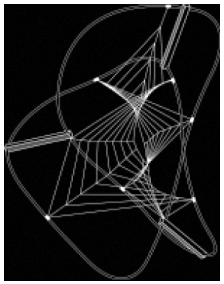
The London School of Economics
and Political Science
Houghton Street
London WC2A 2AE

Tel: 020 7955 7573
Fax: 020 7955 6869
email: m.steuer@lse.ac.uk

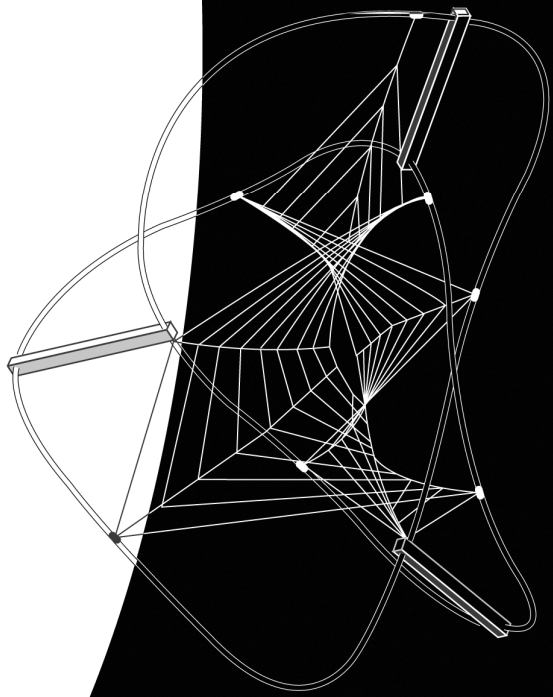


Centre for Philosophy of
Natural and Social Science

LSE Philosophy Papers

Realism about Computational Properties

Emiliano Boccardi



PP/06/08
ISSN 1750-3590

Realism about Computational Properties

Emiliano Boccardi

The property of being the implementation of a computational structure has been argued to be vacuously instantiated. This claim provides the basis for most antirealist arguments in the field of the philosophy of computation. Standard manoeuvres for combating these antirealist arguments treat the problem as endogenous to computational theories. The contrastive analysis of computational and other mathematical representations put forward here reveals that the problem should instead be treated within the more general framework of the Newman problem in structuralist accounts of mathematical representation. It is argued that purely structuralist and purely functionalist accounts of implementation are a-priori inadequate to tackle the problem. An extensive evaluation of semantic accounts is provided, arguing that semantic properties are, unlike structural and functional ones, suitable to restrict the intended domain of implementation of computational properties in such a way as to block the Newman problem. The semantic hypothesis is defended from a number of recent objections.

1. Introduction

Computational structures, such as finite state automata or Turing machines, are a particular kind of mathematical structures.[1] Like many other classes of mathematical structures, they have been used to represent some aspects of the physical world. Just as magnetic and electric radiations are believed to belong to a common genus, thanks to the fact that they are all instantiations of Maxwell's equations, digital computers of the same type are thought to belong to a common genus thanks to the fact that they are all implementations[2] of the same computational structure.

The relatively uncontroversial and successful use of computational concepts in the fields of computer science, engineering, and, not least, in the sciences of the mind, suggests that computational properties are objective (i.e. not dependent on our purposes or on other observer-dependent factors). Just as the solar system instantiates a certain mathematical dynamical system whether or not anyone interprets it as doing so, a computer would be such, according to this realist construal of computational properties, even if no one ever so interprets it, or even if no observer had ever existed. If computational properties were not real – this brand of no-miracles argument goes - how

[1] In particular, they can be argued to belong to a restricted subclass of mathematical dynamical systems. These are abstract structures represented by triplets of the form: $< T, M, \{g^t\} >$, where T is the time set, M is the state space and $\{g^t\}$ is the set of state transitions $g^t : M \rightarrow M$. Abstract computational systems can be argued to belong to this class of mathematical structures. The following is an example of how this can be proved in the case of Turing machines (see M. Giunti 1995). The future behaviour of a Turing machine is determined when 1) the state of the internal control unit, 2) the symbol on the tape scanned by the head and 3) the position of the head, are given. It is therefore possible to take the set of all triplets <state, head-position, tape-content > as the state space M of the system; the set of non-negative integers can be taken as the (machine) time set T; the set of quadruples of the machine is then used to form the set $\{g^t\}$ of state transitions. Each state transition function is such that g^0 is the identity function on M, and $g^{t+1}(x) = g(g^t(x))$. Similar considerations apply to all other computational structures.

[2] Following a standard use, the term "implementation" here denotes specifically the realization of computational structures. When referring to the realization of generic mathematical structures (not necessarily computational), I shall use the term "instantiation".

could computational theories be so successful at modelling and predicting the behaviour of certain systems? Whence do they derive their explanatory power?

Realism about computational properties, however, has been challenged on various grounds. Several attempts to pin down the necessary and sufficient requirements for a system to realize computational properties turned out to place observer-relative constraints.[3] Antirealist arguments typically aim at showing that standard accounts of implementation either fail to describe *all* paradigmatic computing systems as computing, or else they fail to describe *only* computing systems as computing.

These results come in two varieties. According to *strong antirealist arguments*, computational properties are vacuously instantiated by any physical system. I. Hinkfuss, for example, imagined a transparent plastic pail of spring water sitting in the sun and asked himself if the activity of such a system, given some suitable correspondence rules, wouldn't be complex enough as to realize the program that underlies a human mind (see Lycan 1981: 39). J. Searle famously argued that, under a suitable interpretation, his wall implements the Wordstar program (Searle 1992: 209). In the appendix to his book *Representation and Reality* (1988), Putnam proposed a now (in)famous argument to the effect that every open physical system implements every inputless finite state automaton.[4] The soundness of these vacuousness arguments has been challenged (see for example Chalmers 1996, Copeland 1996 and Piccinini 2007), but there is nothing like an agreement as to whether the proposed amendments to the definition of implementation succeed at blocking them.

According to *weak antirealist arguments*, even if computational properties are not universally instantiated, theories of implementation are still too liberal in ascribing computational properties to physical systems. Moore (1990), for example, has argued that a universal Turing machine could be implemented by the motion of a single particle moving in space, bouncing between parabolic and linear mirrors like an ideal billiard ball. Recently, O. Shagrir has forcefully argued that in some cases a computing physical system, short of semantic constraints on its inputs and outputs, can be seen as implementing more computational structures that compete for explanatory power.

[3] Attempts to ground the notion of implementation on that of step satisfaction (see for example Cummins 1989, pp. 91-92), for instance, or on the digital vs/ analog distinction (see Haugeland 1981), or on the instantiation of fundamental vs/ derivative physical laws (Block and Fodor 1972) have all been argued to fail to make a case for computational realism.

[4] The technical result, moreover, has been argued to be extendable to the claim that any open physical system implements any automaton. An argument to this effect can be found in Scheutz (1999).

The purpose of this paper is to put forward a general diagnosis for the problem of excessive liberalism, based on a contrastive analysis of computational and other mathematical representations of reality. Standard manoeuvres for combating the alleged vacuousness of implementation treat the problem of liberalism as endogenous to computational theories. My analysis, instead, allows us to place the problem within the overarching framework of structural accounts of mathematical representations of physical properties, where it is argued to belong, and where a solution to it can be envisaged.

Why are computational properties exposed to vacuousness arguments? Where does this unwanted liberalism come from? Why do other mathematical representations of reality, such as those used in physics, appear to be immune to this disease?

The source of *unwanted* liberalism in the ascription of computational properties, I shall argue, is the *wanted* amount of liberalism. Unlike what happens with the instantiation of other physical properties, one of the chief desiderata of any account of implementation, in fact, is that it must make room for the *principle of multiple realizability* (PMR henceforth): different implementations of the same computational structure need not share any *particular* kind of (first- or second-order) physical properties. A computer, says Johnson-Laird, "could be made out of cogs and levers […]; it could be made out of a hydraulic system through which water flows." [Johnson-Laird (1988): 39]. A complex of logical gates, observes N. Block, could be realized by a system of cats, mice and cheese, where the cats would strain their leashes and open gates in accordance with the rules of the computation to be implemented. According to Pylyshyn "a group of pigeons [could be] trained to peck as a Turing machine" [1984: 57]. "We could be made of Swiss cheese and it wouldn't matter" [Putnam (1975): 291]

What counts, in other words, is the causal *structure*, not the particular properties that realize it. This desideratum, i.e. the requirement that the constraints placed on a physical system for it to be computing satisfy the PMR, constitutes one of the chief allures of computational representations of reality; unfortunately, it turns out to be the most difficult to satisfy within a realist perspective.

How do we ever manage, in general, to build a non vacuous metaphysical bridge between abstract, mathematical structures and empirical ones? Why can't we argue that Newton's law of gravity, or Fresnel's equations, are vacuously instantiated?

As we shall see, all mathematical representations of reality, including as a consequence also computational representations, are subject to the problem of excessive

liberalism, when the burden of fixing the wanted models is left to structural isomorphism alone. This problem, also known as Newman's problem, does not arise in cases where we are able to specify the intended domain of physical properties in the represented domain. In the case of computational theories, however, this property-specific route to the elimination of unwanted models is a-priori blocked by the need to comply with the principle of multiple realizability.

In the literature there are three main strategies for fixing the intended models of computational theories. They can be classified in terms of the kinds of properties that are proposed to restrict the candidate bearers of computational properties. Accordingly, there are structuralist, functionalist, and semantic (or intentional) accounts of implementation, characterized respectively by the assumptions that computational states must be individuated by their structural, functional or semantic properties. The main thesis of this paper is that only a restriction of the intended domain represented by computational structures to intentional items (i.e. to states and groupings of states that instantiate semantic properties, such as content) allows us to block unwanted implementations.

The structure of the paper is as follows. In section 2 I put forward a contrastive analysis of the problem of liberalism. I argue that the problem of liberalism is not restricted to computational theories, but derives from a general problem of purely structuralist accounts of mathematical representations of reality (Newman's problem).

In section 3 the analysis is applied to structuralist theories of implementation. It is argued that standard property-specific combating manoeuvres against the problem of liberalism are not available to computational theories. These must therefore seek to restrict their intended domains in non-structural and non property-specific ways.

In section 4 I provide an evaluation of functionalist accounts of implementation, arguing that functional properties (both of the narrow and of the wide variety) are also a-priori unsuitable to tackle the problem of liberalism.

In section 5, finally, I provide an extensive evaluation of semantic accounts, arguing that semantic properties are, unlike structural and functional ones, suitable to restrict the intended domain of implementation of computational properties. The semantic hypothesis is defended from a number of recent objections.

2. Structuralist representations of physical reality

The instantiation of mathematical structures in physics is prima facie immune from vacuousness arguments. The reason appears to be that in that case we are allowed to make reference to *specific* physical properties and magnitudes. Consider for example the case of Fresnel's equations, describing the propagation of light through a plane reflecting surface: $\frac{R}{I} = \frac{\tan(\theta_1 - \theta_2)}{\tan(\theta_1 - \theta_2)}$. In their intended interpretation θ_1 and θ_2 *refer* to the angles made by the incident and the refracted beams with the normal to the surface, while *R* and *I*, in Fresnel's understanding, *refer* to the amplitude of the vibration of the molecules of ether induced respectively by the reflected and by the incident beam. Being the intensity of a beam of light is a property that is certainly not universally instantiated, let alone by a piece of Swiss cheese! The angles made by the incident and refracted beams, or the intensities of the beams (proportional respectively to R^2 and I^2), are magnitudes specified unambiguously by repeatable measurement procedures. In this sense, these properties can be broadly construed as "observational", in that they reliably produce expected effects in angular translators and photometers.

If it wasn't for these restrictions on the interpretations, the above equation would not represent Fresnel's law. Suppose that θ_1 and θ_2 in the equation above did not refer to the angles made by the incident and a refracted beam, suppose that we were free to interpret them as we please, or that the field of science under theorizing was not even specified: couldn't we then cook up some antirealist argument to the effect that Fresnel's equations are vacuously instantiated?

As a matter of fact, we could. We need not go too far for entertaining conjecture of such a scenario. When mathematical structures are used to model a domain of *unobservable* properties, which, unlike angles and intensities, are not *directly* specifiable through measurement procedures, they can also be argued to fail to capture *only* the wanted models. In these cases, we shall see, the represented domain must be restricted by suitable semantic postulates, if the theory is to avoid vacuousness.

While the interpretation of the variables R^2 and I^2 in Fresnel's equations is constrained by specific measurement procedures (those relative to the observable intensities of the refracted and of the incident beam), the variables *R* and *I* do not refer to any directly observable magnitude. In Fresnel's own interpretation, for example, they referred to the (unobservable) amplitude of the vibrations of the molecules of ether.

Fresnel's ontological interpretation of his equations, which construed light as consisting of vibrations propagating through the all-pervading ether, has been notoriously falsified: ether does not exist! The structure of Fresnesl's theory, including his equations, however, survived this ontological revolution, to reappear, unchanged, in Maxwell's theory, and even in the following relativistic theory. The only difference is that, according to Maxwell's ontological interpretation, it is the (equally unobservable) electric and magnetic field strengths that vibrate, not the molecules of ether: the interpretation of the other variables in the equations (those that refer to observable magnitudes), is left unaltered.

"Fresnel's equations", observes J. Worrall, "are taken over completely intact into the superseding theory – reappearing there *newly interpreted* but, as mathematical equations, entirely unchanged". [Worrall (1996): 160, my emphasis]. According to this structuralist intuition, Fresnel was right, after all, about the fact that *something* was vibrating with a certain amplitude, at certain angles, although he did not, and could not know anything about the *nature* of that something. Concentrating on the reality of structure, rather than on that of the (unknowable) relata, allows us to create a safe niche for philosophers with a residual realist inclination: if the unobservable relata are unknowable, think the structural realists, the structure of their relations is knowable and constitutes the true content of scientific theories.

The sole relation between mathematical structures and reality that is required to be in place, in these cases, is that provided by structural isomorphism. In the case of Fresnel's equations, the isomorphism consists of a mapping from the values of some unspecified (and unspecifiable) magnitudes to the values of the variables in the abstract mathematical structure.

This philosophical manoeuvre is captured by the well know procedure of Ramseyfication. If we describe a theory by the formula $\theta(O_1, ...O_n; T_1, ..., T_m)$, where $O_1, ..., O_n$ are observational or perceptual terms, and $T_1, ..., T_m$ are theoretical terms (those that purportedly refer to the unobservable properties), the structural realist claims that the "true" empirical content of the theory is represented by the Ramsey sentence: $\exists s_1, ..., \exists s_m \theta(O_1, ..., O_n; s_1, ..., s_m)$.

Here enters the spectrum of vacuousness. It is widely acknowledged, in fact, that "there are just too many isomorphisms, and all of them are equally good representations, if representation cannot 'cut through' isomorphism. In fact there are two problems here. Firstly, qua structure, there is nothing to distinguish a data-model of

the simple periodic motion of a pendulum from that of a suitably-described economic cycle. Secondly, even when the subject of the model is fixed, we can define a relational structure on its subject domain, cardinality permitting, in such a way as to guarantee isomorphism." [S. Psillos and R. Hendry 2007: 149]. We shall call *absolute liberalism* the first kind of unwanted liberalism, i.e. that which derives from not having specified the subject domain of the model. The second kind of liberalism, which derives from not having specified the way in which a specific subject domain should be carved, shall be called *relative liberalism*.

The moral I wish to draw from the vacuousness of purely structuralist accounts of mathematical representation, I anticipate, is the following. First, the problem of vacuousness (or of excessive liberalism) is not endogenous to computational theories, but it is common to all purely structuralist accounts of mathematical representation. Secondly, standard combating manoeuvres, which entail a resort to specific context-fixing physical properties, are not affordable in the case of computational properties. I shall argue, in other words, that *any* use of mathematical structures for representing reality would be subject to the same antirealist arguments, if it had to accommodate for the unrestricted PMR. Only an a-priori restriction of the intended implementing medium has any chance of capturing just the correct amount of liberalism in these cases.

The mathematical representation of computational structures, hence the explanatory power of computational theories, stands in exactly the same position. In the case of Fresnel's law, the domain of unobservable entities over which we define the isomorphism that ought to ground the relation of instantiation, is unspecified for epistemological reasons: nothing *can be* assumed to be known about it aside from structure. In the case of structuralist accounts of implementation, instead, the implementing domain is unspecified for reasons intrinsic to the notion of computation, i.e. because of the requirement that implementation be indifferent to any *particular* physical property: nothing *should be* known about it, aside from structure.

The different reasons why the respective instantiating domains are unspecified, however, are irrelevant for our discussion. What counts is that in both cases an unspecified domain, of which only the structure is supposed to be known, is required to carry the burden of instantiation (implementation), with the sole aid of isomorphism. In both cases, in fact, the knowledge that a physical system realizes a given mathematical structure does not elicit any inference about its particular physical nature.

The alleged vacuousness of purely structural properties, a difficulty also known as Newman's problem, has always been one of the major puzzles for structural realist philosophers of science. At the heart of it lies the observation that "any collection of things can be organized so as to have the structure W, provided that there are the right number of them. Hence the doctrine that only structure is known involves the doctrine that nothing can be known that is not logically deducible from the mere fact of existence, except ("theoretically") the number of constituting objects". [Newman (1927): 144].

This critical remark was originally addressed to B. Russell's causal theory of perception. In his book *The Analysis of Matter*, Russell expressed the opinion that the only kind of knowledge that we can have of the unperceived causes of our perceptions is "structural": i.e. knowledge of the structure of causal relations that obtains between them. But of such a structure, Newman objected, "nothing is known (or nothing need be assumed to be known), but its existence". [ibid.: 144] Russell later conceded that this is indeed the case.

Any structuralist account of the relation between mathematical structures and the empirical systems that instantiate them, including of course that between computational structures and their implementing systems, must address Newman's problem. The unperceived causes of our perceptions (and their structure thereby), just as the unobservable properties postulated by physical theories, or like the unspecified properties that implement computational structures, cannot be uniquely represented if the only relation they have with their potential representations is supposed to be the existence of an isomorphic mapping.

There is an ongoing debate about how to best amend structuralist accounts to meet Newman's challenge, and a close scrutiny of the various options offered falls outside the scope of this paper. Here it suffices to mention that virtually all commentators agree on the following point. The way out of Newman's problem requires that "[t]o the general characterisation of theories in terms of state-space structure *we must add other information* […] and that information imposes further constraints on the domain that can possibly be represented by the structure. The theoretical variables for which the simultaneous values and their change are given by the structure are theoretically interpreted: *they refer to physical properties and relations*." [French & Saatsi (2006): 557, my emphasis]. While opinions differ as to what this "other information" should amount to, there is agreement about the fact that, if the theory is to avoid vacuousness,

such information must at least be sufficient to fix the particular domain of physical properties intended to be represented.

3. Structuralist accounts of implementation

According to structuralist accounts, a physical system realizes a computation if the computation *mirrors* its causal structure. The key notion, in this understanding, is that of structural isomorphism. According to D. Chalmers, for example, "a physical system implements a given computation when there exists a grouping of physical states of the system into state-types and a one-to-one mapping from formal states of the computation to physical state-types such that formal states related by an abstract state transition relation are mapped onto physical state-types related by a corresponding causal state transition relation" [Chalmers (1996): sec. 2]. The prima facie allure of this understanding is precisely that reference to the specifics of the intended relata is abstracted away in favour of their causal structure, thus making room for the PMR.

Let us take a closer look at how this account is supposed to work in the particularly simple case of finite state automata. At the abstract, mathematical level, computations are defined as structures that take strings of letters from a finite alphabet and manipulate them according to specified rules to output other strings of letters. An automaton, for example, is specified by three sets *X*, *Y* , and *Q*, and two functions δ and β, where:

1. X is a finite set (the input alphabet)
2. Y is a finite set (the output alphabet)
3. Q is the set of internal states
4. $\delta : Q \times X \rightarrow Q$, the next state function, is such that if at any time *t* the system is in state *q* and it receives input *x*, then at time *t+1* the system will be in state $\delta(q, x)$.
5. $\beta : Q \rightarrow Y$, the output function, is such that when the system is in state *q* it always yields output $\beta(q)$.

An account of implementation must specify how these abstract symbols should be interpreted if the structure is to represent some real property of the physical systems that realize it. According to structuralist accounts, a physical system (P) implements an automaton (A) if *there exist* (in P) specifiable input-, output- and internal states that

satisfy the following (structural) properties. There exists a mapping (f) that maps the internal states of P (Q_P) to abstract internal states of A (Q_A) in such a way that for every abstract state transition $(S_1) \rightarrow \delta(I_1, S_1) = S_2 \rightarrow \beta(S_2) = O_2$, if P is in internal state s_1 and receives input i_1 where $f(i_1) = I_1$ and $f(s_1) = S_1$, this causes it to enter state s_2 and to output o_2 such that $f(s_1) = S_2$ and $f(o_2) = O_2$ (see Chamers 1996: 393).

It is easy to see that this account proceeds from a Ramseyfication of computational statements. The empirical content of a computational theory that ascribes to a physical system the property of implementing *A*, in fact, is reduced to a conjunction of existentially quantified properties that "mirror" its formal structure. As we have seen, "[t]he problem is that this procedure trivializes [the theory]: it threatens to turn the *empirical* claims of science into mere *mathematical* truths. More precisely, if our theory is consistent, and if all its purely observational consequences are true, then the truth of the Ramsey-sentence *follows* as a theorem of set theory or second-order logic, provided our initial domain has the right cardinality- if it doesn't, then the consistency of our theory again implies the existence of a domain that does." [Demopulos and Friedman (1985): 635]

To overcome Newman's problem, we said, it is necessary to add "further constraints on the domain that can possibly be represented by the structure". This standard combating manoeuvre, however, is not available in the case of structuralist accounts of implementation, for the variables featuring in computational structures, those that stand for abstract computational inputs, outputs and states, cannot be (a-priori) theoretically interpreted as referring to (specific) physical properties and relations, on pain of violating the PMR.

Here is an example of the (insurmountable) difficulty that structuralist accounts of implementation must face when confronted with Newman's problem. According to Copeland's structuralist account, the relation between computational structures and their implementations is akin to that between formal theories and their models. It may be conceded, thinks Copeland, that there are interpretations under which a given physical object, Searle's wall, for example, is implementing the Wordstar program; but, it is argued, this is not the case under the *intended* interpretation: "[t]he wall so acted", says Copeland, "only if the referent of 'R' in Skolem's countable model is uncountable!" [Copeland (1996): 353]. But, I argue, if we didn't have any prior (metatheoretical) information about the intended meaning of 'R' in a theory of real numbers (e.g. about

its *real* cardinality), there would be no way to discriminate away Skölem's countable model as *non intended*.[5] Likewise, if we are not allowed to specify any particular non-structural property of the intended model, how is Copeland's wall supposed to "know" that, unlike Searle's, it is not intended to implement the Wordstar program?

The restrictions imposed by the PMR on the possible specifications of the implementing mediums act on two levels on abstraction. At the most abstract level they require that absolutely *any* kind of physical property should, a priori, be a viable candidate for implementing a given computational state. This kind of liberalism, that we have called absolute liberalism, opens the way to strong antirealist arguments (those to the effect that *any* computational structure is instantiated by *any* physical object). These arguments, in fact, exploit the possibility to choose among the huge variety of physical properties instantiated by any macroscopic object. Putnam's own diagnosis of the problem of vacuousness, for example, points at the need to "restrict the class of allowable realizers to disjunctions of basic physical states [...] which really do (in an intuitive sense) have something in common". [Putnam (1988): 100].

But even once a certain kind of property, say electric voltage, is selected as the relevant candidate kind of implementing property, there still remains a degree of arbitrariness as to how to build the groupings of values that are to be mapped onto abstract computational states. Weak antirealist arguments typically exploit relative liberalism, i.e. the possibility to group arbitrarily physical states within the selected class of candidates. M. Scheutz, for example, observes that "if no restrictions are imposed on groupings of physical states, then simple, finite, deterministic physical systems […] can possibly be seen to implement complex, infinite, and non-deterministic computations." [Scheutz (2001): 551].

In a number of recent publications, O. Shagrir (2001; 2007) has proposed a compelling example of the under-determination of implementation that also exploits this second level of arbitrariness. It is worth presenting it briefly. Consider a device that receives 0-100 mV inputs from two channels, and emits voltages within the same range from an output channel. Suppose further that the device emits 50-100mV signals just in case it receives 50-100mV signals on both input channels and it emits 0-50mV signals otherwise. It would be tempting to assign "0" as a label to 0-50mV signals (input and output), and the label "1" to 50-100mV signals. Under this assignment, the structuralist

[5] Cantor's theorem is of course true also in Skolem's countable model.

would be ready to say that the device is implementing an AND gate. Now, without changing these assumptions, suppose further that the device emits 0-25mV when it receives 0-25mV signals on both channels, and 25-50mV when it receives 25-50mV on both channels. Assigning the label "0" to 0-25mV signals and "1" to 25-100mV signals, the very same device can be seen as implementing an OR gate.

Let me summarize what was said so far. On one side, the need to tackle Newman's problem forces any structuralist account of instantiation (including structuralist accounts of implementation) to suitably restrict the candidate domain of instantiating properties (in order to avoid vacuousness, or excessive liberalism). On the other side, the PMR, in the case of implementation of computational structures, places constraints to what these restrictions should amount to. In particular, such restrictions should not make reference to any *specific* domain of physical properties.

The computational realist, in other words, must face the following insidious dilemma. Either (a) she introduces semantical postulates (that restrict the possible interpretations of the abstract structures) in the attempt to individuate *only* the intended models of her theory, thus running the risk of failing to capture *all* the wanted models; or (b) she holds onto the structuralist view of computational theories, in the attempt to capture *all* the intended models, thus running the risk of capturing *also* unintended ones. This dilemma presents structuralist accounts of implementation with a bill that, I argue, they cannot afford to pay.

As anticipated, my view is that the only viable option is to require that the candidate implementing states (or groupings of states) be restricted to items that instantiate intentional properties. Before turning to a clarification and defence of this thesis, however, I will discuss another option, functionalist accounts. I shall argue that they are just as unsuitable to meet Newman's challenge.

4. Functionalist accounts of implementation

Acknowledging that structuralist accounts are unsuitable to capture all and only the wanted models of computational theories, some authors rely on functional analysis. After all, the way computational states are individuated by the relevant community of experts (computer scientists, hardware engineers and computational neuroscientists) relies on the ascriptions of functional roles to the various parts of computing mechanisms.

Functional analyses and explanations make reference to functional properties, abstracting from the specifics of their implementation, and are thus prima facie suitable for satisfying the multiple realizability constraint. Whether something is a carburettor, or whether something is a heart, for example, does not depend on physical makeup: carburettors and hearts are what they are because of the function they serve in the overall mechanisms in which they are embedded (respectively combustion engines and circulatory systems). Similarly, whether something is a memory cell or not, or whether something is an input device or not, according to functionalist accounts, depends on the function the item has in the overall computing mechanism to which it belongs.

Piccinini, in a series of recent papers, has proposed a functionalist account of implementation according to which "the central idea is to explicate computing mechanisms as systems subject to mechanistic explanation. By mechanistic explanation of a system *X*, I mean a description of *X* in terms of spatiotemporal components of *X*, their functions, and their organization, to the effect that *X* possesses its capacities because of how *X*'s components and their functions are organized. […] Computing mechanisms, including computers", he goes on to argue, "are mechanisms whose function is computing". [Piccinini (2007a): 506-507].

Setting aside the serious philosophical difficulties encountered in providing a realist account of functional properties in general, I shall concentrate on a problem that is restricted to computational functional properties. The functional properties of an item are typically defined in terms of the physical effects that that the item *ought to* yield. Thus, for example, carburettors are defined as devices that blend air and fuel for an internal combustion engine, and hearts are defined as organs that pump blood in blood vessels. Although both carburettors and hearts can be made of different stuff, the effects that they (ought to) yield are characterized by making reference to *specific* physical properties, or at least to restricted disjunctions of physical properties.

It could be argued that "blood" or "fuel" are not names of natural kinds: not all bloods, for example, use haemoglobin to carry oxygen; and not all fuels share the same chemical composition. But whether a fluid carries oxygen or whether it enters a combustion cycle as fuel, are objective matters of fact specified by the relevant classes of physical phenomena. Thus, as Craver observes, "[t]he heart cannot expel blood […] without blood, and the expulsion of blood will only circulate it […] if the veins and arteries are appropriately organized." [Craver (2001): 64]. It is obvious that if blood was

defined as whatever it is that is pumped by hearts, then the functional definition of hearts as organs that pump blood would be patently circular.

Computational properties, I shall argue, are a-priori unsuitable for being analyzed in purely functional terms. Because of the PMR constraint, purely functional accounts of computing systems are bound to be either circular or part of an infinite regress of functional descriptions.

The first task of a theory of implementation is that of providing physical counterparts for the letters of the abstract alphabet. The candidate physical counterparts of the variables in other mathematical structures, such as other mathematical dynamical systems, we have seen, are physical magnitudes, such as light intensity, or mass. But what is a candidate physical "letter"? What is a candidate physical "alphabet"?

According to the functionalist account, a set of physical items (a state, or an entity) counts as an implementation of an alphabet (i.e. as the intended interpretation of *X* and *Y*, in our example), if and only if it functions as such for a physical system that realizes the appropriate transformations (i.e. the systems whose states, or groupings of states, realize the set *Q*). But what is it for a set of entities to *function as* a system of letters?

According to Piccinini "[a] system of digits [i.e. physical counterparts to the letters of an alphabet, nda] is individuated by the digits' functional roles within a mechanism." [Piccinini (2007): 510] But how can we define such roles? How are the relevant parts of a computing mechanism to be defined? "Input devices", continues Piccinini, "have the function of turning external stimuli into strings of digits" [ibid.: 514]. The latter are then passed onto other processing components, that "have the function of taking strings of digits as inputs and returning others as outputs according to a fixed rule defined over the strings" [ibid.: 514]. Now, unless we add to these characterizations some necessary and sufficient condition for a mechanism to *function* that way, e.g. for a set of states to function as a system of digits, or for a mechanism to function as a memory cell, these and similar definitions are bound to be circular.

If an item X (a set of states, a class of groupings of states, or a set of entities) is a system of digits if and only if its elements have the function of being output from an input device and input to a processing component, then an input device would be a mechanism that has the function of outputting the outputs of an input device (whatever these are). Similarly, a processing component would be defined as a mechanism that has the function of taking as inputs the inputs of processing components! In sum, either the chain of functional descriptions of the components of a computing mechanism can be

interrupted by introducing an element characterized in terms of property-specific inputs and outputs, or the whole functional architecture will be trapped in a functional marry-go-round.

Piccinini concedes that "[t]he resulting account is not intended as a list of necessary and sufficient conditions, but as an explication of the properties that are most central to computing mechanisms." [ibid.: 508]. But, I argue, any attempt to produce such a list would either make the circularities described above explicit, or it would violate the PMR constraint.

Consider, for example, the following general functional constraint proposed to discriminate computations from other kinds of mechanisms: "[i]n a computing mechanism, under normal conditions, digits of the same type affect primitive components of a mechanism in *sufficiently* similar ways that their dissimilarities *make no difference* to the resulting output." [ibid.: 510]

The crucial difficulty, here, is that while the difference between two inputs is a matter of objective fact, whether such difference also *makes a difference* depends on who or what is to notice the difference. This point was made very clearly by A. Turing in a seminal paper. "[D]iscrete machines", he claims, "are the machines which move by sudden jumps or clicks from one quite definite state to another. These states are *sufficiently different* for the possibility of confusion between them to be *ignored*. Strictly speaking there are no such machines. Everything really moves continuously. But there are many kinds of machines which can *profitably be thought of as being* discrete state machines." [Turing (1950): 36, my emphasis]. The question, of course, is: profitable to whom?

The following example should help to clarify why the requirement that digits of the "same type" affect computing mechanism in "sufficiently similar ways" runs the risk of being vacuously or too liberally satisfied. "[I]f two inputs to a NOT gate", argues Piccinini, "are *sufficiently close* to a certain voltage (labeled type '0'), the outputs from the gate in response to the two inputs must be of voltages *different* from the input voltages but *sufficiently close* to a certain other value (labeled type '1') that their difference does not affect further processing by other logic gates." [Ibid: 511]

There are two problems here. First, a restriction of the candidate label bearers to the magnitude voltage cannot be inbuilt to the notion of implementation, for familiar reasons (absolute liberalism). This problem is not to be considered as an exercise of armchair philosophy. The debated computational status of neurons is particularly apt to

expose this difficulty. Neurons have been always considered as privileged candidates for implementing memory cells. It is certainly possible, in fact, for all practical purposes, to type-identify neurons according to whether they are activated or not. The hypothesis that they act as flip-flops, however, must face a number of challenges. To mention only a few:

1. It is the frequency of firings within a neuron and not the mere presence of action potentials, that is relevant in causally influencing the behaviour of neighbouring ones.
2. The effect of the same type of input to a neuron changes substantially depending on where in the receiving neuron the input is passed.
3. There are properties of neurons that must be arbitrarily disregarded in order to treat neurons as flip-flops (e.g. facilitation, extinction and learning).

What counts is whether these "deviant" properties can be *neglected* or *disregarded* for the purpose of computational analysis. But this depends, in the first place, on the choice of candidate label bearers among the variety of physical properties that neurons instantiate. McCulloch and Pitts, for example, were aware that there are properties that alter the response of neurons and that would disrupt, if taken into account, their treatment as bi-stable devices. Nevertheless, they thought that these properties need not disrupt the formal (computational) treatment of the activity of neurons: "[t]he alterations actually underlying facilitation, extinction and learning in no way affect the conclusion which follows from the formal treatment of the activity of nervous nets, and the relation of the corresponding propositions remain those of the logic of propositions." [McCulloch and Pitts (1943): 352]

Other authors, instead, think that the above mentioned "deviant" properties block a plausible treatment of neurons as binary memory cells: "[t]he principles of computer memories can hardy be realized in biological organisms [because] all signals in computers are binary whereas the neural signals are usually trains of pulses with variable frequency." [Welles (1998): 200]. It is clear that purely functional considerations do not suffice to settle this controversial issue.

Similarly, whether the two inputs to the NOT gate in Piccinini's example are "similar", or whether the respective responses of the mechanism are "sufficiently close" to each other - hence whether the mechanism in question is in fact a NOT gate - depends, in the first place, on what physical magnitudes are selected as candidate label bearers.

In the second place, even if the physical domain of candidate label bearers could be suitably restricted, there would still remain a measure of relative liberalism that cannot be eliminated by adding purely functional constraints. "[E]ven a slight change in the grouping of a single neural property", argues Shagrir for example, "can completely alter the *logical operators* we take the brain to implement. Under a grouping of 0-50mv neural activity into groups of 0-25mv and 25-50mv, the resulting logical operation is AND, but under a grouping into 0-15mv and 15-50mv groups, it is OR." [Shagrir (2005): 240].

This perspectival aspect of causal role ascriptions, when considerations that fix the relevant contexts are not allowed, is largely acknowledged, and is not restricted to computational properties.[6] "Even slight differences in mechanistic context", thinks Craver, "entail different mechanistic role functions. […] Judgements of 'sameness' in these cases depend upon an agreed-upon tolerance of diversity among tokens within types." [Craver (2001): 73]

Piccinini is right, I think, in noting "that the relevant community of scientists can identify a mechanism's functionally relevant components and properties" [Ibid: 508]. But hardware engineers, unlike philosophers of computation, are never faced with the daunting task of individuating the parts of a system without knowing what they are made of, or even without knowing what they are specifically supposed to be doing. What is playing the role of what, in all practical scenarios, is already established in advance: the problem for engineers is that of ensuring that a given part is yielding the desired effects, as these are specified by their physical properties. It is relative to these practical, context-dependent considerations that it is "profitable" to think of a given processing component as performing a certain computational function. "Describing an item's mechanistic role", argues Craver, "is a perspectival affair. This perspectival take on functional ascription should be a reminder that what we take as functional descriptions can be tinged in a very direct way by our interests and biases (see e.g., Amundson 2000; Gould 1981)." [Craver (2001): 73]. How can Piccinini reconcile this perspectival take on mechanistic descriptions with the purported objectivity of the computational properties that they ought to ground?

[6] For a detailed analysis of the problem of liberalism in functionalist accounts see N. Block 1980 and A. Weir (2001). Weir argues that functionalist accounts based on D. Lewis' analysis fall prey of the chauvinistic horn of the dilemma, while other accounts fail to block excessively liberal implementations.

In the next section I consider the third category of accounts of implementation, that of semantic accounts, arguing that it is the only one that complies with the desiderata of computational realism.

5. Semantic accounts of implementation

There are at least three theses that deserve the qualification of *semantic accounts* of implementation. The first one - which I will call Fodor's thesis (FT henceforth) - is the conjunction of the following two claims:

> **FT1**: Computations are defined *over* representations.
> **FT2**: The semantic properties of the representations over which computations are defined do not have an impact on the individuation of computational states.

FT1 must be understood as the claim that all computational states, qua computational states, possess semantic properties (i.e. that they are representations). In other words, given the information that a physical state (or grouping of physical states) implements a computational state, it is analytic to infer that it possesses semantic properties, i.e. that it represents something.

FT1 is not committed to any specific kind of representation. In particular it is not committed to the claim that computational states are (analytically) conceptual representations. Part of the original allure of computational theories was their alleged capacity to vindicate folk-psychological explanations. To this purpose, conceptual representations become a necessary ingredient. But most current computational theories are used to explain cognitive capacities that only require that the implementing system possesses perceptual representational capacities, without presupposing that the system possesses also the concepts that would feature in a description of the content of these representations.

FT2 should be understood as the claim that the *particular* content of the representations over which computations are defined does not have an impact on the individuation of computational states, or on their causal properties. Thus, according to FT, computational states are individuated by their physical (non-semantic) properties, and act upon each other in ways that depend solely on these. The *locus classicus* of FT is Fodor's contention that "computational processes are both symbolic and formal. They

are symbolic because they are defined over representations, and they are formal because they apply to representations, in virtue of (roughly) the syntax of the representations." [Fodor (1980): 64].

The second thesis that goes under the umbrella of semantic accounts of implementation - call it Shagrir's thesis (ST) - is a conjunction of FT1 with the following two claims:

> **ST1**: The contents of the representations over which computations are defined have an impact on the individuation of computational states. But…
> **ST2**: It is only the formal, logical or set-theoretic semantic properties that have an impact on the individuation of computational states.

Formal semantic properties are second-order properties of semantic ones. Consider, for example a representation of the concept *black dog*. Among its semantic properties, is its content: this is, say, the extension of the concept. But a representation of the concept *black dog* has also the (second order) property of representing the intersection of two sets: that of *dogs* with that of *black things*. Second order, formal properties of the latter kind are shared by many representations that do not share any first order semantic properties. It is this kind of properties, and never the former kind that, according to ST2, has an impact on the individuation of computational states. In what follows we shall refer to the former kind of semantic property as *first order content*, and to the second kind (formal semantic properties) as *second order content*.

The third semantic thesis that we shall consider is Burge's thesis (BT). It is the conjunction of FT1, ST1 and the following claim:

> **BT1**: It is the first-order contents of the representations over which computations are defined that have an impact on the individuation of computational states.

Let me summarize this articulation of semantic accounts of implementation. All the theses mentioned share the assumption that computations are defined *over* representations (FT1). It is because of this that they all deserve the qualification of *semantic* accounts. ST and BT share the further assumption that the contents of the representations have an impact on the individuation of computational states (ST1), but

they disagree as to whether it is first-order content (BT) or second-order content (ST) that has such an impact.

Let us now consider in turn the reasons offered in support of each of these theses. The main reason for endorsing FT1 stems from the observation that computations are always computations *of* something, as representations are always representations *of* something: it doesn't make any sense to talk of a computation without assuming that there is something that is being computed; as it does not make any sense to say that an entity is a representation without assuming that there is something that it represents.

Another way to express this intuition is to say that computations are typically conceived of as a kind of information processing. Although there is no agreement as to what carrying information amounts to, it is undisputed that information must be information *about* something. Representations have the essential property of being *about* something, and are thus the most natural candidates as bearers of computational properties.

It is not trivial to turn the above considerations into an argument in favour of FT1. The following, though, is a promising candidate:

1. Computations are individuated by the functions computed
2. Functions are specified semantically, therefore
3. Computations are (at least in part) individuated semantically.

It is worth spending a few words about the premises of this argument. Computations, we said, are always computations *of* a function. To describe the function computed by a system, is to characterize that system in terms of what it "does". Consider for example Marr and Hildreth's (1982) description of what the retina "does" when performing the task of edge detection. According to the authors, the edges of an image are detected by signalling sharp intensity changes. These, in turn, are signalled by zero-crossings of the two-dimensional intensity arrays that are input to the retina (i.e. points where the intensity functions change their signs). Thus, the idea is that the retina receives as inputs the intensity arrays I(x,y) and *computes* the positions of their zero-crossings. "I have argued", says Marr, "that from a computational point of view [the retina] signals $\nabla^2 G * I$ (the X channels) and its time derivative $\frac{\partial \nabla^2 G * I}{\partial t}$ (the Y channels). From a computational point of view, this is a precise characterization of what the retina *does*." [Marr (1982): 337, my emphasis].

So, according to the authors, the retina "solves" the problem of edge detection by implementing the function $\nabla^2 G$. The authors are well aware that the retina "does" a whole lot of other things, but what counts is whether what the retina is seen as doing is also relevant for a computational description of it. "Of course", continues Marr, "[the retina] does a lot more - it transduces the light, allows for a huge dynamic range, has a fovea with interesting characteristics, can be moved around, and so forth. What you accept as a reasonable description of what the retina does depends on your point of view. I personally accept $\nabla^2 G$ as an adequate description, although I take an unashamedly information-processing point of view." [ibid.]

Thus Marr thinks that what individuates $\nabla^2 G$ as an appropriate description of what the retina *does* (among all the other functions that it might instantiate), is the fact that its outputs (the X and Y channels) are reliably correlated with salient features of normal environments, such as the sharp boundaries of distal objects. The outputs of the retina, in other words, carry information *about* the environment; and if it wasn't for this, it would not be possible to take $\nabla^2 G$ as a "precise characterization of what the retina *does*."

Does the above example (or similar ones) provide support for FT1? There are three opinions about it. According to the first one, perhaps the majority view, it does. Thus, for example, Burge, Peackoke, Davies, Churchland, Senjowsky and Shagrir, share the opinion that the above argument supports the thesis that computations are operations *on* representations. They disagree as to whether the content of these representations have an impact on computational individuation; and those who think that it does further disagree about which particular features of content (first- or second-order, broad or narrow) have such an impact. Here, however, our concern is evaluating FT1 in its own right.

The second stance about the argument, advocated for example by Egan and McGinn, is that these examples show that computational states can be seen as essentially representational only in an "unusual way". Egan thinks, for example, that the retina can be seen as implementing the function $\nabla^2 G * I$ whether or not its inputs and outputs carry information about distal stimuli. In some sense, the intended interpretation of the states of the retina is mathematical. "[T]he theory of the computation is intentional in the following sense: it does specify an intended interpretation of a computational process - the intended interpretation is *mathematical.* The topmost level of a computational theory [Marr's "computational level", or Newell's "semantic level",

nda] characterizes the system as computing a series of functions defined on mathematical entities. I am quite happy to say that a computational theory is intentional in this rather unusual sense." [Egan (1995): 187, footnote 8]

As we have discussed at length in the previous sections, however, because of Newman's problem, mathematical representations succeed at representing something (hence succeed at being representations at all), only if constraints are placed on the interpretations of the symbols, over and above the existence of isomorphic mappings. Thus, the knowledge that the retina implements the function $\nabla^2 G * I$, short of further specifications, is no knowledge at all. But, as we have discussed, the kind of constraints that are typically adopted in standard mathematical representations of physical properties are not allowed in the individuation of computational ones. As a consequence, I argue, there is no sense in which, short of further constraints, the retina could be seen as "representing" that mathematical structure.

Finally, there are authors who think that these examples do not provide any evidence whatsoever for FT1. Piccinini, for example, thinks that these alleged arguments from the semantic specification of the computed functions commit the following fallacy. It may be conceded, the counterargument goes, that computations are defined by the functions computed. But functions can be individuated in two ways: as defined (semantically) over the contents of the inputs and outputs that enter the computations (e.g. numbers, when the function computed is an arithmetical operation); or over the strings of inputs and outputs themselves (e.g. the numerals). The latter, according to Piccinini, are strings of symbols individuated by the physical types to which they belong, independently of the content that they may be seen as having, or indeed independently of whether they do have a content at all. The characterization that is relevant for individuating the functions computed, thinks Piccinini, is that based on symbols, not that based on their putative content.

Now, Piccinini is certainly right that a function can be defined over the inputs and outputs themselves, but this is not a different characterization of the *same* function: it is a different function altogether. This is true regardless of whether the arguments of the latter function can be further seen as *representing* those of the former one. If FT1 is true, the individuation of the function defined over the strings of inputs and outputs requires that the strings of symbols be themselves represented. Thus Piccinini's contention that the function defined over the strings of input and output digits does not require that these be themselves represented, begs the question against FT1.

The point I am making can be better appreciated by considering the following example. There are computations that are not amenable to a straightforward semantic interpretation. One such computation is represented by the following machine table[7]:

	@	1
Q_1	1, q_2, R	@, q_4, L
Q_2	1, q_3, L	1, q_4, R
Q_3	1, q_1, L	1, q_3, L
Q_4	1, h, R	1, q_5, R
Q_5	1, q_1, R	@, q_2, R

The input architecture of this machine only possesses two symbols: @ and 1. Its internal structure comprises only five states: q_1, q_2, q_3, q_4, q_5. If started in state q_1 on a blank tape, in spite of its simplicity, the machine has been proved to halt only after 23,554,764 steps. Considering how simple the machine is, this result is rather remarkable! What interests us here is that this machine computes a function that appears to have no straightforward interpretation. What could 1 and @ be systematically interpreted as referring to, in order for the computation to make any sense? Piccinini has argued that cases like this prove that FT1 must be false. If there are machines whose input architecture is not amenable to *any* interpretations, the thesis that inputs and outputs must be representations is reduced ad absurdum. If 1 and @ in the example above cannot be interpreted as representing anything, how could they be representations at all?

Now, because the symbols that feature in the table above have no possible interpretation, the function that individuates the computations must be that defined over input- and output-types: true. Piccinini exploits this fact to argue that there are cases where input- and output-types are not (and could not be) representations. In spite of the absence of representations, the argument goes, the computation above can be individuated and implemented, hence FT1 must be false.

But, I argue, if you did not "see" that the tokens "@" and "1" that you have just read in the table above, are intended to "refer" to different input- and output-types, the

[7] The discovery of this algorithm is due to J. Buntrock and H. Marxen.

computation would not be univocally individuated. Unless we presuppose that FT1 (which is the thesis at stake in the argument) is false, in fact, it can be argued that the only way to fix an intended model for the realization of the computation is by deploying semantic properties that are identical (or isomorphic) to the ones you have just deployed to interpret the machine table above as *that* machine table. It does not suffice that you are able to type-identify @'s and 1's by their shape. The expression "the shape of @", by itself, in fact, is insufficient to individuate any particular feature of that mark of ink. You are also able to type-identify the same two tokens according to categories that cut across the ones to which @'s and 1's belong in the *intended* interpretation of the computation. For example, both @'s and 1's are equally tall marks. If the relevant discriminatory criterion were height, the algorithm specified by the table would be reduced to computing the identity function. Real, physical computing systems stand in exactly the same position as the abstract, inert table above: what a system *does*, from a computational point of view, can only be individuated by bringing in semantic considerations.

Similarly, a description of the process of arithmetical addition presupposes a description of the relevant mathematical symbols (the numerals, the sign "+", etc.), over and above the algorithm to be implemented. It is far from obvious that we could ever perform an addition if we did not have representations of these symbols. Just as an operation over numbers (1,2,… etc.) requires that we manipulate the names of these numbers ("1", "2"… etc.), an operations over the numerals requires that we manipulate the names of these numerals. Indeed it could be argued that we acquire the representations of the numerals, and learn to manipulate them, long before we come to understand what they are intended to represent.

Let me conclude our discussion of the virtues of FT1 with a summary. Arguments in favour of FT1 are, I believe, not definitely conclusive. It is in fact possible that some other properties (other then representational ones), might turn out to serve the same individuating role. Our analysis, however, allows us to draw the following conclusions:

1. The non-semantic individuating properties proposed in the literature are not viable. Both purely structural and purely functional properties, I have argued, do not comply with the relevant desiderata.
2. A-priori arguments against the semantic individuation (like the one from uninterpretable Turing machines), are inconclusive.

Arguments in favour of FT1 sometimes proceed from the claim that content has a role in the individuation of computational states: as only representations have contents, if contents impact the individuation of computational states, computational states are essentially representational (FT1). As I said, according to both ST and BT the content of representations has an impact in the individuation of computational states (ST1). Arguments in favour of ST1 (both those according to which it is first-order content and those according to which it is second-order content that matters) proceed from the observation that computational explanations make an irreducible recourse to content.

Burge, for example, has argued that Marr's theory of vision makes essential reference to the (distal) content of perceptual representation. His aim was to show that the theory is not individualistic (that the content in question is broad, rather than narrow). Here, however, we are only interested in the part of the argument that purports to show that content (rather then other properties of the input architecture) has an impact in the individuation of computational states. According to Burge, "[Marr's] theory makes essential reference to the subject's distal stimuli and makes essential assumptions about contingent facts regarding the subject's physical environment." [Burge (1986): 29]. The information carried by the representations over which Marr's computational theory is defined is individuated by these distal stimuli. If these were different, Burge continues, the information carried by these representations would also be different, even with no variation in the internal physical processes of the implementing system. He thereby concludes that Marr's computational theory explains the relevant behaviour only if the individuation of the postulated (computational) structure is constrained by the (broad) contents of the proximal stimuli.

Various authors have objected (correctly, I think), that while it may be that (broad) content is a necessary ingredient in accounting for the explanatory capacity of Marr's theory (Butler), or for its intuitive understandability (Egan), this does not suffice to show that such content has an impact on the individuation of the computational structure implemented. The locus classicus for testing intuitions about this issue is Davies' example of the Visex system.

The Visex is a (fictional) component of a visual system. As the theory is concerned with explaining some *particular* visual cognitive behaviour, say edge detection, all explanations based on it will make reference to some feature of the visual environment of the system. Now, suppose that there is a subcomponent of the auditory system, call it

the Audex, that is intrinsically identical to the Visex (it is a molecular twin of the Visex). As the Audex is part of the computational structure of the auditory system, in explaining the auditory behaviour of the system, the theory will make irreducible reference to its auditory distal environment. Are the Audex and the Visex, under these circumstances, implementing the same computation, in spite of the fact that their respective representations have different broad contents?

Burge, in compliance with BT1, answers in the negative: the Visex and the Audex implement different computations. Most authors, however, disagree with Burge on this point. Egan, we have seen, thinks that the only "content" that computational states need be assumed to possess, is mathematical content. Thus, according to Egan, the Audex and the Visex implement the same computational structure by virtue of the fact that (by hypothesis) they both instantiate the same mathematical functions. While, as I have argued, Egan thus places computational descriptions at a level of abstraction that fails to achieve an adequate degree of individuation, I think that considering the Visex and the Audex as implementing different computations would place the computational analysis of the system at the opposite, but equally wrong, level of abstraction.

It could be conceded that if the Visex was applied, unaltered, to an auditory slot, the different nature of the distal stimuli would determine a shift from a description of a visual cognitive system to a description of an auditory system, without this implying that the system implements different computations in the two cases. A computational theory of vision, like Marr's theory, in fact, need not be taken to be "computational" in the strong sense that it appeals solely to computational concepts. What makes of Marr's theory a theory of *vision* (as opposed to a theory of audition), in other words, need not be anything that has to do with what makes of Marr's theory a computational theory (see Butler 1996).

In sum, while the above considerations do not rule out (a-priori) that in some cases content may have an impact on computational individuation (ST1), they appear to lessen the plausibility of the hypothesis that first-order content always does (BT1).

We are thus left with the last proposal that we wish to discuss: that it is second-order content that has an impact on computational individuation. Shagrir, who shares with Burge both FT1 and ST1, also thinks that the Visex and the Audex can be seen as implementing the same computation. But this, he argues, is the case only if the mathematical (logical or set-theoretical) properties of the distal stimuli are taken to be identical in the two cases. We have seen that according to Marr's model of edge

detection, what the retina *does* is appropriately characterized by the formula $\nabla^2 G * I$. The formula describes the relation that obtains between the (electrical) activity of the photoreceptors that input the retinal image *I(x,y)* and the (electrical) signals that reach the edge detectors. But this fact alone, argues Shagrir, does not explain why edge detectors carry information about the boundaries of distal objects.

Indeed, I add, this fact alone doesn't suffice to prove that the electrical activity of the retina implements any computation at all. What is crucial, continues Shagrir, is that the mathematical properties in question mirror the mathematical properties of the represented items (e.g. the relation between object boundaries and sharp changes in the intensity of reflected light). This is not only crucial in accounting for the fact that Marr's theory of edge detection is indeed a theory of edge detection, but it is also crucial in determining what the retina *does* (computationally) in a sense that cuts some ice. The retina, in fact, may be seen as having many other implementational capacities. Had the second-order content of the representations been different, the retina may well have been seen as actually implementing another computation.

I think - although Shagrir is not explicit on this point - that the thesis expressed above (ST2) has the consequence that, if the mathematical properties of the photoreceptors/edge-detectors system didn't mirror the mathematical properties of *any* represented domain, the system would not be seen as implementing *any* computation at all. This point deserves some arguing, for it is crucial for assessing whether Shagrir's arguments can be used to argue in favour of FT1 or not.

As we said, Shagrir's argument proceeds from the claim that sometimes the same physical system can be seen as potentially instantiating various syntactic structures. Each of these is a *potential* computational structure. In these cases the second-order contents of the representations select which of these *potential* structures is *actually* implemented. But even if we concede that this is the case, there still remains a crucial question to be answered. Suppose that the physical properties of a system were such as to reduce the number of potential implementations to exactly one: could we then say, contrary to FT1, that that one computational structure is *certainly* implemented by the system?

Piccinini thinks that Shagrir's arguments, while perhaps more stringent that other arguments for a semantic account of implementation, make life a lot easier for functionalists. Shagrir, infact, according to Piccinini, has made a crucial concession. The functionalist notion of implementation, although still slightly too liberal, is seen as

being capable of constraining the number of potential implementations to only a few. Typically, Shargir's examples mention only two. While this is certainly due to the need for clarity, there is no doubt that Shagrir's thesis is very far from the radical vacuousness claims made by Putnam and Searle.

Piccinini's response exploits this alleged concession to blow what he thinks is a fatal stroke at semantic accounts. Functionalist accounts, he concedes, would indeed be too liberal if we construed functions in a narrow sense. If we excluded all reference to distal properties from the individuation of the relevant functions, that is, then Shagrir would be right: functional descriptions would indeed underdetermine the computations *actually* implemented by a system. But, thinks Piccinini, if we allow for a broad construal of functional properties, such liberalism could be avoided.

While it is certainly true that broad individuation of functional properties does not entail individuation based on broad content, I think Piccinini's move does not warrant the assumption that broad functional properties can be safely applied to the individuation of computational ones. Remember, in fact, that the troubles with functional individuations of computational properties that we have discussed in the previous section have to do with the need to comply with conflicting desiderata. None of these, I shall argue, rely on a narrow (as opposed to broad) individuation of functional properties.

I have argued that neither purely structuralist nor purely functionalist accounts of implementation succeed at restricting the class of candidate label bearers of computational states (CLB henceforth) so as to avoid vacuousness or excessive liberalism. The analysis of their common pattern of failure allows us to formulate the following desiderata for an adequate realist account of implementation.

1. If the account is to solve the problem of absolute liberalism, the class CLB must be restricted so as not to contain *any* possible physical magnitudes (or groupings of physical magnitudes).
2. If it is to solve the problem of relative liberalism, for any subclass of CLB whose elements belong to the same type of physical magnitudes, the restriction must be such as not to contain *all* arbitrary groupings of them.
3. Finally, if the account is to satisfy the principle of multiple realizability, the restriction must not constrain the elements of CLB to belong (a-priori) to some *specific* kind of physical magnitudes (or groupings thereby).

As I shall argue, the difficulty encountered by functionalist accounts does not depend on the assumption that functional properties be construed as narrow. The intuition that broad functional properties may serve the purpose of individuating actual implementations, thus rendering semantic properties redundant, stems from the observation that, if the individuation of computational properties is allowed to make reference to properties that lay in the environment, then the fact that computational theories make essential reference to distal stimuli would not entail that these must be *represented*.

According to Wilson's broad account of computation, for example, the problem of absolute liberalism can be blocked by making reference to the particular loci where programs are supposed to be stored. "In response to the grand epistemological, scepticism-mongering question, 'Of the infinite number of programs that a computer could be implementing, how do you know that it is implementing *this* program?', we say: 'It implements this one because it is this one that is *encoded* on the disk we inserted.' (And since a physical disk is simply one type of storehouse for a program, we could replace reference to a physical disk here by reference to anything else a program is stored on.)" [Wilson (1994): 360, my emphasis]

The problem, here, just as with narrow conceptions of implementation, is that unless we further assume that all the potential "encodings" of the program share some (first-order or second-order) properties, we are not in the position to specify where in the disk the program is encoded. Of course Wilson's broad constraints suffice to say that an object *can be* used as a computer, but, I argue, not to say that *it is* being used as such.

I think that the crucial misunderstanding in these attempts (narrow or broad) to block unwanted models, consists in supposing that *encodings* can be individuated without making reference to their representational capacities. As Millikan notes, "[i]t does not help to be told that inner representations are things that have representing (indicating, detecting) as their function […] unless we are also told what kind of activity representing (indicating, detecting) is. [Millikan (1989): 282-283]

The standard notion of "encoding", although this is rarely stated explicitly, is based on sheer factual correspondences. The mere fact that there are factual causal correspondences between some properties of the disk and properties in the proximal environment of the processing unit, is implicitly supposed to be sufficient for individuating the encodings themselves (thereby individuating the program stored). But

it is widely acknowledged that there are just too many factual correspondences between everything and virtually everything else. The mere presence of factual correspondences is only sufficient to claim that a given state has the *potentiality* to be treated *as* an encoding.

Similarly, a broad functional construal of Marr's theory of vision would count some distal features of the environment as parts of the functional architecture. Thus, describing a system as implementing Marr's algorithm would require a (functional) description of distal stimuli. But if the only constraint that we place on these distal features of the environment is that they be in factual correspondence with the proximal stimuli (themselves identified functionally in non property-specific ways), then we have placed no constraint at all. In fact, because of Newman's problem, there will always be, out there, some (unspecified) physical properties that are in factual correspondence with the (unspecified) proximal ones. The broad functional descriptions, in the case of Marr's theory of vision, for example, would have to individuate the intended relation between the boundaries of distal objects and the pattern of intensity of the light reflected by them. But could such individuation be possible if the *potential* encodings of these distal mathematical properties did not *actually* encode them? What if the are proximal stimuli that are in factual correspondence with other, non intended, distal properties, without encoding them?

The broad or narrow individuation of these properties is not the issue here. The existence of unwanted models is due to the absolute liberalism in the choice of the candidate label bearers. If the individuation of computational properties is allowed to range over properties that are instantiated outside the skull, the problem of absolute liberalism is brought out in the environment too. The general problem is that the procedure of Ramseyfication of theoretical computational statements, short of further constraints that ensure that they latch onto their intended models, is going to be exposed to the problem of liberalism, whether the candidate referents of the Ramseyfied properties are allowed to range over distal properties or not.

Another way to expose the indifference of the problem of liberalism to wide or narrow characterizations of function is the following. Wide functionalist accounts include behavioural and sensorial data in the functional architecture, in the hope to block unwanted implementations. But if the principles of functionalism are to be upheld, the properties of being "behavioural" or "sensorial" must themselves be functionally individuated, and this requirement brings us back to the same problem that

broad functionalism was devised to solve: either we restrict the domain of possible implementations, thus falling prey of the chauvinism horn of the dilemma, or we fall prey to the opposite horn, counting as "behavioural", or "sensorial", also what is clearly not behavioural or sensorial. As Skinner once put it, "[t]here must be defining properties on the sides of both stimulus and response or the classes will have no necessary reference to real aspects of behaviour" [Skinner (1938): 35]. The problem is that these "defining properties" cannot be themselves functionally characterized, on pain of vacuousness. We have seen how this is the case, for example, with functional characterizations of encodings.

In sum, purely functionalist accounts of implementation will be dogged by the spectrum of liberalism, wherever we chose to place the relevant candidate domains of implementation.

6. Conclusions

I don't think that there are currently any conclusive positive arguments in favour of semantic accounts of implementation. I argue, however, that placing the issue of liberalism of implementation in the broader framework of Neman's problem allows us to draw the following conclusions.

First, the proposed diagnosis of the problem of liberalism blames the PMR for the proliferation of unwanted models, rather than the discrete (as opposed to continuous) nature of digital systems. Of course the discrete nature of computational systems makes matters worse as far as liberalism is concerned, for it increases the cardinality of the unrestricted set CLB. However, although most vacuousness arguments exploit the discreteness of the set of label bearers, such greater cardinality should not be held directly responsible, in itself, for the problem of liberalism. Even the realization of continuous dynamical systems (e.g. the instantiation of the solutions of Maxwell's equations), we have seen, would be too liberal, if it had to accommodate for the unrestricted PMR.

Secondly, the semantic restriction of CLB, unlike the ones based on purely structural or purely functional properties, can be argued to satisfy the general desiderata of a theory of implementation. Remember, in fact, that while the need to avoid non-intended models requires (as a necessary but not sufficient condition) that the class CLB be restricted so as not to contain *all* sets of physical magnitudes (or groupings of physical magnitudes), the PMR constraint requires that such restriction be not based on

any *particular* kind of (first- or second-order) physical property. Restricting CLB to entities that possess intentional properties complies with these apparently conflicting desiderata. While virtually everything can be used to represent everything else, in fact, not everything is *actually* a representation. Thus, while a semantic restriction of CLB does not constrain (a-priori) its elements to share any *particular* physical property, it ensures that at any given moment the extension of CLB be not the whole universe of physical magnitudes and groupings of states. This, as we said, is exactly as it should be. Notice, for example that the thought experiments of the kind proposed by Putnam or Searle would be blocked at their onset by a semantic restriction of CLB. The states of Searle's wall, or Putnam's microscopic maximal states, in fact, are not candidate label bearers (under a semantic restriction), for they arguably do not possess any representational property.

The third observation that is in order regards the relation between semantic and other accounts of implementation. The semantic hypothesis advertised in this paper by no means entails that possessing representational properties suffices for implementing computational ones. This is clearly false. Rather, the hypothesis should be stated as follows. If the problem of liberalism is to be avoided, a necessary condition for the implementation of computational properties is that the inputs and outputs to the candidate implementing system instantiate *actual* intentional properties, whose transformations can be systematically subsumed under true (counterfactual supporting) syntactic generalizations. The best way to capture such syntactic generalizations may well require the realization of functional properties. Moreover, the isomorphisms postulated by structuralist accounts will be seen to emerge naturally, when such functional + semantic structures are implemented. Therefore, the semantic account should not be construed as alternative to structuralist or functionalist ones, but as complementary to them. For example, a mechanistic characterization of input devices will have to make an irreducible reference to *actual* semantic properties (those that individuate the encodings). Once the set CLB is thus restricted, however, the particular structure implemented may well be individuated by a functional or mechanistic description. When such a functional + semantic architecture has been individuated, it will be possible to map the implementing label bearers onto their abstract counterparts, as prescribed by structuralist accounts. The difference is that while purely structuralist and functionalist accounts provide a top-down characterization of the relation of implementation (from abstract to concrete systems, via Ramseyfication), the semantic

account fixes the relevant constraints at the bottom (semantic) level, thus securing the ground for the relevant computational abstractions. The computational Ramsey sentences will thus be true as a matter of contingent fact, as opposed to being true as a matter of logic. This is exactly as is should be: whether something is a computing system or not, if computational properties are real, must be a matter of contingent fact, not logic.

Finally, although, as I said, it is not possible to prove that *only* semantic properties can suitably restrict the class CLB, the other proposed candidates certainly fail to comply with the relevant desiderata. Moreover, I have argued, standard a-priori objections raised against a semantic restriction of CLB are inconclusive. I have argued elsewhere that teleological theories of content are particularly apt to enforce a restriction of CLB, but as far as the problem of liberalism is concerned, any successful non-functionalist theory of content would be just as suitable to deal with the relevant desiderata. In sum, I argue, semantic accounts of implementation are so far the best bet on the board.

References

Amundson, R. (2000), *Against Normal Function*, Studies in the History and Philosophy of the Biological and Biomedical Sciences, 31: 33-53.

Bickhard, M. H. (2004). *The Dynamic Emergence of Representation.* In P. Slezak H. Clapin, P. Staines, ed., Representation in Mind: New Approaches to Mental Representation, Elsevier, 71-90.

Burge, T. (1986). *Individualism and Psychology.* Philosophical Review, 95:3–45.

Butler, K. (1996). *Content, Computation, and Individualism in Vision Theory* Analysis, 56(3), 146-154.

Chalmers, D. (1996). *Does a Rock Implement Every Finite-State Automaton?* Synthese, 108, 309-33.

- *A Computational Foundation for the Study of Cognition.* url: http://consc.net/papers/computation.html (retrieved winter 2007).

- (1994). *On Implementing a Computation*, Minds and Machines, 4, 391-402.

Copeland, J. (1996). *What is Computation?* Synthese, 108, 335-359.

Craver, C. F. (2001) *Role Functions, Mechanisms, and Hierarchy*, Philosophy of Science, 68(1), 53-74.

Cummins, R. (1989). *Meaning and Mental Representation.* MIT Press, Cambridge, MA.

Demopulos, W. and Friedman, M. (1985). *Bertrand Russell's The Analysis of Matter: Its Historical Context and Contemporary Interest*, Philosophy of Science, 52(4), 621-639.

Egan, F. (1995) *Computation and Content.* The Philosophical Review, 104(2): 181-203

Fodor, J. (1980). *Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology.* The Behavioral and Brain Sciences III(1): 63-109.

French,S. and Saatsi, J (2006). *Realism about Structure: The Semantic View and Non-linguistic Representations* Philosophy of Science, 73, 548–559

Giunti, M. (1995). *Dynamic Models of Cognition*, in T. van Gelder & R. Port (eds), Mind as Motion, MIT Press, 195-225.

Gould, S. J. (1981). *The Mismeasure of Man.* New York: W.W. Norton and Company.

Haugeland, J. (1981). *Mind Design II.* MIT Press/Bradford Books, Cambridge, MA.

Johnson-Laird, P. N. (1988). *The Computer and the Mind*. Harvard University Press, Cambridge Mass.

Lycan, W. G. (1981). *Form, Function, and Feel.* Journal of Philosophy, 78, 24-50.

Marr, D., (1982). *Vision*. San Francisco: W.H. Freeman.

McCulloch, W. S. and Pitts, W. H., (1943). *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics, 5, 115–33.

Millikan, R. G. (1989). *Biosemantics*, The Journal of Philosophy, 86(6), 281-297.

Moore, C. (1980). *Unpredictability and undecidability in dynamical systems.* Phys. Rev. Lett. 64(20), 2354 - 2357

Newman, M. (1928). Mr. *Russells Causal Theory of Perception*. Mind, 37:137–148.

Piccinini, G. (2007a). *Computing Mechanisms.* Philosophy of Science, 74, 501–526.

- (2007b). *Computation without Representation*. Forthcoming in *Philosophical Studies.*

Psillos, S. and Hendry, R. (2007) How To Do Things With Theories: An Interactive View of Language and Models in Science, in J. Brzeziński et al. (eds.), *The Courage of Doing Philosophy: Essays Dedicated to Leszek Nowa*k: 59-115, Amsterdam/New York, NY: Rodopi, forthcoming.

Putnam, H. (1988). *Representation and Reality.* MIT Press, Cambridge, MA.

- (1975). *Mind, Language, and Reality: Philosophical Papers,* volume II, New York: Cambridge.

Pylyshyn, Z. (1984). *Computation and Cognition*, second ed., MIT/Bradford, Cambridge MA.

Russell, B. (1927). The Analysis of Matter. Kegan Paul, Trench, Trubner, London.

Scheutz, M. (1999). *When Physical Systems Realize Functions.* Minds and Machines, 9(2), 161-196.

Searle, J. R. (1992). *The Rediscovery of Mind.* MIT Press, Cambridge, Massachusetts.

Shagrir, O. (2006). *Why We View the Brain as A Computer*, Synthese, 153: 393-416.

- (2001). *Content, Computation and Externalism*, Mind, 110: 369-400.

- (2005). *The rise and fall of computational functionalism.* In Yemima Ben-Menahem (ed.), Hilary Putnam (Contemporary Philosophy in Focus. Cambridge University Press.

Skinner, B. F. (1938). *The Behavior of Organisms.* Englewood Cliffs: Prentice Hall

Turing, A. (1950). *Computing Machinary and Intelligence.* Reprinted in Mind Design II, MIT Press, Cambridge, MA, second ed.

Weir, A. (2001). *More Troubles for Functionalism*. Proceedings of the Aristotelian Society 101 (3), 267–294

Wells, A. J. (1998). *Turing's Analysis of Computation and Theories of Cognitive Architecture*, Cognitive Science, 22 (3), 269–294.

Wilson, R., A. (1994). *Wide Computationalism***.** Mind, New Series, 103(411), 351-372.

Worrall, J. (1989). *Structural Realism: The Best of Both Worlds?*, in Papineau, D. (ed.) The Philosophy of Science, Oxford: Oxford University Press.