Veritas
Revista de Filosofia da PUCRS
e-ISSN 1984-6746
Porto Alegre, v. 61, n. 2, maio-ago. 2016, p. 306-336
http://dx.doi.org/10.15448/1984-6746.2016.2.25574

Realismo Ontológico, Ontologia Matemática e Lógica

Relativistic ontological realism

Realismo ontológico relativístico

***Antônio Carlos da Rocha Costa**

Abstract: The paper introduces the concept of *relativistic ontological realism*, and the corresponding notion of *relativistic realization of objects*. The conditions that allow for the relativistic realization of objects are determined. In particular, the paper indicates the way the imposition of *organizational structures* on already existent objects supports the relativistic realization of new, more complex ones. A type-theoretic presentation of the principles of relativistic ontological realism is given in the language of predicate calculus, making use of the notion of *typed world* and the condition of *effectiveness* in the determination of the existence of objects. Two case studies illustrating the way the notion of *relativistic ontological realism* supports ontological analysis are presented, both concerning the *ontology of computations*, more specifically, the *relativistic realization* of *computational levels* in computational systems. The first case study deals with the realization of *logical circuits* through structural arrangements of electrical circuits. The second, with the realization of *software systems* through computer programming. A conception of the *mechanics of the relativistic realization of objects*, taking the imposition of organizational structures on previously extant objects as its basis, is proposed and briefly analyzed.

Keywords: Relativistic realization of objects. Effectiveness. Ontology of computations. Logical circuits. System of software.

Resumo: O artigo introduz o conceito de *realismo ontológico relativístico* e a noção correspondente de *realização relativística de objetos*. Determinam-se as condições que possibilitam a realização relativística de objetos. Em particular, indica-se o modo como a *imposição organizacional de estruturas* sobre objetos já existentes suporta a realização relativística de novos objetos, mais complexos. Faz-se uma

* Doutorado em Ciências da Computação pela Universidade Federal do Rio Grande do Sul. Atualmente é professor colaborador do Programa de Pós-Graduação em Engenharia de Computação da FURG e do Programa de Pós-Graduação em Informática na Educação da UFRGS. <ac.rocha.costa@gmail.com>.

apresentação tipo-teorética dos princípios do realismo ontológico relativístico na linguagem do cálculo de predicados, com base na noção de *mundo tipado* e na condição de *efetividade* na determinação da existência de objetos. Dois estudos de caso são apresentados, ilustrando o modo pelo qual a noção de realismo ontológico relativístico suporta análises ontológicas. Ambos são relativos à ontologia de computações, mais especificamente, à *realização relativística de níveis computacionais* em sistemas computacionais. O primeiro estudo de caso trata da realização de *circuitos lógicos* através de arranjos estruturais de circuitos elétricos. O segundo, com a realização de *sistemas de software* através da programação de computadores. Uma concepção dos *mecanismos de realização relativística de objetos*, que toma por base a imposição de estruturas organizacionais sobre objetos previamente existentes como base, é proposta e analisada brevemente.

Palavras-chave: Realização relativística de objetos. Imposição organizacional de estruturas. Efetividade. Ontologia de computações. Circuitos lógicos. Sistemas de software.

1 Introduction

This paper introduces the concept of *relativistic ontological realism* to account for the reality of the objects of human action, some pre-existent to it, others existing only as its consequences.

The way relativistically real objects *participate* in the operational structure of human action is one of the factors determining the relativistic reality of those objects. The other is the set of *inescapable properties* the allow them to carry out that participation.

In addition, the paper determines the way the imposition of organizational structures, by human action, on pre-existent objects allow for the relativistic realization of new, more complex objects.

The paper presents these ideas in the following way. In Sect. 2, it introduces the above concepts in an informal way. In Sect. 3, it presents them type-theoretically, making use of the formal language of the predicate logic.

In Sect. 4, the paper presents two case studies concerning the relativistic reality of the *computational levels* of computational systems, and the *computational components* that exist in them. The paper concentrates first, with a certain detail, on the relativistic reality of the most basic of the computational levels, the *bit level*. Next, it concentrates, more swiftly, on the relativistic reality of software.

In Sect. 5, the paper brings a few brief discussions: it attempts to distinguish between fictitious objects and relativistically real ones; then, it discusses the nature of the reality of artifacts; next, it presents the conception of the *mechanics of the relativistic realization of objects*

that presided the elaboration of the conceptual framework of the paper; finally, it places the concept of relativistic real objects in connection to the double nature, *sensibility* and *activity*, of human action. Section 6 discusses complementary aspects of the paper. Section 7 is the Conclusion.

2 The Conceptual Framework, Informally

2.1 Absolutely and Relativistically Real Objects

We take as *primitive* the notions of *object, agent, action,* and *performance of action*. We call *agent action* any action performed by an agent. We take *humans* to be a particular type of *agents*.

We also take as primitive the notions of:

– time and space;
– point in time or space;
– occurrence of an object at a point in time or space;
– feature of an object;
– determination of a feature of an object.

Some additional terms that we also use are[1]:

– operational structure of an action;
– participation of an object in the operational structure of an action;
– mental and non-mental action;
– result of the performance of an action (mental or not);
– determination of the truth of an existential judgment.

We consider that *agents, actions, operational structures of actions, time, space*, and *points of time or space* are *objects*.

We say that an object is *real in an absolute sense* (or, that it is an *absolutely real object*) if and only if the point in time and/or space where it occurs is independent of the operational structure of the action with which such point is determined. Otherwise, we say that the object is *real in a relativistic sense* (or, a *relativistically real object*).

That is, an object is considered to be an *absolutely* real object if and only if the determination of its occurrence, in a certain point of time and/or space, on the basis of a certain operational structure of action, does not change as a result of any change in that operational structure.

We say that an object is *non-real* if and only if none of its occurrence in point in time and/or space can be determined on the basis of a action, whatever its operational structure.

[1] These additional terms belong to a *specific* domain, that of *agents and their actions*. In this paper, however, such domain enters as an auxiliary one: we are examining the *general domain of objects*.

Notice that an object is said to be an *absolutely* real object if and only if its occurrence in any point of time and/or space can be determined to be the same, whatever the operational structure of action that determines it.

Notice also that, for a *relativistically real* object, one can define a notion of *mode of occurrence* (in a point of time and/or space), taking as bases the *types of structure* of actions that can determine that occurrence.

The most *fundamental* example of this type of distinction is that, established by Kant, between *objects of corporeal (or extended) nature,* which occur in time and space, and *objects of thinking (or rational) nature*, which occur only in time (KANT, 2004; see also: COSTA, 2014).

Thus, both *corporeal* and *rational beings* (understood in the Kantian sense) are *relativistically real objects* (in the terminology of the present paper).

Other examples of relativistically real objects are:

– agent actions;
– agents and groups of agents;
– operational structures of agent actions;
– any object that participates, directly or indirectly, in an agent action;
– any object that is considered to exist in a certain operational structure of action and not considered to exist in some other of such operational structures.

An ontology is said to be:

– an *absolutely realist ontology* if it admits at least one absolute real object;
– a *relativistic realist ontology* if it admits only *relativistically real objects*.

And we claim that any common-sense, scientific, or technological ontology is a *relativistically real ontology*.

But, in this paper, we do not attempt to prove this claim. What we attempt to do here is to develop the above notions up to a point that allows them (particularly the notions of *relativistically real object* and of *relativistically realist ontology*) to be expressed in a formal way.

In addition, we justify why the general term *artifact* is the appropriate term to denote the relativistically real objects that constitute the domains of the various *relativistically realist ontologies*, whose general features are determined through the conception of the *relativistic ontological realism*.

2.2 Absolute and Relativistic Ontological Realisms

We take *ontological realism* and *ontological anti-realism* in the sense adopted by Chalmers (2009):

- *ontological realism* is the point of view that it is possible to give *objective answers* to the basic ontological question: *What exists?*;
- *ontological anti-realism* is the point of view that either such answers are not objectively possible, or that there are so many equally admissible that none can be chosen as the right one.

We follow Kant (1998), in his stating that (p. 567): "*Being* […] *is merely the positing of a thing, or of certain determinations in it.*" So, in terms of the concepts formulated in the previous section, we define *existence* as: *occurrence of something in a point in time and/or space*. That is, we interpret Chalmer's question *What exists?* as: *What is occurring now in time and/or space?*

Thus, we say that the form of ontological realism characterized by Chalmers is essentially what we introduced above as *absolute ontological realism*, since it assumes that there is a unique answer to the basic ontological question (that is, an answer that is independent of every possible operational structure of every possible action).

Notice, then, that we may take to be *absolute* every standard variant of ontological realism. In particular, we may take to be *absolute* the relatively recent variant of ontological realism that Roy Bhaskar adopted as the basic ingredient of his *critical realism* (BHASKAR, 2008).

The particular notion of ontological realism that we are introducing in the present paper, on the other hand, is intended to be a *non-standard variant* of ontological realism, in the sense that:

– it acknowledges the possibility of alternative *positive* answers to the basic ontological question: *What exists?*;
– but in such a way that, at any time, criteria may exist to determine the choice of some of those alternatives as the *preferred* ones (possibly, just one).

We call *relativistic ontological realism* such non-standard variant of ontological realism.

2.3 Action as the Foundation of Relativistic Ontological Realism

As mentioned by Chalmers (2009), ontological realism is often traced to Quine (1948), in the form of *scientific realism*: what exists is what is endorsed to exist by the best theories of science.

If one acknowledges that theories of science are not fixed, but may evolve in time (even to the point of being revolutionized), scientific realism should be taken as a *relativistic* ontological realism, not as an *absolute* one[2].

[2] So that the *ontological anti-realism* of Carnap (1950), mentioned by Chalmers (2009), seems not to be so antagonistic to the *ontological realism* of Quine (1948) as that mention seems to imply.

The *relativistic ontological realism* that we introduce here is not of the *scientific realism* brand (although it includes scientific realism as a particular case): it takes *action* as a criterion for choosing the preferred positive answer to the basic ontological question.

That is, the type of *relativistic ontological realism* proposed here endorses as real only those objects that are *accessible* to, or can *participate* in, the operational structure of action (in a sense to be made precise later). As a consequence, both *sensible objects* and *ideas* can be taken as existent objects, as long as they can participate in the operational structure of some action (either mental or non-mental).

The reason why the proposed form of *ontological realism* is said to be *relativistic*, even though it takes a single and precise type of object (*action*) to be the foundation of the reality of any type of object (including action itself), lies in that action is readily acknowledged to be a *variable* object, susceptible to all sort of variations, both in *time* and/or in *space* (which is the reason why *scientific theories*, as products of action, are also susceptible to variations in time and/or space).

That is, action, as a foundation for the concept of *existence* is not an *absolute* foundation, but a *relative* one: for each of the possible forms of action, a different foundation for the concept of *existence* may be established, i.e., a *different variant of ontological realism* may be founded.

In summary: in the sense proposed here, to be *real* is to *participate* in the operational structure of some *action*.

We also remark that:

- *absolute ontological realism* takes reality to be, essentially, a determination of objects (ideas, elementary physical entities, etc.), the reality of the particular type of object we call *action* having to be established as a *derived consequence* of the way it relates to the objects that are assumed to be real (e.g., as a consequence of the reality of human beings);
- *relativistic ontological realism*, on the other hand, takes reality to be, essentially, a feature of objects *determined by* action, the reality of particular types of objects having to be established as a *derived consequence* of the way they relate to action. Objects that do not participate in the operational structure of any action are taken to be not real.

2.4 The Requirement of Effectiveness

As indicated in, e.g., (Martinelli, 2014), an important tendency, in fixing a criterion for determining an answer to the basic ontological question, in absolute ontological realism, is that based on the concept of *effectiveness*: what exists is what can produce effects.

We take this requirement of *effectiveness* as inescapable in any preference criterion applied to alternative positive answers to the basic ontological question.

That is, we take that an object can be considered real only if it is *effective* in the operational structure of any action in which it participates, meaning that that action would necessarily be realized in a different way, if the object did not participate in it.

3 The Conceptual Framework, Type-Theoretically

In this section, we make use of the formal language of predicate logic (assumed to be known by the reader) to determine the notion of *relativistic ontological realism* in terms of what we call *typed worlds* and *conceptual frames of reference* for existential judgments about typed worlds.

We begin with an analysis of what is involved in a question about the existence of an object. The requirement of *effectiveness* of the *determination of the truth of existential judgments* in the operational structure of any action aiming at the solution of that question plays a central role in what follows.

3.1 Has Leonardo da Vinci existed?

Does the surface of a canvas exist? If it does, than Leonardo da Vinci could have painted a picture of Mona Lisa on it. If it does not, Leonardo could not have painted that picture. At least not on the surface of a canvas.

Do layers of paint exist? If they do, then Leonardo could have applied them over something (the surface of a canvas, perhaps) to paint a picture of Mona Lisa. If they don't, then Leonardo could not have applied them on any thing (including the surface of a canvas).

What about the image of Mona Lisa: does it exist? If it does, we can look at it, perhaps in the museum of the Louvre. If it doesn't, we can not look at it, and what the museum of the Louvre says it is showing to us is something else, not the image of Mona Lisa that is supposed to have been painted with layers of paint, applied by Leonardo da Vinci over the surface of a canvas.

And, of course, this sequence of questions can be pursued further: Has Leonardo da Vinci existed? If he has existed, he could have painted a picture of Mona Lisa, etc. If not, he could not have painted a picture of Mona Lisa, etc.

What we have intended with the presentation of these elementary questions is twofold:

– to stress that the *existences* with which we will be concerned, in the following, are the existences of objects that have some form of participation in the operational structure of some agent action;
– to hint at the form of the *question about existence* that will drive the development of the argument.

3.2 Types, and the Question about the Existence of Type Instances

When we ask the question "*Does the surface of a canvas exist?*", we are simultaneously referring to four different entities:

- two *types* of things:
  – the type named "surface";
  – the type named "canvas";
- two *instances* of these types:
  – an instance of the type named "canvas", which we suppose to exist, for the sake of the question;
  – an instance of the type named "surface", which we take as a possible component of a canvas (that is supposed to exist for the purpose of the question), and whose existence we are investigating.

We can, then, represent the meaning of the question by the following formal expression:

$$Q \rhd W \vDash \forall c{:}Canv(\exists s{:}Surf(s \sqsubseteq c))$$

where:

- $Q \rhd E$ denotes that the questioner Q questions if the logical expression E is true;
- W denotes a world to which the questioner refers the question, for interpretation and answering[3];
- $\vDash$ denotes that is such that it satisfies the first order logical formula presented on the right of that sign;
- $Canv$ denotes the type named "canvas";
- $Surf$ denotes the type named "surface";
- for any type X, the expression $x{:}X$ denotes that the variable x ranges over the (possibly empty) universe of instances of the type X;
- $s \sqsubseteq c$ denotes that the entity s is a proper part of the entity c.

Similarly, we may represent the meaning of the question: *Do layers of paint exist?* in the form:

$$Q \rhd W \vDash \forall c{:}Paint(\exists l{:}Layer(l \sqsubseteq p))$$

[3] Which, in this and in the following questions, we will assume to be the daily world of ours.

where:
- *Paint* denotes the type of *volumes of paint*;
- *Layer* denotes the type of *layers of paint.*

Even the question: *Has Leonardo da Vinci existed?* can be put in a form equivalent to that.

To see it, consider, first, the question: *For any population, do individual members of that population exist?*, taken in the form:

$$Q \rhd W \vDash \forall p{:}Pop(\exists m{:}Ind(m \in inst[p]))$$

where:
- *Pop* denotes the type of *populations*;
- *ind* denotes the type of *individuals*;
- for any type T, $inst[T]$, denotes the set of *instances of* T.

We can specify the formal structure of the content of the question about the existence of Leonardo da Vinci in two steps. First we:
- eliminate the quantification $\forall p$, making the variable p assume the value $p{=}p_{FloXV}$, meaning: *the set of members of the population of Florence in the XVth century*;
- eliminate the quantification $\exists m$, making the variable m assume the value $m{=}LdV$, meaning: *Leonardo da Vinci*.

Then we get the formula we were looking for, that formally captures the question about the existence of Leonardo da Vinci:

$$Q \rhd W \vDash LdV \in p_{FloXV}$$

meaning: *Was Leonardo da Vinci one of the members of the population of Florence in the XVth century?*

In fact, any question about the existence of an object seems to admit to be put in this format.

3.3 The Relativity of the Existence of Type Instances

What can one see about the *relativity* implicit in a question about the existence of an object, when the object is considered to be an instance of a type, and the question is put in the form:

$$Q \rhd W \vDash \forall x{:}X(\exists y{:}Y(x \sqsubseteq y))$$

One can see, at least, that:
- The *types* are certainly considered in a relative way: they are considered in relation to the cultural background of the questioner Q

that formulates the question, and to the constitution of the world W that Q acknowledges: depending on his/her cultural tradition, the acknowledged constitution of W etc., the question may be meaningful, or not; may be ambiguous, or not; may be serious or a joke; etc.

- The existence of *objects*, seen as instances of types, is also certainly considered in a relative way: even if the questioner makes a serious question, and the question is unambiguous, the object is considered to be *relative to the types* that are involved in the question. More precisely, relatively to the *sets* of instances of those types that the questioner Q acknowledges, regarding the constitution of the world W.

3.4 The Question about Absolute Existence

To see in a more clear way the last remark above (about the relativity of the objects, when they are regarded as instances of types), notice that, if isolated from type considerations, the form of question about the existence of instances of types becomes:

$$Q \rhd W \vDash \forall x(\exists y(x \sqsubseteq y))$$

which is a *completely different* question: *Is it true that, in the world , for every object there is a another object that is a proper part of it*?

The same different sense belongs to the even more general question, obtained by leaving unspecified (by means of the *underline* character) the world to which the question refers:

$$Q \rhd \_ \vDash \forall x(\exists y(x \sqsubseteq y))$$

which means: *Is it true that, in some world, for every object there is another object that is a proper part of it*?

That is, one sees that to formulate the general structure of a question for the absolute existence of an object, it is not enough to omit considerations of types, and to avoid to see objects as instances of types.

On the other hand, one knows that it is useless to attempt to reach such general structure in the language of predicate logic for, as Kant (1998) stated, in a most reproduced passage: "*Being is obviously not a real predicate"* (p. 567).

Simple examples of futile attempts to do that are:

- $Q \rhd \_ \vDash \exists x(x=x)$, which in fact asks for the truth of a tautology, not for the absolute existence of an object;

- $Q \rhd\_ \models \exists x(\forall y(\neg(y \sqsubseteq x)))$, which in fact asks for the existence of an object that has no proper parts (that is, a *minimal* object), not for the absolute existence of an object;
- $Q \rhd\_ \models \exists x(\forall y(\neg(x \sqsubseteq y)))$, which in fact asks for the existence of an object that is no proper part of any other object (that is, a *maximal* object), not for the absolute existence of an object.

Here, we leave open the problem of *how* (in fact, *if*) the question about the *absolute existence* of objects can be given a faithful formal expression.

3.5 Inescapable Properties of Objects with Relativistic Existences

The most general form of the question about relative existence, however, is the following one:

$$Q \rhd\_ \models \exists x(P(x))$$

meaning: *Is it true that, in some world, there exists an object which is such that the property P is satisfied by that object in that world*?

The property P is any property whose satisfaction the questioner considers to be *inescapable* by any existent entity. We define an *inescapable property* as any property whose satisfaction is a *necessary* consequence of any assertion of existence of the thing in question.

Clearly, inescapable properties are relative to the cultural background of the questioner, which is why that form is the general form of a question about objects with a relativistic existence.

On the other hand, inescapable properties are a scape from Kant's statement, which prevents being from being a predicate. The price of such scape is, of course, having a question limited to relativistic existences.

3.6 Effective Access to Objects with Relativistic Existences

A crucial requirement of any question about the relativistic existence of objects, including the case of question based on inescapable properties, having the form::

$$Q \rhd\_ \models \exists x(P(x))$$

is that the answer to the question should be determined, in some effective way, by anyone attempting to answer it. Which implies the requirement of *effective access* to all the objects of the domain to which the question refers.

In the particular case of questions with typed variables, with the form:

$$Q \rhd\_ \models \forall x{:}X(\exists y{:}Y(x \sqsubseteq y))$$

that domain is the set of instances of the type X.

Of course, anyone may ask a question about the existence of objects that are *not effectively accessible* to anyone attempting to answer the question. But that is an issue that Kant has put aside, in an apparently satisfactory way, more than two centuries ago (KANT, 1998).

Whatever the reader's opinion about Kant's solution to the problem of the *non-effective existence* of things, we restrict ourselves, in this paper, to the consideration of questions of existence that can be answered, positively or negatively, in some effective way.

That is, we restrict the proposed notion of *relativistic ontological realism* to concern only objects that, being effective in some operational structure of some action, turn out to be effectively accessible both to the questioners that question about their existences, and to the answerers that volunteer to answer the questions.

3.7 Conceptual Frames of Reference for Relativistic Existential Questions

The formal structure of the question about the *absolute* existence of objects:

$$Q \rhd \_ \models \exists x(P(x))$$

shows clearly that *even in their most general form*, questions about absolute existences are, in fact, relative: they are relative to the *set of worlds* the questioner admits into consideration.

We call *world specification framework* any means a questioner may make use of, to delimit a set of worlds to be taken as a reference in questions about the existence of objects.

World specification frameworks are given in terms of a variety of *dimensions*, to which the worlds that they specify are submitted. We take here that, at the minimum, world specification frameworks should encompass a *type structure*, that is, a *hierarchically ordered set of types*, which specify worlds in terms of the *hierarchy of types* to which the objects of the worlds should be submitted, when the answer for the question of existence is searched for.

We take a type structure to be a structure of the form $T=(T,\sqsubseteq)$, where is a set of types and the relation $\sqsubseteq\sqsubseteq\ T\times T$ is a partial order, determining the hierarchy of type specialization.

We say that worlds that satisfy such specification frameworks are *typed worlds.*

We assume that, in any typed world, there is a *most general type* of object, which we call . Thus, the most general form of question about the existence of objects in typed worlds has the format:

$$Q \rhd \_ \models \exists x{:}Obj(P(x))$$

meaning: *Is it true that, in some typed world, there is an object that satisfies the inescapable property P?*

Of course, the questioner may question about the existence of objects of some specific type. Thus, if $T \in T$ is a type in the type hierarchy then the following question:

$$Q \rhd \_ \vDash \exists x{:}T(P(x))$$

asks about the existence of an object of type T that satisfies the inescapable property P.

3.8 The Question about the Existence of Types

An important issue one may raise concerning the approach to existential questions that we have proposed in the present paper is the issue of the *existence of types*, themselves. Of course, in accordance with what we have proposed above, we focus on questions about the *relativistic* existence of types.

Formally, the simplest format in which such questions can be put is:

$$Q \rhd W \vDash \exists \tau{:}Type(\tau \in Types(W))$$

meaning: *Is it true that, in the world W, there is a type* τ *(i.e., an object* τ *of the type Type) that belongs to the set of types applicable to W?*

If W is a typed world, the answer is immediately positive, for in any typed world W it happens that $Types(W) \neq \emptyset$.

If W is not a typed world (i.e., a world with no *type structure*), the answer may be either positive (if it happens that types of objects exist in that world, but in an *unstructured way*) or negative (if no type of objects can be identified, at all, in that world).

The most general question about the existence of types has, of course, the formal structure:

$$Q \rhd \_ \vDash \exists \tau(\tau{:}Type)$$

meaning: *Is it true that, in some world, there is a type (i.e., an object* τ *of type Type)?*

We state that the *relativist ontological realism* assumes that *types* of objects exist and concerns itself only with *typed worlds*.

4 Case Studies: The *Relativistic Reality* of the Computational Levels of Computational Systems

Since the beginnings of Philosophy it has been a tradition to take issues in Natural Sciences (Physics, Biology, etc.) and Mathe-

matics to motivate, illustrate or contradict philosophical ideas and theories.

In this case study, we consider a fundamental issue in Computer Science, namely, the question of the *reality* of the computational levels of computational systems, to illustrate the idea of relativistic ontological realism introduced in the present paper.

We present, in Sect. 4.1, the conceptual framework underlying the subject of the case study. Only after that, we proceed with the case study, proper.

4.1 The Conceptual Bases of the Case Studies

4.1.1 *The General Concepts of 'System' and 'System Level'*

We base our general concept of system on Mario Bunge's CESM model (BUNGE, 2014). A *system* is a structure $Sys=(C,E,S,M)$ where:

- C is the set of *components* of *Sys*;
- E is the *environment* of *Sys*, that is, the system of elements that do not belong to C;
- S is the *structure* of *Sys*, that is, a set of relations among the components existent in C;
- M is the *set of mechanisms* of *Sys*, that is, a set of processes performed by the components existent in C, and between them and the components of the environment E.

We take, then, that a system can be structured in terms of a *hierarchy of system levels*, as follows:

- the set of components of Sys is organized in terms of a *hierarchy of systems levels* C_i, for $i\in\{1,...,n\}$, where each system level C_i is a subset of C that is disjoint from the set of components of every system level C_j, where $j\neq i$;
- the structure S is organized in terms of a *hierarchy of sub-structures* S_i, such that each S_i is a set of relations among the elements of the corresponding components in C_i, and a sub-structure S_E, which is a set of relations among the components of C_E and the components of E;
- the set of mechanisms M is organized in terms of a *hierarchy of sets of mechanisms* M_i, such that each set of mechanisms M_i, each performed by a set of components of the corresponding C_i, and a set of mechanisms M_E, each performed by a set of components of C_E and a set of components of E.

Thus, a system structured in terms of a hierarchy of system levels is a structure given by $SL=(\{SL_i\},\sqsubseteq)$ where:

- $\{SL_i\}$ is the set of system levels;

- each $SL_i=(Ci,Ei,S_i,M_i)$ is a *system level*;
- $\sqsubseteq \subseteq \{SL_i\}\times\{SL_i\}$ is the *hierarchical relation* between *system levels* (usually a linear relation);
- each particular system *SL* has its particular definition of $\sqsubseteq$, determining what it means to say that two system levels are such that $(C_i,E_i,S_i,M_i) \sqsubseteq (C_j,E_j,S_j,M_j)$;
- whenever $SL_i \sqsubseteq SL_j$, we say that SL_j is a system level that ishigher than the system level SL_i.

For each specific type of system , a particular justification has to be given for the *reality* or *non-reality* of its system levels. Usually this is done by taking as a basis, for such justification, the *functional significance*, or *insignificance*, of the hierarchy of system levels for the structure and mechanism of the overall system.

For instance:

- one may choose to define a *building* as a hierarchical system where each system level is a *floor*, and the hierarchy of the system levels is given by their pilling over each other; one may, then, determine that each floor is a *real* system level by specifying its functional significance for the structure and working of the building;
- one may choose to define a *classroom* as a hierarchical system where each system level is a *cohort* of students that have in common their month of birth; one may, then, determine that each such cohort is an *abstract* (non-real) system level, existing only as a descriptive device, with no functional significance for the structure and working of the classroom.

4.1.2 *The Issue of the Reality of the 'Computational Levels' of Computational Systems*

The first extensive exposition of the general notion of *computational level* seems to have been that in Bell & Newell (1971).

In that book, computational levels were use merely as abstractions, as descriptive devices (thus, non-realities):

> A system (at any level) is characterized by a *set* of components, of which certain properties are posited, and a set of ways of combining components to produce systems. When formalized appropriately, the behavior of the systems is determined by the behavior of its components and the specific modes of combination used (BELL & NEWELL 1971, p. 3).

That is, a computational level is an abstract structure, a sort of algebraic or relational structure (T,R), where T is the set of *types of components* and R is the set of *rules* for combining components of given types. Also, a computational level has a characteristic *medium*, that is,

a characteristic *type of content* (bits, bytes, numbers, symbols, etc.) that can be processed by components of types *T*.

A hierarchy of computational levels allows for formal descriptions of systems to be given at each of such system level. For that, specific *description languages* could be defined, one for each such system level:

> A system level [...] is characterized by a distinct language for representing the system (that is, the components, modes of combination, and laws of behavior) (BELL & NEWELL, 1971), p. 4).

However, in the practice of computer science (i.e., in the practical analysis and design of computational and programming systems), computational levels are commonly treated by computer architects, programmers and engineers as *real* entities, that are functional in the systems.

That is, sets of system components that, from the perspective formally established by Bell & Newell, are just *describable* in terms of the types of components of a computational level (e.g., software applications, processors and memories, logical gates, etc.) are taken to *constitute* computational levels that effectively exist and operate in the system.

That is, in practice, a system component (e.g., a processor) is not only taken to be of a an element of a type that belongs to a given descriptive system level (e.g., the Processor-Memory-Switch level) and, thus, to be describable in terms of components whose types belong to lower system levels (e.g., types of components of the Register-Transfer level). It is also taken to be a *real* component of the system, *constituted* by components that belong to lower system levels, themselves taken to be *real* levels of the system.

This situation introduces an ambiguity in the term "computational level", between its status of a merely *descriptive device* vs. its status of an articulated set of *real system components*.

A classical statement of the way in which the concept of computational level acquires a sense of real entity in a system, in the context of the practice of computer science, appears in (DIJKSTRA, 1968), the paper that first clearly constituted sets of real system components as computational levels that really exist in computational systems. In the context of the general methodologies for computer programming (WIRTH, 1971) is the paper that first realized that same task.

We submit that such *ambiguity* in the treatment of computational levels (and, in general, in the treatment of any kind of system entity) is a typical sign of their *relativistic mode of existence.*

That is, when facing a *relativistically real entity*, one has the tendency to treat it *nominally,* as a *non-real entity*, as mere a *abstraction.* But, in

the *practice* with that entity, one has the tendency to treat it as *existent*, as a *real entity*.

In the following, we make clear the way computational entities in general, and the computational levels of computational systems, in particular, should be treated as *relativistically real entities*.

4.1.3 *The Concept of Relativistic Realization of a 'Computational Level'*

Computers, when they appeared, in the early 1940's, were built to process numerical information (which is why they are called, precisely, *computers*, that is, machines that calculate).

The idea that computers can process *symbols* (in the mathematical sense of the term: *written marks with definite meanings*) arose in the late 1950's, when computer engineers started to use numbers to realize *symbols* (first mathematical symbols, then general types of formal symbols) through *encoding techniques*, giving rise to *symbolic computations* and *symbolic programming languages*[4].

The *hierarchy of computational levels* that, in the history of computers, originally began with three levels (with three specific media: *computational numbers* realized by *bits*, *bits* realized by *electronic states*, *electronic states* physically grounded), was then extended with a new system level (with *symbols* as medium, realized by *computational numbers*), as shown in Fig. 1.

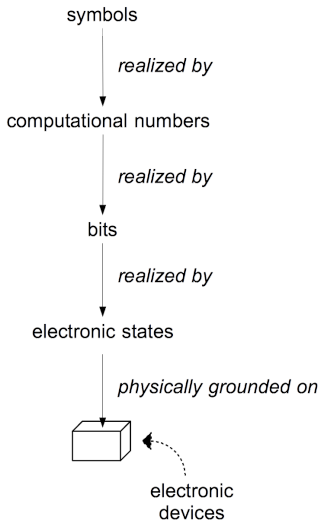


Fig. 1 – The hierarchy of computational levels realizing the symbol level in computers.

[4] The possibility of modeling *symbol processing tasks* in programming languages was the technical development that founded the areas of *Artificial Intelligence* and *Psychology of Information Processing*, from which emerged, later, the general area of *Cognitive Sciences*.

For the purposes of this paper, we take the term "computational realization" to mean: *objectification on the basis of encoding* (i.e., *reification* of items on the basis of their encoding by other, already realized, items).

Saying that the computational level *x is realized by* the computational level *y*, in a computational system, means that, in a recursive way:

- *x* and its workings have been *encoded* by *y* and its workings;
- the encoding of *x* and its workings by *y* and its workings determines the existence of *x* and its workings as a system level, in the hierarchy of computational levels of the system, if either *y* is physically grounded, in the basic physical structure of the computational system, or *y* is computationally realized by some computational level *z* that exists in the hierarchy of computational levels of the system.

Figure 1 illustrates the application of this recursive definition of the *computational realization* to the first three computational levels of any computational system, characterized by processing the three types of media: *bits*, *numbers* and *symbols* (given that the system level of the devices, with *electric states* as medium, is considered to be a *physical level,* not a *computational* one).

In terms of the concepts introduced in Sects. 2 and 3, above, we say that the following *two conditions* determine the *relativistic reality of a computational level* in a computational system:

- the *inescapable property:*
  - to be computationally realized through an *encoding* on a lower system level that is either itself computationally realized or else is *physically grounded* on the lowest system level;
- the *way to participate* in the operational structure of human action:
  - to be *accessible* to techniques and tools of computer use and programming.

Next, we analyze the *relativistic realization* of two types of components that are characteristic of the computational level where bits are the processed medium: in Sect. 4.2, a *basic bit flow control mechanism*; in Sect. 4.3, a *basic bit storage device*.

Both relativistic realizations build on the *inescapable property* and the *way to participate* in the operational structure of human action that were just mentioned.

4.2 Case Study I: The Relativistic Realization of Logical Circuits

We analyze in this first case study the relativistic realization of two elementary components of logical circuits: *bit control flow mechanisms* and *bit storages*.

4.2.1 *The Relativistic Realization of a 'Bit Flow Control Mechanism'*

A clear understanding of the way *bits* (i.e., the *basic computational objects*) and their operators (the so called *logical-gates*) give rise to the most basic forms of *computational processes* is inescapable for understanding how the *bit level* can be taken to be *real* (in fact, *relativistically* real), in a sense that goes well beyond any reality that can be established by a mere procedure of encoding bits by electric states.

For that purpose, we analyze here the computational realization of one of the basic forms of *bit flow control mechanism*.

4.2.1.1 *The Concept of a 'Bit Flow'*

The realization of the notion of *bit flow* (i.e., the idea of *bits flowing in a circuit*) demands the introduction of two *notions* and a *type of entity*, the latter not immediately obtainable from the mere encoding of bits in the electric states of physical systems.

The first notion is that bits can *vary in time*, which is – of course – a notion that can be directly obtained from the mere notion of encoding, by referring to the notion that the electric states that physically ground the realization of bits can vary in time.

A typical example of the form of temporal variation of the value of a bit located in some physical point in space is illustrated in Fig. 2. As usual in computational processes, the time in Fig. 2 is considered to be *discrete*, that is, to evolve as a sequence of *time instants* (t_0,t_1,t_2,...) that occur separated from each other by a fixed *time interval*.

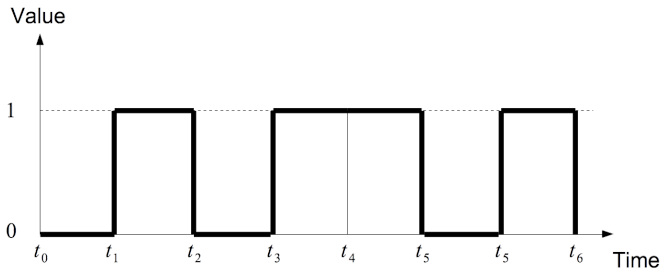


Fig. 2 – An example of temporal variation of the value of a bit.

The second notion needed for the realization of the notion of bit flow is the notion that *bits can flow* from a point in space to another. This notion arises from the replication, at the level of bits, of the notion of *flow of electric currents*. That is, based on the idea that electric states encode bits, *electric currents* (the *flow of electric states* from a point in space to another) can be taken to encode the *flow of bits* in space.

The type of entity that has to be introduced in the conceptual framework that is being built is that of entities that can support bit flows. They are called *bit wires* (or, more commonly, *logical wires*, due to the fact that the bit values *0* and *1* can be interpreted as respectively representing the logical values *True* and *False*).

When a bit flows through a logical wire, from a spatial position *x* to a spatial position *y*, the bit value occurring in the spatial position is taken to be exactly equal, at each time instant, as the bit value that is occurring in the spatial position *x* at that time instant (that is, bit flow is assumed to be instantaneous).

Figure 3 illustrates a logical wire, extended between the spatial points *x* and *y*, with *s* denoting the corresponding *bit flow* (the flowing of a bit through a logical wire is often said to be a *logical signal*). The bit value at *y* is always the same as the bit value at *x*.

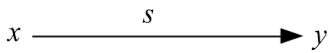


Fig. 3 – A basic example of bit flow.
(Also: a logical signal *s* flowing from the spatial position *x* to the spatial position *y*)

4.2.1.2 *The Relativistic Realization of 'Logical Wires'*

Logical wires can be thought to be *realized* (in the relativistic sense of the word) by *electric wires*.

By having logical wires (relativistically) realized by electric wires, one has that the *systemic function* that the logical wires perform at the bit level (the transmission of the value of a bit from one spatial location to another) is *homologous* to that performed by the *electric wires* at the *electric level* (the transmission of an electric current from one spatial location to another[5]).

The formal basis of such homology is the fact that the diagram in Fig. 4 commutes, that is, the diagram is such that the equation:

$$s_l(encod(e_x)) = encod(s_e(e_x))$$

holds, where:

- the variables *bx*, b_y e_x, and e_y represent, respectively, the values of the *bits* and the *electric states* at the spatial positions *x* and *y*;

[5] By a *functional homology* we mean an analogy of a functional character between two systems, that is, a correspondence between the effects of two systemic functions, each performed in one of the systems.

- the signals (logical signal s_l and electric signal s_e) represent the spatial transportation of the values (bit values and electric state, respectively) from the spatial position x to the spatial position y.

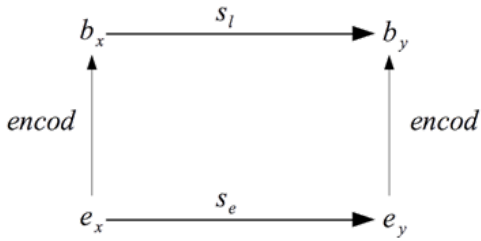


Fig. 4 – The functional homology supporting the realization of a logical wire (carrying the logical signal s_l) by an electric one (carrying the electrical current s_e).

The commutativity of the diagram in Fig. 3 means that the encoding of the electric states into bits, at each spatial position, is compatible with the systemic functions performed by each wire (electric and logical), so that the homology between those systemic functions is guaranteed.

The *functional homology*, thus, satisfies the first requirement of the relativistic realization of the logical wire by the electric wire, namely, that of being the of the the *inescapable property* that logical wires should satisfy.

The second condition, of the participation in the operational structure of an human action is immediate, in the uses logical wires may be put to, in the construction of *logical circuits*.

4.2.1.3 *The Relativistic Realization of 'Bit Flow Control Mechanisms'*

The implementation of means for *controlling* bit flows requires the introduction of another *type of entity*, which is also not immediately obtainable from the mere encoding of electric states of physical systems in terms of bits.

To determine what type of entities are such *control units*, we will start by considering a very basic type of bit operator, usually called a *logical gate*. A logical gate is any entity capable of receiving two logical signals as input and of producing an output logical signal, as the result of the realization of a *logical operation* on those two input signals.

Figure 4 illustrates a logical gate capable of performing the AND logical operation (the graphical sign shown for the AND logical gate is the usual one). The left part gives a graphical picture of the AND logical gate as part of a logical circuit. The signals in the locations x and y are the input signals, and the signal in the location is the output one. The

gate is such that the value of the output signal, at any time, is given by the logical expression $z=AND(x,y)$, as shown in the table in Fig. 5.

x	*y*	*z*
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 5 – The AND gate.

Now, let's consider the step that leads us beyond this type of circuit component, which operates in a way that its tabular definition shows to be *timeless* (in the sense that time enters in no way into the determination of its output from its input).

To reach the possibility of defining circuit components that operate in a *time-dependent* way, we need to allow that signals be attached to the components that generate them in such a way that each such *generator* is capable of *deciding by its own*, possibly taking various conditions into account, *what value* to generate, and *when*.

That is, we need to introduce the idea of *agency*, at the level of logical circuits, with circuit components that can generated logical signals in an *autonomous* way.

In such situation, it becomes meaningful to have a *control unit* (say *K*) capable of controlling the flow of a logical signal from one component (say *A*) to another (say *B*), on the basis of a *control signal* received from a third component (say *C*). The third component *C*, can be thought of as making use of the control unit *K* to control the flow of the logical signal between *A* and *B*.

This is illustrated in Fig. 6, where *x* is the control signal that *K* receives from *C*, s_A is the signal that *A* sends to *B*, and s_B is the controlled signal, that is, the signal that *K* lets *B* receive from *A*, as determined by the control signal *c* that it receives from *C*.

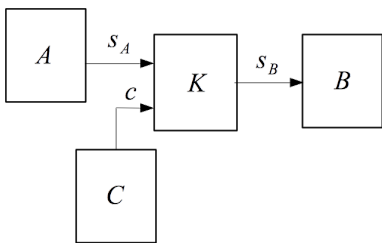


Fig. 6 – Controlling the flow of the signal.

Formally, we may define the functioning of the control unit *K* by:

$$sB = \begin{cases} 0 & \text{if } c=0 \\ s_A & \text{if } c=1 \end{cases}$$

so that if $c=0$, *K* blocks the signal s_A and the component *B* receives just the logical signal with value 0. Whenever *c* becomes 1, the control unit *K* allows the logical signal s_A to go through, so that the component *B* receives a logical signal s_B with the same value as the logical signal s_A.

From the realization point of view, the question we need to make now is: What type of component needs to be introduced, besides the two types already introduced (logical gates and agencies) to allow for the existence of control units like .

Figure 7 gives the answer: none. A simple AND gate can operate as a control unit like *S*, if agencies are available. For, the input signals at *x* and *y* can be treated as the signals originating from the agencies *S* and *B*, that is, from components of the logical circuit that are operationally autonomous.

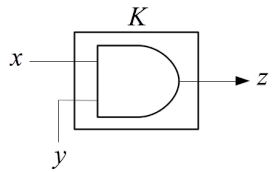


Fig. 7 – The Control gate.

In other words: with the introduction of the autonomy of the components *A* and *B*, a transformation occurred in circuit, that made of an AND logical gate, which operates in a functionally timeless manner, to operate in a time-dependent way, realizing the control unit *K*.

In such transformation, the two inputs, *x* and *y*, which from the perspective of the logical gate AND can be considered as *functionally undifferentiated*, and thus treated in an interchangeable way (regarding the calculation of the output signal), became *functionally differentiated*, one of them (*y* in Figs. 5 and 6) becoming the *controlling* input, and the other (*x* in Figs. 5 and 6) becoming the *controlled* input.

Such *functional transformation* cannot be attributed to any inherent feature of the AND logical gate. it has not emerged in that AND gate as a result of any internal process. It was a result of placing the AND gate to operate in a situation characterized by two features: having signals produced by autonomous circuit components, and being placed in a

circuit where one of the components is considered to be the *autonomous controller component*, and the other to be the *autonomous controlled component*.

The AND gate being placed in such situation constitutes, then, the *inescapable property* that the control unit *K*, implemented by that AND gate, has to satisfy in order to be relativistically realized by such gate.

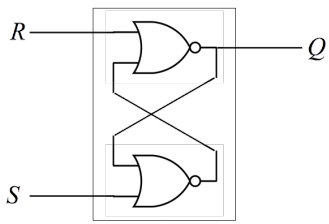
4.2.2 *The Relativistic Realization of 'Bit Storage Devices'*

We analyze now the most basic way of realizing *bit storage devices* with logic gates, namely, through the *articulation of a feedforward-feedbackward loop*.

"Flip-flop" is the usual name given to memory components that are able to store the value of one bit at a time. A *flip-flop* changes the bit value it is storing whenever an appropriate combination of logical signals appears in its inputs. The new bit value to be stored is algo given by that combination of input signals.

Figure 8 shows the structure of a flip-flop (of the type called *SR*), and the table to its right defines its temporal behavior: for each combination of the inputs *S* and *R* (respectively called *Set* and *Reset*), the table indicates the value *Q* that is next stored in the flip-flop, and which is also the value of the output the flip-flop will then produce:

- if $S=0$ and $R=0$, the value stored stays at the value *Q* at which it already is;
- if $S=1$ and $R=0$, the value stored becomes 1, whatever the previous value;
- if $S=0$ and $R=1$, the value stored becomes 0, whatever the previous value;
- if $S=1$ and $R=1$, the behavior of the flip-flop is undetermined.

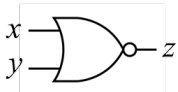


S	R	Q
0	0	Q
1	0	1
0	1	0
1	1	-

Fig. 8 – The SR Flip-Flop.

The functioning of the flip-flop, which goes according to its structure, is the following:

- the two logical gates that constitute the structure of the flip-flop realize the logical operation called *NOR* (that is, *Not-OR*), as defined in Fig. 9;
- whenever $S=1$ and $R=0$, the lower input of the lower *NOR* gate is 1, leading its output to 0 and both inputs of the upper *NOR* gate to 0, thus leading its output (Q) to 1; that reinforces the output of the lower *NOR* gate to 0, because its two inputs become 1, thus stabilizing the flip-flop in a state where $Q=1$;
- whenever $S=0$ and $R=1$, the upper input of the upper *NOR* gate is 1, leading its output (Q) to 0 and both inputs of the lower *NOR* gate to 0, thus its output to 1, which reinforces the output of the upper *NOR* gate to 0, thus stabilizing the flip-flop in a state where $Q=0$;
- whenever $S=0$ and $R=0$, one of two things may happen: either $Q=0$ and the two inputs of the lower NOR gate are 0, so its output is 1, leading the two inputs of the higher NOR gate to 0 and 1, thus stabilizing $Q=0$; or $Q=1$ and the two inputs of the lower NOR gate are 0 and 1, leading its output to 0, so that the two inputs of the higher gate are 0, stabilizing 1. In any case, the value of Q is stabilized and not changed



x	*y*	*z*
0	0	1
0	1	0
1	0	0
1	1	0

Fig. 9 – The NOR Gate.

The feature that allows for the reachability of the two stable states (one where $Q=1$, the other where $Q=0$), and the permanence of the flip-flop in them, is the presence of the *feedforward-feedback loop* that interconnects the output of each *NOR* gate to the input of the other, thus forming a cycle capable of continuously reinforcing the production of the logical values present in each location of the structure, when the flip-flop reaches any one of those stable states, thus keeping the flip-flop in that state until a combination of inputs leads to a state change.

Clearly, it is the inescapable organizational structure of the feedforward-feedbackward loop that gives the SR structure the capacity to store bits. Such organizational structure is, of course, the result of a deliberate design decision, and the existence of such type of object can

hardly be thought of separately from the intention of having a circuit component really existent and capable of performing that function.

4.2.3 *Discussion of the Case Study: "Relativistic Realization" and "Implementation"*

Regarding the way the basic bit flow mechanism and basic bit storage device are realized, notice that, besides the encoding of the electric states of the physical ground of the system in bits, and besides the use of logical wires for the flow of bits, two additional elements have to be introduced in the system, so that control signals and components can appear, namely, *agencies* (that is, active units that interact with each other in autonomous ways) and an *organizational structure*, articulating the means through which the agencies interact with each other.

It is in the context of the structure that organizes the actions of those agencies, together with the inescapable properties that are specific to each type of element of the bit level, that the concept of *relativistic realization* of *bit flow control mechanisms* and *bit storage devices* should be understood.

Notice, also, how different the notion of *relativistic realization* is from the notion of *emergence*. In particular, notice that relativistic realization is based on a deliberation to create some novelty, a deliberation taken by someone that expects that the novelty created be incorporated into the operational structure of the action that that someone performs.

The notion of emergence, on the other hand, at least if taken in its usual sense, means precisely the opposite: an unintended creation of novelty.

In fact, the term "relativistic realization" can be understood as a formal counterpart of the informal term "implementation", which computer scientists (and anyone involved in engineering and technological issues) make use of, to designate the creation of a novelty in a system level (and, in fact, even for the creation of the whole system).

4.2 Case Study II: The Relativistic Realization of *Software Systems*

Software is also something of a relativistic reality. Software exists as a conceptual entity, in the minds of programmers and software engineers, and as a concrete entity, on an underlying computer.

But software is not a state of an underlying computer. It is a set of processes and information structures that that underlying computer can realize, when it is operating from a given initial state.

The initial state determines the set of processes and information structures that the underlying computer can possibly perform, from that initial state on, but it is not the software itself.

It is in this sense that software is a relativistic reality: its existence depends on the state space of the underlying computer, and on the initial state in which that computer is put. But the software is the conceptual structure that, realized by that state space and initial state, determines the system of processes that that computer can realize.

It is the double nature, conceptual structure and physical realization in the state space and initial state of a computer, that constitutes the software.

Formally, we have:

– a *software* for a computer H is a class of structures:

$$Sft_H \subseteq \{(c_H, P_H, s_H) | c_H \in Code, P_H \in Proc, s_H \in StSp\}$$

where:

- $Code$ is the universe of codes for computers;
- c_H is software code acceptable by the computer ;
- $Proc$ is the universe of processes that computers can execute;
- P_H is the set of processes that the computer can realize when executing the software (c_H, P_H, s_H);
- $StSp$ is the universe of state spaces that computers can have;
- s_H is a state at which the computer can be and that is taken as the *initial state* at which the computer should be put to execute the software (c_H, P_H, s_H);
- $c_H \subseteq s_H$, that is, the software code c_H should be a part of the initial state s_H.

A software is a *class* of structures of the form c_H, P_H, s_H, not a single structure, because the same software can be executed in a computer on the basis of different codes and different initial states. Whenever the same software is realized under different set of possible processes in the computer, one says that each such set of possible processes determines a *version* of the software.

As in the case of the flows of bits and the corresponding logical circuits, in digital hardware, as have we examined above, software can only be realized in a stepwise constructive way, starting from basic elements that give rise to more complex ones through the systematic introduction of *organizational structures*.

Such organizational structures construct new software elements from the software elements that have already been constructed, the final element constructed being the final software itself.

The result is a hierarchy of levels of software components, each existing on the basis of the organizational structures that were applied to the components existent in the level below.

The most basic level of the software construction is that of the *machine language*, the software level that is constructed by imposing a *symbolic* organizational structure on the state space of the *architectural level* of the computer, that is, highest hardware level constructed on the basis of the physical structure of the computer, through the imposition of organizational structures of the digital circuit type.

From the physical level of the material components of the computer, through its electrical and electronic levels, and through its digital circuit and architectural level, toward the programming level and the various levels of software construction, there is *a hierarchy of more than 15 system levels* operating to support the execution of the most simple program (a basic text editor, for instance).

Each of those system levels, existing and operating on their own, on the basis of the organizational structures imposed on the components of the lower system level, can only be properly understood in terms of a relativistic notion of existence.

5 The Mechanics of the Relativistic Realization of Objects

On the bases of the conceptual framework introduced in the paper, as well as on the analyzes of the case studies, we submit that the *mechanism of the relativistic realization of objects* amounts to the *imposition of organizational structures* on already existent objects, in a way that realizes the inescapable properties that the relativistically realized object has to have in order to be capable of participating in the operational structure of the agent that creates it.

It is this conception of the mechanics of the relativistic realization of objects that presided the elaboration of the conceptual framework introduced in the present paper.

6 Discussion

6.1 Xephadonts

Assume that we have given some thought to characterize a certain kind of animal: a kind of dinosaur, carnivorous, blue skin, three horns, large wings, small head, long tail, six legs, capable of thought and languge. Let's call it a *xephadont.* Do xephadonts exist? Of course not, their a pure imagination of ours.

Now, let's draw an image of a xephadont, define its mode of behavior, its habitat, its life cycle. Let's program xephadonts as characters in a computer game, running on a website. Let's also connect such computer game to a few email servers and let's give a few xepha-

donts the task of supervising the emails of some of the users of the game.

Let's say that whenever a user sends an email with certain type of subject, the supervisor xephadont gains some points in the game, whenever the subject is of certain other type, the user gains some points.

Let's also establish that whenever a user looses more than a certain quantity of points to her supervisor xephadont, the supervisor xephadont is entitled to fine the user with an amount of money corresponding to half of the user's salary. When the xephadont looses that quantity of points, the user is payed the amount of money that corresponds to her full salary.

If you were playing the game, would you say that your supervisor xephadont exists, or not?

I would say that, given that its full-fledged definition (behaviorally, etc.) serves very well the function of an inescapable property for that kind of entity, and given that it is fully integrated to the operational structure of your game and email behavior, your supervisor xephadont exists, for sure (in a relativistic sense). And also, that you should better be aware of its (relativistic) existence, otherwise you risk to bankrupt, at the end of the game.

6.2 Is Every Artifact a Real Thing? Is Every Real Thing an Artifact?

Not every artifact is a real thing, only those artifacts that, endowed with adequate properties, participate in some operational structure of some human action, for then the realization of the action depends on the way that artifact operates, and this can only happen if the artifact exists.

Not every real thing is an artifact, only those artifacts that, endowed with adequate properties, participate in some operational structure of some human action, for then the real thing becomes more than it is in isolation, it is endowed with new properties, including the property of interfering with the realization of the that action.

6.3 Relativistic Realization of Objects and Human Sensibility and Activity

In chapter where he presented his *Transcendental Aesthetic*, Kant (1998) called *sensibility* our capacity of being affected by objects and of producing *intuitions* as results of such affections.

In no place, in the *Transcendental Aesthetic*, Kant talks about *activity*, our capacity of affecting objects by performing actions.

We submit that, in the same way that intuitions arise as products of our sensibility, when affected by object, objects of our activity arise as products of our intuitions, when that activity is directed to ward those objects.

That is, we submit that intuitions may be both the way we represent objects when they affect to our sensibility, as perceived objects (*objects that are there*), and the way we represent objects, as projected objects (*objects that should be there*), when they are subject to our activity.

The term "real object" is usually taken to mean *objects that are there*, often in this restricted meaning of objects capable of affecting our sensibility. Sometimes the term "real object" is also taken to mean abstract objects only capable of affecting our understanding (but is mostly subject to debate, as a form of platonism, etc.).

What we intend with the use of the term "relativistically real object" is to encompass under a single expression objects *that are there* and *that should be there*. That is, we intend that the expression encompasses objects that both affect our *sensibility* and are affected by our *activity*, that are both *perceived* and *projected*, *sensed* and *acted upon*.

Thus the importance of the inescapable properties (they specify both what is and what should be), of the imposition of organizational structures on objects, and of the participation of the objects in the operational structure of human action (for human action involves both *sensibility* and *activity*).

7 Conclusion

This paper introduced the notion of relativistic ontological realism and the correlated notion of relativistic realization of an object. The articulation of relativistically real objects to the operational structure of the human action dealing with them and the inescapable properties that allow such articulation were put at the basis of those notions.

The way the imposition of organizational structures on previously existent relativistically real objects supports the relativistic realization of new objects was indicated.

The paper claimed, on the basis of two paradigmatic case studies, that the ontology of the computational domain is essentially relativistically real.

Finally, it suggested that every object that is at the same time an object of human sensibility and of human activity is most appropriately seen as relativistically real, if its structure and functioning is determined by the imposition of an organizational structure on its components.

References

BELL, G.; NEWELL, A. *Computer Structures: Readings and Examples*. McGraw-Hill, New York, 1971.

BHASKAR, R. *A Realist Theory of Science*. New York: Routledge, 2008.

BUNGE, M. *Emergence and Convergence:* Qualitative Novelty and the Unity of Knowledge. Toronto: Univ. Toronto Press, 2014.

CHALMERS, D. "Ontological Anti-Realism". In: CHALMERS, D.; MANLEY, D.; WASSERMAN, R. *Metametaphysicis* – New Essays on the Foundation of Ontology. Oxford: Claredon Press, 2009. p. 77-129.

COSTA, A. C. R. *Para a Explicação dos Princípios a Priori da Cognição dos Fenômenos Sociais.* Open publication, 2014. Online at: www.ResearchGate.net

DIJKSTRA, E. The Structure of the Multiprogramming System. *Communications of the ACM*, 11, 5 (1968), p. 341-346.

KANT, I. *Critique of Pure Reason*. 2nd ed. Cambridge: Cambridge Univ. Press, 1998.

KANT, I. *Metaphysical Foundations of Natural Science*. Cambridge: Cambridge Univ. Press, 2004.

MARTINELLI, R. "Realism, Ontology, and the Concept of Reality". *Ethics & Politics*, XVI, 2 (2014), p. 526-532.

QUINE, W. V. O. "On What There Is". *Review of Metaphysics*, 2 (1948), p. 21-38.

WIRTH, N. Program Development by Stepwise Refinement. *Communications of the ACM*, 14, 4 (1971), p. 221-227.

Endereço postal:
Programa de Pós-Graduação em Computação
Centro de Ciências Computacionais FURG
Av. Itália km 8 – Bairro Carreiros
Rio Grande, RS, Brasil

Data de recebimento: 18-10-2016
Data de aceite: 11-11-2016