)\rangle - |0,1\oplus f(0)\rangle + |1,0\oplus f(1)\rangle - |1,1\oplus f(1)\rangle\big) \\ &= \frac{1}{2}\Big((-1)^{f(0)}(|0\rangle\otimes(|0\rangle - |1\rangle)) + (-1)^{f(1)}(|1\rangle\otimes(|0\rangle - |1\rangle))\Big) \\ &= \begin{cases} \pm\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)\otimes\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) & \text{if } f(0) = f(1), \\ \pm\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right)\otimes\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) & \text{if } f(0) \neq f(1). \end{cases}\end{aligned} \tag{3}$$

(4) By applying the last Hadamard gate, one finally reaches:

$$|\psi_3\rangle = \begin{cases} \pm|0\rangle\otimes\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) & \text{if } f(0) = f(1), \\ \pm|1\rangle\otimes\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) & \text{if } f(0) \neq f(1). \end{cases} \tag{4}$$

After measurement of the first qubit of the state $|\psi_3\rangle$ is accomplished (on the standard basis, cf. [23]), one obtains 0 (with probability 1) if f is constant or 1 (with probability 1) if f is balanced.

The first step of the above QC generates a superposed state (Equation (2)), which is taken into the next step to compute the function f in parallel (Equation (3)), generating a superposition in such a way that the relative phase of $|1\rangle$ in the first qubit differs depending on if f is constant or balanced. By applying again a Hadamard gate on the first qubit, quantum interference acts leaving the first qubit in the basis state $|0\rangle$ if f is constant or in the basis state $|1\rangle$ if f is balanced (Equation (4)). Thus, by performing a measurement of the first qubit, we determine with certainty if f is constant or balanced. Note that U_f is used only once in the computation.

The ParTM in Example 8 gives a 'paraconsistent' simulation of the quantum algorithm that solves Deutsch's problem, for the particular case where f is the constant function 1. Instructions i_1 and i_2, executed simultaneously at time $t = 0$, simulate the generation of the superposed state. Instructions i_3 to i_5 compute the constant function 1 over the superposed state, performing in parallel the computation of $f(0)$ and $f(1)$, and writing the results on position 1 of the tape. Instructions i_6 to i_9 check whether $f(0) = f(1)$, writing 0 on position 1 of the tape if there is no multiplicity of symbols on the cell (meaning that f is constant) or writing 1 in another case (meaning that f is balanced). In the present case $f(0) = f(1) = 1$, the execution of the instructions from i_6 to i_9 gives as result the writing of 0 on position 1 of the tape.

Consider a TM $\mathcal{M}'$ a black box that computes a function $f : \{0,1\} \to \{0,1\}$. We could substitute instructions i_3 to i_5 in Example 8 (adequately renumbering instructions and states from i_6 to i_9 if necessary) with the instructions from $\mathcal{M}'$ in order to determine if f is constant or balanced. In this way, we define a paraconsistent simulation of the quantum algorithm that solves Deutsch's problem. In the simulation, $\mathcal{M}'$ is the analogue of U_f and quantum parallelism is mimicked by the parallelism provided by the multiplicity allowed in the ParTMs.

Notwithstanding the parallelism provided by ParTMs, this first paraconsistent model of computation has some peculiar properties which could give rise to 'anomalies' in the process of computation. For instance, consider a TM $\mathcal{M}'$ with instructions: $i_1 = q_1 00 q_2$, $i_2 = q_1 11 q_3$, $i_3 = q_2 0 R q_4$, $i_4 = q_3 1 R q_4$, $i_5 = q_2 1 R q_5$, $i_6 = q_4 \emptyset 1 q_4$, $i_7 = q_5 \emptyset 0 q_5$. If $\mathcal{M}'$ starts the computation on position 0 and state q_1, with a single symbol (0 or 1) on position 0 of the tape, and with symbol $\emptyset$ on all other cells of the tape, then $\mathcal{M}'$ computes the constant function 1. Nevertheless, if $\mathcal{M}'$ begins reading symbols 0 and 1 (both simultaneously on cell 0), then it produces 0 and 1 as its outputs, as if $\mathcal{M}'$ had computed a balanced function. This example shows well how different paths of a computation can mix indiscriminately, producing paths of computation not possible in the TM considered as an oracle (when viewed as a classical TM) and generating undesirable results. Therefore, only TMs that exclusively perform their possible paths of computation when executed on a superposed state can be considered as oracles. TMs with this property will be called *parallelizable*. Note that this restriction is not a serious limitation to our paraconsistent solution of Deutsch's problem, because parallelizable TMs that compute any function $f : \{0,1\} \to \{0,1\}$ are easy to define.

The Deutsch-Jozsa problem, first presented in [18], is the generalization of Deutsch's problem to functions $f : \{0,1\}^n \to \{0,1\}$, where f is assumed to be either constant or balanced.[11] A quantum solution to the Deutsch-Jozsa problem is a direct generalization of the quantum solution to Deutsch's problem presented above: the input register is now constituted of $n+1$ qubits and takes the value $|0\rangle^{\otimes n} \otimes |1\rangle$ (where $|\cdot\rangle^{\otimes n}$ represents the tensor product of n qubits $|\cdot\rangle$); new Hadamard gates are added to act on the new qubits in the input register and also on the first n outputs of U_f, and U_f is now a black box acting on $n+1$ qubits, performing the operation $U_f(|x_1,\ldots,x_n,y\rangle) = |x_1,\ldots,x_n,y \oplus f(x_1,\ldots,x_n)\rangle$. In this case, when a measurement of the first n qubits is accomplished at the end of the computation, if all the values obtained are 0, then f is constant (with probability 1); in another case f is balanced (with probability 1).[12]

The paraconsistent solution to Deutsch's problem can be easily generalized to solve the Deutsch-Jozsa problem as well: the input to $\mathcal{M}$ must be a sequence of n symbols 0; instructions i_1 and i_2 must be substituted by instructions $i_1 = q_1 00 q_2$, $i_2 = q_1 01 q_2$, $i_3 = q_1 0 R q_1$, $i_4 = q_1 \emptyset L q_3$, $i_5 = q_3 0 L q_3$, $i_6 = q_3 \emptyset R q_4$, and the machine $\mathcal{M}'$ is now considered to be a parallelizable TM computing a constant or balanced function $f : \{0,1\}^n \to \{0,1\}$.

Note that the paraconsistent solution to the Deutsch-Jozsa problem does not depend on the assumption of the function to be constant or balanced; in fact, the solution can be applied in order to distinguish between constant and non-constant functions. This can lead to the erroneous conclusion that PartTMs are too powerful machines, in which all NP-Problems could be solved in polynomial time: someone could mistakenly think that, considering an oracle evaluating propositional formulas in polynomial time, we could immediately define a ParTM solving SATISFIABILIY in polynomial time (by defining, for instance, instructions to set values 0 and 1 to propositional variables, invoking the oracle to simultaneously evaluate all possible values of the formula, and then using instructions with inconsistent conditions to establish whether any value 1 was obtained). However, this is not the case, because only parallelizable TMs can be taken as oracles.

As proven in Theorem (14), ParTMs can be efficiently simulated by DTMs. Then, if we had a ParTM solving SATISFIABILITY in polynomial time, this would lead to the surprising result that $P = NP$. To avoid such a mistake, we have to take into account the restriction of parallelizability imposed to oracles in our model: if we had a parallelizable TM to evaluate propositional formulas, it would be easy to define a ParTM solving SATISFIABILIY in polynomial time and, by Theorem (14),

[11] f is balanced if $\overline{\overline{f^{-1}(0)}} = \overline{\overline{f^{-1}(1)}}$, where $\overline{\overline{A}}$ represents the cardinal of the set A.
[12] Calculations are not presented here, for details see [23].

we would conclude $P=NP$. This only shows the difficulty (or impossibility) in defining a parallelizable TM able to evaluate propositional formulas.

On the other hand, Grover's quantum search algorithm and its proven optimality (see [23, Chapter 6]) implies the non-existence of a 'naive' search-based method to determine whether a function is constant or not in a time less than $O(\sqrt{2^n})$. This shows that, in order to take advantage of parallelism, inconsistency conditions on instructions featured by ParTMs is a more powerful property than quantum interference. However, in the case of ParTMs, this feature does not allow us to define more efficient algorithms than otherwise defined by classical means. The reason is that the different paths of computations may mix in this model, and consequently we have to impose the parallelizability restriction on oracles. In the EParTMs model defined in the next section, different paths of computation do not mix indiscriminately as in ParTMs. Thus, no restrictions on the oracles are necessary, and, as it is shown in Theorem (16), this new model of computation solves all NP-problems in polynomial time. This result shows that conditions of inconsistency on the instructions are a really efficient method to take advantage of parallelism, and that this mechanism is more powerful than quantum interference.

3.1.2 Simulating entangled states and relative phases

In quantum theory, if we have n physical systems with state spaces $H_1,\ldots,H_n$, respectively, the system composed by the n systems has the state space $H_1\otimes\ldots\otimes H_n$ (in this case, $\otimes$ represent the tensor product between state spaces) associated to it. Moreover, if we have that the states of the n component systems are, respectively, $|\psi_1\rangle,\ldots,|\psi_n\rangle$, then the state of the composed system is $|\psi_1\rangle\otimes\ldots\otimes|\psi_n\rangle$. However, there are states in composed systems that cannot be described as tensor products of the states of the component systems; these states are known as *entangled states*. An example of a two qubit entangled state is $|\psi\rangle=\frac{1}{\sqrt{2}}(|00\rangle+|11\rangle)$.

Entangled states enjoy the property that a measurement of one state in the component system affects the state of other component systems, even when systems are spatially separated. In this way, singular (one particle) systems lose identity, because their states are only describable in conjunction with other systems. Entanglement is one of the more (if not the most) puzzling characteristics of quantum mechanics, with no analogue in classical physics. Many quantum computing researchers think that entangled states play a definite role in the definition of efficient quantum algorithms, but this is not a completely established fact; any elucidation about this would be of great relevance. In this direction, we are going to show how the concept of entanglement can be expressed in logical terms, and we will define a new model of paraconsistent TMs (EParTMs) in which uniform entangled states are well represented.

As mentioned before, choice functions over the different elements (state, position and symbol on the cells of the tape) of a ParTM, in a given instant of time t, determine a classical TM configuration. Then, a configuration of a ParTM can be viewed as a uniform superposition of classical TM configurations where all combinations of singular elements are taken into account. Ignoring amplitudes, the tensor product of composed systems coincides with all combinations of the basis states present (with amplitude greater than 0) in the component systems. For instance, if a system S_1 is in state $|\psi_1\rangle=|a_{i_1}\rangle+\cdots+|a_{i_n}\rangle$ and a system S_2 is in state $|\psi_2\rangle=|b_{j_1}\rangle+\cdots+|b_{j_m}\rangle$, then the composed system of S_1 and S_2 is in state $|\psi_{1,2}\rangle=|a_{i_1}b_{j_1}\rangle+\cdots+|a_{i_1}b_{j_m}\rangle+\cdots+|a_{i_n}b_{j_1}\rangle+\cdots+|a_{i_n}b_{j_m}\rangle$. This rule can be applied $n-1$ times to obtain the state of a system composed by n subsystems. Consequently, just by interpreting multiplicity of elements as superposed states, ParTMs cannot represent entangled states, because all their configurations can be expressed as tensor products of their singular elements. This is why we define the new model of EParTMs, (cf. Definition 10).

In a ParTM configuration, all combinations of its singular elements are taken into account in the execution of its instructions (and also in the reading of the results). This is because the logic $LFI1^*$, used in the definition of the model, validates the *separation rule* (in the sense that $\vdash_{\mathrm{LFI1}^*} A \wedge B$ implies $\vdash_{\mathrm{LFI1}^*} A$ and $\vdash_{\mathrm{LFI1}^*} B$)[13] and the *adjunction rule* (in the sense that $\vdash_{\mathrm{LFI1}^*} A$ and $\vdash_{\mathrm{LFI1}^*} B$ implies $\vdash_{\mathrm{LFI1}^*} A \wedge B$). Then, for instance, if $\Delta^{\star}_{\mathrm{LFI1}^*}(\mathcal{M}(n)) \vdash Q_1(\bar{t},\bar{x}) \wedge S_1(\bar{t},\bar{x})$ and $\Delta^{\star}_{\mathrm{LFI1}^*}(\mathcal{M}(n)) \vdash Q_2(\bar{t},\bar{x}) \wedge S_2(\bar{t},\bar{x})$, it is also possible to deduce $\Delta^{\star}_{\mathrm{LFI1}^*}(\mathcal{M}(n)) \vdash Q_1(\bar{t},\bar{x}) \wedge S_2(\bar{t},\bar{x})$ and $\Delta^{\star}_{\mathrm{LFI1}^*}(\mathcal{M}(n)) \vdash Q_2(\bar{t},\bar{x}) \wedge S_1(\bar{t},\bar{x})$.

By the previous explanation, if we want to define a model of paraconsistent TMs where configurations are not totally mixed, we have to consider a paraconsistent logic where the separation rule or the adjunction rule are not both valid. There exist paraconsistent logics, called *non-adjunctive*, which fail the adjunction rule,[14] but paraconsistent systems where the separation rule fails have never been proposed before. Moreover, despite the fact that non-adjunctive paraconsistent logics appear to be an acceptable solution to avoid the phenomenon of complete mixing in ParTMs, the notion of entanglement seems to be more related with the failure of the separation rule: indeed, an entangled state describes the 'conjunctive' state of a composed system, but not the state of each single subsystem. Thus, in order to define a model of paraconsistent TMs better approaching the behavior of QTMs, we first define a paraconsistent logic with a *non-separable* conjuction.

By following the ideas in [6] (see also [7]), a paraconsistent negation $\neg_\diamond$ is defined into the well-known modal system $S5$ (departing from classical negation $\neg$) by $\neg_\diamond A \stackrel{\text{def}}{=} \diamond\neg A$ (some properties of this negation are presented in the referred papers). We now define a *non-separable* conjunction $\wedge_\diamond$ into $S5$ by $A \wedge_\diamond B \stackrel{\text{def}}{=} \diamond(A \wedge B)$, where $\wedge$ is the classical conjunction. Some properties of this conjunction are the following:

$$\vdash_{S5} A \wedge_\diamond B \text{ does not imply neither } \vdash_{S5} A \text{ nor } \vdash_{S5} B, \tag{$\wedge_\diamond 1$}$$

$$\vdash_{S5} A \text{ and } \vdash_{S5} B \text{ implies } \vdash_{S5} A \wedge_\diamond B, \tag{$\wedge_\diamond 2$}$$

$$\nvdash_{S5} \big(A \wedge_\diamond (B \wedge_\diamond C)\big) \leftrightarrow \big((A \wedge_\diamond B) \wedge_\diamond C\big), \tag{$\wedge_\diamond 3$}$$

$$\vdash_{S5} (A \wedge_\diamond B) \leftrightarrow (B \wedge_\diamond A), \tag{$\wedge_\diamond 4$}$$

$$\nvdash_{S5} \big((A \wedge_\diamond B) \wedge (C \wedge_\diamond D)\big) \rightarrow \big((A \wedge_\diamond D) \vee (C \wedge_\diamond B))\big) \tag{$\wedge_\diamond 5$}$$

$$\vdash_{S5} (A_1 \wedge_\diamond (A_2 \wedge \ldots \wedge A_n)) \leftrightarrow \diamond(A_1 \wedge \ldots \wedge A_n). \tag{$\wedge_\diamond 6$}$$

Property ($\wedge_\diamond 1$) reflects the non-separable character of $\wedge_\diamond$, while ($\wedge_\diamond 2$) shows that $\wedge_\diamond$ validates the adjunction rule and ($\wedge_\diamond 3$) grants the non-associativity of $\wedge_\diamond$. ($\wedge_\diamond 4$) shows that $\wedge_\diamond$ is commutative, ($\wedge_\diamond 5$) is a consequence of ($\wedge_\diamond 1$) related with the expression of entangled states, and ($\wedge_\diamond 6$) is a simple application of the definition of $\wedge_\diamond$ which will be useful below.

A paraconsistent non-separable logic, which we will call $PNS5$, can be 'extracted' from the modal logic $S5$ (as much as done for negation in [6]) by inductively defining a translation $*: ForPNS5 \rightarrow ForS5$ as:[15]

$$A^* = A \text{ if } A \text{ is atomic,}$$

[13]What we call 'separation rule' is sometimes (quite improperly) referred to as 'simplification rule'. Actually what we have is a separation of a proposition as $A \wedge B$ in terms of A and B, and not necessarily a simplification.

[14]The most famous of them is the *discussive* (or *discursive*) logic $D2$, introduced by Stanisław Jaśkowski in [20] and [21], extendable to first-order logic and with applications in the axiomatization of quantum theory, cf. [13].

[15]Where $ForPNS5$ is the set of propositional formulas generated over the signature $\sigma = \{\neg_\diamond, \wedge_\diamond, \vee, \rightarrow\}$ (defined in the usual way) and $ForS5$ is the set of formulas of $S5$.

$$(\neg_\diamond A)^* = \diamond\neg(A)^*,$$
$$(A \wedge_\diamond B)^* = \diamond(A^* \wedge B^*),$$
$$(A\#B)^* = A^*\#B^* \text{ for } \# \in \{\vee, \rightarrow\};$$

and by defining a consequence relation in the wffs of *PNS*5 as:

$$\Gamma \vdash_{PNS5} A \text{ iff } \Gamma^* \vdash_{S5} A^*,$$

where Γ represents a subset of *ForPNS*5 and $\Gamma^* = \{B^* | B \in \Gamma\}$. This translation completely specifies *PNS*5 as a sublogic of *S*5 with the desired properties.

In the spirit of the LFIs (see [10]), we can define a connective $\bullet$ of 'inconsistency' in *PNS*5 by $\bullet A \overset{\text{def}}{=} A \wedge_\diamond \neg_\diamond A$ (which is equivalent to $\diamond A \wedge \diamond\neg A$ in *S*5), a connective $\circ$ of 'consistency' by $\circ A \overset{\text{def}}{=} \neg_\diamond \bullet A$ (which is equivalent to $\Box\neg A \vee \Box A$ in *S*5), a classical negation $\neg$ by $\neg A \overset{\text{def}}{=} \neg_\diamond A \wedge_\diamond \circ A$ (which is equivalent to $\diamond\neg A \wedge (\Box\neg A \vee \Box A)$ in *S*5, entailing $\neg A$) and a classical conjunction by $A \wedge B \overset{\text{def}}{=} (A \wedge_\diamond B) \wedge_\diamond (\circ A \wedge_\diamond \circ B)$ (which is equivalent to $\diamond(A \wedge B) \wedge (\Box(A \wedge B) \vee \Box(A \wedge \neg B) \vee \Box(\neg A \wedge B) \vee \Box(\neg A \wedge \neg B))$ in *S*5, entailing $A \wedge B$). Consequently, the 'explosion principles' $(A \wedge \neg_\diamond A \wedge \circ A) \rightarrow B$, $(A \wedge_\diamond (\neg_\diamond A \wedge_\diamond \circ A)) \rightarrow B$, $((A \wedge_\diamond \neg_\diamond A) \wedge_\diamond \circ A) \rightarrow B$ and $((A \wedge_\diamond \circ_\diamond A) \wedge_\diamond \neg A) \rightarrow B$ are theorems of *PNS*5; in this way, *PNS*5 is a legitimate logic of formal inconsistency (cf. [10]). These definitions also allow us to fully embed classical propositional logic into *PNS*5.

With the aim to use the logic *PNS*5 (instead of *LFI*1*) in the definition of EParTMs, we first need to extend *PNS*5 to FOL with equality. This can be obtained by considering $S5Q^=$ (the first-order version of *S*5, with equality) instead of *S*5 in the definition of the logic, and adjusting the translation function $*$ to deal with quantifiers and equality. However, for the shake of simplicity, we will consider just $S5Q^=$ in the definition of the model, and we will regard the connectives $\neg_\diamond, \wedge_\diamond, \bullet$ and $\circ$ as definitions into this logic. Then, we will substitute the underlying logic of intrinsic theories $\Delta^\star_{FOL}(\mathcal{M}(\alpha))$ by $S5Q^=$, and through the Kripkean interpretation of $\Delta^\star_{S5Q^=}(\mathcal{M}(\alpha))$ theories, we will define what is an EParTM. Before that, we need to identify which kind of negation ($\neg$ or $\neg_\diamond$) and conjunction ($\wedge$ or $\wedge_\diamond$) are adequate in each axiom of $\Delta^\star_{S5Q^=}(\mathcal{M}(\alpha))$ (we will consider right-associative conjunction, i.e. $A \wedge B \wedge C$ always mean $A \wedge (B \wedge C)$; this is necessary *a proviso* because of the non-associativity of $\wedge_\diamond$, cf. property ($\wedge_\diamond 3$)):

(1) in axioms (A1)-(A5), negations and conjunctions are the classical ones;
(2) in axioms (Ai_{j} (I))-(Ai_{j} (III)), the conjunction in the antecedent is $\wedge_\diamond$, (considering ($\wedge_\diamond 6$)) only the first conjunction in the consequent is $\wedge_\diamond$ (other conjunctions are classical), and negation in (Ai_{j} (I)) is classical;
(3) in axioms (Aα) and (At0), negations and conjunctions are the classical ones;
(4) in axiom (Ath), only the conjunction in the antecedent is $\wedge_\diamond$, all other connectives are classical;
(5) in axioms (Aq_i) and (As_j), all conjunctions are classical, but negations are $\neg_\diamond$ (except in $y \neq x$), and it is also necessary to add the connective $\diamond$ before the predicates Q_i and S_i into the antecedent of the axioms.

We also need to define a notion of *representation* for the configurations of the TMs by worlds in a (possible worlds) Kripkean structure:

Definition 9
Let w be a world in a Kripkean structure. If $Q_i(\bar{t}, \bar{x}), \ldots, S_{j_{-1}}(\bar{t}, -1), S_{j_0}(\bar{t}, 0), S_{j_1}(\bar{t}, 1), \ldots$ are valid predicates on w, we say that w *represents* a configuration for a TM $\mathcal{M}$ at time $\bar{t}$, and the configuration is given by the intended interpretation I presented above.

By considering the choices of connectives and the definition above, worlds in the Kripkean interpretation of $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))$ represent the parallel computation of all possible computational paths of an NDTM $\mathcal{M}$ for the input α:

(1) By axiom (Aα), there will be a world w_0 representing the initial configuration of $\mathcal{M}(\alpha)$;
(2) by axioms (Ai_{j} (I))-(Ai_{j} (III)), if w_t represents a non-final configuration of $\mathcal{M}(\alpha)$ at time t, by any instruction i_j to be executed at time t (on such configuration), there will be a world $w_{t+1,j}$ representing a configuration of $\mathcal{M}(\alpha)$ at time $t+1$.

Configurations represented by worlds for the same instant of time t can be considered *superposed configurations*. In a superposed configuration, a state on position x and a symbol on position y are said to be *entangled* if there exist i,j,k,l ($i\neq k$ and $j\neq l$) such that $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\vdash Q_i(\bar{t},\bar{x})\wedge_{\diamond} S_j(\bar{t},\bar{y})$, $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\vdash Q_k(\bar{t},\bar{x})\wedge_{\diamond} S_l(\bar{t},\bar{y})$ and $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\nvdash Q_i(\bar{t},\bar{x})\wedge_{\diamond} S_l(\bar{t},\bar{y})$ or $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\nvdash Q_k(\bar{t},\bar{x})\wedge_{\diamond} S_j(\bar{t},\bar{y})$. In a similar way, the notion of entangled symbols on positions x and y can also be defined.

In order to exemplify the definition above, consider the two qubits entangled state $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Suppose the first qubit of $|\psi\rangle$ represents the state on position x of an EParTM $\mathcal{M}$ (value $|0\rangle$ representing state q_1 and value $|1\rangle$ representing state q_2), and the second qubit of $|\psi\rangle$ represents the symbol on position y of $\mathcal{M}$ (value $|0\rangle$ representing symbol s_1 and value $|1\rangle$ representing symbol s_2). Regarding only the state in position x and the symbol in position y of $\mathcal{M}$, state $|\psi\rangle$ represents a configuration of $\mathcal{M}$, at time instant t, in which only the combinations $q_1 s_1$ and $q_2 s_2$ are possible, all other combinations being impossible. This is expressed in the theory $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))$ by $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\vdash Q_1(\bar{t},\bar{x})\wedge_{\diamond} S_1(\bar{t},\bar{y})$, $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\vdash Q_2(\bar{t},\bar{x})\wedge_{\diamond} S_2(\bar{t},\bar{y})$, $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\nvdash Q_1(\bar{t},\bar{x})\wedge_{\diamond} S_2(\bar{t},\bar{y})$ and $\Delta^{\star}_{S5Q^{=}}(\mathcal{M}(\alpha))\nvdash Q_2(\bar{t},\bar{x})\wedge_{\diamond} S_1(\bar{t},\bar{y})$.

Taking into account the definition of the inconsistency connective in $S5Q^{=}$, as in the model of ParTMs, we can define conditions of inconsistency in the execution of instructions in the EParTMs. In this case, by the definition of the inconsistency connective in $S5Q^{=}$ and its Kripkean interpretation, condition $q_i^{\bullet}$ will indicate that the instruction will be executed only when at least two configurations in the superposition differ in the current state or position, while condition $s_j^{\bullet}$ will indicate that the instruction will be executed only when at least two configurations in the superposition differ in the symbol on the position where the instruction can be executed.

An EParTM is then defined as:

Definition 10
An *EParTM* is a NDTM such that:

- When the machine reaches an ambiguous configuration with n possible instructions to be executed, the machine configuration *splits* into n copies, executing a different instruction in each copy; the set of the distinct configurations for an instant of time t is called a *superposed configuration*;
- *Inconsistency* conditions are allowed on the first two symbols of instructions (as indicated above);
- When there are no instructions to be executed (in any current configuration), the machine stops; at this stage the machine can be in a superposed configuration, each configuration in the superposition represents a result of the computation.

Note that an EParTM parallelly performs all possible paths of computation of a NDTM, and only such paths. This differs from the previous model of ParTMs, where the combination of actions of different instructions had led to computational paths not possible in the corresponding NDTM.

Following [5], it is possible to define a reversible EParTM for any EParTM without inconsistency conditions in instructions.[16] This way, EParTMs almost coincide with QTMs without amplitudes; EParTMs represent uniform superpositions with no direct representation of the notion of relative phase, but does allow conditions of inconsistency on the instructions. As mentioned before, the notion of relative phase is a key ingredient in allowing interference between different paths of computation in QTMs, which is essential in order to take advantage of quantum parallelism in the efficient solution of problems; however, this method has theoretical restrictions which disable any definition of an efficient (polynomial time) quantum algorithm solving an NP-complete problem by a naive search-based method (see [23, Chapter 6]). On the other hand, conditions of inconsistency on instructions provided by EParTMs are an efficient mechanism to accomplish actions depending on different paths of computation. In fact, Theorem 16 proves that all NP-problems can be efficiently solved by EParTMs.

EParTMs represent an abstract model of computation, independent of any physical implementation. However, if we think from the physical construction of EParTMs, quantum mechanics provides a way to implement the simultaneous execution of different paths of computation, but does not provide what it seems to be a simple operation over the superpositions obtained by quantum parallelism: the execution of instructions depending on differences in elements on the superposed states (which correspond to conditions of inconsistency on instructions of EParTMs). In this way, quantum mechanics does not supply a direct theoretical frame for the implementation of EParTMs, but this definitely does not forbid the possibility of a physical implementation of EParTMs (perhaps conditions of inconsistency could be implemented by a sophisticated quantum physical procedure or by a new physical theory).

We could also modify the definition of EParTMs to capture more properties of QTMs. In this way, conditions of inconsistency in instructions could be avoided, and a notion of ‘relative phase’ could be introduced in EParTMs. This could be achieved by extending $S5Q^{=}$ with a new connective of possibility. Thus, the possibility connective of $S5$ (now denoted by $\diamond_1$) would represent ‘positive configurations’ and the other possibility connective ($\diamond_2$) would represent ‘negative configurations’ (axioms establishing the behavior of $\diamond_2$ and their interrelation with other connectives would need to be added; in particular, combinations of connectives $\diamond_1$ and $\diamond_2$ would have to behave in an analogous way to combinations of symbols $+$ and $-$). The connective $\diamond_2$ could be used to define a new paraconsistent negation as well as a new non-separable conjunction. Thus, by specifying which connectives would be used in each axiom, we could obtain a different definition of EParTMs. In this new definition, a concept of ‘interference’ can be specified; equal configurations with the same possibility connective interfere constructively, while equal configurations with different possibility connectives interfere destructively. Although details are not given here, this construction shows once more how we can define computation models with distinct computational power by just substituting the logic underlying theories $\Delta^{\star}_{FOL}(\mathcal{M}(\alpha))$. In this sense, computability can be seen as relative to logic. Alternatively, we can add a new element on the EParTMs: a sign indicating the ‘relative phase’ of the configuration, and a new kind of instructions to change the relative phase.

It is worth mentioning that the *quantum computational logics* proposed in [14] (see also [15]) are particular cases of paraconsistent logics where the separation rule seems to be not generally valid (although this is not explicitly commented in the cited papers). As this is also the case of *PNS*5 (the logic that we introduced above), the remarkable fact that these common properties have

[16]In the case of EParTMs, it is only necessary to avoid overlapping in the ranges of instructions; the parallel execution of all possible instructions in an ambiguous configuration does not imply irreversibility.

been obtained from different logical approaches on quantum computing suggest that they really are inherent characteristics of such computational models.

3.2 *About the power of ParTMs and EParTMs*

In order to estimate the computational power of ParTMs and EParTMs, we first define what the ‘deciding’ of a language (i.e. a set of strings of symbols $L \subset \Sigma^*$, where Σ is a set of symbols and * represents the Kleene closure) means in these models of computation. In the definition, we will consider multiple results in a computation as being possible responses from which we have to randomly select only one. We will also suppose that ParTMs and EParTMs have two distinguished states: q_y (the *accepting state*) and q_n (the *rejecting state*) and that all final states of the machine (if it halts) are q_y or q_n.

Definition 11
Let $\mathcal{M}$ be a ParTM (EParTM) and x be a string of symbols in the input/output alphabet of $\mathcal{M}$. We say that $\mathcal{M}$ *accepts* x with probability $\frac{m}{n}$ if $\mathcal{M}(x)$ halts in a superposition of n configurations and m of them are in state q_y; conversely, we say that $\mathcal{M}$ *rejects* x with probability $\frac{m}{n}$ if $\mathcal{M}(x)$ halts in a ‘superposition’ of n configurations and m of them are in state q_n. Consequently, we say that $\mathcal{M}$ *decides* a language L, with error probability at most $1-\frac{m}{n}$, if for any string $x \in L$, $\mathcal{M}$ accepts x with probability at least $\frac{m}{n}$, and for any string $x \notin L$, $\mathcal{M}$ rejects x with probability at least $\frac{m}{n}$.

Bounded-error probabilistic time complexity classes are defined for ParTMs and EParTMs as:

Definition 12
BParTM-PTIME (BEParTM-PTIME) is the class of languages decided in *polynomial time* by some ParTM (EParTM), with error probability at most $\frac{1}{3}$. BParTM-EXPTIME (BEParTM-EXPTIME) is the class of languages decided in *exponential time* by some ParTM (EParTM), with error probability at most $\frac{1}{3}$.

Space complexity classes can be defined in an analogous way, considering only the largest space used for the different superposed configurations.

Now, we will prove that ParTMs are computationally equivalent to DTMs, showing how to simulate the computation of ParTMs by DTMs (Theorem 13). As a consequence, we have that the class of languages decided by both models of computation is the same. It is obvious that computations performed by DTMs can be computed also by ParTMs, because DTMs are particular cases of ParTMs. What is surprising is that the simulation of ParTMs by DTMs is performed with *only* a polynomial slowdown in time (Theorem 14) and a constant factor overhead in space (direct consequence of the proof of Theorem 13). Theorems 13 and 14 are inspired in the simulation of multi-tape TMs by one-tape TMs as presented in [19], and show once more how powerful the classical model of TMs is.

Theorem 13
Any ParTM can be simulated by a DTM.

Proof. Let $\mathcal{M}$ be a ParTM with n states and m input/output symbols. Define a DTM $\mathcal{M}'$ and suppose its tape is divided into $2n+m$ tracks. Symbols 1 and 0 on track i ($1 \le i \le n$) and position p of $\mathcal{M}'$ represent, respectively, that q_i is or is not one of the states of $\mathcal{M}$ in position p. In a similar way, symbols 1 and 0 on track j ($n+1 \le j \le n+m$) and position p of $\mathcal{M}'$, respectively, represent the occurrence or non-occurrence of symbol s_j on position p of $\mathcal{M}$. Tracks $n+m+1$ to $2n+m$ are used to calculate states resulting from the parallel execution of instructions in $\mathcal{M}$, and values on these tracks represent states of $\mathcal{M}$ in the same way as tracks 1 to n. The symbol \$ is used on track 1 of $\mathcal{M}'$ to delimitate the area where $\mathcal{M}$ is in any state (i.e. where any symbol 1 appears on some track i associated to states of $\mathcal{M}$). To simulate a step of the computation of $\mathcal{M}$, $\mathcal{M}'$ scans the tape between delimiters \$ in four times. In the first scan (from left to right), $\mathcal{M}'$ simulates the parallel execution of instructions where the action is a movement to right: in each position, $\mathcal{M}'$ writes values in tracks $n+m+1$ to $2n+m$ in accordance with states 'remembered' from the previous step and collects (in the state of the machine, depending on the content of tracks 1 to $n+m$ and the instructions of movement to right of $\mathcal{M}$) the states to be written in the next position of the tape; $\mathcal{M}'$ also moves delimiters \$ if necessary. The second scan is similar to the first one, but in the opposite direction and simulating instructions of movement to the left, taking care in the writing of values so as not to delete values 1 written in the previous scan. In the third scan (from left to right), $\mathcal{M}'$ simulates the parallel execution of instructions where the action is the modification of symbols on the tape: in each position, depending on the content of tracks 1 to $n+m$ and in accordance with the writing instructions of $\mathcal{M}$, $\mathcal{M}'$ writes values on tracks $n+1$ to $n+m$ (corresponding to symbols written by instructions of $\mathcal{M}$) and also on tracks $n+m+1$ to $2n+m$ (corresponding to changes of states from the writing instructions of $\mathcal{M}$, taking care in the writing of values so as not to delete values 1 written in the previous scans). Finally, $\mathcal{M}'$ performs a fourth scan (from right to left) copying values from tracks $n+m+1$ to $2n+m$ on tracks 1 to n and writing 0 on tracks $n+m+1$ to $2n+m$. ■

Theorem 14

The DTM of Theorem 13 simulates n steps of the corresponding ParTM in time $O(n^2)$.

Proof. Let $\mathcal{M}$ be a ParTM and $\mathcal{M}'$ be the DTM described in the proof of Theorem 13 such that $\mathcal{M}'$ simulates the behavior of $\mathcal{M}$. After n computation steps, the leftmost state and the rightmost state of $\mathcal{M}$ cannot be separated by more than $2n$ cells, consequently the separation between \$ delimiters in the first track of $\mathcal{M}'$ is bounded by $2n$ cells. In any scan of $\mathcal{M}'$, in the simulation of a step of computation of $\mathcal{M}$, $\mathcal{M}'$ has to move between \$ delimiters, and a writing operation can be performed in any position, thus any scan takes at most $4n$ steps within the computation (ignoring steps due to scanning of delimiters \$ and their possible relocation). Therefore, the simulation of the n step in the computation of $\mathcal{M}$ takes at most $16n$ steps, i.e. time $O(n)$. Consequently, for the simulation of n steps in $\mathcal{M}$, $\mathcal{M}'$ requires no more than n times this amount, i.e. time $O(n^2)$. ■

Corollary 15

The class of languages decided by ParTMs and DTMs are the same, and languages are decided with the same temporal and spatial complexity in both models.

Proof. Direct consequence of Theorems 13 and 14; it is only necessary to add other instructions to determine if the accepting states are at least 2/3 of the total final states. Clearly, this additional task takes at most a polynomial number of steps (thus preserving the temporal complexity) and does not use new space (thus preserving the spatial complexity). ■

For EParTMs the situation is different: the class of languages decided in both models continues to be the same (DTMs can simulate all paths of computation of a EParTM, writing different configurations in separate portions of the tape and considering the different configurations in the simulation of instructions with inconsistency conditions), but all *NP*-problems can be *deterministically* (with error probability 0) computed in polynomial time by EParTMs (a direct consequence of Theorem 16, since the problem of determining satisfiability of propositional formulas in conjunctive normal form, usually called CSAT, is *NP*-complete). Thus, time complexity of EParTMs and DTMs are equal only if $P = NP$, which is broadly believed to be false.

THEOREM 16
CSAT is in BEParTMs-PTIME.

PROOF. It is not difficult to define a NDTM $\mathcal{M}$ deciding CSAT in polynomial time in which all computational paths have the same depth and finish in the same position of the tape. By considering $\mathcal{M}$ as a EParTM, all computational paths are performed in parallel, obtaining a superposition of configurations in which at least one of them is in state q_y if the codified conjunctive normal form formula is satisfiable, or with all configurations in q_n otherwise. Thus, by adding the instructions $i_{n+j} : q_y^\bullet s_j s_j q_y$ and $i_{n+m+j} : q_n^\bullet s_j s_j q_y$ to $\mathcal{M}$ (where m is the number of input/output symbols of $\mathcal{M}$ and $1 \le j \le m$) we have the acceptance or rejection with probability 1. ■

4 Final remarks

In this article, we generalize a method for axiomatizing TM computations not only with foundational aims but also envisaging new models of computation by logical handling (basically through the substitution of the underlying logic of the intrinsic theories in the computation), showing a way in which logical representations can be used in the construction of new concepts.

The new models of computation defined here use a sophisticated logical language which permits them to express some important features of quantum computing. The first model allows the simulation of superposed states by means of multiplicity of elements in TMs, enabling the simulation of some quantum algorithms but unable to speed up classical computation. In order to overcome this weakness, we define a second model which is able to represent entangled states, in this way, reaching an exponential speed-up of an *NP*-complete problem. Both models are grounded on paraconsistent logic (LFIs). In particular, the only element in the paraconsistent models that cannot be directly simulated in quantum computing is the addition of 'inconsistency conditions' to control the execution of instructions. As this is a key component in taking advantage of paraconsistent parallelism, an important problem is to decide whether it can or cannot be characterized by quantum means.

It seems clear that LFIs can be used to find new kinds of quantum logics, although this is not our concern here. In [22], for instance, weak completeness (relatively to an oracle for arithmetical reasoning) is proved for a new kind of quantum logic, called 'exogenous'. Such exogenous logic, embodying the contents of the postulates of quantum physics, is related to the society semantics and to the possible translations semantics for LFIs (for such semantics see [10]).

In spite of paraconsistent computational theory being only an emerging field of research, we believe that this logic relativization of the notion of computation is really promising in the search of efficient solutions to problems, particularly helping in the understanding of the role of quantum features and indeterminism in computation processes.

Funding

Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brazil, Thematic Research Project (Grant 2004/14107-2); FAPESP Scholarship Grant (05/04123-3 to J.C.A.); CNPq (Brazil) Research Grant (300702/2005-1 to W.C.).

References

[1] J. C. Agudelo, W. Carnielli. Quantum algorithms, paraconsistent computation and Deutsch's problem. In *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, B. Prasad, ed. pp. 1609–1628. India, 2005.

[2] J. C. Agudelo and W. Carnielli. Unconventional models of computation through non-standard logic circuits. In *Unconventional Computation, 6th International Conference, UC 2007, Proceedings*, S. G. Akl, C. S. Calude, M. J. Dinneen, G. Rozenberg, and T. Wareham, eds, Vol. 4618 of *Lecture Notes in Computer Science*, pp. 29–40. Canada, 2007.

[3] J. C. Agudelo and A. Sicard. Máquinas de Turing paraconsistentes: una posible definición Matemáticas: Enseñanza Universitaria, XII 37–51, 2004. Available at: http:// revistaerm.univalle.edu.co/Enlaces/volXII2.html (Last accessed on 31 October 2009).

[4] G. Boolos and R. Jeffrey. Computability and Logic. 3rd edn., Cambridge University Press, 1989.

[5] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, **17**, 525–532, 1973.

[6] J.-Y. Béziau. S5 is a paraconsistent logic and so is first-order classical logic. *Logical Studies*, 301–309, 2002.

[7] J.-Y. Béziau. Paraconsistent logic from a modal viewpoint. *Journal of Applied Logic*, **3**, 7–14, 2005.

[8] J. R. Büchi. Turing-machines and the Entscheidungsproblem. *Mathematische Annalen*, **148**, 201–213, 1962.

[9] W. A. Carnielli, J. M. de Almeida, and S. de Amo. Formal inconsistency and evolutionary databases. *Logic and Logical Philosophy*, 115–152, 2000.

[10] W. A. Carnielli, M. E. Coniglio, and J. Marcos. Logics of formal inconsistency. In *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner, eds, 2nd edn, Vol. 14, pp. 15–107, Springer, 2007, available at *CLE e-Prints* vol 5, n. 1, 2005. www.cle.unicamp.br/e-prints/vol_5,n_1,2005.html (Last accessed on 31 October 2009).

[11] A. Church. A note on the Entscheidungsproblem. *Journal of Symbolic Logic*, **1**, 40–41, 1936.

[12] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society of London. Series A*, **454**, 339–354, 1998.

[13] N. C. A. da Costa and F. A. Doria. On Jaśkowski's discussive logics. *Studia Logica*, **54**, 33–60, 1995.

[14] M. L. Dalla Chiara, R. Giuntini, and R. Leporini. Quantum computational logics: a survey. In *Trends in Logic. 50 years of Studia Logica*, V. Hendricks and J. Malinowski, eds, pp. 229–271. Kluwer, 2003.

[15] M. L. Dalla Chiara, R. Giuntini, and R. Leporini. Logics from quantum computation. *International Journal of Quantum Information*, **3**, 293–337, 2005.

[16] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. Series A*, **400**, 97–117, 1985.

[17] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. Series A*, **425**, 73–90, 1989.

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

[19] J. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*, 2nd edn, Addison Wesley, 2001.

[20] S. Jaśkowski. Rachunek zdań dla systemów dedukcyjnych sprzecznych. Studia Societatis Scientiarun Torunesis, Sectio A I, 57–77, 1948. [A propositional calculus for inconsistent deductive systems]. *Logic and Logic Philosophy*, **7**, 35–56, 1999.

[21] S. Jaśkowski. O koniunkcji dedukcyjnej w rachunku zdań dla systemów dedukcyjnych sprzecznych, Studia Societatis Scientiarun Torunesis, Sectio A I 171–172, 1949. [On discussive conjunction in the propositional calculus for inconsistent deductive systems]. *Logic and Logic Philosophy*, **7**, 57–59, 1999.

[22] P. Mateus and A. Sernadas. Weakly complete axiomatization of exogenous quantum propositional logic. *Information and Computation Archive*, **204**, 771–794, 2006.

[23] M. A. Nielsen and I. L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000.

[24] P. Odifreddi. Classical Recursion Theory. The Theory of Functions and Sets of Natural Numbers. *Studies in Logic and the Foundations of Mathematics*, Vol. 125, North-Holland, 1989.

[25] M. Ozawa and H. Nishimura. Local transition function of quantum Turing machines. *Theoretical Informatics and Applications*, **34**, 379–402, 2000. Available at: http:// arxiv.org/abs/quant-ph/9811069 (Last accessed on 31 October 2009).

[26] R. Sylvan and J. Copeland. Computability is logic-relative. In *Sociative Logics and Their Applications: Essays by the Late Richard Sylvan*, G. Priest and D. Hyde, eds, pp. 189–199. Ashgate Publishing Company, 2000.

[27] A. Tarski, A. Mostowski and R. M. Robinson. Undecidable theories. In *Studies in Logic and the Foundations of Mathematics*, L. E. J. Brouwer, E. W. Beth and A. Heyting, eds. North-Holland, 1953.

[28] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **42**, 230–265, 1936. A correction, Proceedings of the London Mathematical Society, **43**, 544–546, 1937.

Received 22 July 2008