COMPUTATIONAL EXTERNALISM: THE SEMANTIC PICTURE OF IMPLEMENTATION

By

Emiliano Boccardi

SUBMITTED IN FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
LONDON SHOOL OF ECONOMICS
LONDON, UK
JANUARY 2008

THESES
F
8787

1131895

LONDON SHOOL OF ECONOMICS
DEPARTMENT OF
PHILOSOPHY, LOGIC, AND SCIENTIFIC METHOD

Dated: January 2008

LONDON SHOOL OF ECONOMICS

Date: **January 2008**

Author: **Emiliano Boccardi**

Title: **Computational Externalism: The Semantic Picture of Implementation**

Department: **Philosophy, Logic, and Scientific Method**

Degree: **Ph.D.** Convocation: **January** Year: **2008**

Signature of Author

Table of Contents

Abstract

The property of being the realization of a computational structure has been argued to be observer-relative. After contrasting the problematic individuation of states in computational systems with the unproblematic individuation of states in dynamical systems, a general diagnosis of the problem is put forward. It is argued that the unwanted proliferation of models for the relation of implementation cannot be blocked unless the labelling scheme is restricted to semantically evaluated items. The instantiation of mathematical dynamical systems, by contrast, is showed to be immune to analogous skeptical arguments due to the virtuous role of measurements in grounding the relevant abstractions. Naturalized semantic properties are proposed to serve as a surrogate for measurements in grounding the relevant abstractions from the physical to the computational level of description, thus making implementations objective. It is argued that a view of implementation that abandons the pervasive internalist view in favor of a view of implementation according to which inputs and outputs are individuated by their broad semantic properties allows us to accept the validity of observer-relativity arguments while preserving the satisfaction of the desiderata of a theory of implementation, as well as the explanatory power of computationalism as a theory of the mind. The general idea is that of incorporating teleological theories of intentionality within the foundational heart of the notion of computation. An important corollary is that computational properties must be understood as broadly instantiated by relational properties of the implementing system and of its environment. The proposed understanding of implementation is then tested against a number of recalcitrant problems of computationalism. It is argued to be immune to standard objections.

Acknowledgements

I am especially grateful to Dr. Carl Hoefer, for his guidance since the early years of chaos and for having believed in me when I myself didn't. I would like to thank Dr. Emiliano Trizio, Dr. Federico Perelda, and Dr. Andrea Bianchi (for their insightful comments on my work).

Finally, I wish to thank the following:

Francesco Altea, Betsabe Garcia Alvarez, Michele Bajona, Valentina Bonifacio, Guia Camerino, Giovanna Castellani, Gilberto Ciarmiello (whom I miss a lot), Andrea and Isabel Crovato, Piero Dalbon, Franco Franceschin, Lucio Fulci, Dr. Fulghieri, Harvey the Pooka (he knows why), Martin Sarava Holzknecht, Melissa Kozak, Roberto N. B. Loss, Fantina Madricardo, Inti Marconato, Giovanna Massaria, Michela Massimi (for our Sundays), Margherita Morgantin, Nane Moro (for his insights), Valeria Musi (for her love and patience), Matthew S. C. Newman + Stephy and Pepi, Giuseppe Pareschi, Eva Pianalto (for her butterfly effect), Giovanna Pittarello (for her love), Silvia Puppini, La Rivetta, Prof. Giovanni Sambin (for his ideas on semantics), Spartacus, Paolo and Laura + Agata Russo, Massimiliano Vianello, and the Borderlines...

I would also like to mention the GALLETLY S.A.- AUDIO PROJECTS (for their financial support in the final stages of my work).

London, UK
March, 2006

Emiliano Boccardi

Introduction

Before I set out to work on this thesis, I was surveying the literature on the virtues and shortcomings of computationalist models of the mind. The word *computationalism* stands for a broad range of theses about the nature of the mind that have for many years been our received view on the nature of thought. Under its umbrella one can find a spectrum of hypotheses characterized by various interpretations of the claim that to possess a mind is to implement a particular kind of computation. The thesis that mental states *are* computational states is crucial to the definition of the underlying understanding of the mind. The technical appeal of the thesis is that the causal structure of the implementing system can "mirror" the formal structure of a computation thus realizing it. The philosophical allure of the thesis lies in the meaning of the word "formal".

In the 19th century, the discovery of consistent geometries that were not compatible with Euclid's parallel postulate prompted a severe crisis in our understanding of mathematics. Euclid's postulates, in fact, are deeply rooted in our intuitions about the nature of space. The faith in the power of reason (rational thinking) had for long suggested that these intuitively true postulates could not be denied. The peculiar status of the fifth postulate, i.e. the fact that, in spite of its non elementary validity, it could not be deduced from the other four axioms, had troubled mathematicians for centuries. The idea that rational thinking should be rigidly conforming to rules for manipulating explicitly expressed axioms, and that all truth should consist of (or at least be deducible from) formal derivations from these axioms, was already lurking in the background. When consistent non-Euclidian geometries were discovered, this underlying construal of rational thinking finally broke out and declared any non-formal procedures for assessing the truth of a mathematical statement defunct. At the end of the 19th century, prominent mathematicians like Gauss[1], Peano, Frege and Hilbert all worked towards a construal of mathematical truths in terms of a "symbol game" that consisted in deriving all truths from non semantic manipulations of strings of symbols.[2]

This formalist understanding of geometry and logic was to become the received view of rational thinking. The fast spreading wave of formalizations led Whitehead and Russell to apply the strategy to arithmetics and other branches of mathematics. The spirit of the project was also at the base of logical behaviorism in psychology. The upshot of this response to the crisis in mathematical thinking was a glorification of syntax at the expense of semantics. Later developments notoriously lessened the impetus of this programme, but semantics was never completely resurrected from its disrepute.

Computation, in the sense of evaluation of the result of an operation in a finite number of "mechanical steps", offers a perfect opportunity to construct a theory of the mind in compliance with formalist prescriptions.

[1]Gauss' ground breaking work, as a matter of fact, dates back to the first half of the nineteenth century.

[2]As a matter of fact neither Frege nor Russell thought that mathematics was only a "symbol game". But what counts for our discussion is that they thought that mathematical truths could be derived as the result of a symbol game.

The same number, number two, for instance, can have different names: 10 (in a binary representation) and 2 (in a decimal representation), etc. The crucial fact to be noticed is that an operation between numbers, say addition, can be performed by manipulating in a consistent way the names of numbers. After all, when we are first taught to do arithmetical operations, admittedly, we concentrate on conforming to the rules, rather than thinking about numbers. But these syntactic, mechanical rules are specific to the particular means of representation that we chose: the formal recipe for adding two numbers is different if they are represented (named) in a binary or in a decimal format. What counts, however, is that these "name-specific" recipes all allow us to "implement" the same operation.

Notice that semantics, in this picture, is reserved an ancillary role. The operation itself, in fact, as it is non name-specific, must be understood (type-identified) as essentially non name-dependent. Now, normally names are assumed to do a semantic job (i.e. to mean something), but the only property that the symbols that feature in the formal description of the task (addition, in this case) need be assumed to possess, is interpretability.

This picture of computation allows for a neat division of conceptual labor between the implementing machine and the abstract computation: the physical machine only serves the purpose of manipulating the (to it) meaningless names of numbers in compliance with the rules. Such division of labor is well expressed by the claim that the brain is a "*syntactic engine driving a semantic engine*"[3].

Adding numbers isn't but an example of what intelligent behavior is capable of. The basic idea of computationalism is that the brain, or whatever the implementing

[3]Block [11].

medium is, does the same mechanical job that the name-specific recipes do with respect to addition, realizing it through isomorphic causal mechanisms. Name-specific recipes realize addition by operating mechanically on symbols that are systematically interpretable as meaning numbers. The causal structure of a physical system mechanically transforms the symbols in isomorphic compliance with that name-specific recipe, thus realizing physically that operation. That's what the brain could be doing. What if all intelligent behavior, not only calculations, could be understood this way? "Calculemus!", would happily confirm Leibniz, or Hobbes, if they could witness these developments.

> *When a man reasoneth, he does nothing else but conceive a sum total, from addition of parcels; or conceive a remainder, from subtraction of one sum from another; which, if it be done by words, is conceiving of the consequence of the names of all the parts, to the name of the whole; or from the names of the whole and one part, to the name of the other part [...] These operations are not incident to numbers only, but to all manner of things that can be added together, and taken from one out of another. For as arithmeticians teach to add and subtract in numbers, so the geometricians teach the same in lines, figures, solid and superficial [...]; the logicians teach the same in consequences of words; adding together two names to make an affirmation, and two affirmations to make a syllogism; and many syllogisms to make a demonstration [...].*[4]

One of the earliest successful attempts to artificially simulate human cognition

[4]T. Hobbes. *Leviathan: Or the Matter, forme and Power of a Commonwealth Ecclesiastical or Civil.* 1651 ([49], p. 41).

was the "*Logic Theorist*". In 1956 Allen Newell, Cliff Shaw and Herbert Simon[5] showed how their creation (the Logic Theorist), successfully proved 38 of the first 52 theorems proved by Russell and Whitehead in their Principia Mathematica. In their work the authors deployed the following heuristic recipe. Use a language-like symbolic code to represent the world (the objects of the world and the workings that these objects exhibit). This constitutes the "knowledge base" of the machine. Use input devices to appropriately transduce the flux of environmental stimuli incoming from the environment into appropriate symbolic representations of them (these representations should deploy the same code as the one used to form the knowledge base). The machine is then to use both the knowledge base and the transduced input information to produce further symbol structures (according to algorithmic procedures). Some of these "newly formed" symbol structures should then be designated to serve as output. Finally, further transduction should "translate" these output symbol structures into the appropriate behavior.

The Symbol System Hypothesis (SSH), proposed by Newell and Simon in 1976[6], and since then held to be the hard core of the paradigmatic received view of cognition, can be thought as an answer to three fundamental questions about thought. 1) Can a machine think? 2) What is necessary for a machine to think? And, 3) what is sufficient for a machine to think? The answer to the first question, if the SSH is true, is yes: machines can think. The answer to the second question is that symbol manipulation is necessary for thought to take place. Finally, the sufficiency requirement is that a machine is built along the lines suggested by the above recipe.

Enthusiasm about the early successes of the programme set aside all concerns

[5]Newell, Shaw and Simon [1].
[6]Newell and Simon [63].

about the truth of the hypothesis for a long time, let alone concerns over the meaningfulness of the hypothesis. "*Intuition, insight and learning*", said Newell and Simon, "*are no longer exclusive possessions of humans [...]. There are now in the world machines that can think, that learn, and that create*"[7].

Such enthusiasm proved to be far too optimistic. Even the most sophisticated computer today (more then fifty years after the programme was started) is but a toy compared to a machine that thinks, learns and creates.

As I said, before I defined the project that resulted in the present work, I was surveying the critical literature on the shortcomings of computationalism. By the time I began to learn about these issues, a few years ago, in fact, computationalism had been challenged on several grounds, for many years. I was particularly interested by a gradual erosion of orthodoxy in the computationalist camp. Responding to an increasing amount of pressure from rival understandings of cognition, and from some internal objections, in the 80's, 90's, and up to more recent years, the received view of computationalism seemed to me to make an increasing number of concessions to non-computationalist oriented philosophers.

Many of these objections appear, at first sight, to point at a difficulty of computationalism to account for what we may generically call the "semantic capacities of minds". Searle's famous Chinese-room argument, Harnad's symbol grounding problem, or even the more technical problem of transduction, as we shall see, are all examples of this area of concern. At the time, I thought I would try to contribute to this debate. I say these objections *appear* to point at a difficulty of computationalism in treating semantic capacities, because now I think the problem lies somewhere

[7]Newell and Simon [63], p. 6.

deeper: in the very notion of computation.

At first sight, the formalist spirit of the treatment of computation appears promising. It appears, for example, to be able to explain how our brain, or any other implementing medium, performs computations. But we (putting it very naively) are also able to mean numbers, by the names we give them. How do we do that? Any theory of the mind must account for that. I thought, as many have before me, that this was the difficulty that was leading computationalist theories to amend the sufficiency hypothesis in various ways to meet specific objections.

It was then that I discovered a very unsettling feature of computational theories of the mind. I assumed that the original spirit of the computationalist paradigm was formalist, in the sense in which Hilbert's project was. Fodor, for example, confirms this expectation:

> If mental processes are formal, then they have access only to the formal properties of such representations of the environment as the senses provide. Hence, they have no access to the semantic properties of such representations, including the property of being true, of having referents, or, indeed, the property of being representations of the environment.[8]

I was also reassured that things had not changed since, by my frequent encounters with recent writings that explicitly ruled out any doubt:

> It will be noted that nothing in my account of computation and implementation invokes any semantic considerations, such as the representational content of internal states. This is precisely as it should be: computations

[8]Smith [84], p. 231.

> are specified syntactically, not semantically. [...] If we build semantic considerations into the conditions for implementation, any role that computation can play in providing a foundation for AI and cognitive science will be endangered [...].[9]

But one also hears just as frequently that computations must be defined "over" representations, that there is "no computation without representation":

> It is widely recognized that computation is in one way or another a symbolic or representational or information-based or semantical i.e., as philosophers would say, intentional phenomenon.[10]

How is one supposed to feel about that? When I looked back at the historical foundations of the computationalist paradigm, hoping to reconstruct a coherent version of the facts, I was left even more wondering. Virtually all the fathers of computationalist Artificial Intelligence, in fact, made what, to me, sounded like ambiguous claims about the conceptual labor of semantic properties in computation. Fodor himself, for example, wrote that:

> 1. The only psychological models of cognitive processes that seem even remotely plausible represent such processes as computational.
> 2. Computation presupposes a medium of computation: a representational system.[11]

Pylyshyn, another of the fathers of the paradigm:

[9]Chalmers. *A Computational Foundation for the Study of Cognition.* Section 2.2. Unpublished, but section 2 was published as [15].
[10]Smith [84], p. 9.
[11]Fodor [33], p. 27.

> The answer to the question what computation is being performed requires discussion of semantically interpreted computational states[12]

Should we conclude that the "neat division of labor" between syntactic- and semantic-driven processes, or the "syntactic engine driving a semantic engine" picture of the brain, is propaganda? What exactly have semantic properties been doing all these years? Has their job been tacitly exploited without being ever truly recognized? Who is driving: the syntactic engine, or the semantic one?

Well, this is how and where my work began.

To clear the ground for my investigation, I began by looking at where things appear to go smoothly: interpretability. Interpretability is a faint, hardly reminiscent, far relative of interpretation. Arguably, it does not belong to the number of "intentional properties". At first sight, it is all that computations need for being realized.

> A symbol system is a set of arbitrary "physical tokens" [...] that are manipulated on the basis of "explicit rules" that are likewise physical tokens and strings of tokens. The rule-governed symbol-token manipulation is based purely on the shape of the symbol tokens (not their "meaning")[...]. There are primitive atomic symbol tokens and composite symbol-token strings. The entire system and all its parts - the atomic tokens, the composite tokens, the syntactic manipulations both actual and possible and the rules - are all "semantically interpretable": the syntax can be systematically assigned a meaning e.g., as standing for objects, as describing states of affairs.[13]

[12]Pylyshyn [74], P. 58.

[13]This is how Stephen Harnad ([45], p. 337) reconstructs the definition from Newell [63].

This weak requirement, interpretability, is the same that we use in model theory, to express the relationship that holds between a formal language and a structure that satisfies it.

A *signature* is a set of individual constants, predicate symbols and function symbols. Each signature K gives rise to a first-order language, by building up formulas from the symbols in the signature together with logical symbols (including $=$) and punctuation. These constants, predicate and function symbols, are to be conceived as meaningless, non-interpreted items. The rules for building up the formulas of a first-order language out of a signature, consequently, only operate on the base of non-semantic properties of these symbols, like the physical transformation of a physical symbol system operate on the base of the shape of the physical tokens described above. If K is a signature, then a *structure* of signature K, say A, consists of the following items:

1. A set called the domain of A and written $dom(A)$; it is usually assumed to be nonempty;
2. for each individual constant c in K, an element c^A of $dom(A)$;
3. for each predicate symbol P of arity n, an n-ary relation P^A on $dom(A)$;
4. for each function symbol F of arity n, an n-ary function F^A from $dom(A)$ to $dom(A)$.

Notice that the existence of the elements of a structure is totally independent from the items of the signature and viceversa: whether the relevant correspondence exists or not solely supervenes on intrinsic (hence independent) properties of the signature and of the structure. Although sometimes model theorists are tempted to call the items of a signature the "names" of the items of the structure, it is misleading to think of them this way. A constant c in the signature, of course, *could be* used as a name for its correspondent element c^A in the structure, but this doesn't entail that it *is* a name. To assume that c *is* a name for c^A, would unnecessarily impose constraints on c that would "spoil" the formal essence of model theory.

A consequence of this is that the formal language does not have the resources, by itself, for individuating its "intended model". All structures that are isomorphic to the intended one, are indistinguishable from the point of view of a syntactic machine (whether virtual, like a formal language, or physical, like a physical symbol system). Syntactic engines, we may say, are "blind" to semantic properties (which are required to individuate the intended model), like a color-blind person is blind to colors. This indistinguishability of isomorphic structures on the part of a syntactic engine, a feature that I have called the *isomorphism catastrophe*, has, we shall see, far reaching effects on our analysis of computation.

Now, if our physical symbol system (our physical signature) has a structure that corresponds to a certain algorithm (for example a recipe for adding numbers), then we can say that the system "implements" the algorithm. As, moreover, the formal algorithmic structure can be systematically interpreted as referring to numbers, we can say that the physical system computes additions of numbers. Our brain, for example, could be such a physical signature. Notice that semantic properties are totally out of this picture.

But if things were so smooth, there would be no need to say that there is no computation without representation. Perhaps there would be no minds as we know them, without representation, but computation should not be answerable to cognitive science: it cannot be held responsible for the failures of computationalist theories of the mind, more than calculus is responsible for the failure of classical physics. In fact, things are not so smooth.

When I looked at the literature about the foundations of computation, a plethora of arguments pointing at an alleged observer-relativity of the notion of computation convinced me that there might be something wrong with the formal picture described above. No one challenges the notion of computation from a mathematical point of view: the Church-Turing thesis appears to satisfy nearly everyone. But it is the "garden variety" of computation, i.e. the notion of real, implemented computations, that appears to make many philosophers unhappy.

The first chapter of this work presents some of these observer-relativity arguments. I call them *vacuousness arguments*, as they entail that computational constraints are vacuous constraints, at the physical level. Most of them have been counter-argued, and there is nothing like an agreement about the issue. My first response to these arguments, however, has been to ask myself what job the observer was exactly allegedly suggested to be doing. It occurred to me that if I could individuate a precise task for this observer, and if I could subsequently naturalize this task (i.e. explain how a physical system could complete it), then computation could peacefully continue doing its excellent job at explaining the mind.

One of these arguments (Putnam's vacuousness argument, discussed in section 1.4), if sound, would inflict a fatal strike at the neat model-theoretic picture of implementation presented above. Its upshot is that every physical system implements every computation. In our words, it means that every physical system can be thought of as a signature for any possible algorithmic structure. Intuitively, the strategy behind this argument exploits the fact that real physical systems undergo change in a continuous (non-denumerable) way. Uncountably many states are indeed a lot of states, so many that they can be grouped in uncountably many ways to form uncountably many signatures (systems of symbols). Some of these will match, the argument goes, the weak requirement of systematic "interpretability" that we hoped would suffice for computation.

It was while thinking about this that I began to suspect that what the observer might be needed for, could be to do a semantic job: the job left unfinished by mere interpretability. Only some of these uncountably many groupings of states, in fact, would possess real semantic properties. Adding the requirement that real semantics must be in place would block the arguments. In the back of my head I also vaguely suspected that resurrecting semantics from its disrepute could help computationalism to solve its "semantic troubles". But this was a temptation to be refrained, for the moment, as the issue at stake was computation in its own right.

But how could one make sense of these intuitions? One way to expose the problem is by contrasting the allegedly troublesome case of computation with the relatively well-behaved case of dynamical systems in physics. Why, in other words, isn't physics also in trouble? After all, the symbols that feature in a dynamical system (a set of differential equations, for example), are not meaningful in themselves. But model theory seems to apply smoothly to the relationship that there is between a real dynamical system and the mathematical dynamical system that describes its change in time: no one argues that any real physical system instantiates any system of differential equations. What's so good about dynamical systems?

In section 2.2 I propose a contrastive analysis between the case of computations and that of dynamical systems. The result of my analysis is, unsurprisingly, that measurements do the grounding job in the case of physics. Any interpretation of the symbols in a mathematical dynamical system, in fact, must specify what measurement procedures should be applied (without such specification, it does not make any sense to ask whether a physical system is instantiating a certain set of differential equations). This feature of dynamical models, indirectly, does a quasi-semantic job, in that it constrains the symbols (of the structure: the variables, for example) to "refer" to certain magnitudes (in the physical signature).

I then adopted, as a working hypothesis, the idea that the notion of implementation should be grounded on real semantic properties just like the notion of instantiation is grounded on specific measurement procedures, for individuating its models. In particular, I hypothesized (section 2.5.1) that the candidate implementing input and output items be restricted to intentional items (i.e. to items that objectively, and not merely potentially, possess semantic properties). This would make sense of many ambiguous comments about the relationship between computation and representation: there would finally be a clear sense in which there is "no computation without representation"!

As the reader can imagine from the comments quoted above, however, the idea to build semantic properties into the notion of implementation sounds very unpalatable to many. Section 2.5.2 is therefore devoted to freeing my proposal from many potential a-priori objections. Apart from clearing the ground for my proposal, these considerations also set constraints on what restrictions are compatible with a theory of implementation. Most importantly, I argue that the properties suggested as criteria for the restriction of the input architecture of computation (those on which semantic properties supervene), must be extrinsic, relational properties of the candidate label bearers, if the correspondent theory of implementation is to comply with the requirement of multiple realizability.

At this point I believed to have done most of the relevant work. It only remained to apply my strategy to as many theories of content as possible, to test whether at least one of them complied with the desiderata set forth in section 2.5.2. I illustrate some of these attempts in section 3.2.

In doing so, however, I realized that my work had not even began. What I found out, in fact, is that the application of various theories of semantics to my strategy, far from safely grounding computation in the realm of physics, left its pattern of failure unchanged. Theories of content, it appears, present us with a suspiciously symmetrical pattern of failure: either they surreptitiously deploy semantic capacities (thus circularly presupposing what they ought to explain), or they place vacuous constraints, thus falling victim to the same isomorphism catastrophe that haunted non-semantic theories of implementation. Is there some deep reason for these symmetrical patterns of failure? Intuitively, observing that theories of implementation and theories of semantics suffered from an analogous syndrome suggested to me that I might be on the right track.

So it was that I set out to directly address the isomorphism catastrophe as an independent problem (independent from its computational, theory-specific manifestation). Where does the catastrophe come from? Fortunately, I was not the first one to wonder about it. The philosophical literature treats a number of relevant cases. In section 3.3 I consider two vacuousness arguments that approach the issue in a perspicuous way.

One is an objection to Bertrand Russell's causal theory of perception (due to Max Newman, a colleague and friend of Turing's). I treat this argument in section 3.3.1. In the late 1920's, Russell was investigating what knowledge we could have of the external world, i.e. of the unperceived causes of our perceptions, given that we could not assume it to be "direct" knowledge. Russell thought that we could argue to have a "structural" knowledge of these unperceived events: the kind of knowledge a blind person has of a photo, when someone describes it to him. Newman's objection consists in arguing that such alleged structural knowledge is no knowledge at all, as relational structures are compatible with any suitably numerous field. Russell conceded that Newman was right.

The other, more recent vacuousness argument I consider (section 3.3.2), is Putnam's so called model-theoretic argument against metaphysical realism. This consists of an extension of Skölem's paradox: the apparently paradoxical fact that there is a countable model of real number theory (it is a consequence of Skölem's theorem). Putnam uses it to argue that metaphysical realism is not tenable. Put plainly, it allegedly shows that our theories, no matter how complete they may be, do not have the resources to individuate their intended models, if these are conceived as "real", external and independent from them.

These arguments suggested to me that indeed the problem with our current theories of implementation and of semantics may lay somewhere deeper (i.e. non theory-specific) then what I had previously suspected.

It occurred to me that all the arguments that I had at hand (vacuousness arguments against state-to-state theories of implementation, against internalist theories of semantics, against Russell's theory of perception and against metaphysical realism), as well as the respective theories that they aim to criticize, all make an implicit assumption: that syntactic properties supervene on intrinsic properties of their bearers. I call this pervasive construal of syntactic properties: *syntactic internalism.* This entails that, granted (ex hypothesis) the validity of vacuousness arguments, we are faced with two options: either we accept their skeptical conclusions, or we drop the assumption that syntactic properties are intrinsic to their bearers.

In section 3.3.3 I argue that dropping this implicit assumption allows us to block the skeptical arguments. I then proceeded to investigate the tenability and the consequences of this option.

Before I continue telling the story of this work, I would like to make a brief digression that I hope will help the reader to "digest" more intuitively the route that I decided to take.

Imagine a flag painted so as to reflect light of continuously increasing wavelengths, from left to right. The range of wavelengths is such as to encompass three adjacent colors of the spectrum: say yellow, green and blue. Looking at it, a normal human being would see a flag made of three differently colored sectors. Now, being a three-sector flag is a property our flag has in common with many other flags. Another flag, painted red, white and green, for example, is also a three-sector flag. What do these two flags have in common? The colors? No. They have in common the property of being sectioned into three differently colored parts. This is a syntactic, multiply realizable property of some colored flags.

In a first approximation, we may suppose (like many people do) that the colors of the spectrum must be correlated with intrinsic properties of objects. In this case we would say that all three-sector flags have in common the fact that they are painted with three substances that have three different properties. This is a view about three-sectorness that corresponds to what I have called syntactic internalism.

A syntactic internalist about colored flags, if asked to explain why someone (a color-blind person, for example) does not categorize flags in the same way as he does, would answer that the color-blind person cannot appreciate the fact that there are three different properties in the flag, because of his malfunctioning eyes: the problem is in the color-blind person's eyes, not in the flag.

But if he was asked to check that indeed his hypothesis is correct, he would be surprised to find out that no rigid partition of wavelengths into bands does justice to his perception of colors. He would be looking in vain for three properties in the flag that reflect light of continuously increasing wavelengths. The number three, it appears, has been lost somewhere between his eyes and the flag. It cannot be in his eyes alone, for he shows to be able to systematically pick up the same objects, when asked to select three-sector flags. But it can neither be in the flags themselves, for when he analyzes the intrinsic properties of them, he does not always discover three things.

What happened to the number three? One can take an anti-realist stance, and argue that, as neither the eyes nor the flags can be showed to be three-sectored, three-sectorness is not a real property after all. This is the stance that skeptical arguments want us to take.

But, I argue, there is another option. We can hold onto our realist stance, at the price of abandoning the internalist description of three-sectorness. Colors, according to this view, are secondary, relational properties of objects, environments and cognitive systems (including our eyes). If we were able to describe precisely how these interact with each other, somewhere in the physical description of the interaction we would find our lost number three.

Three-sectorness is a syntactic property, in that it does not depend on any *particular* color for its instantiations (implementations, in the analogous case of computations). But this does not entail that three-sectorness inheres in the flag, in a color-independent way. In sum, three-sectorness is a property that depends on colors (albeit not on some particular set of three colors), and colors are relational properties of their bearers and of our brains. It follows that three-sectorness also is a secondary, relational property of flags and of our brains.

This realist view of three-sector flags is an example of what I call: syntactic externalism. We could say, reversing in a parody Block's slogan, that this is a "color engine" driving a syntactic engine.

Before I began this digression, I was saying that we need not accept the anti-realist conclusions of skeptical arguments. We can maintain that they are valid, just like it is true to say that three-sectorness is vacuously compatible with various intrinsic property of the flag. But if we give up the internalist assumption, computations need not be construed as enjoying a second class ontological status.

Computation depends on semantic properties (albeit on no *particular* semantic properties), and semantic properties are secondary, relational properties of their bearers, like colors. It follows that computational properties are also secondary, relational properties of their implementing systems. I call this view of computation: *computational externalism.* Brains, according to the consequent picture of computationalism, are semantic engines driving syntactic engines. I argue for the semantic view of syntactic properties in 3.3.3, 3.3.4, 3.3.5 and 3.3.6.

I began my journey from the safe requirement that the symbols input and output to a computational machine be interpretable, and things appear to have gone very far: interpretability is not sufficient, if vacuousness arguments are assumed to be sound, to rule out unwanted models. Interpretability, like satisfaction in model theory, in fact, is autonomously sufficient only in so far as among the models there are no unwanted models. If, like vacuousness arguments suggest, among the models of our symbol systems, there are unwanted ones, only real semantics, interpretation, can rule them out: symbol systems, in fact, do not have the resources in themselves to discriminate wanted from unwanted models (as it happens in the case of Sk¨lem's paradox). So much for my strategy.

Before I could apply this strategy, however, one final piece was missing that would complete the puzzle: a theory of content that complied with the desiderata of syntactic externalism. Section 4.2 is devoted to arguing that teleological theories of content would do the job. As some would find an irreducible resort to historical, evolutionary properties in defining implementation unpalatable, I discuss both an etiological version (section 4.2.3) and a non etiological version (section 4.2.4) of teleological theory of content.

Finally, to exemplify how my strategy works in a real scenario, I applied it to the computation of a logical gate (section 4.4.1), showing how externalist computations are immune to vacuousness arguments. In section 4.4.3 I formalize my proposal, providing a precise definition of the implementation of a finite state automaton (one of the equivalent ways to describe a computational machine).

Before proceeding to test the applicability of the semantic picture of implementation proposed, however, two mutually related objections had to be addressed. The model theoretic approach adopted in this work appears to be at odds with the standard practise of application of computational concepts by the relevant community of experts. Is there another approach to implementation that is more "friendly" to standard practises and to the intuitions of computer scientists? And if yes, should we conclude that what v-arguments really show is that the methodological stance that stands behind them is inadequate? A second, related issue, is the applicability of my semantic picture of implementation to more complex computational structures, such as universal Turing machines. Both issues are treated in chapter 5. It is generally argued that the semantic picture is not as "unfriendly" to standard practices as it prima facie seems to. The adoption of a model theoretic approach is defended and an outline for the applicability of the semantic labelling scheme to universal Turing machines is provided.

With an objective notion of physical computation in place, I was now ready to investigate what consequences it would have in grounding a computational theory of the mind. I discuss applications of my theory of implementation to the symbol grounding problem (section 6.3), to combatting John Searle's famous criticism (section 6.4), to the issue of syntactic constitutivity in connectionist models (section 6.5), and to the debate over cognitive architecture (section 6.7). I argue that the definition of physical computation provided in section 4.4.3 allows us to formulate a computational theory of the mind that is free from standard, recalcitrant shortcomings (I call this theory: *Teleological Computationalism*).

Chapter 1

Computational realism and its discontents

1.1 Introduction

The objectivity of implementation, i.e. the possibility to specify the necessary and sufficient (physical) conditions a real dynamical system must satisfy for it to be an implementation of a given computational structure, has been questioned several times. Many of these concerns are now familiar to most philosophers of mind and of Artificial Intelligence. These doubts about the objectivity of implementation have been voiced for so long now, and from so many different standpoints, that, I believe, it is not senseless to explore the consequences of such arguments, if they are assumed to be sound. This is the purpose of this treatment.

The first chapter of my work, then, is dedicated to the introduction of some of these arguments. Far from doing justice to an increasingly large literature on the issue, the following pages introduce but a few examples of them.

Section 1.2 is dedicated to the contrast between the notion of satisfaction of a function, as it is understood by the physical sciences, and the stronger notion of computation of a function. The challenge, on the part of computational realists, is that of providing necessary and sufficient physical constraints that allow us to discriminate the case of computation from that of mere satisfaction. Skeptical arguments, we shall see, all point at the alleged vacuity of the proposed constraints. If the additional constraints suggested for the realization of computation pose vacuous constraints at the physical level they fail to provide a viable principle of discrimination.

Section 1.3 deals with the intuition that to implement a computational system, its syntactic structure must be mirrored by the causal structure of the implementing object. I concentrate on the simple case of Finite State Automata.

Section 1.4, finally, explores some popular arguments to the effect that the notion of causal "mirroring" of an abstract formal structure cannot be grounded in physicalistic terms alone. Again, we shall see, syntactic constraints are argued to be vacuous constraints, when analyzed at the physical level.

1.2 Satisfaction and computation of a function

1.2.1 Computation = satisfaction + ?

What (physical) properties of a real dynamical system (S) are necessary and sufficient for it to be implementing a computational structure (A)? Indicate the predicate "S *implements* A" by $Imp(S, A)$. Our question can be rephrased as follows: what are the truth conditions of $Imp(S, A)$?

We shall assume a formal theory (A_f) that specifies an architecture and an algorithm (virtually) implemented by it. Such algorithm takes the arguments of the function f as inputs and produces its values as outputs. The relation between A_f and S (the real dynamical system) will be described by a mapping from the terms of A_f to (real) parts of S. These parts of S will have to be specifiable and identifiable by their spatio-temporal properties alone. Following a now common terminology[1] we shall call such mapping from A_f to S a *labelling scheme.*

[Labelling scheme] *A labelling scheme (L) for a computational structure (A) consists of 1) the specification of which parts of a physical system (S) are to be the "label-bearers" (an interpretation function) and 2) an unproblematic way to individuate the label born by each part at any time.*

Within this notation, $Imp(S, A_f)$ is true iff there exists a labelling scheme L such that the pair $< L, S >$ is a model of A_f. In other words, our (preliminary) notion of implementation can be expressed by: *$Imp(S, A_f)$ is true iff all sentences in A_f are true of S under L.*

In what follows, we shall contrast $Imp(S, A)$ with the relation that holds between a mathematical dynamical system (M) and a Real Dynamical System (RDS henceforth) that instantiates it (S).[2] We shall indicate the predicate "*S instantiates M*" by $Inst(M, S)$.

Suppose we have a function (f) and a physical system (S). We ask ourselves whether S computes f, that is, we ask whether $Imp(S, A_f)$ is true. We have seen how this depends on whether there exists a labelling scheme (L) such that the pair $< L, S >$ is a model of A_f.

The most obvious requirement is that S instantiate the function f. In fact, if a

[1]It can be found, for example, in Copeland [19].

[2]In the literature the expression "dynamical system" is used to denote both a (at least potentially) real entity, one that undergoes change in space time, and the mathematical description of it. Borrowing the terminology from M. Giunti [41], I shall call the former a Real Dynamical System (RDS) and the latter a Mathematical Dynamical System (MDS).

physical system S does not satisfy a function f, for sure neither it computes it (i.e. for any computational structure A_f, and any RDS S: $Imp(S, A_f) \Rightarrow Inst(S, f)$). As all RDS's can be described as instantiating some function, the opposite, however does not hold (or the notion of computation would be indistinguishable from that of instantiation).

[Satisfaction] A physical system S satisfies a function f if and only if: 1) it causally associates the physical arguments of f (W_i) with its physical values (W_o)[3] and 2) it can be seen as instantiating another function g whose arguments and values are outside (spatially) the system S.

It is easy to see that satisfaction of a function is essential for a system to compute. The first requirement expresses the thought that real computing systems are a kind of physical system, that is a kind of system that can be described by a mathematical dynamical system. The second requirement expresses the thought that computing systems are also a kind of information-processing systems: that there is no computation without representation[4].

If these desiderata are necessary to capture our pre-analytic notion of computing system, they are certainly not sufficient: any physical system satisfies a function. In fact, any physical system can, in principle, be described by a mathematical dynamical system, and anything can be thought of as representing something else (the arbitrary

[3]The physical values and arguments are the values taken up by the relevant real magnitudes. These should not be confused with the real numbers that can be assigned to them by measurements and that are the values of the variables that feature in the mathematical structures instantiated.

[4]See section 2.6.1 for a discussion of the claim that "*there is no computation without representation*".

nature of symbols makes the fortune and the disgrace of computationalism). So, to capture the nature of a computing system we must further constrain either or both the requirements of satisfaction. We must either say that not all physical systems count as computing systems (by imposing constraints on the mathematical dynamical systems that describe them), or that not all interpretation functions are acceptable for a system to be computing. The chief computationalist strategy is to add the further requirements that: 3) The process mediating the arguments and the values of g be algorithmic and 4) The function g be recursive (Turing computable)

Our question ("Does S compute f?"), thus, boils down to asking whether conditions 1), 2), 3) and 4) provide our formal theory (A_f) with a labelling scheme such that all sentences in A_f are true of S under L. Let us indicate the "translation" of a sentence α in A_f by $[\alpha]_L$. We are then asking whether the (as yet abstract) conditions for S computing f allow us to define a labelling scheme L such that $\forall \alpha \in A_f : [\alpha]_L$ is true of S.

It is relatively easy to produce an interpretation for the sentences in A_f that correspond to the requirement that S satisfies f. It is in fact sufficient that for each abstract input term i and output $o = f(i)$ there exist magnitudes in S such that:

$[i]_L = W_i$ be reliably, causally, associated to $[o]_L = W_o$, where W_i and W_o are the values of the magnitudes.[5]

[5]As I shall discuss later, the notion of satisfaction is less straightforward than this. For the sake of clarity, however, I decided to postpone a more detailed discussion the notion of satisfaction to chapter 2.

On the contrary, it has been argued to be impossible to produce a (non observer-relative) labelling scheme[6] for the requirement that the processes mediating the arguments and values of $[f]_L$ be algorithmic.

All arguments to this effect (that the property of being "algorithmic" is not a property that appears to be ever "objectively" instantiated by a physical process) entail the claim that, at the physical level, the computation of a function is indistinguishable from its instantiation. It entails, for example that the solar system computes the solutions to its equations of motions[7]. I shall call these arguments: "indistinguishability arguments", or "i-arguments".

The rest of this section will be devoted to the treatment of some i-arguments. I shall mention only a few.

1.2.2 ? = Step Satisfaction

To capture the notion of algorithmic process some have argued that one must think of them as sequences of instantiated basic functions. These basic functions, although necessarily recursive (or the whole function wouldn't be computable) must be "directly" satisfied. They are computable (in the sense that they are recursive), but they are not computed. Some have argued that this is the same as requiring that the causal structure implementing the function be analyzable into a series of causal steps.

> Computing reduces to program execution, so our problem reduces to explaining what it is to execute a program. The obvious strategy to exploit

[6]One of the requirements of a labelling scheme, in fact, is that it be such that it allows computation to select (objectively) a class of Real Dynamical Systems.
[7]Fodor discussed this claim in [33], p. 74.

> is the idea that program execution involves steps, and to treat each elementary step as a function that the executing system simply satisfies... Program execution reduces to step satisfaction[8]

But this requirement is trivially satisfied by any physical system (which is the same as saying that it is no physical constraint). Given any physical system satisfying a certain dynamics, we can always analyze its behavior as a series of step satisfactions: any arbitrary choice of the time series would do just as well. Take for example a non elastic bouncing ball. Given a starting height h_n, after one bounce the ball reaches a height $h_{n+1} \cdot (1-c)$, so the system (B) satisfies the function defined recursively by $f_{n+1} = f_n \cdot (1-c)$. Does it also compute it? According to the proposed definition it does, for 1) B satisfies f, 2) B satisfies an algorithmic function (in the sense of step satisfaction) and 3) B satisfies a Turing computable function.

What makes B a computing system (as opposed to a system that merely satisfies a function), according to this understanding of computation, is the fact that the process can be analyzed into steps (the individual bounces). With a trivial change of the interpretation function, however, B can be seen as satisfying also the function $g(h) = (1-c) \cdot h$ that assigns to the input h the output $(1-c) \cdot h$. This process comprises only one bounce (one step). Is B computing the function g? If we follow the proposed criterion we should ask if the process can be analyzed into more than one step. If "step" means bounce, then the answer is no, and B is not a computing system. But why should step mean bounce? We can group the states of B when the ball is falling (call them F) and the states of B when the ball is rising (R). Now, the fall of the ball reliably causes the ball to rise back. The process has been analyzed into two

[8]Cummins [20], pp. 91-92.

steps (F and R) so now the system is again computing the function g. Surely the very act of naming the states of a system cannot ground an absolute notion of physical computation. The idea of step satisfaction as necessary for physical computation was an attempt to provide a physical counterpart to the notion of algorithm: such an attempt, however, fails, for the proposed property proves to be observer relative (dependent on arbitrary descriptive choices of the modeler).

1.2.3 ? = Digital

Acknowledging the difficulty in providing a suitable account of computation by making reference to the continuous/discrete distinction, some authors prefer to use the notion of digital (as opposed to analog) systems. According to these proposals, a RDS (real dynamical system) computes, as opposed to merely instantiate, a function if and only if it is a digital system. The standard definition of a digital (analog) system is one that "ranges over discrete (continuous) sets". However not all the arguments of the function satisfied are relevant for our definition of "digital": only those that play a direct causal role in the satisfaction of the function. So an analog watch whose arms move in discrete steps doesn't cease being analog, for the relevant arguments range over a continuous set. Similarly a digital computer doesn't become analog if we make the inputs to it continuous. Most authors agree that Goodman's notion of notational scheme captures the relevant notion of discreteness. According to Goodman's treatment[9], the requirements for being a symbol (digital) scheme are (a) syntactic disjointness (each token must be a token of at most one type) and (b) syntactic finite differentiation (tokens of different types must not be arbitrarily similar). The following is an example of digital scheme: two line segments X and Y are

[9]Goodman [42].

tokens of the same type iff:

$$n + 1/2 < L_X, L_Y < n + 1 \quad \text{for some} \quad n \in N$$

Notice that two distinct types are never infinitely close to each other and that no token can belong to more than one type. An example of an analog scheme is:

$$n < L_X, \quad L_Y < n + 1$$

In this case tokens of (adjacent) types can be arbitrarily close to each other. Haugeland[10] applied Goodman's theory of notational scheme to physical systems. It turned out that, under Goodman's understanding, digital computers could not be considered as digital. Pulse detectors in standard digital computers are in state + if subject to a pulse greater than 2.5 volts and in state - for pulses smaller than 2.5 volts. This contradicts the requirement of syntactic finite differentiation. In fact there is no theoretically possible way to categorize a pulse of exactly 2.5 volts. As a matter of fact pulses are always either close to 5 volts or to 0 volts, so that there is never real ambiguity in sorting them. However, by Goodman's definition, the scheme is analog.

Hagueland proposal is to adapt Goodman's theory to the realm of physical systems by requiring that syntactic disjointness and syntactic finite differentiation be "relative to our current technology and scientific practices", rather than to sheer theoretical possibility. What is relevant for our discussion is that the only account that seems to do justice to the digital/analog distinction turns out to give up the hope to ground the notion of digital physical systems on purely intrinsic properties: whether a physical system is or isn't digital does not depend solely on what system it is (or on what

[10]Haugeland [47].

mathematical dynamical system describes its behavior) but also on other (external) factors.

An alternative account of digital systems which (if correct) would overcome the above difficulty is that proposed by Block and Fodor[11]. The intuition behind their proposal is that analog processes are "low-level processes" that can be (directly) subsumed under some physical law. In other words a process is analog if its types represent numbers that quantify some primitive (or quasi-primitive) magnitude. More precisely Block and Fodor suggested that a system is analog if its input-output behavior instantiates a physical law. The notion of a system directly instantiating a physical law is, however, very problematic.

The instantiation of basic physical laws is always subject to ceteris-paribus clauses. We are therefore faced with a dilemma. In one sense any system, at all times, is instantiating some basic physical laws (or else it would be violating them). However, whether it does so directly or indirectly depends on our means of observation, not on the nature of the system. In this sense, then, all systems would be analog.

On the other hand, outside a carefully controlled experimental set-up, no physical law is ever immediately instantiated. In this sense, no physical system would be analog. Finally, if we resort to counterfactual statements regarding ideal experimental set-up's to mitigate the absurd conclusion that no system can be analog then, once again, we give up the hope to ground the notion of analog system on properties of the system alone. Hagueland's proposal is consistent with the above considerations: *any suitable notion of analog (digital) system must make reference to observer-relative factors.*

[11]Block and Fodor [12].

Concerns about the alleged observer-relativity of the digital/analog distinction are widespread in the literature. As early as 1951 Pitts, for example, put it this way:

> Actually, the notion of digital or analogical has to do with any variable in any physical system in relation to the rest of them, that is, whether or not it may be regarded for practical purposes as a discrete variable.[12]

1.3 Implementation of a computational structure

We might hope to resort to the notion of implementation (realization) of a computational system to render the notion of computation objective. A physical system would be really computing if it realizes a computational structure (a Turing machine, or an automaton, for example). The notion of realization of a computational system, however, has also been threatened by observer-relativity arguments. If one is particularly worried about these claims, then, he or she will not find this solution viable. Let us take a closer look at this potential solution to our problem.

1.3.1 States, state transitions and automata

I shall analyze the simple case of finite state automata. First, what is a state? Given a certain physical system whose behavior is known to be deterministic, is it reasonable to expect that it will always respond to a certain type of input with the same type of output? Clearly not. If we print some command on our computer, for example, we expect the output to depend on its internal state (e.g. things change if the computer is turned on or off). For such a deterministic system a state is defined as:

[12]Pitts [69], p.34.

[State] A representation of the past activity of a dynamical system that is sufficiently detailed to determine, together with the current input, what the next output and state will be.

There is an ambiguity in this definition that ought to be dissolved: given a certain state, what is exactly "the next state"? Intuitively it is the state of the system in the next time step, but this notion ought to be clarified. Suppose that the evolution of an aspect of a real system is described by a set $\{g^t\}$ of functions that are solutions to certain differential equations. We can always consider the values of g^t at discrete intervals of T (natural numbers are real numbers after all). In this case a discrete series of time steps is embedded in the continuous series of real time steps. The expression "next state and output of the system" in this case means state and output of the system at time $t + 1$. Our ordered discrete set of times, in this case, inherits the metric from the real continuous time series, so that between time $t + 1$ and time t there is a definite amount of time, function of t and $t + 1$ only.

This, however, is not the only way in which a system can be a discrete time system. In some cases the internal dynamics is so fast at reaching equilibria that one can describe the evolution of the system by restricting the set M of states to states of equilibrium (possibly discrete). The transition function, this time, describes how the system evolves from one equilibrium to another, given certain conditions. In this case the real lengths of the intervals is irrelevant. There is a lower bound (dependent on how fast the dynamics is) but above this threshold the time series serves merely as an ordered set.

That the observer is lurking in the background, is something that can be appreciated even at this preliminary stage. Consider a head-or-tail game. Suppose that a player wins if she scores two consecutive heads. The real time lapsing between one flip and another is irrelevant. This is because when a coin reaches a flat surface it takes a negligible time to reach a stable equilibrium (head or tail). If coins took a hundred years to settle at a stable position, however, people wouldn't use them to make up their minds. The point that I want to stress is that what is negligible depends on who (or what) is supposed to be negligent.

Setting aside, for the moment, these premature concerns, let us continue with our description of computational structures. Given any (non empty) state space M, any function $g : M \mapsto M$, and letting T be the non negative integers, we can specify a cascade by constructing the state transition function in the following way: let g^0 be the identity function on M, and let $g^{t+1}(x) = g(g^t(x))$ for all $x \in M$.

Confining ourselves to discrete-time, time invariant systems with finite sets of inputs and outputs, we give a formal definition of finite state automaton (FSA):

[**Automaton**] An automaton is specified by three sets X, Y, and Q, and two functions α and β, where: 1) X is a finite set, the set of inputs

2) Y is a finite set, the set of outputs

3) Q is the set of states

4) $\delta : Q \times X \mapsto Q$, 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) β: $Q \mapsto Y$, the output function is such that when the system is in state q it always

yields output $\beta(q)$. The automaton is said to be finite if Q is finite.

The above definition must be taken as a timeless formal description of a real system. The purpose of this section is to understand what (physical) constraints we must impose on a system for it to implement a FSA. The conclusion, like that of the previous paragraph, is that no amount of physical constraints on the system alone can make of it an implementation of a FSA.

1.3.2 How to implement a Finite State Automaton

The notion of implementation rests on the notion of causal structure "mirroring" a formal structure. In mathematics, structural identity is expressed by a suitable bijective mapping. According to 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*"[13].

So, given the above definition of FSA, a physical system (P) implements a FSA (M) if there is a mapping (f) that maps the internal states of P (Q_P) to internal states of M (Q) such that for every state transition $(S, I) \mapsto \delta(S, I) = S' \mapsto \beta(S') = O'$, if P is in internal state s and receives input i where $f(i) = I$ and $f(s) = S$ this causes it to enter state s' and to output o' such that $f(s') = S'$ and $f(o') = O'$.

[13]D. Chalmers, *A Computational Foundation for the Study of Cognition*: unpublished paper. URL: http://consc.net/papers/computation.html.

Notice that the mapping f allows us to group physical states into physical state-types (by grouping all physical states that are mapped to the same formal state together) in such a way that between physical state-types and computational states there is a relation of implementation that is one-to-one and onto. It is in fact possible to define a map f^* from groups of physical states (Q_P/f)[14] to computational states (Q). It is one-to-one because function f was defined in such a way so as to group physical states into groups that match the formal states of the automaton. It is onto because of the clause "for every state transition" in the above definition. It is legitimate to ask whether f^* is such that between Q_P/f and Q there is a relation of isomorphism. This is the case only if f^* is such that the following holds:

[Iso-Left] For every computational state transition $f^*([s_i]) \Rightarrow f^*([s_j])$ the following causal state transition also holds: $[s_i] \mapsto [s_j]$[15].

[Iso-Right] For every causal state transition $[s_i] \rightarrow [s_j]$ the following computational state transition also holds: $f^*([s_i]) \rightarrow f^*([s_j])$.

Now, it is clear that [Iso-Left] holds (for that is how we defined our notion of implementation). As to the reverse, things are more complicated. We could add [Iso-Right] to the requirement in our definition of implementation, but this would leave out all simpler computations and would reduce the number of automata implemented

[14]The set Q_P/f is the set of groups of physical states that can be constructed from the set Q_P of states using the relation of equivalence induced on it by the mapping. Two physical states are equivalent iff they are mapped onto the same computational state.

[15]Here the notation $[s]$ refers to the class of states that are equivalent to s, i.e. the class of states that are mapped by f onto the same computational state $f(s)$

to exactly one. For the moment, however, I shall assume that we require that there be a relation of isomorphism between Q_P/f and Q. So, our tentative definition is the following:

[Implementation] Given the automaton A specified by $\langle X, Y, Q, \delta, \beta \rangle$, and a physical system S described by $\langle Q_S, T, \{g^t\} \rangle$, S implements A iff there is a function $f^* : Q_S \times X_S \times Y_S \mapsto Q \times X \times Y$ such that: [Iso-Left] for every computational state transition $I_1, S_1 \rightarrow \delta(I_1, S_1) = S_2 \mapsto \beta(S_2) = O_2$ there exists a causal state-type transition $([i_1], [s_1]) \rightarrow [s_2] \mapsto [o_2]$ such that $f^*([i_1]) = I$, $f^*([s_2]) = S_2$, and $f^*([o_2] = O_2$. While X_S and Y_S are the set of physical inputs and physical outputs, $[i]$, $[s]$ and $[o]$ are physical input-types, state-types and output-types defined as all those inputs, states and outputs that f maps onto the same computational inputs, states and outputs.

And: [Iso-Right] for every causal state-type transition $([i_1], [s_1]) \rightarrow [s_2] \mapsto [o_2]$, A is such that $I_1, S_1 \rightarrow \delta(I_1, S_1) = S_2 \mapsto \beta(S_2) = O_2$, where $f^*([s_1]) = S_1$,$f^*([s_2]) = S_2$ and $f^*([o_2]) = O_2$.

Requirement [Iso-Right], as we have anticipated, is not as straightforward as [Iso-Left]. In fact, whereas the existence of a causal state-type transition of the kind $([i_1], [s_1]) \rightarrow [s_2] \mapsto [o_2]$, required by [Iso-Left], is ensured if there exists an underlying causal state transition, [Iso-Right] amounts to the requirement of the existence of a computational state transition in M that corresponds to every given physical state-type-transition. It is tempting to say that, as the elements of Q_P/f are "artificial" groupings of physical states, the causal nature of a transition of the kind $([i_1], [s_1]) \rightarrow [s_2] \mapsto [o_2]$ is inherited from the "natural" causal transitions occurring between the elements of Q_P. Now, there is no safe way to make precise sense of the notion of natural (as opposed to artificial?) state-type transition. However when we quantify over "all state-type transitions" we must make precise sense of what we are saying. What counts as a state-type transition? Do all state-transitions count as tokens of state-type transitions? It is tempting to say yes, or else we would have to give a circular definition of what it is to be a state-type transition, rendering [Iso-Right] tautological. So, given a function f, and any state-transition, there corresponds a state-type transition (obtained through the mapping f). Requiring that not only should the physical system "reflect" the state transitions of M ([Iso-Left]), but that M should also mirror the causal structure of P ([Iso-Right]), seams an unreasonably strong requirement. Chalmers for example, in his definition of implementation, only requires [Iso-Left]. So, we have seen that whether a physical system P implements a FSA depends on the existence of an implementation function f such that [Iso-Left] (and possibly [Iso-Right]). I am now going to ask whether this requirement can be grounded on physical terms only. In other words, the rest of this chapter will try to answer the question: given a generic physical system, what properties of it are necessary and sufficient for it to implement a given FSA M?

1.4 Vacuousness arguments and their consequences

1.4.1 V-arguments

The notion of realization (implementation) of a computational structure has been subject to a variety of observer-relativity arguments. This other category of arguments, let's call them *vacuousness arguments*, aims at showing that if computations are ever instantiated, they are always, vacuously instantiated. As we shall see, these arguments rest on the claim that any real physical system is causally rich enough to be described as implementing any computational structure. If v-arguments are taken seriously, they make computations devoid of any physical interest and (still worse for our concerns), make computational functionalism vacuous, or absurd. V-arguments all point at the (alleged) fact that the physical properties of a system S are insufficient to ascribe syntactic properties to it. "*If computation is defined in terms of the assignment of syntax then everything would be a digital computer, because any object whatever could have syntactical ascriptions made to it*"[16]. In our formalism, then, such arguments amount to the claim that:

[V-arguments] *Given any formal (computational) theory A, and any physical system S, there always exists a labelling scheme L such that the pair $< L, S >$ is a model of A.*

Suggestions that go in this direction abound in the literature.

> On the standard (Turing's) definition of computation it is hard to see how to avoid the following results: 1. For any object there is some description of that object such that under that description the object is a digital computer. 2. For any program and for any sufficiently complex object, there is some description of the object under which it is implementing the program. Thus for example the wall behind my back is right now implementing the Wordstar program, because there is some pattern of molecule movements that is isomorphic with the formal structure of Wordstar. [...]

[16]Searle [82], p. 207.

> I think it is possible to block the result of universal realizability by tightening up our definition of computation. [A] more realistic definition of computation will emphasize such features as the causal relations among program states, programmability and controllability of the mechanism, and situatedness in the real world.[17]

A number of allegedly problematic cases have been proposed, all suggesting that Turing's analysis should be amended, if the notion of computation is to be given any empirical content. In 1978, for example, Pinckfuss proposed what is now known as the case of Pinck's pail:

> Suppose a transparent plastic pail of spring water is sitting in the sun. At the micro level, a vast seething complexity of things are going on: convection currents, frantic breeding of bacteria and other minuscule life forms, and so on. These things in turn require even more frantic activity at the molecular level to sustain them. Now is all this activity not complex enough that, simply by chance, it might realize a human program for a brief period (given suitable correlations between certain micro-events and the requisite input-, output-, and state-symbols of the program)? And if so, must the functionalist not conclude that the water in the pail briefly constitutes the body of a conscious being, and has thoughts and feelings and so on?[18]

A particularly precise v-argument (to be discussed in the following) has been proposed by Putnam. Our diagnosis will concentrate on the contrast between the

[17]Searle [82], p.209.

[18]This is how Lycan puts it in [54], p. 39.

"troublesome" individuation of (computational) states in a real computer (intended as a physical system that computes), and the apparently "safe" individuation of physical states in a RDS.

After having briefly outlined this argument, I shall propose a preliminary diagnosis. It will be argued that (chapter 2), unlike the case of instantiation of mathematical structures, where measurements allow us to individuate the states of a system at a given time independently of all other states, computational structures suffer from not being able to individuate their implementing states on an individual (non relational) basis.

I shall further argue that Turing's analysis cannot be made safe unless the labelling scheme provided for it makes reference (maps) to items that possess (naturalized) semantic properties.

1.4.2 Putnam's v-argument

Does the above definition of implementation provide a labelling scheme that satisfies our desiderata? In the appendix to his book *Representation and Reality*[19], Putnam proposed a now (in)famous argument to the effect that every open physical system implements, in the sense expressed above, every FSA. This, if true, would clash with our (pre-analytic) intuition that not any physical system computes. In particular it would endanger the computationalist hypothesis (and indeed the argument was thought of by Putnam as a reductio ad absurdum of computational functionalism). The conclusion Putnam draws from the result, in fact, is that computational functionalism is not tenable (save accepting a ridiculous form of panpsychism).

The argument is given the form of a theorem, and is composed of two steps. In

[19]Putnam [73].

the first one it is argued that:

[**Putnsm's theorem**] *All physical open systems implement every imputless FSA (FSAs with no input or output)*

The result is then extended to FSAs with input and output. Putnam granted that the specification of particular physical inputs and outputs prevents a straightforward extension of the result. However, Putnam argues, as the result remains valid for the internal description of any FSA, the claim that cognitive properties are coextensive with the implementation of particular FSA structures tantamount to say that they are coextensive with the implementation of the specified input-output functions, thus conflating computational cognitivism with behaviorism.

> Thus we obtain that *the assumption that something is a "realization" of a given automaton description (possesses a specified "functional organization") is equivalent to the statement that it behaves as if it had that description.* In short, "functionalism", if it were correct, would imply behaviorism! If it is true that to possess mental states is simply to possess a certain "functional organization", then it is also true that to possess mental states is simply to possess certain behavior dispositions![20]

Putnam's technical result, moreover, has been argued to be extendable to become a v-argument: i.e. to argue that any open physical system implements any automaton.[21]

[20]Putnam [73], pp. 124-125.

[21]See the next section for a proof of such extension.

Proof. The proof rests on two (physical) principles: 1) The Principle of Continuity, stating that the electromagnetic and gravitational fields are continuous except at most at denumerably many points (under the assumption that the only source of fields are point particles) and 2) The Principle of Noncyclical Behavior, stating that a physical system is at different maximal states at different times. The latter principle follows from the observation that no open system can be perfectly "shielded" from electromagnetic and gravitational signals coming from "natural clocks" (such as radioactive atoms). In other words, it is claimed that the electromagnetic and gravitational signals stemming from (for example) a piece of radioactive matter, being ever changing, induce every open system that is not shielded from them to enter ever changing maximal states. According to Putnam, the principle holds in its generality because there are natural clocks from which no open system can be shielded. Suppose an (inputless) automaton is described by a machine table that goes through the states $ABABABA$. The abstract machine goes through the above states as "machine time" goes by. We wish to prove that any open physical system implements the table in real time. We must then find a pair of state-types of the system such that, during a specified time interval (say from 12:00 to 12:07), the system will realize the above machine table. We must show that, given the sole laws of physics, an omniscient mind would predict that if (for example), the system has been in state A from 12:02 to 12:03, it will be in state-type B from 12:03 to 12:04. Indicating with $St(S,t)$ the maximal state of S at time t, we can construct state intervals by grouping the maximal states the system goes trough in each interval of time $t_i \leq t < t_{i+1}$, thus obtaining, in our example, seven state intervals $S_{i(1<i<7)} = \{ST(S,t) : i < t < i+1\}$. The principle of noncyclical behavior guarantees that all such state intervals are disjoint. Now, by letting $A = S_1 \vee S_3 \vee S_5 \vee S_7$ and $B = S_2 \vee S_4 \vee S_6$ it can be easily checked that in the time interval considered the system goes through the state table prescribed by the automaton. That the system reliably causes state A to be followed by B is ensured by having the groupings being collections of maximal states (thus maximally determining the state of the system at each following time). The generality of the result is finally ensured by the arbitrary choice of the machine table (ABABABA in our case).

□

1.4.3 Extensions of Putnam's result

The extension of the result to FSAs with input and output can be obtained by the following labelling scheme.[22]

[22]The proof of this result can be found in Scheutz [77].

We must show that:

Proposition 1.4.1. *For any open physical system S, for any $Q_i \in Q$ and for any I and O_i there always exists a labelling scheme L such that $[(Q_1, I)©\delta(Q_1, I) = Q_2 \mapsto \beta(Q_2) = O_2]_L$ is true of S.*[23]

Proof. Consider again a sectioning of a (any) time interval ($Int = [t, t']$) into a sequence of consecutive sub-interval: Int_0, Int_1... such that $Int_0 = [t, \frac{t-t'}{2}]$ and $Int_k = [t + \sum_{j=1}^{k} \frac{t'-t}{2^j}, t + \sum_{j=1}^{k+1} \frac{t'-t}{2^j})$

Map the maximal state of S during interval Int_0 onto formal state $Q_0 \in Q$ (this is to be the initial state). Then for each interval state Q_S^k, defined as the grouping of maximal states of the interior of S during interval Int_k, define I_k as the interval state of the boundary of S during interval Int_k. Let I_k label the input i after k computational steps. Now, designate the state interval of the interior of S during interval Int_{k+1} as the "label bearer" of the successor state Q_2. Finally, take $[©]_L$ to mean "causes".

It only remains to group together all interval states of the interior and of the boundary of S that map to the same constant of I or Q. The definition of implementation given at the beginning of this paragraph ensures that the map thus constructed is 1-1 and onto. This, in its turn, ensures that each sentence $[(Q_1, I)©\delta(Q_1, I) = Q_2]_L$ is true of S.

□

Note that if the above extension of Putnam's argument were sound, it would prove that, given any formal computational theory A, and any physical system S, there always exists a labelling scheme L such that the pair $< L, S >$ is a model of A. These extensions, of course, limit the room for combatting manoeuvres[24]

Putnam's own diagnosis of the problem points at the "arbitrary", albeit legitimate, grouping of physical states, i.e. at the need for all states grouped together to form the label bearer for a computational state to have "something in common".

[23] *The sentence is intended to mean that at receiving input I, a computational system in state Q_1 transits to state Q_2 and and finally outputs the symbol O_2. The symbol © stands for the abstract notion of causing, determining a transition.*

[24] See Scheutz [77] for a number of similar labelling schemes.

> [We] must restrict the class of allowable realizers to disjunctions of basic physical states [...] which really do (in an intuitive sense) have 'something in common'.

The problem is that there are constraints on what this "something in common" could be:

> [...] this 'something in common' must itself be describable at a physical, or at worst at a computational level: if the disjuncts in a disjunction of maximal physical states have nothing in common that can be seen at the physical level and nothing that can be seen at the computational level, then to say they 'have in common that they are all realizations of the propositional attitude A', where A is the very propositional attitude that we wish to reduce, would just be to cheat.[25]

The dilemma has been recently expressed very clearly by Scheutz:

> Physical states of an object are normally defined by the theory in which that object is described. As it happens with classical fields, there might be too many states that could potentially correspond to some abstract, in this case computational, state. In order to exclude certain unwanted candidates, one has to define an individuation criterion according to which physical states are singled out. This criterion, however, is not defined within the physical theory that is used to describe the object, but rather at a higher level of description. In the worst case, this will be exactly the computational level, namely in the case that none of the potential

[25]Putnam [73], p. 100.

"lower level" theories can define a property in their respective languages such that the set of states conforming to that property corresponds in a "natural" way to the computational state. The potential circularity is apparent: what it is to be a certain computational state, is to be a set of physical states which are grouped together because they are taken to correspond to that very computational state. Every state-to-state view of implementation must, therefore, avoid being 1) vacuously broad (because physical state type formations are too liberal), and 2) circular (because individuation criteria for physical states are not provided at a level lower than the computational one). In the case of physical fields, one is left with a very pessimistic prospect: there are more than countably many different possible physical states according to the state space of fields (for every interval of real-time). Which of those correspond to a physically possible object, and which correspond to a given object in a "natural way"? Since there are even more possible mappings from physical states onto abstract states, it seems totally implausible if not impossible to specify finite criteria that single out the right mappings. The only way we could find such a mapping is either by pure chance or by using higher level properties that constrain possible objects significantly and hence the plethora of mappings. If we are lucky, then the number of mappings will be so constrained by these properties that we can actually write down the definition of a (correspondence-)function. But again, this "will work" only by using properties defined at levels of description higher than physical fields, yet lower than the computational level of description (which must not be used in defining a mapping from physical states to computational states, if the task is to find out what kind of computation a given physical system implements).[26]

The solution that I shall propose here, I anticipate, will be to claim that the "something in common" be the (physical) properties on which semantic ones supervene.

In the present work the soundness of this (and similar) arguments shall not be discussed. As I said, other arguments to the same effect: "*that computational properties are not intrinsic to physics, so that computational descriptions are observer-relative*" can be found in the literature[27]. The soundness of such arguments has been challenged[28], but most authors concede that they "*point out the need for a better understanding of the bridge between the theory of computation and the theory of physical systems: implementation.*"[29]. I shall argue that, instead, the consequences of such arguments (even if they were assumed to be sound) would not be as dramatic for the computationalist stance as they have been suggested to be.

This "salvage" of computationalism will be achieved at the price of abandoning the (at the moment pervasive) construal of computational properties as being intrinsic properties of a real dynamical system, in favor of an externalist notion of implementation that is consistent with these critical results while retaining the explanatory power of the computationalist hypothesis.

[26]Scheutz [77], p. 169.
[27]e.g. Searle [81].
[28]Chalmers [15], [16], and Scheutz [78].
[29]Chalmers [15].

Chapter 2

The individuation of states in computational and dynamical systems

2.1 Introduction

Most of the arguments presented in the previous chapter have been counter-argued. The general realist strategy has been to argue that the skeptical conclusion follows from a mistaken inference, or from a false premise. The structure of skeptical arguments (the doctrine that I have called: *computational anti-realism*) is the following:

1.a Mental properties are implemented computational properties (sufficiency hypothesis).

2.a Being the implementation of a computational property is observer-relative (v-argument).

3.a Mental properties are not observer-relative. Hence:

4.a Mental properties cannot be computational properties.

The typical combatting manoeuvre denies that being the implementation of a computational property is observer-relative. This is a stance that I call: *computational realism* (2.a is argued to be false).

My strategy, instead, proceeds from the denial of an implicit premise that is common to both the proponents of observer-relativity arguments and to their opponents: the assumption that if syntactic or computational properties are real at all, they must be realized by some intrinsic physical property of the implementing system alone. I call this view: *syntactic internalism.* More precisely, the computational anti-realist's strategy, I argue, should be articulated in the following way:

1.b Mental properties are implemented computational properties.

2.b If computational properties supervene on some non-vacuous disjunction of physical properties, they supervene on a disjunction of intrinsic physical properties of the implementing system (syntactic internalism).

3.b Computational properties do not supervene on a non-vacuous disjunction of intrinsic physical properties of the implementing system (v-argument). Hence:

4.b Mental properties cannot be computational properties.

Computational realists, so far, have not questioned premise 2.b (syntactic internalism). Combatting manoeuvres, in fact, concentrated in the attempt to show that 3.b is false.

In advocating computational realism I shall take, in this work, the opposite stance: I shall assume that v-arguments (hence proposition 3.b) are sound. But I shall argue that syntactic internalism is not the only viable option (i.e. I argue that 2.b is false). In other words, I shall argue that v-arguments, if assumed to be sound, do not entail the bankruptcy of computational realism *tout court*, but only of internalist versions of it.

This strategy is articulated in two steps. First, it is argued that, if v-arguments are taken to be sound, computational realism must endorse the claim that semantic properties are (really) instantiated by the implementing system. Second, it is argued that the necessary semantic properties cannot supervene on intrinsic properties of the implementing system alone. The two claims, together, imply that computational properties are naturalizable only if implementation is construed as supervening on relational physical properties of (1) the implementing system and (2) of its environment. These properties must be such sufficient for externalist semantic properties to be instantiated. I call this version of computational realism: *computational externalism.*

This chapter is dedicated to arguing for the first claim: that computational properties can be conceived as being real (i.e. grounded on physical properties, under physicalistic assumptions), only if intentional properties are factually instantiated by the implementing system.

The argument will proceed from a contrastive analysis of the notion of instantiation: contrasting the "safe" instantiation of a mathematical structure by a physical object, with the troublesome realization of computational properties by the implementing object. The analysis (section 2.2) concludes that what makes mathematical models "safe", i.e. free from observer-relativity objections, is the possibility to specify measurement procedures that ground the relevant mathematical abstractions. It is hypothesized that computational models, lacking this feature, cannot appropriately bridge the gap between the implementing physical system and its abstract, syntactic properties, unless the system is endowed with intentional properties (section 2.3).

A preliminary proposal is then put forward (section 2.4.1): it is to guide our exploration, as a working hypothesis, for the rest of this treatment. Finally some comments are made (section 2.4.2) in order to rid my proposal of possible a-priori objections.

2.2 The individuation of dynamical models

On the notion of instantiation

The ones mentioned in the previous chapter are but a few examples of i-arguments and v-arguments. Each of them can be, and has been, counter argued. What I wish to emphasize here, however, is that, if these arguments were to be taken seriously: (1) Turing's analysis of computation would be insufficient to allow for an honest labelling scheme, i.e. it would be devoid of empirical content; and (2) all attempts to provide an honest labelling scheme on the base of Turing's analysis must avail themselves (surreptitiously) of observer-relative properties.

One may wonder why the relation of instantiation of a function f by a physical system S ($Inst(S, f)$) is not so haunted by similar arguments: why, in other words, are we so sure that there is a straightforward way to provide (in a non observer relative way) for the truth conditions of $Inst(S, f)$?

We said that a necessary condition for the obtaining of $Inst(S, f)$ is that S "causally associate the physical arguments of f (W_i) with its physical values (W_o)". But we have been too quick in assuming this notion to be unproblematic: what does it mean exactly, that a system "causally associates the physical arguments of f (W_i) with its physical values (W_o)"?

Let us start by analyzing the case of classical kinematics as an example. Suppose, that is, that our function (f) is to represent the displacement of an object (S) as time goes by.[1] What does it mean to say that the position of the object "really" instantiates a certain (say linear) function?

It is obvious that a mere change in the way we measure time and/or space, would change the function that is being instantiated. Moreover, measurements are not expected to be infinitely precise, so that a certain amount of error must be tolerated when measuring the value of a magnitude. So, at best, we can say that the conditions for the obtaining of $Inst(S, f)$ are relative to a (arbitrary) selection of the units of measurements and of their errors. In principle, for example, the same magnitudes can be argued to instantiate any (linear, in our hypothesis) function.

In sum, the conditions for the obtaining of $Inst(S, f)$ are relative to the choice of (1) a frame of reference (where it makes a difference) and (2) the units of measurement. The interpretation function for our formal structure will have to specify both of them. The relevant factor, however (relevant in deciding why the notion of instantiation is not itself subject to v-arguments), is that both the units of measurement, the choice of a frame of reference (where it applies), and the errors of the measurements can be specified in objective terms within a physicalistic language. Ideally, moreover, there is no lower bound to the extent of the errors: a perfect measurement, i.e. one

[1]Let us neglect, for the moment, the fact that a real object occupies a region of space only in a finitely approximate sense.

that corresponds exactly with the values of the function instantiated, is by definition without error. The notion of instantiation of a function, therefore, is safely grounded in objective terms through an isomorphism that maps specific physical magnitudes onto abstract mathematical ones. This is why the objectivity of the abstractions that ground the notion of instantiation has not been questioned so often.[2]

The difficulty that v-arguments claim to have expressed is that the way in which computational states are individuated seems unsuitable for allowing a coherent notion of their realization. Note that the allure of computational descriptions is precisely that they allow for multiple realizations. In order to make any sense, however, computational structures must not allow for "universal realizations": in distancing themselves from excessively chauvinistic physical identifications of computational state-types, orthodox understandings of implementation seem to go one step too far away from real dynamical systems.

To better appreciate the direction in which this step was taken, contrast the identification of computational states with that of physical ones (as provided by dynamical system theory). What properties must a real dynamical system have in order for it to be the instantiation of a mathematical dynamical system? Indicate the predicate "*S instantiates M*", where S is a physical system and M a mathematical dynamical system, by $Inst(M, S)$. We are then wondering what interpretation function L would make the pair $< L, S >$ a model for M. The case of dynamics is not different from that of kinematics mentioned above.

Let $M_i(t)$ indicate the time evolution function of magnitude M_i. Consider n such

[2]The exact nature of the relationship between the physical world and mathematics is, as a matter of fact, a very complex issue. But here we are not concerned with a precise metaphysical description of the physical world, but rather with the applicability of the notions that physics deploys.

magnitudes $M_1 ... M_n$. Suppose that their time evolution functions can be expressed by a functional, parametric relation with their respective initial values $x_1 ... x_n$: for each i such that $1 \leq i \leq n$ the time evolution of M_i is $M_i[x_1 ... x_n](t)$.

Let

$$P =< T, M_1 \times ... \times M_n, \{g^t\} >$$

be a dynamical system where T is the set of values of the magnitude time and the cartesian product the set of values of the magnitudes M^i. For every $t \in T$,

$$g^t(x_1 ... x_n) =< M_1[x_1 ... x_n](t) ... M_n[x_1 ... x_n](t) >$$

is the set of state transitions. **M is the system generated by the magnitudes** $M_1 ... M_n$

The structure defined above, to qualify as a dynamical system, moreover, must comply with the following constraints.

[**Dynamical system**] *A system M generated by the magnitudes $M_1 ... M_n$ is a* ***dynamical system*** *iff the following holds:*

1. $M_i[x_1 ... x_n](0) = x_i$ and

2. $M_i[x_1 ... x_n](t + w) = M_i[M_1[x_1 ... x_n](t) ... M_n[x_1 ... x_n](t)](w)$

Under what conditions is a real dynamical system an instantiation of a dynamical system M generated by magnitudes $M_1 ... M_n$? Suppose that we are given a (mathematical) dynamical system M, without being told what the symbols that feature in it are meant to refer to, so that any semantic ascription (any interpretation scheme) is allowed that is coherent with the theory of dynamical systems. Could we say that any real dynamical system is a realization of M? In other words, does a mathematical dynamical system succeed in selecting a class of real dynamical systems?

To start with, the set T must refer to real time values, or M wouldn't be a mathematical dynamical system.[3]. Secondly, as we shall see, reference to measurement procedures must be made.

What feature of measurements makes Galilean models "objective", when contrasted with computational ones? Intuitively, the role of measurements in the above account of mathematical modelling can be expressed by saying that they allow for an independent objective evaluation of time dependencies: measurements bridge the gap between the abstraction of the model and the concreteness of the real system, thus blocking the isomorphism catastrophe. The value of a magnitude can be evaluated both by a formal calculation ($M_i(\bar{t})$) and by a measurement ("the value of M_i at any time $\bar{t}$"), and the match between the two grounds the notion of instantiation. The two evaluations are independent because it is possible for them to mismatch. The gluing factor (gluing formal calculations and direct measurements) appears to be time. The time involved in dynamical modelling is real time.[4]

So, it seems like the crucial instantiating mappings (the mirroring of our pretheoretical talk) are: (1) that between the set T of values the parameter t can take up and the set of values that real time takes up and (2) that between the values of the abstract magnitudes and the results of measurements. Computational models,

[3]T might well be a discrete set, provided that its elements represent time values.

[4]The crucial factor is that, unlike all other physical magnitudes, time is not time-dependent: all other magnitudes (including constant magnitudes), vary as time varies (constant magnitudes are constant with respect to varying times). Time, by contrast, although it is associated with a set of values (like all other magnitudes), is not associated with an evolution function (unless we don't want to think of its evolution function, trivially, as the identity function for all dynamical systems).

instead, represent time in abstraction from its real properties, making time indistinguishable from any other ordered set.

Is this abstraction responsible for the impossibility, in computational models, to set the relevant constraints on the notion of implementation that would render it objective? That is, should we then conclude that the alleged non objectivity of the notion of implementation available to computational models is due to their excessively abstract representation of time? Some have argued, in other domains of enquiry (other than the concern with the objectivity of computation), that computational models of the mind are inadequate precisely for this reason.

> Cognitive processes always unfold in real time. Now, computational models specify only a postulated sequence of states that a system passes through. Dynamical models, by contrast, specify in detail what states the system passes through, but also how those states unfold in real time. [...] Since cognitive processes unfold in real time, any framework for the description of cognitive processes that hopes to be fully adequate [...] must be able to describe not merely what processes occur but how these processes unfold in time.[5]

What is it exactly that is missing when we represent time as an ordered set? What mathematical property of the set of the reals is lacking in a generic ordered set, that is responsible for the non objectivity of computational properties? Is it its cardinality?

The following thought experiment can help us here to expose the virtues of measurements. Imagine that you were trying to measure a magnitude without using your

[5]van Gelder [92], p. 18.

hands: simply by looking at it. For example, imagine that you were trying to measure the distance D between two objects A and B simply by looking at them and at other objects. How far can you go? At best, you would be able to build an ordered set of distances, by judging the relative magnitude of D with respect to other distances (although impaired by the impossibility to use your hands, in fact, your sight is sometimes enough to judge if a distance is greater or smaller than another). The resulting structure (the structure of such ordered set), however, would be compatible with any suitably numerous set of objects, hence the information that a set of objects is (order-)isomorphic to such structure cannot have any (non vacuous) empirical content. The very notion of measurement, instead, implies the possibility to select a unit of measurement (the meter in Paris, for example), and multiply it as many times as it takes to cover the whole distance. The "trick" is that the repetition (multiplication) of this operation allows us to define D as the sum of these units (the abstraction in this case consists in idealizing measurement procedures)[6]. I don't think we would be going too far if we used the above thought experiment as a metaphor: trying to render the relation of implementation $Imp(S, A)$ objective, that is, trying to specify a labelling scheme that would make the pair $< L, S >$ a model for A, is like trying to measure a distance between two objects simply by looking at them.

As for the set of magnitudes $M_1 ... M_n$, any interpretation is allowed: in principle, in fact, the same dynamical system can model different sets of magnitudes of the same RDS. The choice of semantic values for the symbols $M_1 ... M_n$, however, is constrained in various ways. The following has been proposed[7]:

[6]A detailed analysis of the necessary role of measurement procedures in fixing the empirical content of abstract mathematical structures has been proposed by Trizio in [89].

[7]Definition proposed by Giunti in [41].

Consider all the sets of n magnitudes of a RDS. Some of these sets of magnitudes will satisfy, with infinite precision, conditions (1) and (2) (see definition above). Each of these sets will thus generate a mathematical dynamical system P. Any MDS M thus generated by a finite number of magnitudes is a **Galilean dynamical model of the real dynamical system.**

The relation of instantiation, in the case of a Galilean dynamical model of a real dynamical system, can thus be unpacked in the following way:

1. The aspect of the RDS that the MDS models is the change of the magnitudes $M_1 ... M_n$ that generate it.
2. Such change is exactly modelled by the functions $M_1(t)...M_n(t)$, and, most importantly for our analysis,
3. The model is adequate if and only if the value of M_i at any time $\bar{t}$ is $M_i(\bar{t})$.

So, not any real dynamical system can be said to realize a given dynamical system M: in fact not all (arbitrary) choices of magnitudes comply with the three conditions above.

The honest labelling scheme for $Inst(M, S)$ should then be the following:

- the expression: "the value of M_i at any time $\bar{t}$" should be interpreted as referring to the value of the magnitude relative to certain rules for measuring it
- the expression "$M_i(\bar{t})$", by contrast, refers to the result of the calculation, independently of any real measurements.

An interpretation (L) of the formal theory (the MDS) succeeds (or fails) in producing a model only if some of its terms ("the value of M_i at any time $\bar{t}$", in our case) are interpreted via the specification of idealized measurement procedures. However the case is virtuous because, given a RDS, there is no reason to think that there always exist measurement procedures that satisfy any MDS within arbitrarily small error factors. The specification of measurement procedures, in other words, binds the variables in mathematical dynamical systems to real magnitudes in real dynamical systems.

Can computational properties be naturalized in spite of their "silence" as to what surrogate for measurement procedures should be applied? In what follows I argue that, if we stick to a state-to-state correspondence view of the computationalist hypothesis, that is, if we think that computational properties should be thought of as characterized by an isomorphism with the causal structure of a real system, the naturalization programme cannot succeed. I propose to utilize naturalized semantic properties to play, in computational theories, the same role that measurements play in physical theories: to bridge abstract modelling structures with concrete implementing ones.

2.3 The individuation of computational models

2.3.1 From physics to computation: a practical guide

Accepting the validity of skeptical arguments forces one to think of computation as an observer relative property. Computers would be computers *for us*, because we use them as such, and not because they belong to a type of physical systems. In principle, then, anything could be used *as* a computer. But what does it mean to use a physical system as a computer? And if our brains are computers, who is the observer? And, most importantly, is whether something is being used as a computer or not, a fact that can be characterized in objective physicalistic terms, or is it itself an irreducibly observer-relative property?

We shall answer these questions by first briefly commenting on the requirements a physical system must comply with, in order for it to be *usable as a computer.* We shall than put forward an hypothesis as to what are the objective matters of fact that must obtain for a physical system to *be* a computer. The contrastive analysis sketched above blamed the abstract representation of label bearers for the alleged impossibility to individuate an honest labelling scheme. To better appreciate the consequences of such abstractions, consider a simple example.

A delay circuit outputs the same incoming (input) signals after a certain delay d_1. Very intuitively it can be thought of as realizing the identity function: $f(X) = X$. The label bearers for the input architecture are the values of the magnitude voltage measured at the input and at the output of the circuit. The abstract function $f(X) = X$ disregards completely real time. The MDS that models the circuit is in fact, say, $f(x,t) = g_x(t-d)$ (where $f(x,y)$ is 0 for all $y < d$ and $g_x(t)$ is the function that describes the value of x at time t)[8]. This is not an identity function (which no real system will ever instantiate, for $d > 0$ for all real systems). It seems reasonable, however, that some formalism should account for the fact that any two such circuits (consider another similar circuit with delay d_2), are doing a similar job.

The point I wish to stress is that the difference between a system that implements the identity function and one that doesn't is not qualitative, but vague and relative

[8]In general real time dependencies would be much more complex than this: for example the delay will depend on the magnitude of the input, thus reintroducing the problem of units of measurement discussed above. But this simplified example suffices to expose the issue. Notice how in this case the problem of units of measurement is bypassed by having the same magnitude measured both at the input and at the output of the circuit.

to an observer.

Suppose the second circuit was such that it output the same voltage input with a delay of billions of years. We would not even be sure that the universe will still exist when d_2 will have elapsed. Would that circuit be an implementation of the identity function?

Similar practical considerations[9], marking the distance between concrete physical properties and abstract computations, should be made about the domain of applicability of the magnitudes, and about the errors (in measuring real time, or the value of the input, and the value of the output) which should be tolerated before saying that the system is not implementing the given function, etc. Unlike the case of instantiations, the errors can never (not even ideally) be eliminated: if the errors in measuring the output, for example, were reduced to zero, no real system would implement the identity function. If it was too big, two different inputs would be undistinguishable.

The advantage of such abstractions, however, is quite obvious: they allow us to formalize talk of *functionality*. The two circuits in our example, although different, in a certain sense *function* in the same way. Moreover anything else, a neuron for example, could be argued to function that way: this leaves room for the multiple realizability principle that constitutes one of the chief allures of computational functionalism. Abstracting from the specifics of a physical system allows us to build syntactic isomorphisms between various real dependencies and a timeless function from reals to reals (numerical values input and output)

In line with uncontroversial intuitions, moreover, to use a system as a computer, it

[9]In some cases the output also depends on previous inputs, so that a crude representation of time must be reintroduced in the model. Of course any information about real time by now has been lost. Time, in these more complex but typical cases, is measured in steps, not seconds. Steps, unlike seconds, only possess an ordered structure.

must be possible to describe its behavior by finite, discrete functions. A digital system in fact must realize a discrete function with finitely many inputs and outputs (digits) in order to realize, for example, a logical gate. A further step of abstraction must then be taken to account for this. Here again, we shall argue, semantic properties make an irreducible contribution[10] to computation: in order to reduce sets of uncountable inputs and outputs to finite sets, in fact, the respective domains must be segmented, and to this end, I shall argue, arbitrary (semantically individuated) encodings must be introduced.

Consider for example a circuit that realizes (in the abstract observer-relative way) the logical function XOR, i.e. the function: $f(X,Y) = 0$ if $X = Y$, otherwise $f(X,Y) = 1$ (where '0' and '1' are the two "digits"). Suppose the (token-specific) domain and range of applicability are both $[0,100]$ Volts. There must exist two isomorphisms (for the inputs and for the outputs) between disjoint intervals of $[0,100]$ and the reals. For example, they can both map 0 onto the interval $[3-0.1, 3+0.1]$ and 1 onto $[90-0.1, 90+0.1]$. Firstly, such arbitrary segmentations place further limits on what errors should be tolerated: if the input and output errors were equal to 90 Volts, for example, the isomorphisms proposed would be unapplicable. Secondly, it should be obvious by now that the isomorphism catastrophe is again lurking in the background: provided that there are enough values (and uncountably many is more than enough) there always exist, in principle, such isomorphic mappings.

Reference to specific measurement procedures is by now totally abstracted away. So, we can concede that these desiderata set constraints to what properties a system

[10]Here the word *irreducible* should not be taken as entailing the claim that semantic properties enjoy an irreducible ontological status. It should, instead, be understood as referring to the claim that only the properties on which semantic ones supervene can ground the relevant computational abstractions.

must have in order to be usable as a computer, but such constraints are relative to the user. Now, for a system to be objectively a computer, these constraints, and the (arbitrary) mappings, must be objectively implemented. And for constraints and arbitrary mappings like these to be objectively implemented, I shall argue, representational properties must be deployed. I shall not here get into the technical details of such abstractions.[11] It suffices to notice how they all place observer-relative constraints on a physical system. What range of delays should be accepted, what domains and ranges of applicability, what errors should be tolerated, what isomorphic mappings must be in place, are all observer relative constraints: they depend on who wants to do what with these physical systems.

2.3.2 The quest for the honest model

The problem of finding an appropriate labelling scheme for computational structures (either by amending Turing's analysis or by specifying the conditions of applicability of the notion of implementation) has been extensively addressed in the literature. Copeland, for example, granted that any physical system S may be a model for any computational structure A, but he conceived of these excessively liberal models as "deviant", hence as irrelevant with respect to the issue of the empirical content of the computational hypothesis. Interestingly, he likened these "deviant" models to non-standard models in mathematical logic. His treatment is quite apt for grounding the intuition that semantic properties should be brought at the heart of the notion of implementation. Quite obviously, not all models of a formal theory are "about" the intended objects of the theory.

[11]See Scheutz [77] for a detailed treatment of the practical constraints on a physical system for it to be usable as a computer.

As an example of how non-intended models fail to be "honest", consider the case of the so called Sk¨lem's paradox. According to the L¨owenheim-Sk¨lem's theorem, any satisfiable formula in a theory is true in some countable model (a model whose universe is at most countable). When we apply the theorem to the theory of real numbers, it appears to produce a paradox. The theory of real numbers, in fact, contains as an axiom (or as a theorem: Cantor's theorem) a sentence whose intended meaning is that the cardinality of the reals (Card R) is greater than that of the naturals (Card N). Yet the L¨owenheim-Sk¨lem theorem ensures that this axiom is true in a structure that countenances nothing but natural numbers and countable sets.[12] How can a sentence referring to a set larger than any set of natural numbers be true in a universe where there are no sets other than sets of natural numbers?

In rebuking Searle's polemical claim that his wall is implementing the Wordstar program, Copeland replies that *"the wall so acted only if the referent of R in Sk¨lem's countable model is uncountable!"*[13]. Copeland uses the notion of non-standard (dishonest) model of a theory in a looser sense than that of mathematical logic: it is not required that non-standard models be non isomorphic to standard ones. This apparently innocuous extension of the notion of non-standard model, I think, blocks a-priori any chance to ground a labelling scheme in physicalistic (non surreptitiously semantic) terms. All (isomorphic) standard models (in the sense of mathematical logic), are in fact indistinguishable from the point of view of the theory of which they are models. This is the source of a philosophical problem that I call: the *isomorphism catastrophe.*

[12]Putnam, as we shall see (section 3.3.2) uses this result to draw skeptical conclusions about metaphysical realism.

[13]Copeland [19], p.24.

Consider a theory of European geography[14], whose constants are the names of European cities, and whose only predicates are relational (binary) predicates: "*is-north-of*". In the intended interpretation, a sentence of this theory is true iff in the real world it is true that the intended interpretation of the term that occupies the first place of the predicate is a city whose location is north of the city intended by the name that occupies the second place. So, for example, the (formal) sentence "*Moscow is-north-of London*" is true, because Moscow is north of London.

Now, it is obviously possible to interpret the formal theory otherwise. For example, we could take the term "*Moscow*" as referring to number 10 and the term "*London*" as referring to number 1. The predicate "*is-north-of*" can be interpreted as "*is greater than*". Under this (non-intended) interpretation, the sentence "*Moscow is-north-of London*" ($10 > 1$) is still true, but the sentence is no longer about European geography. Copeland dubbed the "intended" labelling scheme for a computational structure: an "honest" labelling scheme. So, under this treatment all "dishonest" models of a computational structure (A) do not count as cases of "computers". The solar system allegedly computing the solutions to its equations of motions, for example, would be "computing" under a dishonest interpretation of computation.

The example of the dishonest model of the theory of European geography is quite apt for exposing the problem. What are the grounds, we may ask, for the claim that the mapping from the formal theory onto numbers is a "non-intended interpretation"? Well, it is quite easy: numbers are not cities, for they possess different properties, and these differences can be expressed in physicalistic terms alone. What a theory is intended to be a theory of is something that is decided (intended) before we propose

[14]This is Copeland's example.

it. Thus a "theory of real numbers" must be a theory of real numbers. If one of its models ascribes some properties to real numbers that we know to be false of them, this is enough ground to say that that model is not "honest". Now, we know what a theory of European geography should be a theory of because we have a clear (extra theoretical) understanding of what geography is (in the real world), we just need a theory to tell us what sentences are true, under that pre-intended interpretation. Our preexistent knowledge of the meaning of the word "geography" or "Moscow", or "real number" gives us positive means of identification and discrimination of the models. Such observer-relative "intentional means" allow us to tell whether a real entity E belongs to the intended model, independently of what relations E bears to other elements of the universe (I can tell whether an entity is the city of London or number 1, without knowing whether London is north of Moscow or not). Similarly, we know that the set of reals has a greater cardinality than that of the naturals even without knowing that Lovenheim-Sk¨lem's theorem holds.

But what Turing's analysis is a theory of is precisely what we are trying to understand. Structural properties of the kind "intended" by Turing's analysis are unsuitable (if i- and v-arguments are assumed to be sound) to provide us with such positive means of identification as the one expressed above. A structure of relations such as that that identifies computational states in Turing's analysis, like the structure of ordered unspecified measurements in our example, is totally uninformative as to what real physical structure would implement it, with the exception of the information about the minimum amount of complexity that the system must possess (for example, the number of states of a candidate implementing system cannot be smaller than that of computational states to be implemented).

The concerns expressed above can be summarized by the following claim.

Given a model $< S, L >$ of a formal computational theory A, it is impossible (without the deployment of semantic capacities) to prove that $< S, L >$ is the "intended" one (i.e. to give a principled notion of "honest model"), unless at least one of the following conditions holds:

1. $< S, L >$ is not isomorphic to all other models $< S_i, L_i >$ and the cardinality of the intended model is known.

2. The truth of any sentence α of A can be assessed in the model individually (independently of the truth of other sentences of A under that interpretation).

The first requirement states the trivial fact that the independent knowledge of the cardinality of the intended model (which is a relational property that can be deduced by deploying structural information alone) can be used to rule certain models out as non-intended, when these can be shown to be non isomorphic to the intended one (e.g. in the case of Sk¨lem's paradox).

The second requirement demands that the interpretation function be such that the truth of the predicates of the theory be assessable independently of the truth of other predicates of the theory.

Consider for example a theory T whose set of terms $\{a, b, c\}$[15] is intended to refer to any three different objects belonging to an ordered set, and whose only unary

[15]In what follows I shall use ordinary braces to refer to ordered sets.

function ($O : \{a, b, c\} \rightarrowtail \{1, 2, 3\}$) is intended to output the position of the element with respect to the other two (e.g. $O(a) = 1$ iff $a < b$ and $a < c$). The only unary predicates of our theory are: $o(a) = 1$, $O(b) = 2$, $O(c) = 3$. The set $\{1, 5, 10\}$, for example, is a model for the theory. The set $\{Johns, Smith, Watson\}$, whose elements belong to a phone-book, is also a model. Which one is the intended model? Well, without any further specification there is no way to select a model of our theory as the "intended one", from within the theory.

As the truth of $O(5) = 2$ in our example cannot be assessed by some property of number 5 alone (independently of the holding of its relational properties with respect to the other two numbers in the set) this interpretation of the theory does not satisfy condition 2. Of course, we may interpret our theory in a "wild" way, as referring to entities that are not numbers (like the phone book example). We can do this because structural, relational properties (such as the relations that hold in an ordered set), do not (per se, i.e. without deploying our capacity to restrict semantically our intended interpretations) belong to any specific universe. This fact appears obvious when we restrict (by a semantic act) our universe to real numbers, and run the same argument over this restricted universe. It is impossible, under the above interpretation, to select any particular triplet of ordered numbers as the "intended" model. One way to do so (without changing the interpretation of the predicates) would be by adding the requirement that numbers a, b and c be interpreted as numbers 1, 5 and 10 respectively. In other words, the only way to restrict our universe without restricting the defining properties so as to become definite descriptions of their bearers, is by deploying our extra-theoretical, semantic (intentional) capacities: i.e. to conceive of the interpretation function as a "baptism" of the terms in the theory. Any such move, however, would require that semantic properties be in place.

Consider, by contrast, the following "virtuous" example. If the interpretations of $o(a) = 1$, $O(b) = 2$, $O(c) = 3$ in our theory were, respectively, $a^1 = 1$, $b^3 = 2$ and $c^5 = 3$ (as these expressions are interpreted in standard number theory), then, whether a sentence in the theory (say $O(b) = 2$) is true in the model, will depend solely on a property of b (on whether it is true that $b^3 = 2$), independently of whether $O(a) = 1$ or $O(c) = 3$ are also true. This interpretation complies with condition 2, without any need to deploy semantic properties of the symbols.

The fact that the expression "the value of M_i at any time $\bar{t}$" can be interpreted as referring to the value of the magnitude at that time (relative to certain rules for measuring it), independently of the relation that holds between the value of M_i at time $\bar{t}$ and the value of M_i at any other time, is what satisfies condition 2 in the case of mathematical dynamical theories. The case of physics, it appears, evades the prohibitions of condition 2, because measurements do the "dirty baptizing job" in a mechanical way: without (apparently) deploying any semantic capacity.[16]

The theory of computation provided by Turing, if v-arguments were sound, would not satisfy condition 2, hence the quest for an "honest" model for a theory of computation is argued to be hopeless.

[16]Of course, semantic capacities must be deployed to select what kind of measurements should be tested (matched) against our calculations. However, the specification of measurement procedures does not itself require further semantic capacities to individuate its intended models: once we have made that decision, in fact, the intended model no longer depends on our intentions, but on mind-independent matters of fact. It could be argued that physics also appears to fail at capturing its intended models. Maxwell's equations, for example, apply equally well if applied to the oscillation of molecules of ether, or to classical or relativistic fields. So it appears like physics also has its difficulties in fixing its models. This argument, however, would miss the point completely. If our measurements under-determine our theories, this is a problem for philosophers of science, but it does not entail that we cannot express what it takes to be a model of a theory in physicalistic terms alone.

2.4 The individuation of dynamical and computational models: a contrastive analysis

2.4.1 Rigid and liberal realizations of mathematical structures

Both the relation of instantiation of mathematical dynamical systems and that of implementation of computational structures are cases of empirical realizations of mathematical structures. The latter, unlike the former, has been argued to fail to suitably fix its models. Regardless of what one thinks of v-arguments, it is interesting to ask what explains (a priori) the difference between the two cases. The analysis proposed above, I believe, allow us to address this issue.

The relations between mathematical structures and the empirical ones that realize them can be classified according to the extent to which the models fix some physical properties of the empirical relata (the realizations). Consider the simple case of concepts whose extensions contain nothing but physical objects. As an example, consider the concept (A) *pair of objects whose mass is 5 kilos.* For each pair of objects, it is a priori true that if the mass of one of them is not 5 kilos, the pair doesn't satisfy the concept. Now contrast A with the concept (B) *pair of objects that stand two meters away from one another.* There is no physical property such that if an object does not possess it, then that object does not belong to a B-pair. Whether there actually are B-pairs one of whose elements does not weigh 5 kilos is an empirical matter of fact. Consider, finally, the concept (C) *pair of red objects.* Not only, like in case of B-pairs, there is no intrinsic physical property such that, if an object does not possess it, then it doesn't belong to a C-pair, but there is no property of the pair of objects (relational or intrinsic) that is necessary and sufficient for two objects to be a C-pair.

Notice that the relations of instantiation of the three concepts are different under a relevant respect. Even if the world was such that A, B and C had the same extension (which would be the case if all and only the objects that stand two meters away from one another weighed 5 kilos and were red), in fact, the following two claims would still hold true:

1. If the mass of an object is not 5 kilos, then necessarily the object does not belong to an A-pair.

2. There is no physical property such that if an object does not possess it, then it is necessarily the case that the object does not belong to a B-pair or to a C-pair.

To mark the difference between these two kinds of concept, I will say that A is rigidly instantiated, while B and C are not. The case of satisfaction of concepts is rather trivial: as concept A makes explicit reference to an intrinsic property of its satisfiers[17], it comes as no surprise that (a priori) its satisfiers share that property. Concept B, by contrast, explicitly mentions only a relational (albeit grounded) property of its satisfiers. Concept C, finally, only mentions a relational ungrounded property shared by its satisfiers. It is then obvious that if all B- and C-satisfiers share some intrinsic properties, this is a contingent happenstance. We may say that the instantiation of relational ungrounded properties is more "liberal" than that of relational grounded properties, and that the latter is more liberal than that of the intrinsic properties which ground them: intrinsic properties are rigidly instantiated

[17]As I have discussed in the previous sections, the property of a magnitude taking up certain (particular) values should better be understood as a relational property. Weighing 5 kilos, for examples, is a relational property of an object with respect to another object that weighs 1 kilo. What is relevant in the present context, however, is that unlike the case of B-concepts and C-concepts, once a unit of measurement is arbitrarily picked up, the (classical, non relativistic) value of the magnitude mass is fixed by intrinsic properties of the object.

by there instances.

The case that is been analyzed here is not different. Just like not all concepts restrict the properties of their satisfiers in the same way, different mathematical structures restrict in different ways the empirical properties of the structures that realize them. The instantiation of a mathematical dynamical system M is rigidly realized by certain real dynamical systems. In fact, if the result of the measurement of a magnitude of a RDS (S) (say M_i) at a given time $\bar{t}$ is not $M_i(\bar{t})$ (as prescribed by the dynamical system M), then necessarily S does not instantiate the MDS M, whatever values the magnitude takes up at other times.

Thus, once the symbols in a MDS M have been interpreted as referring to certain magnitudes, and the relevant framework and units of measurement have been chosen, the information that a physical system S instantiates M suffices to infer the outcome of the relevant measurements. Consequently, the models of the relation of instantiation for a given MDS rigidly pick up (a priori) all and only the RDS's that share the relevant physical properties. We may say that the concept of *realizer of a MDS*, for a given MDS, is rigidly satisfied by the elements of its extension.

Contrast this with the case of implementation. Consider a certain real dynamical system S. Each magnitude of S instantiates some mathematical dynamical system. We have seen that which MDS is being instantiated by each magnitude depends on the outcome of the relevant measurements (potential or actual). Suppose that we also have the information that S implements a certain computational structure A. Under a state-to-state correspondence picture, this entails that there exists a mapping from groups of states (hence from groups of values of magnitudes) of S onto the computational states of A. For each one of these values, it is in general not true that, had that magnitude taken up another value, the system S would have not implemented the given computational structure. Whether changing the value of a magnitude also changes the computational state implemented by it, in fact, never depends only on what values the magnitudes of the candidate implementing system take up: it always also depends on relational properties of these magnitudes. In particular it depends on whether taking up a different value also entails a change in the group of computational equivalence to which the value belongs. This is never a fact that can be established a priori on the base of the intrinsic properties of the implementing object, for, as we have seen, what group a value (or a state) belongs to, is a fact that is relative to factors that are external to the realizing system. We may then say that the concept of *realizer of a computational system*, for a given computational system, is not rigidly instantiated by its satisfiers.

It may be objected that, due to the finite precision of any measurement, the same is true for the instantiation of mathematical dynamical systems. Suppose that a RDS has been observed to instantiate a given MDS. In all real circumstances, it is strictly speaking not true that, had the relevant magnitudes taken up different values, the system would not have instantiated the same MDS. There is, in fact, a certain tolerance due to the non eliminable errors of the measurements. There are, moreover, cases of physical instantiation of mathematical structures that inherently allow for multiple realizations. Think for example of the instantiation of Boyle's laws by a collection of molecules (even under ideal circumstances).

The way in which a mathematical structure under-determines its instantiations, however, remains conceptually different from the way in which computational structures under-determine their implementations. Even when a physical abstract structure (or property) is instantiated by a disjunction of real physical properties, it is instantiated rigidly by them.

In these cases, admittedly, it is not true that had the value of a relevant magnitude (such as the velocity of one particular molecule) been different, the system would have not instantiated the same mathematical structure. In the case of Boyle's laws, for example, it is not true that, had the velocity of a certain molecule been different, the temperature of the system would have been different. Any change in the velocity of any molecule can be "compensated" by an opposite change in the velocity of some other molecule.[18] So, within certain limits, the value of the velocity of a molecule is not, per se, relevant for instantiating Boyle's laws. But in the system containing the ideal gas the magnitude corresponding to the average velocity of all molecules is singularly responsible for the temperature taking up certain values. In the case of implementation, instead, no magnitude has ever so much responsibility.

Similarly, the non eliminable errors in the measurements of real physical magnitudes do not make the instantiation od MDSs liberal: we say that a real magnitude instantiates a given variable when the results of measurements fall within a certain (preestablished) range of precise values.

2.4.2 The logical space of vacuousness arguments

Thus, while the realization of the relation of instantiation rigidly constrains the physical properties of the instantiating physical systems, the realization of the relation

[18]This is true within the boundaries imposed by the lower limit of total average velocity, which, ideally, is 0.

of implementation doesn't. It is this liberalism of the relation of implementation, I argue, that creates the logical space for vacuousness arguments. This should not be taken as a plausibility argument for the validity of these arguments. Rather, by this analysis I wish to stress the following points.

Mathematical dynamical systems can afford a state-to-state correspondence view of their instantiations thanks to the relation that obtains between physical properties (or states) of a RDS, and the real values that can be systematically assigned to the relevant magnitudes. Measurements are such as to reveal these systematic correspondences: there is a systematic correspondence between a system being in a certain state and the relevant magnitudes (thereby the relevant measurements) taking up certain values. The mathematical states of a MDS, once they are interpreted, are such as to rigidly "describe" (or designate) the physical states that instantiate them. Thus, we can say that a RDS does not instantiate a given MDS when the relevant measurements do not match with the correspondent calculated values. This guarantees that the notion of instantiation could not possibly be argued to be a priori vacuous. This, in fact, would require that all magnitudes always take up all real values, even when a unit of measurement and a frame of reference have been chosen.

Secondly, because of the liberalism of implementations, instead, there isn't (and there should not be) any systematic fixed relationship between the physical states of a RDS and the computational states or inputs or outputs that they implement. Computational states, even when interpreted, do not rigidly designate any specific physical properties (or states) of their realizers. In particular, they do not designate any particular value taken up by any magnitude. The concept *realizer of a computational structure*, we said, is not rigidly satisfied.

If a symbol is realized in a memory cell by a certain voltage level, for example, it is never true that, had the voltage taken up a different value, then the physical system would have not realized a memory cell. This is the case not only because there is a certain tolerance for errors (an ordinary digital computer doesn't behave differently if the voltage level is 4.999V instead of 5V), but also because the same symbol (or the same computational state) could have been realized by a voltage level of 10V (or any other level, for that matter). We could, in fact, build our digital computers in such a way that binary coding symbols were realized by voltage levels that take up values that are either close to 5V or to 1000V, instead of being close to either 0V or to 5V.

These considerations, I repeat, do not entail that "anything goes", i.e. that under all circumstances, any physical state can realize any computational state. Not even if v-arguments successfully made their point, would this be the case. The considerations simply entail that, unlike what happens in the case of instantiation, the notion of implementation is not a priori immune to vacuousness arguments.

The proponents of v-arguments do not claim to have empirically shown that the notion of implementation is vacuous. This would require, at least, to show that any physical object can be used as a digital computer. The skeptical arguments, instead, claim to have shown that the state-to-state correspondence notion of implementation is structurally, a priori inadequate to combat vacuous realizations. The standard picture of implementation, we have seen, is not such as to be a priori immune to these arguments. For this reason, standard combatting manoeuvres consist in arguing that the skeptical claims are false. The strategy I propose to adopt, instead, is to retreat implementation to a safer position, whereby vacuousness arguments would be a priori blocked. The model for such a safe notion of implementation will be the virtuous notion of instantiation of a MDS.

Of course, as we want to maintain the principle of multiple realizability, we shall not be able to retain the same role that measurements play in that case: if we constrained computational states to correspond to specific physical properties, or equivalently to certain magnitudes taking up certain values, in fact, we would spoil their abstractness. We shall then have to provide for a surrogate of measurements for blocking the alleged unwanted models of the relation of implementation. The semantic properties of the realizers, I shall argue, can do the job.

The claim that semantic properties should play the same role of measurements should be understood as the claim that semantic properties, like measurements, allow us to block (a-priori) v-arguments: i.e. that they make the concept of implementation rigidly satisfied by the elements of its extension. Another way of understanding the analogy between measurements and semantic properties is to think that while measurements ground the notion of physical instantiation of the mathematical concept of variable, semantic properties must provide for the realization of the mathematical concept of string of symbols, as it is used in computability theory. An example should help to clarify this point.

Suppose that the notion of instantiation of a MDS did not constrain the magnitudes of the realizing system to take up certain values (or equivalently, to certain measurements yielding certain results). Suppose that all that was required for a system to instantiate a given MDS was that there be a one-to-one correspondence between physical states and the values taken up by the variables of the MDS. What could block the conclusion that any physical system instantiates any function? Under the assumption that real physical states undergo change in a continuous way, in fact, it will always be possible to establish such a one-to-one correspondence. The requirement that measurements yield certain results, in these cases, block the vacuousness of instantiation.

The state-to-state correspondence notion of implementation, if v-arguments are right, stands exactly in the same position as the absurd state-to-state correspondence view of instantiation imagined above. If my analysis is correct, the requirement that the label bearers of computational items possess representational properties allows us to block v-arguments like the requirement that measurements yield to certain results allows us to block the absurd conclusion that every physical system instantiates any MDS.

2.5 Towards an intentional theory of implementation

2.5.1 Intentional theories of implementation: a necessary evil?

It is thanks to measurements that physics can afford a state-to-state correspondence view of its mathematical models. Short of a surrogate for measurements (that ground physics from the bottom up, from real to mathematical systems), I argue, only a restriction of the implementational basis to naturalized semantic items could block skeptical arguments.

Given the observer relative conditions under which a RDS can be said to be *usable as a computer*, under what further conditions can we say that it *is being used* as such? We are now in the position to express in an vague and informal way the strategy of my treatment. I shall argue that a system that is usable as a computer is in fact being used as such if and only if the label bearers of its input architecture are, objectively, representations: i.e. they possess naturalized semantic properties. Intentional properties, in this picture, ground the (otherwise vacuously obtaining) isomorphic mappings that are required to sustain the relevant abstractions, thus fixing (in an objective way, if intentionality can be naturalized) what specific constraints, what ranges of applicability, tolerance of errors, etc., should apply.

According to v-arguments, for example, any real system implements a XOR function, provided that we pick up the appropriate label bearers. If we require that the input-tokens and the output-tokens (X-tokens and Y-tokens), be instantiations of properties on which semantic ones supervene, we block v-arguments, while grounding the progressive abstractions in an objective way. The delay of the implementing system, or the errors in measuring X and Y, for example, should be tolerated only so long as they do not disrupt the natural semantic properties of the label bearers (the properties of the label bearers on which semantic ones supervene). It might well be that, under a certain interpretation, the same system also implements, say, an AND gate (it suffices for this that the right AND-gate conditions for digitality obtain). But if such an interpretation maps to tokens that do not, as a matter of fact, possess semantic properties, I argue, the system cannot be said to *really* implement the AND gate. Bearing this working hypothesis in mind, we proceed to investigate its plausibility as a solution to skeptical arguments.

2.6 A priori objections to the semantic restriction of the implementational basis

2.6.1 No computation without representation: either trivial or false

The proposal sketched above will strike some as trivial, and others as trivially implausible. On one side, in fact, it is part of the received view of computation that representational items should be taken into account. Both Fodor and Pylyshyn[19], for example, assume that there is "no computation without representation". The idea behind this claim stems from a number of relatively uncontroversial intuitions. One, for example, is the observation that computations are partly identified by the functions they compute. Functions, in their turn, are partly identified by the interpretations of their inputs and outputs. Interpretations, in their turn, are identified by representations with semantic properties. Hence the claim that there is no computation without representation.[20]

Other intuitions corroborating this "semantic view of computation" are derived from the fact that computationalism is our best theory of the mind. Mental states can be individuated by their semantic properties. Hence computational states must be endowed with representational properties too, if the sufficiency hypothesis is to be upheld. To those who do not object to the claim, my thesis that representational items should be built into the notion of implementation could sound trivial.

The claim, however, has never been made explicit, or properly argued for. Many, in fact, tend to conceive of this as a terminological issue. In particular, those who

[19]In their early seminal works [33] and [74].

[20]This claim is discussed below, in section 2.5.1.

think that the notion of computation (and implementation) is not in danger[21], tend to interpret the claim in a somewhat weak way. There is no computation without representation, according to this weak interpretation, in the sense that, for most practical purposes, computations, and computational theory, are invoked in the explanation of interpreted symbol manipulation: it is not that computations need representations to be implemented. Rather, computations, without representations, would be of little practical use. In what follows I shall argue that, if one takes observer-relativity arguments seriously, then one has to endorse a stronger version of the claim that there is no computation without representation: the view that, without representations, computations lack a principle of individuation.

This stronger claim, however, will strike many as extremely implausible. Partly because of an old, positivistic disgust for intentionality, and partly because of the firm conviction that the notion of computation stands in perfect good shape, many authors would object that the stronger claim doesn't make any sense. Such potential bankruptcy of my thesis is expressed, for example, by Chalmers in a note to a paper on the foundations of computational theories of the mind:

> It will be noted that nothing in my account of computation and implementation invokes any semantic considerations, such as the representational content of internal states. This is precisely as it should be: computations are specified syntactically, not semantically. Although it may very well be the case that any implementations of a given computation share some kind of semantic content, this should be a consequence of an account of

[21]Although the number of concerns expressed about the notion of computation is rapidly increasing, not only among the opponents of computationalism, I suppose that most theorists belong to this category. See chapter 5 for a critical discussion of alternative views of implementation.

> computation and implementation, rather than built into the definition.
>
> If we build semantic considerations into the conditions for implementation, any role that computation can play in providing a foundation for AI and cognitive science will be endangered, as the notion of semantic content is so ill-understood that it desperately needs a foundation itself.[22]

My thesis must be understood as a strict contradiction of the above claim. As we shall see, views such as that expressed above follow from the adoption of a received picture of syntax, according to which syntactic properties supervene on the intrinsic properties of whatever it is that bears them. I shall criticize this view in the following pages, to argue that it unduly restricts our room for maneuver. It is certainly true that the notion of semantic content is poorly understood but, I shall argue, the inclusion of naturalistic theories of semantic content into our theories of computation might be eventually forced upon us.

Those who, like Chalmers, think that vacuousness arguments pose no threat to the notion of computation will find my proposal very unpalatable, if understandable at all. I agree with them that if observer-relativity arguments were all fallacious, having semantic properties built into the theory of implementation would constitute a useless, uncomfortable risk that is not worth taking. Moreover, many of the reasons that have been proposed for endorsing a semantic view of implementation, have been argued to be inconclusive. I shall briefly comment on some of these potential counterarguments, to explain that my reasons for endorsing a semantic view of implementation fall outside their scope.

[22]D. Chalmers. *A Computational Foundation for the Study of Cognition.* Section 2.2. Unpublished, but section 2 was published as [15].

2.6.2 Semantic properties lie on the wrong level of analysis

There are constraints on what kind of restriction of the implementational basis of computation are compatible with a theory of computation and with a computationalist theory of the mind. Before turning to my proposal, then, it is in order to get rid of some potential preliminary concerns. Most attempts to block v-arguments, we have seen, consist in various ways in which we might define an honest labelling scheme on the base of some state-to-state correspondence. Our contrastive analysis suggested that the lack of a surrogate for measurements explains the pattern of their failure. It is quite obvious, however, that if we positively restricted the set of candidate label bearers, this would block v-arguments.

One reason why this strategy has not been pursued is that it does not appear to comply with the chief desideratum of a solution: the individuation criterion for providing a labelling scheme must be defined at the same level of description as that of the theory that is used to describe the implementing object.

What is needed is a characterization of states at a level of description that is such as to naturally correspond to the computational states implemented. This level of description must not be the computational level itself (the potential circularity is obvious).

The solution I propose, however, escapes this concern. The proposed individuation criterion, in fact, is defined at the physical level: if semantic properties can be naturalized, restricting the set of label bearers to entities that possess intentional properties is tantamount to restricting them to the physical properties proposed as a supervenience basis for intentional ones. Thus, the proposed individuation criterion is defined at the same (physical) level as that of the theory that describes the physical object that is candidate for implementing a given computational structure.

2.6.3 Computation is not answerable to cognitive science

It could be argued that the theory of computation stands in no need for being restricted to the domain of cognitive systems, just like any other branch of mathematics doesn't. The symbol string "$2 + 2 = 4$" (together with the entire Peano-arithmetic symbol system) would be semantically interpretable as meaning $2 + 2 = 4$ whether or not anyone ever so interpreted it or even whether or not anyone with a mind ever existed. As "interpretability" is all computations need for their realization, the objection goes, the resort to intentional properties (which are mental properties) unduly restricts the scope of applicability of computational properties. If computation and the mind (including its intentional properties) have something in common, it is because mental properties can be argued to be computational properties. However, it is the mind that is explained by computation, not viceversa: computation, we may say, is not answerable to cognitive science.

This line of argument, obviously, presupposes that the notion of implementation of a computational structure is unproblematic (i.e. that v-arguments are all not sound). In this work, as we said, instead, v-arguments are assumed to be sound.

> It is, alas, one of the scandals of cognitive science that after all these years the very idea of computation remains poorly understood.[23]

It is reassuring, for my proposal, that comments like the one quoted above are increasingly common in the literature. It is yet more reassuring to observe how, unlike many contemporary "fundamentalists of computationalism", as we may call

[23]Clark [18], P. 159.

those who refuse to discuss the possibility that computation might be ill-founded, some of the fathers of computationalism, like Pylyshyn, for example, acknowledged that the realization of a computation must depend on the instantiation of intentional properties, thus coming very close to the suggestion put forward in this work:

> The answer to the question what computation is being performed requires discussion of semantically interpreted computational states.[24]

Notice that Pylyshyn assumes that computational states must be interpret-ed, over and above being interpret-able, for the objectivity of their implementation. Who is doing the interpreting?

Apart from mentioning occasions in the literature that support my thesis, however, here it shall not be possible to give a comprehensive answer to this a-priori worry. I can only advise that the reader bear with me till the end of chapter 4, where I shall dispel the pervasive idea that syntactic properties (such as the ones that ground computational properties) are intrinsic to their bearers[25]. This will allow me to make a positive case for a semantic view of computation. For now, I am only concerned with rejecting a-priori objections.

The objection under consideration is this. If my case for a semantic view of implementation proceeded from some difficulties of a computationalist theory of the mind in attributing content to mental states, then there would be something wrong with it. If computationalism proves inadequate as a theory of the mind, so bad for computationalism: why blame computations? An example of such bad arguments

[24]Pylyshyn [74], P. 58.

[25]e.g. the idea that the symbol string "$2+2=4$" has, among its intrinsic properties, that of being semantically interpretable as meaning $2+2=4$, "whether or not anyone ever so interpreted it or even whether or not anyone with a mind ever existed".

would be the following:[26]

- Computational states are hypothesized to realize mental states (computationalism)
- Mental states are individuated semantically (i.e. by their contents). Hence:
- Computational states must be individuated semantically.

I do not object to this: I think the argument is inconclusive. This, however, does not constitute a case against my semantic view of computation. In this treatment, in fact, the version of mental state computationalism that is afforded by my notion of implementation is only mentioned in the last chapter to illustrate one of its virtues, not as an argument for its plausibility. My case for a semantic view of implementation will only proceed from considering a number of shortcomings of the current understanding of implementation: primarily the need to combat the unwanted proliferation of the models.

2.6.4 Symbols would not be "interpretable" if they couldn't be individuated non-semantically

Arguments in favor of a "semantic view of computation" sometimes proceed from the observation that computations are individuated by the functions computed, and these are individuated semantically. A schema for these arguments is the following:

1. Computations are individuated by the functions they compute

[26]Piccinini attributes variants of this line of argument to Pylyshyn ([74]), Burge ([13]), Peacocke ([65]), Wilson ([96], p. 161). I do not commit to his interpretation of these authors. Here I am concerned with mentioning and combating the potential counter-argument, regardless of whether someone has ever advocated a semantic view of computation on the basis of the fact that mental states are individuated by their contents, or not.

2. Functions are individuated by the pairs $< domain - item, range - item >$ that are denoted by the inputs and outputs of the computation. Hence:
3. Computations are individuated by the semantic properties of their inputs and outputs.

I believe these arguments are either vague or inconclusive. Piccinini argues (rightly, I believe) that these arguments fail to acknowledge that functions can also be individuated by the pairs $< input - type, output - type >$, and that it is this latter means of individuation of functions that bears on the individuation of computations. But he goes on to argue (wrongly, I think) that computations can be interpreted as calculating functions because the pairs $< input - tipe, output - type >$ can be seen as *denoting* the pairs $< domain - item, range - item >$, while they are themselves individuated non-semantically. Thus, it is concluded, for computations to be interpretable at all, states, inputs and outputs must be individuated independently of their semantic properties: because "*functional (non-semantic) individuation of computational states is a prerequisite for talking about their content*". The argument rests, I believe, on a wrong interpretation of the claim that symbols are "independent" of their meanings.

It is certainly true that symbols are independent of their meanings, but the only reasonable interpretation of this claim, is that the *intended* meaning of a symbol cannot be deduced by its intrinsic properties alone. As a consequence of this truism, it is possible that the symbols be individuated independently of their intended meanings (for example, we can individuate words of our language even without knowing what they mean). However, the point here is not whether it is possible in principle to individuate particular symbols by descriptions that do not mention (literally) any semantic property, but whether symbols can be so individuated from the standpoint of the computing machine itself, for the purpose of computation. The following example should help showing that Piccinini's objection begs the question against the semantic view of computation.

Suppose I had never heard of numbers and arithmetics. You might well teach me to perform calculations (by complying with the appropriate syntactic rules), without telling me what these symbols (1, 2, 3... +, etc.) mean. The argument I am criticizing thereby concludes that a semantic view of computation is not viable. As I am not told what the symbols mean at all (they are meaningless to me), it is concluded that computational items need not be semantically individuated. However, showing that I would perform arithmetical operations in a way that is Turing indistinguishable from the way you do it, does not suffice. The semantic view of computation that I shall advocate in this work, in fact, does not presuppose that the "intended" semantic properties must be in place, but that some (any) isomorphic set of semantically evaluated items must be in place. In our example, for instance, my thesis would require that the inputs and outputs to my brain instantiate semantic properties that naturally denote the symbols that I deploy ("1", "2", "3"... "+", etc.).

So it does not suffice to show that I can pass the Turing test without knowing about numbers. It must be further proved that I can compute without having any semantic capacity at all. The fact that ordinary calculators appear to pass the Turing test does not prove me wrong. In fact, I can argue that they pass the Turing test only relative to our representational capacity.

These lines are not intended to support the case for a semantic view of computation, but only to make logical room for it. My point, here, is that the fact that functions can be individuated independently of the meanings of their input- and output-types does not falsify my view of computation.

I think that the intuition driving this kind of argument against the semantic view of computation rests on a too "anthropomorphic", chauvinistic view of epistemic capacities. The idea behind these arguments, in fact, is that symbols can be individuated by their shape (not meaning). This is certainly true, but I argue that the ability to discriminate symbol-types on the basis of their form, without mentioning what representational capacities are also in place, is almost vacuously instantiated. Another example should help to clarify my view.

I am able to type identify one-pound coins. If you ask me to sort the coins in my pockets over and over again, I will show to be able to systematically select the class of 1-pound coins. If you ask me how I do it, I would mention, among various things, the fact that I know what "one-pound coin" means. You could object that, as even a cigarette machine is able to type-identify one-pound coins, my resort to intentional capacities is unnecessary.

Notice, however, that the mere fact that there are systematic factual correspondences between one-pound coins and a certain physical object (whether my brain or the cigarette machine), does not suffice for a perspicuous explanation. It could be argued, in fact, that virtually anything is systematically causally affected by one-pound coins in a way that can be described as "type identification". The patterns of reflected light of a moving one-pound coin, for example, establish factual correspondences with the chemical makeup of a film plate. The gravitational force of a coin also establishes factual correspondences with virtually anything else, etc. Sure these absurd examples are practically irrelevant, but practically irrelevant to whom? The point I am making here is that this "factual correspondence view" of discriminatory and individuation capacities, makes them too widely instantiated to be used in the understanding of non specifically human computation or cognition.

2.6.5 Not all computations are interpretable

Another potential objection, related to the one discussed above, proceeds from the observation that there are computations that are not amenable to a straightforward semantic interpretation. One such computation is represented by the following Machine Table[27]:

	@	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 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?

I shall not here question this point: I am going to assume that this computation

[27]The discovery of this algorithm is due to J. Buntrock and H. Marxen, as cited by A. Wells in [94]. See section 6.6.2 for further comments on the workings of this Turing machine.

is not amenable to any straightforward interpretation. I am going to ask, instead, what consequences the existence of this (and similar) machines has with respect to the semantic view of computation. The potential objection to my thesis, it should be clear, is that there appears to be an incompatibility between the claim that computational input architectures should be individuated semantically and the observation that there are computations whose input architecture is not even interpretable. The potential reductio ad absurdum would run as follows:

1. My thesis requires that the input architecture of computational systems be individuated by the properties on which the semantic properties of its inputs and outputs supervene.
2. If a system of symbols possesses semantic properties it is a fortiori interpretable.
3. There are computations whose input architecture is not systematically interpretable. Hence:
4. My thesis is reduced ad absurdum.

Piccinini, for example, mentions the Turing Machine considered above as a counter example to semantic views of computation. He concludes that these examples show that:

> Sometimes it is useful to describe TMs without assigning any interpretation to their inputs and outputs. [...] The identity of individual TMs is determined by their instructions, not by the interpretations that may or may not be assigned to their inputs and outputs. The whole mathematical theory of computation can be formulated without assigning any

interpretation to the strings of symbols being computed.[28]

I think these arguments ignore the most relevant fact: i.e. that if you did not understand, if you did not "see" that the marks "@" or "1" that you have just read above, "refer" to input- and output-types, the argument could not be run. If v-arguments are sound, the only way to fix an intended model for the realization of the above computation is, I argue, by deploying semantic properties that are identical (or isomorphic) to the ones you have just deployed to interpret the Machine Table as a Machine Table. It does not suffice that you are able to type-identify @'s and 1's by their shape. In fact, you are also able to type identify entities according to categories that cut across the ones to which @'s and 1's belong in the computation above. For example, both @'s and 1's are equally tall marks. If the relevant discriminatory criterion were height, the computation above would reduce to an instance of the identity function.

If v-arguments are sound, real physical systems are in no better position than the inert Machine Table above, when it comes to realize the computation.

It is precisely because they are semantically individuated that symbols can be interpreted as meaning something else. In sum: (1) nothing can be semantically "interpreted", I argue, if it has not previously been semantically individuated (interpretation requires that both what "stands in for" and what is to be stood in for, be represented); but this does not entail that (2) if a system of symbols is (essentially) semantically individuated, then it must be systematically interpretable in an "interesting way". The arguments that I criticized in this section only show that (2) is false, thus leaving the logical space for my thesis intact. It should not come as a surprise

[28]Piccinini [68] p. 6.

that not all computations are amenable to perspicuous interpretations. Expecting this to be the case, in fact, would be like expecting a random selection of words from an English dictionary to mean something.

But the fact that some (most) random selections of words do not mean anything, does not entail that words are not individuated semantically through their representations (the strings of marks). In the worst case, such as that of the Turing machine mentioned in this section, the representations that individuate the input architecture of a computational system will denote nothing but the physical properties on which the symbols themselves, and their ruleful manipulation, supervene. A reason for this (although a conclusive argument to this effect will have to wait till the end of this treatment), I believe, follows from the contrastive analysis that I have proposed in this chapter. Only mathematical dynamical systems are amenable to objective, non-semantic instantiations, because time dependencies, in spite of the arbitrary choice of measurement procedures, succeed in leaving certain systems out from their models.

The models of $Imp(S, A)$, instead, need semantics to "finish up" the individuation job (i.e. to ground the relevant relations of isomorphism, escaping the vacuous proliferation of the models).

2.6.6 How can connectionist networks compute?

An apparently paradoxical consequence of my treatment regards computation in connectionist and neural networks. Notoriously, computations in these models are performed by the collective activity of several units (nodes), each of which computes a non-linear function of the linearly summed inputs from other connected units. Representation in these models occurs, or better emerges (if it does occur or emerge at all) at the higher level of the pattern of activity of these units. This picture appears to require that computation and representation be two independent phenomena, for computation is thought to occur locally, while representation can be distributed over several units.

A consequence of my treatment, then, appears to be that connectionist models could not be considered as computing, contradicting what most advocates of connectionism claim. More precisely, the class of transformations of the activity of a computing unit that leaves the computation performed intact, according to our view, should be such as to ensure the maintenance of the representations input and output.

But what is input and output to a unit, according to connectionist models, are not representations, but non-symbolic numerical values of magnitudes. Notice, however, that the (numerical) function computed by each unit is not given by the unit's physical properties alone, but is relative to the specifics of the activity of all other units. If a neuron, for example, took a year to respond according to the function that it is allegedly computing, it would count as a non-firing unit. So, to decide whether a neuron is or isn't implementing a certain timeless function, we must check what are the consequences of its activity with respect to the activity of other units. Ultimately, to judge whether a single neuron is implementing a function or not we must fix what the whole network is thought to be computing.

Now, at this level, representations may or may not be input and output, depending on objective matters of fact such as the ones discussed above. According to my intentional theory of implementation, if there are representations input and output to the network at this level, then we can check whether a certain computation is or isn't being implemented, and relative to this we can judge where a neuron is or isn't computing its function. If, instead, there are no representations input and output, or if there are representations whose maintenance does not match the constraints that support the computation, then the connectionist network is nothing but an approximation to the underlying dynamical system instantiated, and the architecture proposed is, in this case, strictly speaking, dynamicist.[29]

2.6.7 Implementation must make room for multiple realizability

Another potential a-priori objection to my proposal is the following: not only would a random restriction be arbitrary and uninformative, but it would also make computationalism devoid of its explanatory capacity. Part of the allure of computationalism, in fact, is that it is compatible with functionalism: computational properties only require physical properties, but no *particular* physical property, to be instantiated (neurons, for example, couldn't be argued to implement the same identity function of our example, if we restricted the set of candidate label bearers to voltages in copper wires).

If we restricted the set of candidate realizing tokens to a certain kind of physical states or variables (for example to certain voltages in metal wires), the generality of v-arguments would be blocked: maximal states, and their arbitrary groupings, for instance, would be a priori excluded from our candidate implementations of computational states. But Turing's analysis of computation was devised precisely to abstract away from non-essential features: the particular physical properties of the computer, Turing thought, are totally irrelevant in determining what operation is

[29]It should be noted that the theory of implementation that I shall develop in the course of this work, does not provide arguments in favor or against computationalism as such: it just provides the correct grounds on which to judge whether an architecture is computationalist or not. Whether our mind is or isn't computational, and to what extent it is, if it is, then, is left to empirical findings to say.

being computed. An arbitrary restriction of the candidate labelling schemes would contradict these desiderata. Isn't then my proposal to restrict the input label bearers to intentional items a-priori inadequate?

Let us take a closer look at this alleged problem. The argument seems to be the following:

1. Computational properties must be (in principle) implementable by any kind of physical properties.

2. Any restriction of the set of candidate label bearers to a particular sub set (individuated semantically or otherwise) would leave some kinds of physical properties out as a-priori illegitimate.

3. It follows that no a priori restriction of the set of candidate label bearers can be built into the notion of implementation.

I argue that 2. is false, hence 3. doesn't follow. Given a certain set of physical entities B, in fact, a restriction of it to a subset B_1 can be obtained in two ways: (1) by specifying the intrinsic properties of an entity that are necessary and sufficient to belong to B_1, or (2) by specifying the relational properties of an entity that are necessary and sufficient to belong to B_1. For example the set B of all classical physical bodies can be restricted to B_1 in the following two ways:

(a) A physical body belongs to B_1 iff its mass is m_0; or

(b) A physical body belongs to B_1 iff its weight is m_0 (where the weight of a body is the total gravitational force exerted on it from all other bodies in B).

Notice that while restriction (a) a priori excludes all elements of B whose mass is different from m_0, restriction (b) doesn't. Any element of B, in fact, can in principle be made to weigh m_0 kilograms, its weight depending on the spatial distribution of all other elements.

Accordingly, I shall call a restriction of a set of physical entities B to a subset B_1 an *intrinsic (extrinsic) restriction* if the necessary and sufficient conditions for an element of B to belong to B_1 are some intrinsic (relational) physical properties of it.

Point 2, in the pseudo argument above, is only true of intrinsic restrictions. All that the principle of multiple realizability requires, in fact, is that no particular kind of physical property is excluded a priori from implementation. This entails that if, as we have hypothesized, we are to render the notion of computation objective by restricting the set of candidate input label bearers (CILB henceforth) to representational items, the naturalized theory of representation adopted must be such as to induce an extrinsic restriction on the set. In other words, the properties suggested for an individuation criterion must be extrinsic, relational properties.

Having prepared the ground for our general strategy, and bearing in mind the relevant constraints, we shall now set out to discuss how it can be made explicit and how specific theories of semantics can be used to make of it a concrete proposal.

Chapter 3

Intentional theories of implementation and their pattern of failure

3.1 Introduction

The proposal put forward at the end of the last chapter, that the honest labelling scheme should restrict the set of candidate label bearers for the input architecture of computational structures to intentional items, is nothing more than a working hypothesis.

Firstly, in fact, I have provided no compelling reason why semantic properties (rather than some other class of properties) should play in the notion of implementation the role that measurements play in the notion of instantiation of mathematical dynamical systems.

Secondly, unless we specify what properties semantic ones supervene on, my proposal would amount to explaining an ill-understood class of properties (real computations) on the basis of an even foggier one (intentional properties). Having defended my proposal from a priori objections, has only made logical space for it, but does not add anything to its plausibility.

The purpose of this chapter is to argue that the proposal is indeed very plausible. As there appears to be no agreement yet about what properties intentional ones supervene on, however, I had no option but to try my proposal on various theories of semantics. My heuristics, then, will be the following: try the proposal on a number of theories of intentionality and test it against the consequences of doing so for a theory of implementation. The result, then, will be a different theory of implementation for each theory of intentionality that is applied to my proposal. I call such potential theories of implementation: *intentional theories of implementation.*

The idea behind this heuristics is that, if some of these intentional theories proved to comply with the desiderata of a theory of implementation, this would provide a-posteriori grounds for the plausibility of my proposal.

Unfortunately, this will have to wait till the next chapter: the theories of intentionality considered here (section 3.2), in fact, shall all prove to fail. As I think I already know what kind intentional theory of intentionality can be successfully applied to my proposal, one may wonder why I should lead the reader into a series of failed experiments.

The reason for doing so is two-fold. For one, the intentional theories of implementation discussed in section 3.2, as we shall see, present a pattern of failure (correspondent to the pattern of failure of the respective theories of intentionality deployed) that is suspiciously similar to that of orthodox theories of implementation. This, for reasons to be discussed, provides indirect ground for my proposal of building intentional properties into the notion of implementation.

Secondly, and most importantly, this pattern of failure exposes the implicit commitment of orthodox theories of implementation to *syntactic internalism*: the thesis, I repeat, that syntactic-computational properties supervene on intrinsic properties of the candidate implementing object.

The pattern of failure is the following. Where a theory of implementation (or a theory of intentionality) characterizes its models on the basis of structural isomorphisms alone, it is argued to pose vacuous constraints, or to deploy surreptitiously semantic capacities: i.e. it appears to be unable to rule out unintended models (sections 3.2.1 and 3.2.3). On the other hand, where a theory of implementation (or a theory of intentionality) adds further requirements in the attempt to block unwanted models, it is argued to be too restrictive: i.e. to not comply with the desiderata of a theory of implementation (intentionality): section 3.2.2.

> None of naturalization proposals currently on offer are successful. We have seen a pattern to their failure. Theories that are clearly naturalistic fail to account for essential features of semantic properties [...]. In attempting to avoid counterexamples semantic naturalists place restrictions on the reference (or truth condition) constituting causes or information. But in avoiding counter examples these accounts bring in, either obviously or surreptitiously, semantic and intentional notions and so fail to be naturalistic.[1]

Recall that I wish to argue that: first, (1) if v-arguments are taken to be sound,

[1]Loewer, [53], p. 121. For a treatment of this problem in Conceptual Role Theories of semantics, see Eliasmith ([29]), Block ([10]), Field ([32]), Lycan ([55]), and Fodor and Lepore ([38]). As for Causal Theories of Semantics: Stampe ([87]), Dretske ([26]), Millikan ([58]). For Teleological Theories see: Millikan ([58]), Millikan ([59]), and Bickhard ([8]).

computational realism presupposes that semantic properties are (really) instantiated by the implementing system.

Second, (2) that the necessary semantic properties cannot supervene on intrinsic properties of the implementing system alone. The fact that the only potentially viable theories of implementation (intentionality) are argued to make a surreptitious deployment of intentional capacities, provides ground for the first claim: my proposal, in fact, can be thought of as a move to render the surreptitious deployment of intentional capacities explicit.

But why do isomorphism theories systematically fail to fix their intended models (I call this unwanted feature: the *isomorphism catastrophe*)?

The second part of this chapter (section 3.3), aims at providing an answer to this question. The analysis concludes that syntactic internalism is to be held responsible for the isomorphism catastrophe, thus providing ground for the second claim above, as promised. The conclusion is drawn from a critical analysis of two arguments that exploit the isomorphism catastrophe to reduce ad absurdum the respective isomorphism theories: Newman's objection to Bertrand Russell's causal theory of perception (section 3.3.1), and Putnam's model theoretic argument against metaphysical realism (section 3.3.2). I show that both arguments are implicitly committed to an internalist view of syntactic properties (section 3.3.3).

I argue, consequently, that they should be thought of as a reductio ad absurdum of syntactic internalism.

3.2 The failure of intentional theories of implementation

3.2.1 Conceptual role theories of implementation

It has always been natural to construe meaning (semantics) as some kind of relation between language (representations) and the world. However, concerns about "Fregean" cases (expressions referring to the same thing in the world while having different meanings), led some to argue that the content fixing conditions are to be looked for within language itself. What fixes the meaning of an expression, according to these *internalist* views, is not some actual relation between that expression and some fact of the world, but rather the role the same expression has with respect to other expressions in its language. Similarly, the candidate label bearers of computational states (within a state-to-state correspondence view of implementation), are determined by their reciprocal causal roles, rather than by some actual relation obtaining between each of them and their implementing counterparts. This approach, in the case of semantics, is sometimes called "conceptual role semantics", and it comes in a variety of versions.

If one of these theories successfully naturalized intentional properties, then our solution to the problem of implementation (a "conceptual role theory of implementation", in this case) would amount to requiring that the input labelling scheme mapped abstract inputs and outputs onto conceptual roles. The abstraction from the physical level to the computational one would be grounded by the requirement that variations within the required amount of tolerance do not disrupt the intentional status (or meaning) of the conceptual roles of the input label bearers: that is, we would require that the role of the implementing input or output with respect to other possible inputs and outputs be robust enough to be maintained under any physical transformation allowed by the computational abstractions.

Notice how this would allow us to preserve the internalism of state-to-state correspondence theories of computation, as both computational states, and the intentional properties on which these must be grounded would supervene on intrinsic properties of the implementing system itself.

What I here call "conceptual role theory of implementation" should not be thought of only as a fantasy case invented to proceed with my argument. One of the first philosophers in the computationalist camp to systematically worry about mental content , Gilbert Harman, in fact, devised a theory of the mind that combined Computational Functionalism about mental states with a Functional Role theory of semantics[2]. In his account, unlike the fictitious theory of implementation under discussion here, there was a virtuous division of conceptual labor between the two theories: on the one side, computational roles implemented the functional roles that instantiated semantic properties. On the other side the mental states, individuated by the computational roles of their realizers, were provided with content by the very fact that they instantiated the relevant functional roles.

Such division of labor presupposes that (1) functional and computational roles can be individuated independently form each other and (2) each of them can be objectively realized by the same properties of the implementing systems alone. Although we have, and will further criticize these assumptions, Harman's theory is doubtlessly very elegant: it is internally consistent, and conceptually very economic. What is relevant here, is that if either the thesis that functional role suffices for content, or the

[2]Harman [44].

thesis that computational roles can be individuated non-semantically were dropped, Harman's elegant construction would become circular.

As I shall argue, both theses must be dropped, for very similar reasons. I have, and will further argue against the thesis that computational roles can be individuated non-semantically. Here I deal with the second thesis.

Among the difficulties that conceptual role theories of semantics (like all other internalist proposals) must face, the only one that need interest us here is that the conceptual role of a representation fails to fix the appropriate relations with facts of the world. The meaning of an expression, together with the appropriate knowledge about the world, in fact, must ensure that the reference of that expression is fixed. The fact that the conceptual role of a representation is independent from the world has led some to argue that such theories are a priori inadequate to foot the bill. The sole information that there exists an isomorphism between a conceptual structure (for example the inferential structure of a net of beliefs) and the relational structure of facts obtaining in the world, is insufficient to fix the truth conditions of each conceptual role. We are looking for the truth conditions (a model) for $Ref(P, S_I)$: the intentional relation that obtains between a physical system in a certain state (S_I) and a proposition (P).

The concern that is being voiced here can be expressed by saying that if the only condition that we postulate is that the state of S (I) can be mapped onto P by a function that establishes an isomorphism between the structure of S and that to which P belongs, then the sentence $Ref(P, S_I)$ has too many models. As we want our condition to fix content, this proliferation of models is fatal for the proposal. The case of European geography mentioned in the last chapter is a good example of how things could go astray in fixing the relevant model.

Many authors, moreover, consider inferential roles, and their semantic properties, as part of the explanandum, so they see conceptual role theories as circularly presupposing the existence of states endowed with intentional properties.

> To be told that a state or structure has the semantic content that P if it plays the same inferential role as my belief that P plays in my cognitive economy is to leave one wondering what make my neural structures play an inferential role or participate in a cognitive economy.[3]

Notice how this problem (the proliferation of models) is the same that state-to-state theories of implementation have been argued to be subject to. The same isomorphism catastrophe that haunted state-to-state correspondence theories of computation, now threatens to make conceptual role theories of intentionality vacuous. If this cannot be overcome, a conceptual role theory of implementation, or any purely internalist theory of implementation, would be infected by the isomorphism catastrophe of its label bearers, thus failing to block v-arguments. As we shall see, this reappearance of the isomorphism catastrophe, is no accident.

Many conceptual role theorists[4] acknowledged the impossibility to fix truth conditions on the base of conceptual roles, and conceived theories of meaning (dual-aspect semantics) according to which there must be a component that accounts for the relations between representations and the world. Again, this requirement seems to demand a double bill that cannot be payed. Fodor and Lepore, in their critique of Block's two-factor theory of semantics, put it this way: "*we now have to face the*

[3]Dretske [25], p. 59.

[4]See for example Block [10], Field [32] and Lycan [55].

nasty question: What keeps the two factors stuck together? For example, what prevents there being an expression that has the inferential role appropriate to the content 4 is a prime number but the truth conditions appropriate to the content water is wet?"[5]. In short, the aspect that takes care of the inferential role is (and should be) totally irrelevant when it comes to fix the truth conditions of the externalist aspect (their independence is precisely the allure of dual-aspect theories). As a consequence, the fact that they are systematically matched in the right way stands, again, in the need for an explanation.

Things being so, the possibility to apply dual-aspect semantics to an intentional theory of computation entirely depends on what grounds the semantic properties of the externalist aspect. The dual-aspect semantics manoeuvre, without further specifications as to how the externalist aspect fixes content, and how the two aspects are kept together, is thus inapplicable for our purposes.

3.2.2 Causal theories of implementation

If abstract, formal properties (e.g. correspondence in an isomorphism relation) are unsuitable for fixing the models of our theories of semantics, why not looking at some more "mundane" constraint, such as at the physical causes of representations.

There is in fact another influential understanding of representations that concentrates on their causes. Can we indicate some physical condition that is sufficient to fix the content of representations? Attempts to explain the content of (mental) representations by referring to the external conditions that cause them are known as "causal theories of semantics". They can be seen as the natural enemy of conceptual role theories of intentionality. If representation-tokens of type I are always and

[5]Fodor and Lepore [38], p. 170.

only caused by events of type C, than the semantic properties of I, i.e. the fact that they represent C's, supervene on the fact that C's cause I's. If one such theory of semantics succeeded, then our correspondent proposal would be an intentional theory of implementation that mapped the input architecture onto items that stand in the appropriate physical relations with their causes.

Natural signs are promising candidates for a non observer-relative, content-fixing causal condition. An expanding metal indicates (proto-) means that temperature is rising; it does so whether or not an external observer acknowledges the fact. A ringing door-bell (proto-) means that there is someone at the door; it means so even if there is no one at home to appreciate that.

Unfortunately, representations must be able to misrepresent: they must indicate their truth conditions independently of whether these obtain or not. The problem for a crude causal theory of semantics (a problem often called the *problem of error*), then, is this: if the necessary and sufficient condition for I's to represent C's is that C's cause I's, then if R mistakenly represent something (that is not in the intended extension of C's) as a C, this something is also (by definition) a C. A way to express the problem is saying that the crude causal theory does not leave any logical space for error, for any candidate error is turned into a non-error by the very fact that it has occurred. This crude causal theory manoeuvre, then, avoids the isomorphism catastrophe at the price of failing to achieve sufficiency: again no one appears to foot the double bill.

Although no one has ever really advocated a crude causal theory of semantics, thinking of it serves the purpose of exposing clearly the problem of error, which all causal theories have been argued to be incapable to solve. It is instructive to think about what the failure to solve the problem of error would entail for a causal theory of implementation. According to a crude causal theory of semantics, we have seen, for an item I to represent something it is sufficient (and necessary) that that something causes I. As a consequence, the labelling scheme that is afforded by a causal theory that does not make room for error, would map inputs and outputs onto some intrinsic properties of the representations. There would be no way, in fact, to tell "right causes" apart from "wrong causes", so that the constraints imposed by causal theories for an item to be representational $Ref(S_I, P)$) would boil down to requiring that the system enters a particular physical state (I), independently of what relations hold between I and other similar items in S.

But we have seen (section 2.5.7) how a minimal requirement for our proposed solution to be applicable is that the restriction of the candidate input label bearers be based on non-intrinsic properties.[6] As a consequence, causal theories of semantics as we know them do not appear to be suitable for our purpose.

3.2.3 Isomorphism theories of implementation

In the first part of this treatment I discussed some difficulties in grounding the notion of implementation onto some suitable (non observer-relative) notion of structural isomorphism. The story that was told in the previous sections can be summarized as follows. The computationalist stance has been seen by some to suffer from a tradeoff: in order to avoid vacuousness (i.e. to fall victim of the isomorphism catastrophe), it must constrain its implementations so as to block arbitrary realizations; but theories

[6]Of course if a theory of semantics doesn't make room for error, that theory is not a theory of semantics after all, so it would not even be a candidate for the labelling scheme of an intentional theory of implementation. But as there is not yet an agreement about where it is best to look for a naturalized theory of semantics, it is instructive to test our intentional theories of implementation on as many available naturalization proposals as possible.

that appear to succeed at that can be argued to make a surreptitious deployment of semantic properties. On the other hand, in order to avoid a chauvinistic conflation of computational properties with physical properties of the implementing medium, it must not constrain its implementations too much; but theories that comply with this desideratum have been argued to place vacuous constraints. The alleged problem is: there doesn't seem to be a notion of implementation that foots the double bill.

Interestingly, there are in the literature attempts to provide a theory of representational content on the base of structural isomorphisms. It can be argued, however, that, just like their counterpart state-to-state correspondence theories of implementation, such accounts either surreptitiously avail themselves of a non naturalized observer, or they are subject to the isomorphism catastrophe. Cummins' account of misrepresentation ([21]), for example, explains the robustness of representational content (thus apparently solving the problem of error) by introducing the notion of target. Representations have a "target" (which is the element that is invariant under misrepresentation), to which they apply. Misrepresentations are accounted for as cases in which a representation is applied to a target with respect to which they are false. True representations are accounted for by resorting to the notion of structural isomorphism.

The paradigmatic example is that of a toy car running in a maze. The turns of the car (its wheels) are regulated by a peg that slides in a card that is inserted into the car; such card is "read" by moving the card through the position of the peg. Different cards represent different mazes; the content of these representations is fixed by the slot pattern. Representation is successful when the slot pattern is isomorphic to the maze. In this case the card guides the toy car through the maze. Misrepresentation happens when such structural isomorphism does not hold. In these cases, although the content of the representation is robustly the same, the car doesn't get through. It has been objected that

> The normativity in this example [...] is carried under the assumption that the car is supposed to run through the maze, and that assumption is made by the observer, not by the car. If the goal were to hit the wall of the maze at a certain point, then the card that gets the car through the maze would no longer be correct. [...] Cummins assumes that the notion of structural isomorphism is relatively unproblematic, but it is in fact seriously problematic. There is no "fact of the matter" about what the relevant structure is in any material entity. Suppose that the card inserted into the car were not "read" by a peg in the slot, but rather, by a head responding to the pattern of magnetic domains along the edge of the slot. Now the slot pattern per se is irrelevant.[7]

We have, once again, switched from a position that fails to achieve sufficiency (in that case by not being able to solve the problem of error) to one that is made hostage of the isomorphism catastrophe.

If this problem cannot be amended, an isomorphism theory of implementation would suffer, predictably, from the very same problem which our intentional theories of implementation were devised to solve: it would either be subject to the isomorphism catastrophe, or it would surreptitiously deploy non-naturalized semantic properties to achieve sufficiency.

[7]Bickhard [9], pp. 9-10.

3.3 Preliminary diagnosis: the source of the isomorphism catastrophe

3.3.1 Newman's argument

We have proposed to block the isomorphism catastrophe in theories of computation by imposing that the input label bearers possess naturalized intentional properties. Now we find ourselves again in troubles, for the catastrophe seems to have come in again from the back door: many proposals to naturalize semantics seem to be hostage to the very same dilemma. Where do isomorphism catastrophes originally stem from?

Concerns about the lack of informativity of structural isomorphisms are not new in the philosophical literature. A precursor of all arguments hinting at the isomorphism catastrophe, in fact, dates back to the beginning of last century, and can be considered the father of all v-arguments.[8]

In 1927 Bertrand Russell claimed that "*wherever we infer from perception, it is only structure that we validly infer; and structure is what can be expressed by mathematical logic*"[9]. To this, a friend and colleague of Turing's, Max Newman, contested that: "*no[] information about the aggregate A, except its cardinal number, is contained in the statement that there exists a system of relations, with A as field, whose structure is an assigned one. For given any aggregate A, a system of relations between its members can be found having any assigned structure compatible with the cardinal number of A*"[10]. Russell later conceded that this is indeed the case.

It is worth taking a quick look at Newman's argument. Having decided that it is

[8]A more recent, related line of argument is Putnam's model-theoretic refutation of metaphysical realism, to be discussed in the following pages ([71]).

[9]Russell [75], p. 254.

[10]Newman [64], p. 140.

rational to believe in an external world (at the expense of "solipsism" and "phenomenalism"), Russell was in the business of enquiring what knowledge, if any, we can have of it. The "external world", as conceived of by Russell (in that work), consists of all and only unperceived causes of our perceptions (of all our actual as well as possible perceptions). Russell concluded that, if no direct (by acquaintance) knowledge can be achieved of such causes (on pain of contradiction), nevertheless we can achieve indirect (inferential) knowledge of them. The only kind of knowledge that we can have of the unperceived causes of perception is, according to Russell, "structural": i.e. it is knowledge of the structure of relations that obtains between the unperceived causes. Of such relation, Newman claims, "*nothing is known (or nothing need be assumed to be known), but its existence*". Russell takes, it is conceded, the generating relation to be "causal continuity", i.e. the fact that events close to each other in a spatial map of the structure of relations are to be taken as representing unperceived events that are "causally continuous" to each other. However, Newman continues, "*if Mr. Russell's principles are to be upheld this statement must be merely the definition of causally continuous: if anything were directly known about its nature we should know something not structural about the external world.*"[11]

The form of our alleged knowledge of the external world, in Russell's own words, is that we know that "*[t]here is a relation R such that the structure of the external world with reference to R is W*".

Newman's objection is 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

[11]ibid. pp. 144-145.

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.[12]

The argument rests on the claim that: "*given an aggregate A, there exists a system of relations, with any assigned structure compatible with the cardinal number of A, having A as its field*". Clearly, then, the universality of the argument rests on the assumption that no restriction (intrinsic or extrinsic) be placed on the potential generating relation. The relation, in other words, must be understood in its most general sense: a relation is the class of all sets $(x_1, x_2, ...x_n)$ satisfying a given propositional function $\phi(x_1, x_2, ...x_n)$.

For example, given any four objects (this is Newman's example): a, α, β and γ, a relation that holds between a and α, a and β, a and γ, but no other pairs, is the set of all couples x and y, satisfying the propositional function: x is a, and y is α, β or γ. As this propositional function is satisfiable by any set of four objects, the claim that four objects are the field of a structure of relations as the one assigned above is totally uninformative. As this is true for any aggregate of entities and any assigned structure of relations, Newman concludes that structural knowledge is no knowledge at all.

The argument can only be combatted by restricting the set of propositional functions that count as "real", or "relevant" relations (or "honest relations", in our terminology) : for example by restricting the domain of possible relations to "real" as opposed to "fictitious" relations, where a "fictitious" relation is one "*whose only property is that it holds between the objects that it does hold between*" (such as the relation

[12]ibid. p. 144.

of the example above). As Russell is describing what knowledge we can have of the external (real) world, restricting ourselves to real relations is a respectable move after all.

This restriction, however, succeeds in combating the argument only in so far as it cannot be argued that, given a structure W of real relations between the elements of a given aggregate A, there is a system of real relations between the objects of A having any assigned structure W_1. In other words, the restriction to real relations blocks the argument only if the argument cannot be shown to apply to structures of real relations as well. But, Newman argues, things are precisely so.

Consider, in fact, an aggregate A and a structure of real relations between its elements (W) that is supposed to be known. The structure provides us with all necessary means to name the objects of the aggregate (each object can be given the same name as its correlate in the map corresponding to W).

Consider again, for example, the case of the four objects. The structure generated by the "fictitious" relation "x is a, and y is α, β or γ" can be also generated by the real relation: "denoted by letters of different alphabets". So, it appears, for any structure W generated by a fictitious relation R there is a real relation R_r that generates the same structure. Hence the restriction to real relations does not suffice to block the argument.

Other attempts to restrict the candidate implementing relations (for example one based on the distinction between "important" and "unimportant" relations, much alike to the notion of honest models in Copeland's treatment) are argued by Newman to be equally insufficient. The pattern to the failure of all such attempts is the following:

1. We hope to block Newman's argument by restricting the set of candidate relations from R (the set of all possible relations) to C, the set of "honest" relations.

2. The "honesty" of C's, however, ought to be judged using the only information we have about unperceived events.

3. But the sole information that an aggregate is the field of a structure of relation W is compatible with any allegedly honest further information about the nature of the generating relation. Hence:

4. No proposed restriction of R is suitable for blocking Newman's argument.

Notice that Newman suggests that structural theories of perception all suffer from a double bind that is very similar to the one we have already discussed in the cases of the theories of implementation and intentionality. In his own words, the double bind can be expressed as follows:

> The argument that has here been used [...] proceeds from a denial that there is a classification of relations [...] with these properties: a) the classification is applicable to relations between unperceived events; b) if C is the class to which the generating relation of the world-structure W is held to belong, it cannot be logically demonstrated that there is another relation of the class C that generates an assigned structure W_1.[13]

We are now in the position to express an a-priori concern about such approaches along the lines of Newman's argument. The information that the represented is

[13]ibid. p. 147.

"isomorphic" to the representation, in fact, provides us with no information about the represented, hence about what the world should be like for the representation to be "true" (or false). We have seen how, if no restriction is placed on the generating relations, the claim that a domain is isomorphic to the representing one is totally uninformative, hence unsuitable to ground semantic notions (such as "true", "valid", or "correct").

The only way to make isomorphism informative would be to restrict the class of generating relations so as to leave all unwanted instantiations out. However, it can be argued, following Newman's analysis, that this would be a solution only if we can give a principled way to distinguish "honest" from "dishonest" relations in such a way that:

1. the classification is applicable to relations between non-representational events (i.e. between non semantic, natural properties) and
2. if C is the class to which the generating relation of the represented domain-structure W is held to belong, it cannot be logically demonstrated that there is another relation of the class C that generates an assigned structure W_1

Before concluding our digression, it is worth to notice that the double bill that v-arguments claim is impossible to pay can be rephrased as follows:

It is impossible to distinguish in a principled way "honest" from "dishonest" models of the implementation relation in such a way that:

1. the classification is applicable to physical, non-computational structures and

2. if C is the class to which the generating relation of the real physical structure W is held to belong, it cannot be logically demonstrated that there is another relation of the class C that generates the same structure W having as field any assigned (sufficiently complex) aggregate A.

I argue that the difficulties that the theories of computation encounter in paying such double bill are inherited from the symmetric double bind that haunts many implicit understandings of intentionality (on which, we have argued, real computation must be grounded). This treatment of the isomorphism catastrophe is also apt for further justifying the proposed solution: semantics, like measurements in physics, provide us with a non-structural, direct link between the abstract model and the represented domain.

3.3.2 Putnam's model-theoretic argument

There exists in the literature a contemporary, model-theoretic argument hinting at the isomorphism catastrophe. It is due, unsurprisingly, to Putnam, and goes under the name of *model-theoretic argument against metaphysical realism.* We have seen how, if Newman's objection to Russell's theory of perception were sound, knowledge of structure would be no knowledge at all. This is because the alleged object of such knowledge (the model of the theory, we would say today), is radically undetermined.

Putnam's version of the argument entails, as one of its consequences, a denial that any descriptivist theory of reference could ever succeed in supplying content-fixing conditions. It goes without saying that accepting the conclusion Putnam draws from this argument would radically undermine my strategy for solving the problem of implementation: an obvious consequence of the argument is, in fact, that no naturalistic theory of intentionality could ever succeed.

The similarity between this argument and the v-argument against the state-to-state theory of implementation which our treatment was devised to combat is striking, so the argument is particularly apt for further corroborating the overall claim that computation cannot be naturalized unless intentionality can be naturalized too. Recall that Putnam's v-arguments to the effect that any object S is a model for $Imp(S, A)$, for any automaton A, proceeds from the claim that, (1) given any automaton and any object, there always is an non-intended model and, (2) there is no principled way to tell which is which by adding suitable constraints to the theory.

Similarly, Putnam's model-theoretic argument proceeds from the claim that, (1) any sufficiently complex theory (in the sense in which the term is used in mathematical logic) has non-intended models and (2) there is no principled way, no amount of additional information (not even physical information), to determine which one is the intended model. If such additional information is the physical information that would allegedly be necessary and sufficient to fix the content of representation (the information provided by a perfect naturalistic theory of intentionality), then the result leads to a reductio ad absurdum: the "supertheory" comprises all the information that is sufficient to determine the reference of its symbols (ex hypothesis), but it is still insufficient to fix its intended model (i.e. its content). We can conclude that the sufficient information added was not sufficient, contradicting the hypothesis. Hence no naturalized theory of intentionality can, in principle, achieve its own goals.

The argument consists of an extension of Sk¨lem's theorem.[14] Recall that Sk¨lem's theorem leads to an apparent paradox (known as Sk¨lem's paradox). We would like

[14]What follows refers to the version proposed by Putnam in [72].

any model of a theory of real numbers to map the expression "*the set of real numbers*" to the actual set of real numbers: naturally, we would like a theory of something to be about that something.

As we have seen, however, any satisfiable theory has a countable model which comprises, of course, nothing but countable sets. In this model, therefore, the expression "*the set of real numbers*" cannot refer to the set of real numbers (which, by Cantor's theorem, is uncountable). Moreover, the presence in the theory of a sentence like "$Card(R) > Card(N)$" (Cantor's theorem) is of no help in identifying the intended model, for it holds true in all the models, whether countable or not.

The naturalist's move, at this point, is to claim that if the natural conditions that instantiate the intentional relation ($Ref(\text{“}R\text{”}, R)$, in our case) were added to the theory, this would suffice to determine that when the theory of real number mentions "real numbers" it refers to real numbers, and not to some other non-intended set.

Putnam's model-theoretic argument aims at blocking (a priori) any such move. The "trick" is to show that also sentences that appear to be empirically decidable are bound to be under-determined as to what is the intended model that satisfies them, even if all the relevant physical information is provided.

The sentence used for the proof is about sets as understood from the perspective of set theory. There is an ongoing, old debate about the nature of sets. G¨del proved that a model for set theory as we know it (axiomatized by ZF, for example) comprises nothing but "constructible sets". These are all the sets that can be recursively constructed out of simpler sets, according to fixed rules.[15]

[15]A more intuitive way to define constructible sets is as sets that can be outputted by a transfinite computation: this is an ordinary Turing machine computation but with a tape of arbitrary ordinal length and arbitrary ordinal time in which to operate on it.

The question is whether the class of all such sets (L) exhausts the class of all sets (V), i.e. whether there exist sets that are not constructible.

It has been proved, moreover, that the question cannot be settled from within set theory itself, for the axioms of set theory are independent from $L = V$. Thus there are models of ZF set-theory in which $L = V$ is satisfied and others in which it is not. Now, either there actually are non-constructible sets, or there aren't. Again, we would like set theory to be about real sets, so, if a theory has both a model that satisfies $L = V$ and a model that satisfies $L \neq V$, then, depending on whether there actually are non-constructible sets or not, the intended model must be that which corresponds to the "truth" (like the intended model of real number theory is that in which the set R is, really, uncountable).

There is, it appears, a way to discover the "truth". Imagine a machine that constantly measures some physical magnitude (any magnitude would do) and outputs its results every second or so. Imagine, further, that such machine never stopped, for an infinite amount of time. The set thus produced, let us call it X, is clearly not constructible. Aha! So there are non constructible sets after all! Then it should be easy to tell an honest model from a non-intended one: if a model does not contain X and all its unconstructible classmates (whose existence we have "proved"), then it cannot be an intended model. So it would appear that requiring that X be in the models, as an additional constraint to standard set theory, should select the intended model.

However, Putnam thinks that it can be proven that even a theory of sets plus $L = V$ as an additional axiom has a model that contains X (and its classmates).

That is, he proposed a proof for the following theorem: *ZF plus $V = L$ has an ω-model which contains any given countable set of real numbers.*[16] Thus, being a model that contains X (which is a set of countably many real numbers) is compatible with being a model that satisfies $V = L$. Therefore, no amount of additional constraints appears to rule out the unwanted indeterminacy or reference in the vocabulary of set theory: any additional constraint is "just more theory".

The reason why such argument would be fatal for my proposal is quite simple. We can argue as follows[17]:

1. Suppose we have a perfect naturalised theory of intentionality, according to which for a model of set theory to be the intended one, a certain number of (relational, physical) facts (described by A,B,C) must obtain .

2. Then, a supertheory of sets, to which A,B,C are added as constraints, must guarantee that all unwanted models are ruled out.

3. But we have seen that any theory of sets whatsoever, regardless of any additional theoretical constraints, has unintended models.

4. Therefore 1 must be false.

Because of its generality (no particular theory of intentionality has been assumed), this would entail that the naturalization of intentionality cannot be carried out: truth (any truth, even the whole of truth) is not sufficient for reference. If all naturalistic theories of intentionality fall victim to the isomorphism catastrophe, then my maneuver to save computation from it by grounding implementation on naturalized semantic

[16]Putnam's proof of this theorem has been argued to be mistaken (see Bays [3]). Putnam himself acknowledged that we cannot claim to have shown that set theory is semantically indeterminate. His proof, however (Bays concedes this point) shows that semantic indeterminacy is possible. As the possibility of semantic indeterminacy suffices for the purposes of this chapter, I shall here assume, for the sake of the argument, that set theory is semantically indeterminate.

[17]Putnam does, for example.

properties would be circular.

I could argue, in this case, that Putnam's arguments further corroborate my hypothesis that a proper analysis of computation must expose the implicit commitment to naturalized intentionality (the sameness of the symptoms speaks for the sameness of the cause), but my proposal would fail its optimistic purposes. Put plainly, if the possibility to naturalize computation is dependent on the possibility to naturalize intentionality, an enemy of naturalistic intentionality is also my enemy.

Indeed, a full-blown attack to metaphysical realism such as the one represented by Putnam's model-theoretic argument undermines not only my proposal, but the very essence of most philosophical programmes on the board. In particular, it undermines not only computationalism as a theory of the mind, but also any other theory of the mind that purports to be physicalistic in some way or other. For these reasons a challenge to metaphysical realism falls entirely outside the scope of this treatment. Fortunately, the two arguments: the v-argument presented at the beginning of this work and the model-theoretic one mentioned above, need not be accepted or refuted together. We may well accept that computational theories as we know them are unsuitable for ruling out unwanted models, while denying that this is true of any imaginable theory.

We shall argue that the surreptitious assumption behind all v-arguments is that syntactic properties are intrinsic to their bearers. Newman's objection to Russell, for example, was carried out under the assumption that acquaintance with (knowledge of) facts of the external world could not be but structural knowledge (i.e. it could not be knowledge by direct acquaintance). The idea is that the same structure is intrinsic to both (1) our representations (with which we have direct acquaintance) and (2) their unperceived causes. The sole additional knowledge of the sameness of these structures does the rest: this is how we would come to know something true about a world with which we have no direct acquaintance with.

Similarly, I have argued, Putnam's objection to computationalism need be accepted (if at all) only under the assumption that computational (structural) properties be conceived as intrinsic to physical objects (as indeed most current understandings of computation do). In that case, the sameness of structure between the causal organization of the model and the formal organization of the computational structure, in the theory of implementation that we have criticized, hopes to guarantee the honesty of the models (the objectivity of implementation). We have seen that these pictures are exposed to the isomorphism catastrophe.

Putnam, however, digs the knife deeper, arguing that the same problem applies to the very idea that there is a world of objects and properties "waiting" for a theory to name them and to "speak" the very same truths that, anyway, already obtain among them: this is metaphysical realism. Objects and properties, instead, according to Putnam, are born with names to start with.

I argue that we can learn a less dramatic lesson from these arguments: structural properties are *real* only if coupled (grounded) with *real* intentional properties. So v-arguments can be accepted only if no real intentional properties are assumed to be in place. The model-theoretic argument, for example, is carried out under the assumption that no one is there to interpret the symbols. But the unwanted model that Putnam proves to always exist, is only "unwanted" if there is one that is "wanted". And nothing can be wanted or unwanted in itself: there must be someone there to do the wanting.

More precisely, the unreasonable hope that Putnam proved to be unsatisfiable, is that first-order logic, set theory and model theory, considered as mere uninterpreted systems of symbols, could individuate and eliminate "unwanted models" that are structurally indistinguishable from each other. Predictably, no uninterpreted system of symbols, not even one that could be interpreted as describing the facts that naturalize intentionality, could do that. This should not come as a surprise. After all, if intentionality (as naturalists think) is a natural property, it is not reasonable to expect that a mathematical description of it would share the same causal properties with it.

I think that the temperature of a gas, for example, is a natural property that can be described by a dynamical system that represents the positions and velocities of a large number of atoms. Should I expect to be able to boil some water by throwing into it a written copy of this set of differential equations? And if not, should I conclude that temperature is not a natural property after all? Hoping to make a difference as to the indistinguishability of the models by throwing into the models of set theory an inert set of real numbers (X) that can be interpreted as referring to the conditions of instantiation of intentional properties, is, I argue, just as desperate.

3.3.3 The reductio ad absurdum of syntactic internalism

I argue that, far from providing an a priori argument against natural semantics, the model-theoretic argument can be thought of as a reductio ad absurdum of syntactic internalism.

Consider two tokens of the same theory, T_1 and T_2 (say two identical copies of ZF set theory + "$V = L$" + *Constraints*). One can argue as follows:

1. Semantic properties supervene on structural properties alone, therefore

2. Sameness of structure entails sameness of semantic properties.

3. By Putnam's argument, sameness of structure is compatible with different semantic properties ($L = V$ and $L \neq V$, for example). It follows that:

4. The information that there is a structural isomorphism between T_1 and T_2 is compatible with T_1 and T_2 having different semantic properties, contradicting 1.

5. T_1 and T_2 can only differ as to: (1) their structural properties or (2) the physical make up and embedding of the tokens that instantiate them. Hence,

6. If semantic properties supervene on some other class of properties, then

7. By 5. we must conclude that the semantic properties of T_1 and T_2 supervene on the physical makeup and embedding of their tokens.

Now, the efficacy of Putnam's argument is dependent on its capacity to overcome objections like the one presented above. The general strategy is the following:

8. The only way to impose constraints to the physical makeup and embedding of a token of system of symbols, is by introducing a description of these constraints.

The reason why conclusion 7. above seems to force the case for a naturalistic theory of semantics, is because of the occurrence of the expression: "supervene on their physical makeup and embedding". Now, The occurrence of this expression does not, itself, add any physical constraint to the system of symbols. Hence the following holds:

9. The uninterpreted description of these constraints does not carry, in itself, any information as to what it is a description of (it is, itself, compatible with unwanted interpretations of the whole system of symbols that preserve its truth).

It would then be of no use to point out that the description of the physical constraints must be placed at a higher semantic level than the symbol system that we wish to interpret. In fact, whatever semantic level or meta-level that description is placed at, what counts is that at that level the symbols are not themselves intrinsically interpreted, so that Putnam's argument can be made to climb up to meet the challenge. This leads to the apparent counterintuitive consequence that it is impossible (or meaningless) to impose physical constraints onto anything, if this means to impose physical constraints to the represented domain.

I think, however, that this further move is unwarranted, and that it descends from a hard-to-die familiar prejudice: the prejudice according to which syntactic properties are intrinsic to the symbol systems that instantiate (implement) them. As I have argued, syntactic properties, such as the relational property of isomorphism between symbol structures, far from being capable to survive without "real" semantic properties, presuppose them. So Putnam's argument can, at best, be considered as a reductio ad absurdum of an internalist conception of syntax.

Again, in fact, the skeptical conclusion is carried under the assumption that syntactic properties, such as those that a structure must possess in order for it to be the model of a theory, supervene on intrinsic properties of the structure and of the theory (a view that I call *syntactic internalism*). But theories, and structures, do not possess any syntactic property intrinsically, for syntactic properties, I have argued, presuppose semantic ones, and the latter are not intrinsic to either our theories, or to their models.

The skeptical conclusion of Putnam's argument, is carried under the assumption that a system of symbols (a theory), when abstracted from its physical manifestations (as it should, if we are to identify it at the appropriate level of analysis), only possesses "syntactic properties". These are assumed to be the only properties of a theory to "survive" through the steps of abstraction that it must undergo to climb the ladder from its physical manifestations (its tokens) all the way up to the Empireum where its individuation belongs. It is only natural, then, to presuppose that these syntactic properties would be intrinsic to whatever possesses them.

Having assumed syntactic internalism, if Putnam can manage to show that syntactic properties do not suffice for the determination of content, then he is right to conclude that no amount of theory could either, for any additional piece of theory would add nothing but more syntactic properties.

According to the view put forward in this work, instead, the relevant abstractions are grounded on externalist semantic properties. Thus, syntactic properties are only instantiated when certain *cross-semantical properties* (i.e. properties that are instantiated when facts obtain that appropriately relate representations with their representeds) are also instantiated.[18]

So Putnam cannot argue that "no amount of theory would do the job", for this presupposes (hence it cannot informatively entail), that no physical theory can be systematically interpreted (via the specification of relevant measurement procedures, as I have discussed), as referring to cross-semantical facts.

This analysis of Putnam's argument is in keeping with a more general objection

[18]Notice that the claim that the abstractions that ground syntactic properties require that semantic properties are in place, does not entail the (absurd) claim that syntactic properties supervene on some particular set of semantic properties. Using again our example, this is the same as pointing out that the fact that three-sectorness is not independent from the presence of the secondary properties (colors) that ground it, does not entail that three-sectorness is a property that depends, for its instantiation, on some particular set of colors.

that has been raised against it:[19]

> Only if the words that occur in the formulation of the L¨owenheim-Sk¨olem theorem [...] have the "intended" interpretation, could the latter be used to deduce what Putnam deduces. [...]. Obviously, we are not to deny the truth of the metatheorems that Putnam uses. But why should we assume that the metatheory is exhausted by them? In particular, why could we not see the move of the naturalist we considered in the previous section as a move at this level, namely as the proposal of supplementing the metatheory with the formula "x refers to y if and only if $R(x, y)$, or something similar? If we maintain that what our naturalist is doing is just adding more theory, we not only misunderstand, but also cheat him. After all, the formula contains a variable that ranges over expressions of the language and a predicate as "referring to: it therefore seems to be an assertion about the theory, and not an assertion of the theory. If so, it should have the same status as the metatheorems that Putnam uses in his reasoning.[20]

The following sections are dedicated to further specify the claim that syntactic properties cannot be intrinsic to what possesses them.

3.3.4 Notes on the semiotic vocabulary

Let us start from the bottom of the semantic ladder: markers. Some identifiable patterns of energy are interpreted as markers. Take the following, for example:

[19]More general in that it is not committed to my externalist construal of syntactic properties.
[20]Bianchi, [5], pp. 12-14.

P

It is recognizable as a letter of the alphabet. By virtue of what is it so? In other words, when we say that "the sign is type-identifiable as a token of letter P", what do we mean, exactly? Is it an objective matter of fact? Suppose a Greek reader contested that it is not a token of letter P, but a token of letter Rho. Could we argue conclusively that he is wrong? Of course not. The following four different interpretations of the claim that P is a token of letter P have been argued to coexist ambiguously within standard type-identifications of markers:

1. "P" has been interpreted by someone as an instance of letter P
2. "P" was intended by someone as an instance of letter P
3. There exists a linguistic community for which any instance of a pattern of energy equivalent to "P" is a token of letter P
4. "P" is in principle interpretable as a token of letter P

One step higher up on the ladder are words. We shall ask again by virtue of what does a complex of markers such as

PAIN

get assigned to a particular semantic type. Again there doesn't seem to be a non observer-relative way to assign PAIN to a particular type (A French speaker, for example, might contend that PAIN belongs to the same semantic type as BREAD).

Again, the following interpretations have been suggested:

1. "PAIN" has been interpreted by someone as meaning pain
2. "PAIN" was intended by someone as meaning pain
3. There exist a linguistic community for which any instance of a pattern of energy equivalent to "PAIN" means pain
4. "PAIN" is in principle interpretable as meaning pain

Finally, syntax, as well, shares the ambiguities of its fellows markers and words. Take for example the expression

$$CAT$$

Is it an instance of the same syntactic type as the predicate C(A,T), or is it of the same syntactic type as the unary predicate C(AT)? Or is it a constant in a theory of animals (CAT). The situation is perfectly symmetric as that described above: there are four possible interpretations of the sentence "CAT is of syntactic type S".

A first conclusion to be drawn from this analysis of the semiotic vocabulary is that marker-types, semantic-types and syntactic-types all require semantic capacities to be identified (any proper analysis of them must make irreducible use of mental-semantic vocabulary).

The above analysis is true of semiotic items, i.e. of symbols. Is it reasonable to suppose that it would not hold true also of internal, mental, symbols? A good deal of effort has been spent in trying to prove that mental, inner symbols, cannot be subject to an analogous observer-relativity.

The reason for this is that assuming that mental symbols and their structure and semantics are also observer-relative, would leave us wondering about what could explain the semantic and syntactic properties of the observer.

I, instead, argue that this apparent circularity can be bypassed by giving up syntactic and semantic internalism. It does not come as a surprise to my view, then, if we have concluded that computational models (when identified by their syntactic properties alone) make surreptitious use of semantic capacities. It should also not come as a surprise, then, that the pattern of failure of natural computation resembles so closely that of natural semantics: any theory (of computation or of semantics) that implicitly presupposes that syntactic properties are intrinsic to their bearers, I argue, has to face the nasty choice between being circular (presupposing what it purports to explain) or falling victim of the isomorphism catastrophe. The vacuousness arguments discussed in this chapter, then, can be thought of as a reductio ad absurdum of syntactic internalism.

Secondly, the fact that syntactic types, just as semantic types are observer-relative, provides further ground for the claim that the naturalization of computation must proceed from the naturalization of intentionality. It is in fact clear that if the "honest model" for a formal theory is not individuated by an overt deployment of semantic capacities (such as the preliminary assignment of cardinality or the artificial restriction of the universe), it must be individuated by covert ones. This is exactly how we manage (ideally) to build our computers and softwares following purely syntactic recipes: by putting at the disposal of these systems our semantic capacities.

Finally, the analysis corroborates a claim anticipated at the end of the previous section: syntactic facts, such as the property of sharing the same structure or being the model of a theory (not necessarily the intended model), presuppose, for their intelligibility, semantic properties, hence they cannot ground it. Without the deployment of semantic capacities, in fact, two systems of symbols are not only incomparable as to their structure, but are not even type-identifiable as systems of symbols.

Whether something is or isn't the model of a given theory, therefore, is not a property of that theory, unless we understand the expression "that theory" as referring to an abstraction that presupposes, to be safely and systematically performed, that semantic properties are deployed and maintained constant.

So, strictly speaking, the model-theoretic argument, if assumed to be correct, would only show that semantic properties do not and could not supervene on structural properties alone, on pain of contradiction. This is not to say that Putnam's model-theoretic argument should not be taken care of, for I think that it points at the need for a better understanding of the notions of truth, reference and reality.

For the purposes of this work, however, all that matters is that (1) Putnam's v-argument and his model-theoretic argument need not be accepted or refuted together, (2) that the argument can be used to object only to internalist theories of intentionality, and, most importantly, (3) the argument can be thought of as a reductio ad absurdum of syntactic internalism.

3.3.5 The encodingist paradigm

It is interesting to ask what has obscured this simple fatal objection to internalism. I believe that what has obscured the untenability of syntactic internalism is an internalist preconception about the nature of semantic properties. In fact, if semantic properties supervened on syntactic ones, then syntactic internalism would be viable: syntax would still presuppose semantics, but the latter would supervene on the former, thus making logical space for syntactic internalism.

Most theories of semantics, like most theories of implementation (for example the state-to-state correspondence view that we have criticized) adopt what has been termed the *encodingist paradigm*: the view that cognition consists of encodings and of operations on them.[21] The adoption of this paradigm, I argue, is what obscured the circularity of most attempts to naturalize computation and intentionality.

By and large, we conceive of our epistemic contact with the world as being mediated by our senses. These are thought to "encode" information about the environment that is thereby made available to the mind for processing and planning of further epistemic contacts.

This simple picture constitutes the basis for most of our current scientific understanding of cognition and semantics. What does it mean to "encode" information? Intuitively, it means to transform (transduce) information into a format that is practical to manipulate, while preserving the possibility to unpack, decode it whenever is needed.[22] One simple way to analyze what "encoding" means is to think of them as "stand-in"'s. For example "..." stands-in for "S", in Morse code.

According to this notion, encodings are systematic transformations of representations. As such, they require that representations to be stood-in for (e.g. "S" in the Morse code) be already in place. Alternatively, factual, possibly nomic correspondences, can be taken as defining encodings. So, for example, we often hear that some neural activity encodes some property of the light that hits the retina. Yet there is no one there to know that those properties are in factual correspondence with those neurons. To say that they "encode" those properties leads to an equivocation.

The same light that hit the retina might permanently impress a film in a camera,

[21]The notion of *encodingist paradigm* and a detailed discussion of its shortcomings has been proposed by Bickhard ([6], [7],[8] and [9]).

[22]Familiar notions of encodings are the Morse code, computer codes, stenographic codes, G¨del numbering, etc. See Bickhard [6], [7],[8] and [9] for an extensive treatment of the notion of encoding.

establishing factual correspondences with the film just as it does with a population of neurons. In this case, though, we would not be ready to say that these properties are "encoded" in the film. For there to be an "encoding", or anyway, for a factual correspondence to have any epistemic value, these correspondences must be known; and for them to be known we need to have a representation of both what is to be coded and of what is to stand-in for it. In other words, factual correspondences don't carry any information about what these correspondences are with. Failure to appreciate this shortcoming left many causal theories of semantics wanting for an account of misrepresentation or, circularly, led them to deploy ready-made semantic properties.

The encodingist myth, moreover, prevented to appreciate the fact that isomorphisms, per se, are totally uninformative as to what they are isomorphisms with. Because of their pattern of failure, internalist (encodingist) theories of semantics can be thought of as a naturalized-intentionality counterpart to state-to-state correspondence theories of implementation.

In both cases the surreptitious deployment of semantic properties consists in supposing that our semantic capacities, that are totally transparent to us, and that serve the purpose of individuating the intended models of our theories, can be "transferred" to our symbol systems "for free", i.e. without transferring also the physical states of affairs that ground our semantic capacities and, through them, the syntactic properties of the systems of symbols that we produce. It should be clear then, why with this treatment we are trying to make explicit the (semantic) job that was previously implicitly taken for granted.

The assumption that syntax is among the intrinsic properties of representations, a commitment that I have termed "syntactic internalism", forces one to implicitly buy an encodingist construal of semantics. To see how things are so, consider the following claim made by Fodor in his book "*The Mind doesn't Work that way*":

> Syntactic properties of mental representations are ipso facto essential (because the syntactic properties of any representation are ipso facto essential).[23]

This is a clear endorsement of syntactic internalism. In a note to this claim, Fodor further argues that it is uncontroversial among those who advocate a syntactic theory of the mind. Those who contend that syntactic properties are not essential to representations (the connectionists, for example), Fodor observes, reject altogether the claim that mental properties supervene on syntactic ones, and fall thus outside the number of those who advocate a syntactic theory of the mind.

Now, my claim that syntactic properties are not intrinsic to representations is not meant to reject the claim that mental properties supervene on syntactic ones. The proposal put forward here, in fact, is committed to what we may call *syntactic externalism*: a view that, if viable, would strictly contradict the claim mentioned above.

So, let us take a closer look at the reasons Fodor puts forward for his claim. In the same note, he continues with what he apparently takes to be a quasi-reductio ad absurdum of any view that rejects the claim that syntactic properties are essential to representation while retaining a syntactic theory of the mind. Consider a representation whose content is that "*John Loves Mary*". Imagine that someone, Fodor argues, contended that the fact that it is a complex representation (i.e. that it is constituted

[23]Fodor [37], p. 25.

by the simpler elements *John*, *Loves* and *Mary*) be not among its essential properties. He, or she, Fodor argues, might

> prefer not to recognize that the "John" that appears in "John loves Mary" is the same word as the "John" that appears in "Mary Loves John". [...] You might then wish to embrace the view that English contains two different words "John" - as it were, "John-subject-of-a-verb" and "John-object-of-a-verb" - both of which are spelled "John" and both of which mean John. This analysis might reasonably be considered perverse; clearly it flouts plausible intuitions about the individuation of English words.[24]

Clearly Fodor thinks that if one gives up the idea that syntactic properties (for example constitutivity) are intrinsic to the representations that bear them, then either (1) one also rejects the claim that mental properties supervene on syntactic ones, like connectionists do[25], or (2) one is forced to accept the absurd proliferation of words described above.

This, I argue, reveals Fodor's implicit commitment to the encodingist paradigm. The argument for the absurd proliferation of words, in fact, can only proceed from the assumption that the individuation of words supervenes on some intrinsic property of them, as encodings do. Otherwise, how could he conclude that "John-subject-of-a-verb" and "John-object-of-a-verb" would (absurdly, I agree) stand in the need for two different words?

Syntactic externalism (the view that I advocate), instead, has it that the claim that mental properties supervene on syntactic ones and the claim that syntactic properties

[24]Fodor [37], p. 108.

[25]See section 5.5 for a discussion of the constitutivity of representations in connectionist models.

are not among the essential properties of representations need not be contradictory. In order to naturalize the job done by the implicit observer in standard accounts of implementation, then, we shall have to look at theories of intentionality that abandon the encodingist view of semantics, on pain of circularity. This will allow us to make explicit the view that I have called: syntactic externalism.

3.3.6 Varieties of syntactic externalism

The thesis that syntactic properties should be better understood as extrinsic, relational properties of their bearers stands in need for some further clarification. As a general statement, it would not sound new to the philosopher of mind. There are views in the literature, in fact, that bear some apparent resemblances with the doctrine that I have called *syntactic externalism*. The following is a typical example:

> Mental state tokens are brain state tokens. But the properties by virtue of which mental state tokens are classified into syntactic categories are not intrinsic features of those brain states; they are not features which depend exclusively on the shape or form or "brute physical" properties of the states. Rather, the syntactic properties of mental states are relational or functional properties - they are properties that certain states of the brain have in virtue of the way in which they causally interact with various other states of the system.[26]

Stich calls this understanding of the syntactic properties which computationalism ascribes to mental states: *fat syntax*. This view of the syntactic properties of brain

[26]Stich [88], p. 310. Another example of such broad individuation of computational properties is Wells' interactive approach (Wells [94]) discussed in section 6.6.2.

states is clearly externalist. Notice, however, that it has nothing to do with what I have called *syntactic externalism* (apart from being compatible with it).

Firstly, it is not a view of syntax: it is a view of the syntactic properties attributed to brain states by a computationalist theory of the mind.

Secondly it does not question the assumption that isomorphism suffices for the instantiation of syntactic properties. It simply excludes that such isomorphisms exist that capture generalizations about the workings of the mind and those of the brain alone.

This is certainly in keeping with our analysis. But it is only the first step to be taken. I have argued, in fact, that isomorphisms, per se, are incapable in principle to ground the relevant abstractions, whether these are construed as pertaining to states of the brain alone (the "skinny standard of individuation"), or according to a "fat" standard of individuation.

The view of implementation that I am advocating here endorses, as a necessary further requirement, the claim that an externalist theory of content (a teleological theory) is needed to ground (broadly) syntactic properties.

My arguments in favor of external syntax, in fact, are not meant as a criticism to a particular theory of the mind, but to an internalist, syntactic construal of implementation tout court (regardless of what properties are suggested as a basis of instantiation for syntactic ones). External syntax - this is how my semantic view sounds like in a slogan - meets broad content.

I am particularly fussy about marking the distance between my syntactic externalism and other, theory-specific versions of it, because I believe that failure to appreciate the difference would endanger the plausibility of my proposal. This concern is warranted by comments like the following:

> I will not defend an individualistic version of the functional view of computation, however, because I endorse a wide (non-individualistic) construal of functional explanation. For present purposes, it is important to distinguish between wide individuation and individuation based on wide content. Individuation based on wide content is one type of wide individuation, but wide individuation is a broader notion. Wide individuation appeals to the relations between an entity and its environment regardless of whether those relations are of a kind that warrants content ascription to the organism. For my purposes, of course, what is needed is wide individuation that does not appeal to content.[27]

Also Stich, we have seen, makes the case for a wide individuation of syntactic properties, but, like Piccinini, he argues for a brand of externalism that does not entail "wide individuation based on wide content". I, instead, have made the case for wide individuation based on wide content. Failure to appreciate the difference, could lead to the following misconceiving objection.

The arguments proposed here could be thought of as successfully making the case for a broad individuation of syntactic properties in theories of the mind. But they could be argued to fail to make the more specific case for a broad semantic individuation of computational properties in real dynamical systems.

"Fine!", someone could object to my proposal, "you showed us what we already suspected: mental states are not narrowly individuated. Semantic properties, let us

[27]Piccinini [68], p. 13.

grant it for the argument's sake, are a way to individuate them. But there is plenty of other ways to individuate broadly these properties: why semantics?".

The arguments I have provided, I think, prove that there is no other appropriate way to broadly individuate syntactic properties but by having them grounded on broad semantic ones. This is the view that I have called *syntactic externalism.* In other words, the trouble with syntactic internalism that I have argued to be intractable, does not depend on some specific application of it to a given domain. Given the generality of the problem, no simple (non-semantic) extension of the supervenience base of syntactic properties would save them from having unwanted models.

The reason for this, it should be clear, is that the extension that I think I have argued to be necessary, cuts across two different semantic levels: the level of representations, and the level of the represented domain. An extension of the supervenience base that comprises properties that are "external" to the implementing system, but that does not cut across the two semantic levels (Stich's *fat syntax*, for example), would be just as hopeless as an internalist picture, as far as blocking the isomorphism catastrophe is concerned.[28] That things are so, can be intuitively appreciated by considering again the example of colored flags and three-sectorness. If we conceded that three-sectorness must be understood as supervening on relational, external facts, but we did not require that these facts suffice for color perceptions to be instantiated, our externalist move would be useless: number three, would still be wanting in the picture.

[28]The arguments pointing at the incapacity of isomorphisms to rule out unwanted models, in fact, do not depend on the choice of some particular domain of instantiation: they hold valid, if they are valid, even if one urges that syntactic properties should be individuated broadly. A "long arm" supervenience basis, in other words, is compatible with unwanted instantiations just as much as a narrow one.

My arguments, as I have already mentioned, proceed from the assumption that v-arguments are sound. I have provided some reasons to suppose that things are so, but ultimately, if someone thinks that v-arguments are mistaken, he or she would find my conclusions unconvincing. But even if, parting from my personal convictions, we suspended the judgement on the validity of v-arguments, I could still claim that, given the total lack of agreement in the literature, assuming them to be sound, is no riskier than assuming them to be wrong.

Moreover, while I believe to have successfully argued that if v-arguments eventually prove to be sound, a semantic view of implementation will be forced on us, there is no convincing argument to the effect that, if v-arguments will prove to be all wrong, this would entail that my view would thereby be refuted (although I agree that it would lose much of its allure).

Chapter 4

Teleological theories of implementation: blocking the isomorphism catastrophe

4.1 Introduction

I have argued that *syntactic internalism* is to be held responsible for the isomorphism catastrophes of orthodox theories of computation (as well as of isomorphism theories of semantics).

I have also argued that the surreptitious deployment of intentional capacities has often obscured the unwanted proliferation of the models.

Without further specification, however, this analysis does not appear to be of any interest. Knowing that we are to abandon the internalist view of syntax, in fact, does not help us to imagine what we should replace it with. Of course, for a mere terminological reason, we should expect to embrace an externalist view of syntax, and this is precisely what in this chapter I shall argue we should do. But what does that mean exactly? And how are we to proceed? Let me start with an analogy that I discussed in the general introduction to this work, and that I believe will help our intuitions on the issue.

Notoriously, colors are properties that, contrary to popular belief, do not supervene on intrinsic properties of their bearers. A "red" apple does not reflect "red light", and it is misleading to think of things that we see as objectively colored at all. Rather, the apple simply absorbs light of various wavelengths to different degrees, in such a way that the unabsorbed light which it reflects is perceived *as* red. An apple is perceived to be red only because normal human color vision perceives light with different mixes of wavelengths differently, and we have language (concepts) to describe that difference. One may formally define a color to be the whole class of spectra which give rise to the same color sensation, but any such definition would vary widely among different species and also to some extent among individuals intraspecifically (also depending on environmental factors).

To understand which particular color perception will arise from a particular physical spectrum requires knowledge of the physiology of the retina and of the brain.

This does not show, however, that colors enjoy a kind of second-class ontological status. Mental properties are trivially "not independent of us", but are not thereby less qualified to be part of reality. Moreover, although the neurophysiology of color perception is not yet perfectly understood, it is compatible with the above considerations that colors supervene on purely physical, albeit non-intrinsic (relational) properties.

Now, imagine a flag partitioned into three differently colored sectors. The three colors are yellow, green and blue: they are adjacent in the spectrum. Our language, and intuition, mislead us into saying that the flag *is* composed of three different sectors, rather than saying that it *looks* composed of three sectors. Following this intuition, under physicalistic assumptions, we would hypothesize that this syntactic property (that of being a three-element compound)[1] supervenes on some isomorphic intrinsic property of the flag.

Suppose, then, that we decided to chose three wavelength bands as a model for the supervenience base of our internalist theory of "three-sectorness". Someone, let us call him Hilary, could object that, whatever choice we have made, there will be unwanted models for the same property (i.e. other wavelength bands that satisfy the same property). As a matter of fact, even a perfect continuum of wavelengths would satisfy the property, for the flag would still appear segmented into a yellow, a green and a blue sector to a normal human being.

Is Hilary entitled to draw the conclusion that the properties of our perception of that flag do not, at least in part, supervene on some isomorphic physical property? Of course not. Hilary can only argue that the properties of our perception do not supervene on some intrinsic syntactic property of the flag.

But the number three that features in the proposition "*the flag is composed of three sectors*", must be correlated (mapped) to some (relational) physical property of the light reflected and of our cognitive dynamical system. *Number three*, so to speak, must feature somewhere in the description of this relational property (or set of properties). Resorting to this *number three* is a legitimate explanation of the fact that we see *three* sectors. Thus we do not have to abandon the idea that the flag is syntactically structured in a certain way: we just have to give up the hope to explain such structure by resorting to an isomorphism with intrinsic properties of the

[1]In this introduction the expression "syntactic property" is to be understood in a broad sense, as meaning any property that holds true regardless of what particular semantic properties are instantiated.

perceived object.

This account of the constitutivity of the flag is an example of syntactic externalism. Neither must we give up the idea that syntactic properties are independent from their implementations. A different flag, say red, white and green, would share the syntactic property of being three-sectored with the previous one, by virtue of instantiating isomorphic relational properties. It remains true, however, that under a proper treatment of colors, the number of sectors of a flag is "observer-relative". Such observer relativity, however, is safely implemented by physicalistic, externalist properties alone.

Let us now turn to our understanding of implementation. I have argued that we are to abandon the internalist view of implementation in favor of what I call: *computational externalism.* I have proposed that intentional properties should ground the notion of implementation. Similarly, colors ground the notion of instantiation of three-sectorness. I have also suggested that in order to uphold a syntactic theory of implementation, hence of the mind, we shall have to abandon internalist theories of semantics. Likewise, an internalist theory of colors (i.e. the false view that colors supervene on intrinsic properties of their bearers), is not compatible with the externalist explanation of three-sectorness proposed above.

The first section of this chapter (sec. 4.2) discusses a class of theories of semantics that comply with the above desiderata: *teleological theories of content.* Some of these externalist proposals (those discussed in section 4.2) make irreducible reference to historical, evolutionary concepts. As this would have consequences for a theory of implementation that some might find unpalatable, I also discuss a non-etiological version of these theories (section 4.3). I do not wish to commit to the viability of these theories of content. It suffices, for my purposes, to argue that if one of these proposals would prove to be viable, then it could be used to uphold a realist stance about computation, hence to salvage computationalism from skeptical arguments.

Teleological theories of content are applied to a concrete example of intentional implementation (the implementation of a logical gate), thus providing a first informal application of computational externalism to a concrete example (section 4.4.1). The example shows that the notion of implementation has been successfully immunized from vacuousness arguments.

Finally I make a concrete, formal proposal of an externalist honest labelling scheme for the implementation of finite state automata (section 4.4.2). The general theory of implementation proposed is shown to successfully block unwanted models, while upholding the computational sufficiency hypothesis (notwithstanding the assumed validity of v-arguments).

4.2 Teleological theories of semantics

4.2.1 General concepts

When a proposed solution to the problem of naturalizing intentionality doesn't avail itself surreptitiously of semantic concepts, it appears to be unable to rule out unwanted instantiations. In our terminology, this amounts to saying that the proposed physicalistic interpretations for the sentence $Ref(S, P)$ fall victim of the isomorphism catastrophe, or fail to achieve sufficiency. Of course, we could again say that not all models of our theory of semantics are "honest", but this would be a solution only if we could specify (using physicalistic concepts alone) what counts as an "honest" model. Teleological theories of semantics, we shall see, pretend to have done so.

The “trick” teleological theories of semantics adopt, is to make “honest” models depend on the notion of “proper functioning”.

> These theories generally begin with some more basic theory of the relation between a true thought, taken as embodied in some kind of brain state, and what it represents, for example, with the theory that true mental representations co-vary with or are lawfully caused by what they represent, or that they are reliable indicators of what they represent, or that they “picture” or are abstractly isomorphic, in accordance with semantic rules of a certain kind, with what they represent. The teleological part of the theory then adds that the favored relation holds between the mental representation and the represented when the biological system harboring the mental representation is functioning properly, that is, functioning in accordance with biological design, or, perhaps, design through learning.[2]

Teleological theories, then, do not (qua teleological theories) propose a solution to the problem of intentionality that is in opposition with other, alternative solutions. They rest on specific theories of the relation that must obtain between a system S and a true proposition P for $Ref(S,P)$ to be the case, adding what conditions must obtain for $Ref(S,P)$ to be the case when P is not true. The conditions added by a theory, to qualify it as a teleological theory, must make irreducible reference to some naturalized notion of purpose, or function.

Borrowing Millikan’s words to express this common feature of teleological theories, we would say that a model for $Ref(S,P)$ must determine what would have had to

[2]Millikan [60], p. 4.

have been the case with the represented (P) or the world had the system (S) produced or harbored that same representation when functioning properly.

Before turning to specific proposals, it is worth spending a few words on the notion of "proper function". Teleological concepts are concepts that make reference to some ends or goals, and these are intentional concepts, and as such don't seem prima facie to be suitable for a naturalization of intentionality. The notion of function deployed in teleological theories, however, is not itself purposive. In most cases the notion of function deployed is "borrowed" from biology. When we say that the function of a heart is to pump blood, for example, we are claiming that it is "adapted" to pump blood. When a heart is pumping blood we say that it is doing so because it was "designed" to do so, but we are not implying that such design is purposive, because we believe the process underlying such "design" to be driven by a mechanical, purposeless, non-intentional force: natural selection.

> The concept of a biological function is defined in terms of natural selection (Wright [[97]], Neander [[61]]) along the following lines: it is the function of biological system S in members of species Sp to F iff S was selected by natural selection because it Fs. S was selected by natural selection because it Fs just in case S would not have been present (to the extent it is) among members of Sp had it not increased fitness (i.e. the capacity to produce progeny) in the ancestors of members of Sp.[3]

Although, as we shall see, not all teleological theories derive their notion of function from natural selection, whatever notion of purpose they deploy, they will have

[3]Loewer [52], p. 9.

to make sure that such notion is not "literally" purposive (i.e. that it does not surreptitiously rest on intentional properties).

Teleological theories can be thought of as a general strategy to solve the problem of intentionality that consists of two steps: 1) the indication of what is the relation between a true representation (some item, state or feature (R) of a system (S)) and its represented (P), and 2) the claim that it is the proper function of S to produce such relation. In our formalism the teleological strategy consists of:

1. The specification of a model $< S, L >$ for Ref(S,P), when P is true, and

2. The claim that it is the proper function of S to instantiate $< S, L >$

Accordingly, teleological theories can be classified by the various proposed models for Ref(S,P), when P is true.

4.2.2 Indicator semantics

In 1981 Fred Dretske wrote a very influential book (Knowledge and the Flow of Information[4]) where he proposed a (non teleological) causal theory of content that he later rendered teleological[5] on the lines of the above characterization of "teleological theory".

His original idea was that an item R represents P if R "indicates" P, where R *indicates* P iff (if there is an R, then P).[6]

This understanding of intentionality, we have seen, is not viable, for it entails that if S represents P then P must be the case. If we want to make room for error, it

[4]Dretske [24].

[5]See Dretske [26] and [27].

[6]Note that it is not necessary that P directly causes R, for they could have a common cause. Moreover, says Dretske, neither is it necessary that *if R then C* be nomological (universal), for it is sufficient that it be true locally.

must be possible that S represents P and P is not the case. Dretske then applied a teleological strategy requiring that S represents P iff its function is to indicate P. So the idea is:

1. If P is true, Ref(S,P) is true just in case there is an item (a state, for example) R of S such that *if P then R.*
2. R has the function of indicating P.

The "infallible" notion of indication (if P then R) has been made compatible with representational error by making it the function of the representation to indicate its represented. As items happen to not function properly, the teleological move has made room for error.

> The fundamental idea is that a system, S, represents a property, F, if and only if S has the function of indicating (providing information about) the F of a certain domain of objects. The way S performs its function (when it performs it) is by occupying different states s_1, s_2,...,s_n corresponding to the different determinate values f_1, f_2,...,f_n, of F.[7]

A Problem for indicator semantics: functional indeterminacy

There is a species of marine bacteria that use internal magnets (magnetosomes) to direct themselves. Magnetosomes align parallel to the earth's magnetic field. As the field lines in the northern hemisphere point downwards (towards the magnetic north) the bacteria, in the northern hemisphere, direct themselves accordingly. Apparently the "function" of this awkward means of orientation is that of keeping the bacteria

[7]Dretske [28], p.2.

away from oxygen-rich surface water by directing them towards more oxygen-free (hence less toxic to them) waters. In other words, the function of magnetosomes is that of prompting the organism to engage in a surface avoiding behavior. If a magnet oriented in the opposite direction as that of the earth magnetic field is placed near the bacteria, or if northern bacteria are placed in southern waters, they direct themselves towards death.

If we were, as Dretske puts it, looking for "nature's way of making a mistake", we have found one. The marine bacteria, we said, can be "tricked" into lethal environments by using artificial magnetic fields.

It seems natural to say that the function of magnetosomes is to direct the system towards oxygen-free sediments, and that sometimes the magnetosomes are not in the conditions to perform their proper function. If magnetosomes have the function of indicating oxygen-free sediment (as opposed to lethal, oxygen-rich surface), then the magnetosome represents oxygen-free sediment, under our theory of representation. When the bacteria are fooled to their death, they misrepresent. However, we could as well say that the magnetosomes have the function of indicating the direction of geomagnetic north. Worse still, we would be legitimated to say that the function of magnetosomes is to indicate the local magnetic north, in which case we could not count as "error" the case of tricked bacteria.

More generally, the problem is that if an inner state of a system S has been selected for indicating a distal feature (P) of its environment, it will also have been selected for indicating the more proximal features that carry information about P. The indeterminacy of the proper indicating function of an item conflicts with the determinacy of representational content, thus threatening the informational version of teleosemantics. This is an instance of a class of problems that go under the name of: *functional indeterminacy problems.*

Dretske's solution[8] has been to amend his theory by adding that a system is capable of representing a distal feature of its environment only if it is also capable of learning any number of alternative epistemic routes to that same distal feature. As this is not true of all disjunctions of more proximal stimuli characteristic of a given epistemic route, it is argued, this blocks the indeterminacy argument. The solution has been criticized[9], and partially abandoned by Dretske himself.

4.2.3 Benefit and Consumer-Based semantics

In 1984 both David Papineau and Ruth Millikan proposed teleological theories according to which the content of a representation is determined by the uses to which it is put. It is therefore the "users" of representations and their proper functions, that will be relevant for determining their content. There are at least two a priori reasons for concentrating on the users of representation.

For one, teleological theories all rest on the notion of teleological functions, and these are selected effects. So if we are interested in the "effects" of representations we should look at the consumers of representations, for they are the ones most directly concerned by them.

Secondly, something qualifies as a representation only if it is used as such. In fact, if whatever fixes the content of a representation was not determined by its use, something could count as a representation without representing anything, which, according to Millikan, is nonsense.

[8]Dretske [26].
[9]Loewer [52].

Papineau's theory

Papineau's idea is to ground the normative properties of intentionality on the normative properties of "desires". A representation can be true of the world or not like a desire can be satisfied or not. We are, as usual, trying to express the truth conditions of Ref(S,P). The general idea of teleological theories, we have seen, is to ground the intentional relation on some notion of (typically biological) function. According to Papineau, if a desire D, a belief B (a certain state of S, for example) and a proposition P (a state of affair) are such that the obtaining of P is sufficient for an action based on D and B to satisfy D, then the content of B is P.

This provides for naturalistic truth conditions of Ref(S,P) only if it is possible to naturalize the notion of satisfaction of a desire (which is, as it stands, a semantic notion). Papineau's notion of "satisfaction of a desire" is, roughly as follows: if q is the minimal state of affairs such that it is the biological function of D to operate in concert with beliefs to bring about q then D is the desire that q.

Here "minimal states" are meant to avoid indeterminacy. In fact, suppose that the desire D is meant to be the desire of a frog to eat a fly. As "eating a fly" is co-instantiated with other states of affairs (such as "opening the mouth"), we could say that, by the same token, the content of the desire of the frog is opening its mouth. Papineau thinks that only the most specific of these possibly co-instantiated behaviors (i.e. eating a fly) counts as the content of the desire, for there have been occasions when D was selected for even if it did not cause the frog to open its mouth.

Millikan's theory

Millikan proposed, over the past two decades, a Consumer-Based theory of semantics according to which the content of a representation is determined by the performance of the proper functions of its consumers. Proper functions, as Millikan has it, are "*effects which, in the past, have accounted for selection of its ancestors for reproduction, or accounted for selection of things from which it has been copied, or for selection of ancestors of the mechanism that produced it according to their own relational proper functions*".[10]

Millikan argues that if the (proper) function of a system is to produce representations, then these must function as such for the system itself. She then suggests that the system should be conceived as made of two parts: a part that produces representations and one that consumes them.[11] The consumer part is thought to be responsible for a candidate representation actually acquiring the status of representation, and for fixing its content.

But how can we spell out what it is for a part of a system to act as a representation consumer? Intuitively, a system uses an internal item as a representation if its proper response to it can be fully explicated only if certain conditions obtain in the environment. In other words, a system uses an item as a representation if the response to it that was naturally selected served its adaptive function depending on certain conditions obtaining in the environment. Secondly (this definition alone

[10]This is how Millikan defines the notion informally in [57], p. 3. A more detailed treatment of the notion of proper function can be found in the first two chapters of [58]. Like most teleological theories of intentionality, Millikan's theory favors an etiological notion of function, according to which the function of an item is determined by the history of selection, or by the past selection of the items of that type.

[11]The consumer and the producer of a representation can be different systems or different time slices of the same system (before and after the representation has been tokened). This allows us to treat inner and external representations as tokens of the same type.

would be too liberal), these conditions must vary depending on the form of the representation. That is, changes applied to the form of the representation (mathematical transformations applied to it) must correlate to changes in the represented conditions according to specifiable rules.[12]

The function of the producer part is, in this view, to produce signs that are true relative to the consumer reading of them. Any indication or information about the environment that these signs might carry is irrelevant if the consumer part does not read them as such.

> First, unless the representation accords, *so* (by a certain rule), with a represented, the consumers normal use of, or response to, the representation will not be able to fulfill all of the consumers proper functions in so responding - not, at least, in accordance with a normal explanation. [...] Second, represented conditions are conditions that vary, depending on the form of the representation, in accordance with specifiable correspondence rules that give the semantics for the relevant system of representation.[13]

An example that illustrates what Millikan has in mind is that of honey bees. Honey bees, notoriously, perform dances to indicate to other bees the location of a source of nectar. This is how Millikan's theory of representation would apply to the example. First, as we discussed, representing systems consist of a producer and of a consumer part. Interestingly, in the example the producer part and the consumer part are not confined within the same organism (unlike the case of bacteria). In fact,

[12]See section 6.5.8 for a discussion of this feature of intentional icons as applied to my treatment of implementation.

[13]Millikan [59], p. 224.

the producer part is located in the dancing bee, while the consumer part is located in the other nectar-seeking bees.

Second, in order for the dance to count as a representation, it must satisfy the first condition above. Unless there is a factual correspondence between the particular features of the dance and the location of the nectar, the other bees will not find the nectar (thus failing to fulfill their function in responding to the dance). Sure, even if such correspondence was not in place, the bees might find some nectar on their way, by accident. This is why the condition has been made conditional on normal explanations. Third, varying tempos of the dance, and angles of its long axis, correlate with the distance and direction of the nectar. This satisfies the second condition above.

The central concept of Millikan's theory is that of *indicative intentional icon.* These are (physical) structures that "stand midway" between producer and consumer mechanisms (specifiable in terms of biological function). Most semantically evaluated items, such as indicative sentences in natural languages, thoughts, bee dances or beaver splashes are thought to deploy intentional icons. The idea is that the consumer mechanisms modify their activities responding to the state of the icon in a way that only leads systematically to the performance of the consumers' biological functions if a particular state of the world obtains. Such state is the content of the representation (of the icon in that state). The icon is supposed to map onto the world according to a rule or a function (we shall later discuss what kind of mathematical function). Given how the consumers respond to the icon the function is such that if the represented is in state s_1, then the icon is supposed to (in the biological sense) be in corresponding state i_1.

Notice that, as with all other teleological theories, Millikan's does not claim that the intentionality of a representation is determined by the proper function of the consumer: that a given icon represent a fact "*is not determined by their proper functions[...] It is a matter of HOW this fact-representation performs whatever function it happens to have*"[14].

The naturalization of the notion of "proper function" proposed above represents a significant departure from previous attempts. In fact, rather than grounding the intentional relation in a way that excludes the observer, it naturalizes the observer by naturalizing the consumer function of a representational system. "*Teleological theories of content*", says Millikan, "*thus separate "intentional" signs and representations, those capable of displaying Brentano's relation, quite sharply from natural signs. Even when intentional representations are true, neither the fact that they represent nor what they represent is determined by any current relation they actually bear to their representeds. The representational status and the content of the intentional representation are both determined by reference to its natural purpose or the natural purpose of the biological mechanisms that produced it, and these purposes are determined, it is typically supposed, by history, by what these mechanisms were selected for doing, either during the evolution of the species or through earlier trial and error learning*".[15]

It is possible, I shall argue in the following pages, to adapt the above treatment (or any theory of intentionality that has the same consequences) to a framework for computationalism that comprises the naturalized observer as an essential part.

[14]Millikan [57], p. 8.
[15]Millikan [60], p. 3.

Teleological isomorphism theories

Millikan often claims that her theory is meant to be an isomorphism theory. For example, we have seen how her theory grounds intentionality on the notion of natural purpose. Natural purposes, however, are not a trademark for intentionality: body organs have natural purposes, behaviors copied by other's behaviors can have proper functions, artifacts copied from earlier exemplars (because of the effects the latter had) can have natural purposes. Millikan clearly states that natural purposes become associated with intentionality only when they are "explicitly" represented. The relevant notion of representation, says Millikan, is kin to the mathematical notion of representation.

> According to the mathematical notion, a structure consisting of a set of abstract entities along with certain designated relations among them is said to represent another such structure if it can be mapped onto it one-to-one. Similarly, an intentional representation corresponds to the affair it represents as one member of a whole set of possible representations. These bear certain relations to one another such that, ideally, the whole structure maps one-one onto the corresponding structure of possible representations. [...] The forms of the representations in the system vary systematically according to the forms of the affairs it is their proper function to bring about.[...] The explicitness of these representations of natural purposes results from contrast with alternative purposes that could have been represented instead by contrasting representations in the same representational system.[16]

[16]Millikan [57], p. 5.

Notice that this notion of isomorphism is more plausible than standard notions, in at least two respects. According to an isomorphism theory (any one), representation amounts to the preservation of similarities and differences such that the structure of relations in the represented domain is "mirrored" by the structure of relation in the representing domain. Remember for example how the notion of implementation ($Imp(S, A)$) was thought to be grounded on the notion of mirroring. The causal structure of S was (allegedly) mirrored by the formal structure of A. Moreover, the relations that formed the two isomorphic structures in that case, were first order relations, i.e. the causal structure of S was supposed to be identifiable by "intrinsic" properties of S alone: we have called this assumption the "internalist" theory of implementation. According to the present version of a "resemblance theory", instead, the relevant resemblances are second order: they are themselves relational.

Secondly, it appears (and this will be crucial for our discussion) that teleological isomorphism theories have an advantage over crude isomorphism theories. It is in fact often argued that isomorphism theories are vacuous. The general objection is that "everything maps on to everything by some rule or other". The advantage of teleological theories is that they allow for a natural distinction between "relevant" mappings and "irrelevant" mappings, the "relevant" (honest) mappings being those that the system was designed (in the non intentional sense specified above) to use (in the sense in which a user part of a system uses a representation).

> An appeal to teleological functions can be combined with various ideas to form hybrid theories. [...] it's worth mentioning that such an appeal can also be combined with isomorphism theories (e.g. Cummings 1996),

> If we combine the idea that representations are isomorphic with their representeds with the idea that psychosemantic norms depends on the norms of proper functioning, we can generate several proposals: for example, the proposal that the relevant mappings are those that the systems were designed to exploit.[17]

Notice that this, if true, would allow us to block the isomorphism catastrophe, for unwanted models would be ruled out as mappings that the system under study has not been designed to exploit.

4.2.4 Fodor on misrepresentation

The introduction of teleological concepts was an attempt to solve, among other difficulties, the so called "disjunction problem"; this is exemplified by a red face (candidate representation) being caused by several, disjoint, factors, without any means to point at one as the intrinsically "intended", "honest" content. If X is the intended content of a representation R, and a Y happens to cause R, how can we ground the notion that X is the "correct", "intended", "honest" content of R, while Y is misrepresented by R? That is, what prevents us from saying that the content of R is really "*X or Y*", thus making error inconceivable? Fodor contends that teleological accounts are not a viable solution.

> [...] The appeal to teleologically Normal conditions doesn't provide for a univocal notion of intentional content. [...] It's just not true that Normally caused intentional states ipso facto mean whatever causes them. So we need a non-teleological solution of the disjunction problem.[18]

[17]Neander [62], sec. 3.
[18]Fodor [36], p. 230.

Fodor's idea is to account for representational error by a resort to asymmetric dependencies. Incorrect representations (such as a horse being mistakenly seen as a cow) depend on correct instances (such as a cow being seen as a cow) in a asymmetric way: correct cases would occur even if incorrect cases didn't, while incorrect cases are dependent on there being correct ones. Such asymmetric dependency is grounded on a (non-specified) asymmetric causal dependency. A large number of objections and alleged counterexamples has been suggested, many of which have been addressed by Fodor himself[19]. One that, to my knowledge, has not been addressed, and that, if correct, would be fatal for our purposes is the following.[20].

Fodor's solution to the disjunction problem might be internally consistent: it allows, from a theoretical point of view, to discriminate between "correct" and "mistaken" instances of representation. However, the point is not only to convince philosophers (external observers) that there can be a principled way to distinguish between representations and misrepresentations, but to allow representational organisms to detect errors autonomously. An organism has no way to know what asymmetric dependencies hold between its representations. So there is no sense in which the organism can "know" what its representations are "intended" to be about. How can an organism, for example, having represented a horse as a cow, recognize its error once it has got close enough to the horse to see that it was not a cow, if it has no information about what the content of its representation was in the first place?

Moreover, one would like asymmetric dependencies to be explained by a theory of content, rather then having them as the foundation of it. Fodor himself admitted that

[19]Fodor [35].

[20]See Bickhard's analysis of Fodor's theory in [8].

> the treatment of error I have proposed is, in a certain sense, purely formal. [...] It looks like any theory of error will have to provide for the asymmetric dependence of false tokenings on true ones.[21]

It has been suggested that the mere possibility to specify in a less formal way the nature of the asymmetric dependency is threatened by the risk of bringing in the observer.

In fact, it can be questioned whether the dependency relations that Fodor requires are naturalistic. Firstly, these are not themselves the subject of any known natural science so Fodor cannot claim, as the teleo-semanticist does, that he is explaining a semantic notion in terms of a scientifically respectable notion; i.e. biological function. Further, it is not obvious that the synchronic counterfactuals that Fodor appeals to when explaining asymmetric dependence have truth conditions that can be specified non-intentionally. Why is Fodor so certain that the counterfactual (synchronically construed) if cow $\mapsto$ "Cow" were broken then cow-picture $\mapsto$ "Cow" would also be broken is true? Perhaps if the first law were to fail "Cow" would change its reference to cow-picture and so the second law would still obtain. If so, then while "Cow" refers to cow ADT would say that it refers to cow-picture. Fodor cannot respond by saying that in understanding asymmetric dependence the counterfactual should be understood as holding the actual reference of "Cow" fixed since that would be introducing a semantic concept into the explanation of asymmetric dependence.

[21]Fodor [34] p. 110.

4.2.5 Are etiological theories of intentionality apt for inducing extrinsic restriction?

At the end of chapter 2 (section 2.5.7) we concluded that if a restriction of candidate input label bearers (CILB) were to be compatible with a theory of implementation, the properties that determine the restriction should be extrinsic to the implementing system. Our proposal, we anticipate, will be to restrict CILB by requiring that its elements be "indicative intentional icons". As anything, in principle, can be an intentional icon (anything can instantiate the relevant relational properties), a restriction of CILB that consists in requiring that its elements be intentional icons is an extrinsic restriction. This, of course, does not mean that, given an organism, and a particular environment, anything in this environment could count as a candidate input label bearer. This is precisely as it should be: we want to rule out unwanted models, while allowing for multiple realizability.

Some have argued that, as etiological accounts of function cannot be cashed out in terms of the present state of the instantiating system they are causally epiphenomenal, i.e., causally inert. We are after real causal constraints on representations, of the kind exemplified by Galilean dynamical models; if this difficulty cannot be amended this fact could threaten our proposal. Some authors suggest that biological function could be cashed out in non-etiological and non-teleological terms. Here it suffices to say that, as teleological functions are often considered as selected effects, they can also be considered as selected dispositions: certain traits are selected because they produce certain effects in response to certain causes. This, of course, does not make teleological functions a set of current dispositions, but a set of selected dispositions. In the next section we consider a theory of intentionality that is non-etiological, but that, if sound, would be equally suitable for our purposes. I do not wish to commit to either of these treatments, or, still less, to the claim that either of them as they stand at the present date are free of objections. What I want to stress is that, if either of them proved to be correct, it could be used to salvage computationalism from v-arguments, and to construct a computational theory of the mind that is free from standard objection.

4.3 The interactivist picture of intentionality

4.3.1 Control, Functional Indication and Functional Goal Directedness

According to the view of representation that I shall introduce in this section, Bickhard's interactivist theory, intentionality is grounded on the notions of function, control and goal directedness. As for all other teleological theories, then, it is in order to make sure that they are not, themselves, observer relative. The problem, we have seen, is that standard accounts of goal directedness make reference to the representation of the goal to be met. The following technical notions have been devised for this purpose.[22]

If a system **A** exerts an influence on another **B**, I shall call the effects that **A** has for **B** the **functions of A relative to B**. This definition allows us to retain multiple realizability while avoiding any reference to representation.

If, in particular, the outcome of some specifiable process in **A** influences the course of the processes in **B** then **A** exerts **control** on **B**. In other words, a subsystem **A** exerts control on **B** if differences in the processes of **A** trigger differences in the

[22]If one already believes that it is possible to free goal directedness from representations without making reference to natural selectionist concepts, this section can be skipped.

processes of **B**. A persistent complex of such control relationships among a number of subsystems constitute a **control structure**.

A particular instance of a control relationship (hence of a control structure) is a switching relationship: we say that a control relationship is a **switching relationship** if the influence A exerts on B consists in causing B processes to enter (or exit) a state of quiescence. Given a certain switching control structure, the flow of conditions that switches on the components of the system is called a **control flow**. As only components (subsystems) that are "*on*" can cause other components to switch on, the control flow is the flow of activation of the system.

Finally: the notion of indication. Given three subsystems A, B and C, A is an **indicator** of C for B if A exerts control on the switching processes in B such that it makes it possible (although not necessarily causes) that the control flow activates C. In other words, an indicator (A) is sufficient (not necessary) for the possibility that B switches to C.

The notion of control can be used to build a non observer relative notion of goal directedness. Consider a system (switch) A that either switches control flow to B, or switches it away from the $A-B$ subsystem. If the internal conditions of A that determine which of these possible routes the control flow takes are themselves controlled by the environment, then the $A-B$ subsystem is a **goal directed** subsystem. If the environmental controlling system leads away from the $A-B$ system, then it is a satisfier of the goal; if, instead, it leads to the activation of B, then it is not a satisfier. Intuitively, the activation of B corresponds to the message "*goal not met: try again*"; if, instead, the control flow switches away from the $A-B$ system, that corresponds to the message "*goal met: go on*". This minimal notion of goal directedness makes reference solely to internal, functional properties of the system, rather than to the external, semantic properties of the representation of the goal.

4.3.2 The origin of epistemic properties

One of the problems that one has to face in trying to naturalize the intentional relation is that physical, factual, correspondences are either in place, or not there at all: they are never, themselves, wrong. Consider again a mathematical dynamical system and a real dynamical system. We have discussed under what conditions the MDS adequately describes the RDS. Suppose that it doesn't: that is, suppose that our MDS, although qualifying as a dynamical system, doesn't comply with the constraints that would make of it a Galilean dynamical system. What we say, in these cases, is that the MDS fails to describe the RDS. We don't think that it is the RDS that is wrong about something: RDS's are what they are, they can never be "wrong", hence they can neither be "right".

By contrast, if a representation misrepresents a horse as a cow, it is not the horse's fault, or the cow's, nor can misrepresentation be blamed on some causal path from the horse to our cognitive system. We might be able to explain how the cognitive system made a representational mistake, but ultimately, it is the cognitive system that misrepresents. Semantic properties require a degree of normativity that is hard to naturalize. Our cognitive systems, in fact, are real dynamical systems. As such, they can't but comply with the laws of physics. How can they be wrong about something? In other words, where can we look, in the natural order of things, to find epistemic properties? When the universe was born, presumably, nothing was being right or wrong about something. To be sure, now we can be right or wrong about what was happening back then. But back then nothing had such epistemic properties.

At some point, either gradually or abruptly, something must have come about that had epistemic properties. What?

A promising domain of physical properties that has been suggested as a candidate is that of far from thermodynamical equilibrium systems. Physical systems tend to reach states of equilibrium. This fact is described in dynamical system theory by the presence of attractors. Such attractors can be single points, (such as is the case for a marble rolling in a bowl), or a periodic orbit (a steady oscillating behavior such as that of the moon around the earth). However, this is not the only kind of stability that we observe in nature. Some systems, in fact, seem to have stabilized into a far from thermodynamical equilibrium state (FTE hence forth). If isolated from the rest of the world, these systems would naturally tend towards equilibrium, thus ceasing to exist (as systems, that is). Their stability is dependent on the presence of counteracting forces. Clearly, these systems are open systems: they have to be, or they would not exist. A fridge is an example of such a system: its far from equilibrium stability depends on us providing it with a continuous energy supply.

More interestingly, some FTE systems, although dependent on environmental counteracting forces for their existence, make their own contribution to their self maintenance. An example of such systems is a burning piece of wood. The heat created by it generates a convection dynamics that allows us to provide the system with more and more oxygen-rich air, which, in turn, reacts to create more heat. A burning piece of wood, if deprived of its environmental conditions for existence, however, ceases to exist (qua burning piece of wood) without reacting in any way: it is not provided with any means to respond to changing environmental conditions so as to maintain its far from equilibrium state. Other systems, on the contrary, are endowed with more or less efficacious means to respond to a changing environment so as to secure their self maintenance. This property is called recursive self-maintenance (RSM).

The typical example as to how this programme might be carried out is the case of marine bacteria that we have discussed. This, Dretske argues, is a perfect example of natural misrepresentation. The direction of magnetosomes bears information about the direction of oxygen-free water, it is its function to convey this peace of information, so a magnetosome pointing downward means that oxygen-free water is downward. Notice that it continues to do so even if things are not so, allowing for misrepresentation. The idea is that the content of a representation is which ever fact about the world that the representation has the function of indicating. This function must be understood in terms of the internal information-gathering cognitive economy of the organism.

These marine bacteria, that we often encounter in the literature, are an example of RSM systems. They are FTE systems, like all biological systems (they cease to exist if severed from their environment) and they have means (magnetotaxis) to respond to changing environments so as to preserve their far from equilibrium state.

It should be clear, by now, where we are getting at. We wondered, some time ago, where in the natural order of things, we should look for physical systems that can be wrong about something. RSM systems seem to be the right place to look at.[23]

It has been suggested[24] that a theory of semantics should be based on RSM systems.[25]

[23]Our bacteria, for example, can be "tricked" into a lethal environment, that is, they can be wrong about things.

[24]Bickhard [8] and [7].

[25]As, to our current knowledge, the extension of the concept of a RSM system is that of an organism

4.3.3 Dynamical presuppositions

If a system is controlled by the environment, the final state it ends up in when it interacts with it implicitly categorizes environments (it selects the class of equivalence of all those environments that would cause it to enter that same state). Thus, for example, a metal bar implicitly categorizes environments that have the same temperature, i.e. it functions as a differentiator of environments. Now, consider an environmental differentiator that (for the sake of simplicity), can only end up in two final states, **A** or **B**. Suppose that a certain goal is active (in the sense specified above): the differentiator controls the activity of a goal directed subsystem. Depending on what state the differentiator ends up in, the goal directed subsystem selects a different interactive strategy.

Suppose that in our case the goal directed subsystem only has two available interactive strategies, $\mathbf{S_1}$ and $\mathbf{S_2}$. Each of these, when invoked, is iterated until when a certain internal outcome is induced (signalling that the control flow should continue elsewhere). Suppose, moreover, that when the differentiator ends up in state **A** the goal directed subsystem indicates (in the sense specified above) strategy $\mathbf{S_1}$, and when the differentiator ends up in state **B** then strategy $\mathbf{S_2}$ is indicated. The differentiator, we said, implicitly categorizes environments as **A-type** environments and **B-type** environments (through the factual correspondence of certain environments causing it to end up in state A or B). Similarly, the whole system that we are considering categorizes **A-type (B-type)** environments as $\mathbf{S_1}$**-type** ($\mathbf{S_2}$**-type**) environments.

being alive and healthy, one might ask why not to use the latter concept instead of the former. This simplification, however, would bring with it some unwanted chauvinistic consequences: in the current use of the expression, "being alive" entails having particular physio-chemical properties. So, demanding that representation be be a prerogative of living creatures would be tantamount to excluding the possibility of artificial representation, hence of artificial intelligence.

It is argued that such cases of elementary implicit predications (e.g. "**B-type** environments are **S_2-type** environments") constitute the basis for the emergence of representation. Implicitly, in fact, these propositions presuppose that certain facts obtain in the world: swimming in the direction indicated by the magnetosome, for example, contributes to the maintenance of the far from thermodynamical equilibrium of the bacterium only if the direction indicated is away from the surface. Following Bickhard I shall call the environmental conditions under which such predications are true: **dynamical presuppositions**.

To better appreciate how the notion of dynamic presupposition might ground the naturalization of representation, consider the relationship between the implicit categorization of a differentiator ("This is an **A-type** environment") and the environments that it categorizes. Notice that it doesn't make any sense to ask whether it is true or not that a certain environment, in isolation from the differentiator, is an **A-type** environment. Being an **A-type** environment is not an intrinsic property of environments. And, most importantly, it is not a property represented by the differentiator. We have seen how various theories of semantics (causal theories of semantics in particular) failed their naturalizing job for their incapacity to account for representational error. We are now in the position to say that they fail because they are attempts to ground semantic properties on environmental differentiation alone.

Contrast this with the emergent semantic properties of the dynamic presuppositions defined above. Does it make any sense to ask whether a dynamic presupposition is true? In our simple example: can we ask whether it is true that "**B-type** environments are **S_2-type** environments"? It does. In fact, things can go wrong, and strategy $\mathbf{S_2}$ can fail to induce the internal outcome indicated by the goal directed subsystem (just like the marine bacteria in Dretske's example can mistakenly follow the magnetic field towards death). This possibility for an implicit predication (dynamical presupposition) to be wrong, opens the way for a naturalization of representation.

4.3.4 Interactivist theory of semantics

We said that a viable research programme as for the naturalization of representation is one that seeks a natural notion of function. The semantic theories that adopted this strategy often tried to define in natural terms what it is for an internal item of a system to *have a natural function.* Once in possession of such a notion, they defined what it is to *serve a natural function* in relation to what it is to have a natural function (to serve a function is to succeed in functioning). This time, instead, we will shall reverse the logical order of these concepts.

[**Natural function**] An internal item of a RSM system is said to *serve a natural function* if it makes a contribution to the system's far from thermodynamical equilibrium.

As we have seen, a RSM system detects features of the environment and responds accordingly (in the attempt) to secure its far from equilibrium state. These responses, depending on the complexity of the system, may be simple triggers (such as in the case of our bacteria) or they may set indications of multiple possible future interactions. These responses (whether simple or complex) implicitly "assume" that the indicated behavior is appropriate (that is, that it contributes to the self-maintenance of the system). Clearly, these indicated interactions (or better, types of interactions) will actually be appropriate depending on certain conditions obtaining in the environment.[26]

We can then propose the following:

[Dynamical presuppositions] An indication of future action in a RSM *dynamically presupposes* that the conditions for the appropriateness of that action obtain.

In other words, a process can be said to dynamically presuppose the conditions (these can be internal or external to the system itself) that must obtain for it to serve its proper function. It is in terms of these dynamical presuppositions that an item can be said to have a certain function.

The important thing to notice is that dynamical presuppositions can be true or false: if the dynamical presupposition of an indicated action is false, the indicated action will fail. This property of dynamical presupposition is then an elementary epistemic property. Bickhard suggests that it should be used to ground the notion of representational content:

[Representational content] The dynamical presuppositions of an indicated action constitute its *representational content.*

If a RSM systems indications for action are ever to be appropriate, the system must make a number of relevant environmental distinctions. These environmental distinctions cannot but be a result of the contacts the system has with its environment.

[26]The appropriateness of the "swim in that direction" indication in our bacteria depends, for example, on that direction actually being that of decreasing oxygen-rich water.

Isn't this picture of epistemology equivalent to the simplistic picture that I claimed to be responsible for the recalcitrance of the problem of conflicting desiderata? According to that picture our epistemic contact with the world consists in encoding the relevant information so that it can be processed and used for selecting actions.

The problem with that picture was that encodings presuppose representations, hence they cannot explain how representations come about. The view that I am considering, now, distinguishes *contact* with the world from *content.* The contact with the world provided by the senses, in this view, and the environmental distinctions that it provides to the system, have no content whatsoever. Environmental distinctions don't carry any information as to what they are distinctions of: they, and the processes that allow them, have no representational property. It is the dynamical presuppositions of the indications for action that have the relevant epistemic properties: these could be in place even if no contact with the world was taking place, except that in this case they would probably all be false. We may want to call this alternative picture, a *two-factor epistemology.*[27]

[27]The perspective that we are considering seems to have the counterintuitive consequence that only processes whose appropriateness can be assessed individually can count as representations. For example, my representation of a bull charging me, can quite understandably be put in some good use in maintaining my organism far from its thermodynamical equilibrium. But what of my representation of a peaceful cow? How can that *serve its proper function*? Our representational apparatus is (presumably) to be understood within a complex arrangement of embedded goal-directed sub-systems. Fortunately, as we have seen above, the notion of goal that we want needs not be representational. All that we need a goal-directed system to do is to respond differently according to the indications of the system's error detecting devices. For example, it can respond with a "proceed to other interaction" indication if the current interaction is appropriate, and with "try again" indications if it is not appropriate. In other words, the goal-directed subsystem needs not represent the contents of its indications, just as the environmental-distinction making subsystem (the contact-making subsystem) needn't.

4.4 Teleological theories of computation: an honest proposal

This long incursion into the field of naturalized intentionality had the purpose of looking for naturalized semantic properties that complied with the desiderata put forward at the end of chapter 2. We are now in the position to draw a conclusion.

Teleological theories of intentionality, whether of the etiological variety or other, do not treat semantic properties as encodings, or as any sort of intrinsic property of the physical structures that instantiate them (we have seen how, for example, the dance of a bee is not, by it self, or in virtue of some intrinsic property, a representation of nectar location). As I shall argue, for this reason they are apt for grounding syntactic properties (in particular computational properties) in a way that abandons the pervasive internalist picture of them. We shall call any such theory of computation: *computational externalism.*

There are various theories of semantics that can be said to be "teleological", and they imply very different (possibly opposed) strategies of naturalization. For our purposes, however, what counts is that if any of them proves to be sound, it would allow us to induce in the set of CILB an extrinsic restriction, while bypassing the isomorphism catastrophe: they allow us to ground syntactic constraints to a representational labelling scheme in a way that complies with the desiderata of a theory of computation.

The supervenience base for an item being a teleological representation (whatever theory one prefers) will be a large disjunction of relational properties. Within each type of relational properties belonging to this disjunction, moreover, a certain degree of tolerance will be objectively introduced. Various instances of the same bee-dance, for example, will present slightly different lengths of the long axis. How much tolerance is acceptable, before the dance misrepresents the location of the food? The answer to this question depends on a number of complex states of affairs concerning the dance itself, the response of the food-seeking bee, and a lot of other environmental facts. What is important for us, however, is that such constraints are objective: whether an item ceases to be a representation, or changes its represented, is a matter of objective fact.[28]

As I shall now discuss in more details, *I propose to ground the observer-relative constraints that sustain the computational abstraction on the objective conditions for the maintenance of the semantic properties of intentional icons.*

4.4.1 Teleo-computation of a logical gate: an intentional theory of computation at work

Before putting forward a precise proposal, an example should help to see how our strategy works in the case of logical gates. Suppose (this is a fictitious example) that the interactive strategies for the maintenance far from thermodynamical equilibrium of our marine bacteria were not simple triggers, as in fact they are. Our bacterium now has two options for its survival: either it swims in oxygen-free *and* cold waters, or, alternatively, it opts for oxygen-reach *but* warm waters.

Two magnitudes serve the relevant functions: the little magnetic bar will differentiate environments according to the oxygen-free/oxygen rich axis, assigning value $X = 1$ ($X = 0$) to oxygen-free (oxygen-rich, respectively) directions. Another subsystem will take care of the temperature of the water, assigning value $Y = 1$ ($Y = 0$,

[28]Although it might be a matter of degrees.

respectively) to warm (cold) environments. The response of the bacterium (the output) will be simply: *swim in that direction* ($f(X,Y) = 1$) or *do something else* ($f(X,Y) = 0$).

An orthodox (internalist) computationalist explanation of the behavior of the bacterium would describe it as implementing the XOR logical function: $f(X,Y) = 0$ if $X = Y$, otherwise $f(X,Y) = 1$ (where '0' and '1' are the two digits). But we have seen that between a galilean model of the system (the bacterium) and the abstract XOR function, there is a gap of abstraction that cannot be bridged unless a representational labelling scheme is given. For example, the sensor (subsystem) differentiating environments according to their temperatures, if described by a galilean model, will present a delay factor, and it will respond gradually to varying temperatures according to some complex MDS. Moreover, it will not respond to the temperature of the water with infinite precision: there will not be an exact correspondence between the countably many values of temperature and the states of the differentiator subsystem. Hence an error must be allowed. The response of the real system, finally, will not be the same at all possible temperatures of the water. A MDS describing the behavior of such a system, will have to model all these constraints.

The abstract input function X, instead, neglects all such practical constraints. What delays and errors should be tolerated? If the bacterium responded differentiating environments a couple of hours after having swam through them, would it still implement a XOR function? And what if the temperature-detecting subsystem only responded to waters of 100 degrees and above (we assume that at that temperature the bacterium would cease to exist as such)? These are all familiar concerns.

My proposal, in this case, is to require that the tokens that bear the labels for the input architecture (X, Y and $f(X,Y)$ in our example) be teleological representational items: they provide for the honest representational labelling scheme. We have seen how teleological theories comply with the desiderata for such a scheme. This, then, is what our teleo-computationalist bacterium looks like.

[**Teleo-Computation**] A physical system S (the bacterium) implements a computation A (i.e. $Imp(XOR(X,Y),S)$), if and only if:

1. The candidate label bearers for the input architecture of A (X, Y and $f(X,Y)$) are teleo-representations (representations as naturalized by teleological theories), relative to the user (in this case the bacterium itself) and to its historical and current environments[29]. And

2. The system complies with the desiderata for a digitally supporting system, relative to the preservation of the intentional properties (point 1) that identify the honest labelling scheme.

So, for example, if we ask ourselves whether a candidate computation A is really being implemented by S, we should check that the practical conditions this implies for the system comply with the conditions for the maintenance of the representational status of its input architecture. There is in fact a certain amount of tolerance before a representation changes its status, or its represented. It is not necessary that the long

[29]Only its current environment, under non-etiological treatments.

axis of the dance of a bee, I repeat, be exactly (100%) correlated with the direction of the nectar, or that the magnetosome be exactly aligned with the earth magnetic field, or that the temperature of the water instantly influences the internal state of the bacterium: the representing system will robustly tolerate, with respect to its maintenance, a certain amount of error. The extent of such tolerance is an objective fact (if some theory of intentionality proves to be adequate) and can thus be used to ground practical constraints in objective terms.

Our hypothesis is that the practical constraints that made computational schemas observer-relative in the previous sections (the conditions for the digitality of the system) must be those inherited from the conditions for the maintenance of the representational labelling scheme.

4.4.2 The honest labelling scheme for Finite State Automata

Let us see how the solution proposed above works in the case of finite state automata. What groupings of physical states are allowed? Recall that the state-to-state picture of implementation that we have criticized requires, for a physical system S to implement a FSA A ($Imp(S, A)$), that there exists a mapping (f) from the internal (physical) states of S (Q_S) to internal computational states of A (Q) such that, for every state transition $(Q_1, I)©\delta(Q_1, I) = Q_2 \mapsto \beta(Q_2) = O_2$, if S is in internal state s_1 and receives input i (where $f(i) = I$ and $f(s_1) = Q_1$), this "causes it" to enter state s_2 and to output o_2 such that $f(s_2) = Q_2$ and $f(o_2) = O_2$.

Now, to formalize my proposal, I shall need to define a number of equivalence relations.

For all physical inputs ($i \in X_S$), states ($s \in Q_S$) and outputs ($o \in Y_S$) of S, let:

$$i_j =_f i_k \text{ iff } f(i_j) = f(i_k)$$

and

$$o_j =_f o_k \text{ iff } f(o_j) = f(o_k)$$

and

$$s_j =_f s_k \text{ iff } f(s_j) = f(s_k)$$

The ones defined above are the relations of equivalence induced on physical items by the function that maps (groups) them onto computational ones. Now, the physical conditions discussed at the beginning of this chapter for an item to be an indicative icon (i.e. to possess naturalized semantic properties) allow us to define also the following relations of equivalence. Intuitively, the following equivalences group together all physical items that map onto the same intentional properties.

For all $i_j, i_k \in X_S$ and $o_j, o_k \in Y_S$ let:

$$i_j =_{Ref} i_k \text{ iff } \exists P : Ref(P, i_j) \Leftrightarrow Ref(P, i_k)$$

and

$$o_j =_{Ref} o_k \text{ iff } \exists P : Ref(P, o_j) \Leftrightarrow Ref(P, o_k)$$

For all computational outputs, let:

$O_j =_{Ref} O_k$ iff $o_j =_{Ref} o_k$ for all o_j and o_k such that $f(o_j) = O_j$ and $f(o_k) = O_k$

And for all physical states $s \in Q_S$ let:

$$s_j =_{Ref} s_k \text{ iff } \beta(f(s_j)) =_{Ref} \beta(f(s_k))$$

where β is the output function.[30]

The above two sets of relations of equivalence, $=_f$ and $=_{Ref}$, group together physical items that are mapped, respectively, onto the same computational items and onto the same representational items.

The proposed solution to the problem of implementation requires that the label bearers for the input architecture be intentional icons. This, in our formulation, is tantamount to requiring that for each item the two relations of equivalence are matched. Formally, then, our proposed notion of implementation is the following:

[Implementation] Given the automaton A specified by $\langle X, Y, Q, \delta, \beta\rangle$, and a physical system S described by $\langle Q_S, T, \{g^t\}\rangle$, S implements A iff:

[30]Given the definition of representational equivalence of two computational outputs, the right-hand side of the last biconditional must be understood as compactly denoting the representational equivalence of each element of $f^{*-1}(\beta(f(s_j)))$ (the class of all physical outputs that are mapped by f onto the same computational output $\beta(f(s_j)))$ with each element of the class $f^{*-1}(\beta(f(s_k)))$. The fact that we have introduced the notion of representational equivalence of two computational outputs ($O_j =_{Ref} O_k$), thus, should not be taken to entail the claim that abstract entities, such as computational inputs and outputs, can have representational properties independently of the physical properties that implement them. The definition is just a convenient notation that denotes a large conjunction of representational equivalences between the respective implementing physical outputs.

1. There is a function $f : Q_S \times X_S \times Y_S \mapsto Q \times X \times Y$ such that: [Iso-Left] for every computational state transition[31] $I_1, S_1 \rightarrow \delta(I_1, S_1) = S_2 \mapsto \beta(S_2) = O_2$ there exists a causal state-type transition $([i_1], [s_1]) \rightarrow [s_2] \mapsto [o_2]$ such that $f^*([i_1]) = I, \quad f^*([s_2]) = S_2, \quad \text{and} \quad f^*([o_2] = O_2$. And

there exists a causal state-type transition $([i_1], [s_1]) \rightarrow [s_2] \mapsto [o_2]$ such that

$f^*([i_1]) = I, \quad f^*([s_2]) = S_2, \quad \text{and} \quad f^*([o_2] = O_2$. And

2. Th[illegible]
such that:

$$i_j =_f i_k \;\Rightarrow\; i_j =_{Ref} i_k \tag{4.4.1}$$

$$o_j =_f o_k \;\Rightarrow\; o_j =_{Ref} o_k \tag{4.4.2}$$

$$s_j =_f s_k \;\Rightarrow\; s_j =_{Ref} s_k \tag{4.4.3}$$

Some considerations are in order. The above constraints are to be thought of as empirical constraints on the candidate implementing system, not as providing an alternative definition of automaton. They ought to tell us under what conditions a physical system, type-identified by the MDS that it instantiates, implements a finite state automaton.

The requirement that functional equivalences entail representational equivalence reflects the intuition that the realizers of a given computational property must have something in common, a part from being all realizers of that computational property. If v-arguments were sound, in fact, the mappings from physical to computational properties would be arbitrary (i.e. they would allow arbitrary disjunctions of physical properties to be mapped onto the same computational one). The requirement

[31]For the sake of clarity here, like in other similar occasions, I use particular indices for denoting different inputs, outputs and states. Of course these must be thought of as ranging over the whole sets of inputs, outputs and states.

that there be a mapping, in general, is a syntactic constraint on the arguments to be mapped, and is therefore silent as to the homogeneity or heterogeneity of the arguments. For example, the requirement that there be a one-to-one mapping from groups of words of a language to the natural numbers, doesn't constrain the groups of words that are mapped to be meaningful (or even syntactically correct) sentences. G¨del numbering scheme, for example, can be used equally well to identify syntactically incorrect arrangements of elementary symbols. Thus the mere information that an arrangement of symbols corresponds to a number does not guarantee that the arrangement is a well formed formula. The state-to-state correspondence picture of computation, if v-arguments are right, is equally incapable of ensuring that functionally equivalent states have something in common. In the case of MDSs, we have seen, the information that a magnitude takes up a given value (or a given set of values) instead, ensures that all possible realizers have something in common (a part from being all realizers): they all share the physical property that corresponds to that magnitude taking up that (or those) particular values.

In addition to the syntactic constraint provided by functional equivalence (by a state-to-state correspondence view of implementation), the schema proposed here also requires representational equivalence.

Secondly, it will be noticed that the schema constrains the requirement of functional equivalence, but that it is itself not constrained by representational equivalence. This reflects the idea that if representational properties must be instantiated whenever computational ones are, the contrary does not hold. It is certainly possible that an item is a representation without being a computational item. Notice, moreover, that if requirements 4.4.1, 4.4.2 and 4.4.3 were bi-conditionals, this would have the absurd consequence that different computational states could not yield to the same computational output.

It is easy to test the proposal against Putnam's skeptical result: the seven state intervals $S_{i(1<i<7)}$ in Putnam's proof, for example, fail to meet requirement 4.4.3 above. Similarly, the computational item I_k, defined as the (maximal) interval state of the boundary of S during interval Int_k, in the proof of the extension of Putnam's result to automata with inputs and outputs, fails to meet requirement 2.

These skeptical arguments are blocked because the mere fact that the electromagnetic and gravitational signals (for example) induce every open system that is not shielded from them to enter ever changing maximal states, has now no relevance whatsoever in determining the computations implemented. These environmental distinctions, in fact, bear no information as to what they are distinctions of: very likely, they do not preserve the equivalence relations above. It is only the dynamical presuppositions that individuate what physical processes or items do possess intentional properties (if, for example, we take the interactivist theory of intentionality to be our preferred one).

The above formalization is also apt for exposing the analogy with the case of physics. Recall that a result of our analysis was that physics escapes skeptical arguments because measurements allow us to individuate states independently of their relational properties. It is the match between abstract calculations and concrete measurements, we said, that blocks unwanted instantiations in the case of physics. Similarly, in my proposal, it is the match between the abstract computational relation of equivalence (that can be formally "calculated") and the concrete intentional relation of equivalence defined above, that blocks the unwanted models.

Chapter 5

Other approaches to the problem of implementation

5.1 Introduction

In this work I have discussed several v-arguments, but very few counterarguments. This was justified because my intention was to explore the consequences of these arguments, if they are assumed to be valid. The opinions that one can find expressed in the literature, however, are not simply divided between those who think that v-arguments are correct and those who think that they are not. There is also a large number of authors (probably the majority among computer designers and cognitive scientists), who think that there is no such thing as a "problem of implementation", and that v-arguments, just like the various failed counterarguments, are irrelevant because they address the issue in the wrong way.

Throughout this work I have adopted (without discussing it) a particular methodological approach. The general question that I set out to answer was: what does it take to implement a computational structure? Since the very first few words of this thesis, I rephrased this question as: what are the models of $Imp(S, A)$? The main v-arguments that I have discussed are model theoretic arguments. Let us call this general methodological stance the *model-theoretic approach* (MTA henceforth) to the problem of computational individuation. I have never discussed neither the virtues nor the shortcomings of this stance. It is now time to do so.

An assessment of the general validity of the MTA is of the uttermost importance for the tenability of my thesis. To realize this it suffices to notice that the v-arguments that my semantic picture was designed to block rest on a MTA to the problem of implementation. Although, as we have discussed, most responses to these arguments question their validity without undermining their methodological premises, it is certainly possible to argue that what v-arguments, if they were sound, would show is not that there is a "problem of implementation", but that MTAs are inadequate to address the issue of computational individuation. As this intuition is probably shared by most of the relevant community of experts, it deserves special attention.

In section 5.2 I consider whether MTAs bear with them some implicit metaphysical assumptions. It is argued that they do not, per se, commit us to any metaphysical claim about the nature of computation.

In section 5.3 I discuss the way in which the concept of implementation is used by the relevant community of experts. This appears to be at odds with the view that is affordable by an MTA to implementation. Some implications of this for the present thesis are discussed. The conceptual soundness of the alternative view (the *functional view*) of implementation is then challenged (section 5.4), and a partial justification for the adoption of a MTA is given.

Finally, a shortcoming of my treatment of implementation is that it appears to be applicable only to finite state automata (as opposed to some more complex computational structures, such as Turing machines). This difficulty is tackled in section 5.5, where I provide an outline of how the semantic view should be applied to universal Turing machines.

The thesis of this chapter is that there is no conceptually sound alternative understanding of implementation at the moment. This, however, does neither entail that we are forced to adopt a MTA to implementation, nor that standard computational explanations are not valid. Moreover, It is conceded that if a functional view could be provided, it would be preferable to the semantic one. The adoption of a MTA in this work is defended on methodological grounds. The semantic view, however, is argued to be less "unfriendly" to the intuitions of expert practitioners then it appears to be. The semantic picture, in fact, does not contradict the functional characterization of implementation: it adds semantic constraints to the functional ones in order to restrict in a multiply realizable way the implementing physical structures.

5.2 Some general issues concerning model theoretic approaches

A first question that ought to be addressed is whether the MTA bears some implicit metaphysical assumptions. On one side it appears not. To ask what is the model of a sentence in a formal language is the same as asking what are the truth conditions of that sentence once it is interpreted. In this sense, to ask what are the models of $Imp(S, A)$ is the same as asking what it takes for a physical system to implement a computation. In this sense, then, the MTA doesn't entail any metaphysical commitment.

However, it is implicit in the approach (given some widely shared assumptions) that the truth conditions of the sentence $Imp(S, A)$ must be cashed out in physicalistic terms. Computations (and thereby computational properties), in fact, are not typically construed as enjoying a peculiar metaphysical status: not only do they supervene on the physical properties of their realizers, but they are thought to be grounded in them. The principle of multiple realizability only entails that different physical objects can implement the same computational structure, just like objects made of different staff can share the same shape. The multiple satisfiability of, say, the concept of *spherical object* doesn't force upon us a "platonic" assumption about the metaphysical status of sphericity.

Although the MTA, per se, doesn't bear with it any particular metaphysical assumption, it implicity commits us to the claim that it is possible to specify the relevant causal structures by using nothing but the language of fundamental physics. This is the methodological assumption behind the MTA as it is applied to the implementation of computational structures. This assumption amounts to the claim that the computational properties of an object must be deducible from its microscopic, complete physical description. Let us call this claim the *deducibility assumption* (DA henceforth). The fact that DA is virtually uncontroversial in the relevant philosophical literature has obscured a potential unwarranted use of it. DA, in fact, can be interpreted in two ways.

The metaphysical version of DA (the one that is relatively uncontroversial), is the claim that the computational properties of a physical system are in principle deducible from its physical properties. The methodological version of it, instead, is the much stronger claim that it must be possible to specify the deductive bridge from, say, the values of all fields in a region of spacetime to the computational properties of that region. This claim is unwarranted and it is probably false. Throughout this work, the failure to specify a model for $Imp(S, A)$ within the standard picture of implementation has been assumed to be a reductio ad absurdum of the picture itself. One could argue that, in stead, these failures only show that the MTA is not appropriate to address the issue. Such a criticism to my thesis (and to any other work that adopts the MTA) is not necessarily committed to a denial of DA in its metaphysical interpretation: it only needs to deny its methodological version.

Suppose that, for some reason, a community of philosophers were investigating the relation between the structure of body organs and their physical realizations. Suppose that, endorsing a MTA, one of these philosophers asked what are the models of $Imp(S, H)$, i.e. of the sentence "*the physical system S implements a heart*". Suppose, moreover, that all the proposed models of $Imp(S, H)$ proved to be inadequate (either too strict or too liberal) to capture the concept of heart. Would anyone think that all explanations that make reference to hearts (such as: "the death of the old man was caused by a heart attack") should be considered as unwarranted? Wouldn't we rather think that these philosophers are simply approaching the issue in the wrong way? Isn't it relevant that surgeons do not appear to face any problem at individuating hearts in their patients?

Notice that if one thinks in this case that it is the philosophers that are reduced ad absurdum (as anyone would) he or she needs not deny the claim that hearts do their jobs by virtue, and only by virtue of their physical properties.

Similarly, one could object that assuming that v-arguments are sound (like I have done) does not entail that there is a problem of implementation, but that there is a problem with the MTA to implementation. This chapter addresses this potential objection.

Before considering some other possible approaches to implementation I wish to make some preliminary remarks. As we shall discuss in what follows, a concern about MTAs is that very rarely they capture the relevant concepts in a familiar way. A prima facie implausible consequence of MTAs to implementation (of the standard variety or of the semantic variety defended here) is that they propose to individuate the realizers of computational structures in a way that is alien to that of the relevant community of experts. Computability theorists and computer designers, in fact, have very effective means to tell whether something is a computing system or not, or for individuating the realizers of computational structures, inputs, outputs, and states. Admittedly, these means are nothing like the ones afforded by any MTA. In particular, it is not clear whether a MTA could in principle ever yield an analysis that would be consistent with the practise of computer science or of computationalist cognitive science. Can this fact be used to object in principle to MTAs?

On one side it appears like this is a problem. The most competent users of technical concepts are certainly the relevant experts. It is also doubtless that the domain of concepts that pertains to computer theory has been used for several decades with most profitable results. Any analysis of these concepts that completely disregarded the way in which they are typically used would then be questionable a priori. On the other side, however, different communities of scholars use the same domain of concepts in different ways and with different goals in mind. A concept may be perfectly usable for practical purposes and yet fail to meet the desiderata of a philosophical analysis. One of the most striking cases of this happening is the relatively "uncontroversial" use of the the concept of *quantum state* by the community of physicists vis a vis the difficulties encountered by philosophers of physics in providing a satisfactory analysis of it. In a sense, I argue, the unfamiliar flavor of the result of a conceptual analysis is no a priori argument against it. However, as we shall see, there are reasons to think that the use of a MTA should be well justified.

In what follows I briefly discuss how computer designers and cognitive scientists use some concepts of computability theory, considering whether their praxis could (or should) ground an alternative, philosophically satisfactory picture of implementation.

5.3 Vacuousness arguments: the view of computer-designers

5.3.1 The ontology of computer science

Let us begin by considering how computer programmers and designers would describe (and thereby explain) how programs are executed by a machine. At the highest level of analysis is an instruction written in a certain programming language. This can be something like UNTIL P TRUE DO ——— ENDUNTIL. In the convention known by the programmer, this (once appropriately input in the machine, and if the computer functions properly), should result in the computer doing whatever the programmer writes in the blank space, until the value of the variable P becomes TRUE. At any step in the response of the machine, then, the value of the variable P must be calculated, and the process should not stop until when this value is TRUE. Clearly the machine will also respond to the input doing other things, that are not described as part of the process. It will, for example, activate a fan to cool the central processor. What is relevant to the programmer, however, is that the computer produces the right output: anything that is not directly related to ensuring that or to explaining how that is achieved, should be considered as accessory.

The machine, of course, needs not "understand" what the instruction means, in order to execute it. Not only must the instruction be written according to known conventional rules, but it must be "encoded" as a (typically binary) string of bits. This is done by the computer as soon as the programmer prints on the keyboard the instruction. The effect of printing the instruction, in fact, is that the voltage in a number of memory cells (one for each bit of the instruction string) is set to certain levels according to the input information. These levels (i.e. the exact values of the voltage) are not relevant per se. Any level would do just as well, so long as the functional organization of the machine remains unaltered.

Normally the instructions, thus encoded, are not immediately executable by the machine. There are, in fact, a number of operations that can be said to be primitive, in that they are not further analyzable into more elementary sub-operations. These can be a transfer of bits from one register to another, or an operation, logical or arithmetical, on those bits. The string of bits that encodes the instruction that the programmer has just entered, then, must be transformed into another one that encodes the same instruction by encoding a functionally equivalent large number of primitive sub-instructions. Typically this "translating" job is done by a compiler and the language apt to encode primitive instructions is called *assembly language.*

There are two things that I wish to comment on at this early stage of the explanation. First, the string that is input to the compiler and that that is output in assembly language can be said to encode the same instruction only in a loose sense. An example can help us to appreciate the difference between them. Suppose that a friend told you to pass her the salt. To execute this instruction, of course, you will have to execute a number of steps, such as, for example, look around on the table until you see the white container, move the bottle of wine out of the way, reach for the white container, grab it and pass it. So, in that particular circumstance, your friend might as well have told you: "look around on the table until you see the white container, move the bottle of wine out of the way, reach for the white container, grab it and pass it to me". Now, in a loose sense, she would have asked you to execute the same instruction, but literally, of course, they are very different. For example, if you didn't understand well the language of your friend, you might know what "salt" means, but not what "white container" means, in which case you would react differently to the two instructions. Similarly, different machines might be designed to execute different primitive operations.

The second thing that I wish to comment on regards the use of the word "translation". Translations, in fact, are literally meaning-preserving transformations of (meaningful) strings of symbols from one language to another. Here, instead, what is preserved is an input-output behavior. Moreover, neither the input nor the output literally refers to the respective instructions: they are simply interpretable as referring to those instructions by virtue of the fact that they are apt to prompt a chain of events in the machine that lead to the execution of those instructions.

So far, in our story, there is nothing that could help us understand what computer designers think implementation is. So far, in fact, we have described no real happening. To say that a string of bits that encodes a given instruction has just been transformed into another string that encodes a set of primitive instructions, is not the description of a physical process. The description of a physical process sounds more like, for example: "the two billiard balls have collided at time t". The description of the functioning of a compiler, instead, doesn't mention any balls, collisions, or voltage levels.

The next mechanism in our story represents a step in the direction of physical implementation. Although the strings that encode instructions in assembly language "refer" to primitive operations, we have seen, they do not suggest what should happen physically to execute them. Assembly language strings, then, must be further transformed into a suitable language, called *machine language*, that "refers" (in the sense specified above) to concrete physical happenings in the machine. The mechanism responsible for this second type of transformation of strings is called the *assembler.*

The insertion of a string that encodes a machine language instruction in a special register, finally, causes a series of physical events in the computer. These events can consist in the transfer of strings of bits from one register to another, in the creation of strings in a previously empty register, etc. Computers are built in such a way that, under ordinary circumstances, a chain of physical happenings that realize the instructions encoded in a binary string (m) of machine language implements the instructions encoded by the assembly language string (a) that was transformed into m by the assembler. In its turn, the string m implements the instructions originally entered by the programmer and that where transformed into m by the compiler. So, through a chain of implementations, the story goes, the computer implements the instructions that are input by the programmer, thus executing them.

In telling the story of an implementation I have only used the language that one can find in the relevant technical literature. We were wondering how the concept of implementation is used by computer scientists. This story, I think, is pretty clear about it. Notice that computer scientists don't feel the need to use two different words when referring, on the one side, to (1) the relation that obtains between a string in assembly language and the string in machine language that implements it, and, on the other side, to (2) the relation that obtains between a string in machine language and the chain of physical happenings caused by its insertion in the appropriate register: it appears that the word "implementation" suffices for both uses. This intuition provides the basis for the alternative picture of implementation that I will discuss in the following pages. Let me spell in more detail what this intuition amounts to.

According to the view that I am considering, an object is a computer if and only if its physical structure can be functionally analyzed into predefined parts and relations between these parts. These parts and processes are the intended referents of the strings of machine language: memory cells, locations (addresses) of memory cells, input or output configurations of voltage levels, changes of configuration of voltage levels among memory cells, patterns of activation of units, etc.

We are trying to extrapolate a general theory of implementation from the intuitions and practices of computer designers. The first thing to notice, summarizing what we said so far, is that this idea of implementation is grounded on a specific ontology. Not everything is a candidate input, or output, or a candidate memory cell, or a candidate string of digits, or a candidate central processor. The following is a good example of how some items of this ontology are characterized:

> Some systems manipulate inputs and outputs of a special sort, which may be called strings of digits. A digit is a particular or a discrete state of a particular, discrete in the sense that it belongs to one (and only one) of a finite number of types. Types of digits are individuated by their different effects on the system, that is, the system performs different functionally

> relevant operations in response to different types of digits. A string of digits is a concatenation of digits, namely, a structure that is individuated by the types of digits that compose it, their number, and their ordering (i.e. which digit token is first, which is its successor, and so on). [...] Strings of digits in this sense are, to a first approximation, a physical realization of the mathematical notion of string.[1]

Of course the fact that competent scholars use a language that implicitly refers to a peculiar ontological domain doesn't show, by itself, that such independent ontological domain exists in its own right. For now, however, I will set aside this issue, to further explore the alternative notion of implementation.

5.3.2 Computer architecture

There is another aspect of MTAs to computation that is at odds with the practise of experts. In the relevant technical literature the concept of implementation is never divorced from a specific computer architecture. The relevant question is never: "does this object really compute something?" It is rather: "does the object compute this computation by implementing that architecture?"

For an object to implement a modern van Neumann architecture, for example, not only must the candidate computer contain a structure that is functionally equivalent to a memory and one that is equivalent to a control unit, it must also contain structures that realize access to this memory in a specific way (the so called *random access*). There must be, for example, hard wired bidirectional links between each memory unit and the control unit, and the control unit must contain a structure that

[1]Piccinini [66], section 5.

implements a memory address register (MAR). Consider for example, the following description of how a random access to a memory is realized:

> When the control needs to access a particular memory unit, the address is put in the MAR and sent out as a signal on the address lines. Each memory unit is equipped with decoding hardware which responds to the address signal . Every memory unit receives the address signal and the one that matches it prepares for action. [...] The action taken is either to send what is in the memory location to the control (a read operation by the control) or receive something from the control for storage in the memory location (a write operation).[2]

To argue that a wall implements a van Neumann machine, one would have to argue that there is a structure within the wall that is functionally equivalent to a MAR, or that some parts of it act as memory cells, or that another part acts as the control unit, and that each memory cell can be accessed in roughly the same amount of time (as a random access typically ensures). Moreover, when computer scientists or computational cognitive scientists ask themselves if a physical structure implements a given architecture, in most cases, they are debating over which architecture is being implemented, and not, abstractly, whether an architecture (any one) is being implemented at all. So, for example, if it is hypothesized that the cerebral cortex implements a van Neumann architecture, the opponents of this view would argue that, say, a Turing machine architecture is being implemented, instead. In Turing architectures, for example, there are two distinct memories. One is the symbolic memory of the tape, while the other takes care of determining which choices should be

[2]Wells [95], p. 171.

made under certain external and internal conditions. In a van Neumann architecture, on the contrary, this distinction is practically irrelevant.

Thus, when one wonders whether the brain implements a certain computational structure, attention is payed to the details of that structure in contrast to those of other potential structures (e.g. to whether the encoding mechanisms in the brain distinguish data from instructions), rather than requiring that the brain comply to some abstract desiderata that result from a model theoretic approach to the issue.

5.3.3 Universal machines

Finally, I wish to mention a third aspect of computer science and cognitive science that is typically neglected by MTAs. A crucial aspect to explain the fortune of computers in our life, as well as in providing models to the sciences of the mind, is the fact they can be controllable. If all Turing machines, in a sense, function in the same way, there is a sense in which Universal Turing machines (Henceforth UMs) deserve a special place, both at a theoretical and at a practical level. Unlike ordinary Turing machines, universal Turing machines respond to a part of the tokens written on the tape by executing the instructions (computing the functions) that are encoded in those tokens.

A first point to notice is that postulating that an object is a UM adds constraints to its functional architecture. The ontology of computer science, in this case, in fact, is particularly restrictive. Standard universal Turing machines, for example, are made of a number of basic functional components, such as the *leftmost* and a *rightmost symbol finders*, a *single symbol erasers* and a *symbol copier.* As I have already mentioned, computer designers expect an implementing object to implement each basic component of the implemented architecture. Thus, to someone who claimed that a wall implements a universal Turing machine, a computer designer would ask, for example: "which part of the wall implements the leftmost symbol finder?". In other words, the design of UMs adds constraints to the functional organization of its candidate implementations. These considerations, however, are substantially not different from the issues treated in the above two sections.

But there is yet another aspect of UMs that is particularly relevant for our discussion, and that deserves an independent treatment. Universal Turing machines are so called because they can "simulate" all other machines. This means that when the input to a UM contains, together with the usual data, also the standard description of a particular Turing machine encoded in the appropriate way, it behaves exactly as if it were that particular machine. This is why we can use the same computer to simulate the most various phenomena, such as a chess player, a calculator, or the motion of the ocean. This flexibility is also at the heart of the allure of computationalist models of the mind. Also the mind, in fact, appears to be able to respond in very flexible and intelligent ways to various stimuli, always exploiting the same cerebral structures. The programmability of UMs is so important to computer designers that some have proposed to reserve the name "computer" for universal computers only: "computing mechanism that are not programmable deserve other names, such as calculators, arithmetic-logic units, etc."[3]

What is relevant for our discussion is that such flexibility creates an asymmetry between UMs and the mechanisms that it can simulate. If we grant that any physical mechanism (provided that its workings are well understood) can be simulated (hence described) by a computational structure, then we must also grant that any physical

[3]Piccinini [67], p. 304.

mechanism implements some computation: it implements at least the computational structure that simulates it. In this work, v-arguments have been used to make the case for a semantic picture of implementation. But another possible response is to say: "fine, it might be that a wall implements some computational structure under some interpretation (for example it implements a computational model of the motion of its atoms), but can we also use it to simulate other virtual machines?". In other words, one could concede that under some interpretation or other, any object implements some computation, but deny that any object implements a UM.

> The key idea that underpins the idea that everything is a computer depends on the representational capacities of universal machines. It turns out, as we now understand, that the functioning of any machine whose elementary operations are clearly understood can be described in terms of a program and simulated on a universal computer. [...] Does this licence the conclusion that the brain is a universal machine and that psychology can be studied independently of hardware considerations? It does not, because the argument does not show that the brain is a universal computer. [...] The argument may be strong enough to show that the brain is a computer of some kind but it does not show that it is a universal machine rather than a finite automaton or a fixed function Turing machine.[4]

5.3.4 The functional picture of implementation

In sum, if a theory of implementation is to respect the way in which computer designers or computationalist cognitive scientists individuate computational states and processes, it must comply with the following two desiderata:

[4]Wells [95] p. 193.

1. It must specify how each item in the (functionalist) ontology of computer science is implemented, and

2. It must specify how different architectures are implemented.

To a large extent, these considerations are not made explicitly: these views must be inferred from the implicit use of computational concepts by the community of experts. As it often happens, the relevant concepts as they are used by competent practitioners of a science are not straightforwardly spendable in the domain of the foundations of that science. As a matter of fact, the domain of the foundations typically exists precisely because some concepts are believed to be ill founded. Of course, this by no means entails that anything goes. Even though often the analysis of a conceptual repertoire yields to some more or less radical modifications of the original notions, it goes without saying that its results should better not be completely at odds with the standard use of the unanalyzed repertoire.

This preference for an analysis that respects the standard use of a conceptual repertoire can also be given a more rigorous flavor. Program execution explanations, one could argue, are really explanatory. The reason why the same tokens that I'm typing on my computer now appear in the pdf document after the file has been processed, is that my computer implements the relevant program. If they fail to so appear, by contrast, it is because the computer has failed to implement the program. If I try to type the same tokens on the wall, the wall fails to output a pdf document. Any analysis that concludes that the concept of program execution is vacuous, then, must be flawed. Any analysis that fails to mention the entities that typically feature in a program execution explanation, must be flawed.

This argument, without further explanations, is not generally valid. If it were valid as it is, if, that is, one could easily jump from explanatory power to conceptual soundness, no one would have spent years and years on any debate over scientific realism. By the same token, for example, one should argue that luminous ether exists because Maxwell's equations are explanatory when applied to molecules of ether. Nevertheless, it must be admitted that it has some appeal and that it deserves some attention.

Moreover, and this, I think, is a more serious objection, one can find in the literature attempts to build a notion of implementation that fulfills these desiderata. In various works, for example, Piccinini has proposed a non model theoretic approach to implementation. The view that resulted from his analysis, the *functional view of computation*, complies with the above desiderata. I think that such a view, or any other that respected more closely the intuitions of the relevant experts, would be for this reason preferable to the one that I have advocated. I do not exclude that in the future some tenable non model theoretic approach to the problem of implementation will succeed. In this case, my work would loose much of its interest, and I would probably abandon the conclusions that resulted from it. At the present, however, as I shall argue in the next section, I don't think there exists in the literature a satisfactory functional account of implementation. I therefore believe that we should suspend our judgment on the matter and wait to see how the various approaches manage to cope with the respective difficulties.

5.4 A critique of the functional account of implementation

5.4.1 Where do MTAs to implementation come from?

A preliminary question that one needs to ask is why MTAs have been adopted in the first place. Why, that is, has anyone decided to take such an unnatural route to understanding what real computations are. I think it all started when the concept of computation was applied to domains that did not originally pertain to it.

As the possible uses of computers grew, old designs were adapted to complete the new tasks, often radically changing the mechanisms exploited for implementing certain functional parts, or even the entire architecture implemented: the design of computers, with their increasingly various architectures has evolved following the standard problem solving routes that one observes in any other engineering tradition. But for how innovative a new machine can be, it will always be similar to some previous one in some relevant respects.

The wheels of a modern car are the product of a long history of wheels that dates back from before the invention of engines. Their design has co-evolved with that of all other parts of terrestrial means of transport, and it is this history that keeps the functional organization of cars together: when the design or the material out of which a part of a machine is changed, while most other parts remain the same, we say that what is preserved is the part's function. As this, in principle, can be done with all parts, we say that the car has a certain functional organization, i.e. that it can retain its architecture and workings while undergoing material changes in all its parts. This is all well, when our problem is building cars or computers.

The trouble begins when we suddenly abandon completely the history of a certain kind of engineered items, and hope to apply the same functional categories to a domain that is alien to the original one. The problem is relatively straightforward when this applies only to a part of the whole item. An alien object, even if it doesn't belong to a history of wheels, is a wheel if it can be attached to a wheelless car in such a way that the result is similar enough, in the relevant respects, to a car. But how can we ask whether an alien object is a car, regardless of the uses one can make of it? What are the relevant similarities that should be taken into account?

Now, the notion of program execution has been applied as an explanatory tool outside of its original domain of application. Although in most cases the architectures proposed for modelling cognition were and are tested by using ordinary computers, where the concept applies in a smooth way, they were meant as genuine models for the cognitive sciences, and for the relation that obtains between the brain and the mind. It is therefore of the uttermost importance to check that the concepts and explanations that proved to be so successful when applied to the domain of computer design, i.e. the conceptual repertoire that I have called the ontology of computer science, can be safely applied in the wild.

One could object that there is no need to check whether the ontology of computer science can be applied in the wild, for, by its very nature, computational explanations never make any reference to the details of the implementing mediums: computational explanations abstract from the details of implementation. As there is no privileged domain of applicability, of course computational explanations can be applied also in the wild! But if the arguments that I have discussed in the first two chapters of this thesis are sound, this is simply a myth. What is controversial, of course, is not whether computational structures are abstract entities (which is certainly true), but whether the relevant abstractions can, themselves, be abstracted from their standard domain and practices of applicability (these, as we have seen, for example, have been argued to include human purposes).

The notion of clock, for instance, abstracts from any specific implementing medium or architecture, but no one would seriously apply the notion of clock in the wild physical world if not in a metaphorical sense: this is because the abstractions that are relevant for fixing the notion of clock do not abstract from human purposes.

Whatever one thinks about v-arguments, it is philosophically safe to say that the abstract nature (in the relevant sense) of computation is an hypothesis, not an a priori truth: let's call it the abstractness hypothesis (AH henceforth). Those who believe that AH is true, while believing that no MTA to implementation has succeeded[5], coherently, will believe that what explains the failure of MTAs is their methodological inadequacy.

Now, model theoretic approaches were introduced precisely to frame the debate over observer-relative arguments.[6] So anyone who thinks that they are not methodologically adequate must show that there exists an alternative picture of implementation that allows to block these arguments, at least in their non model-theoretic versions. We have seen that the only alternative picture of implementation available at the moment is the functional picture. So the question becomes, does the functional picture allow to block v-arguments? I will address this issue in the next sections.

[5]We mentioned in the first chapter that some scholars, e.g. Chalmers, accept the methodological validity of MTA, while denying that v-arguments are correct. The view that I am discussing here, in stead, accepts that MTAs render the problem of implementation intractable, but deny that we should worry about it.

[6]In a sense, Putnam's MTA allows to make observer-relativity objections more precise: it can be thought of as a technical version of Searle's argument

5.4.2 Do v-arguments entail that Turing's analysis is wrong?

I have argued that MTAs to implementation entered the scene of the foundations of computer science and AI when the concept of implementation was applied to a domain that was not the original one. We said that the possibility to do so was taken for granted thanks to the firm belief in the truth of AH. One could argue that the truth of AH could not be questioned without also questioning the validity of uncontroversial analyses, such as Turing's analysis of computation.[7] Turing's analysis,. in fact, was not meant as a study of the computational capacities of artificial machines[8], but as an analysis of real human computing.

So, it appears, the abstractness of computation (AH) was exploited, and thus conceptually tested, since the very beginning of computational cognitive science. Or should we think that there is something wrong in Turing's analysis? In other words, should those who doubt about the validity of AH also doubt about the foundations of computational functionalism? As a matter of fact, we have seen, many of those who question AH (whether adopting a MTA or not), do also question the tenability of computational functionalism (Searle and Putnam are good examples); but this is by no means forced upon them.

Turing machines, in fact, are not real machines, they are abstract mathematical (logical) entities. The states of a Turing machines are logical states, its inputs and outputs are logical inputs and outputs. Turing described the psychological states of a human computer by means of the logical states of his machine. This was an astonishing achievement, and it would be so even if brains or computers did not exist. His

[7]Famously, the analysis was first published by Turing in 1936 ([90]), and was to become one of the most influential papers of the twentieth Century.

[8]Artificial machines of that kind didn't even exist at the time.

analysis, in other words, is silent with respect to the relation between computational structures and their implementations: one could argue that even angels, when they compute an addition, instantiate a Turing machine. In a sense, what Turing did, was trading a psychological description for a structurally identical, more abstract description (one that could also be used to characterize effective procedures).

Moreover, it should be noticed that what Turing has modelled, i.e. a human computing an arithmetical calculation, is a behavior that we typically type-identify by using our representational apparatus. Under what conditions do we say that a human is executing a given arithmetical calculation on a sheet of paper? Surely we do not only require that what the human writes and reads on the sheet of paper be characterizable as belonging to a finite number of symbol types. We also require that the symbols written and read actually refer (with respect to our representational system) to natural numbers. We could express this by saying that it is essential to the concept of *human executing a calculation*, that he or she is capable of manipulating symbols that (to us) refer to numbers. Turing's analysis suggests that the means of execution of humans do not require the use of this representational repertoire (because each step requires nothing but a simple formal symbol manipulation). These means for executing a calculation, therefore, can be simulated by a machine. The soundness of Turing's analysis only depends on whether the behavior of the machine is or isn't indistinguishable from that of a human computer, from the perspective of the human interrogator in the simulation game. As he succeeded at that, his analysis is untouched by any v-argument.

To argue that Turing was wrong, one would have to argue that when humans multiply two numbers using a pen and a sheet of paper, in spite of all appearances, they do not go through a number of simple stupid steps, but guess the final result by means of some complex intelligent capacity. We have seen that the concept of step is hard to characterize in physicalistic terms, but it is certainly not hard to characterize at the level of psychological explanation.

The validity of Turing's analysis, in sum, does not entail that it is possible to deduce from it a coherent notion of implementation. The reason why Turing's analysis appears to contain a notion of implementation is that its fundamental elements, although defined at a functional level, have a physical "flavor". The notion of machine, that of state, or that of tape, or input or output are all functional notions that appear to possess a straightforward physical correlate. But Turing himself acknowledged that the notion of implementation could not be made independent from an observer. In characterizing the realization of his machines, in a famous paper of 1950, Turing described them as essentially belonging to the category of discrete state machines.

> These 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.[9]

What does Turing mean by "profitable"? Does the fact that they can only be *thought of* as being discrete state machines (DSM) entail that they are not *really* DSM? In replying to a potential objection to his analysis, Turing makes it clear that

[9]Turing [48], p. 36.

what is relevant for the application of computational concepts in the wild, is indistinguishability of behavior. The objection in question proceeds from the observation that the physical processes in the brain must be continuous. How can Turing's analysis, then, be applied to human cognition? To simplify his point of view, Turing contrasts the physics of a continuous machine (a differential analyzer), with that of an (ideal) discrete state machine (a digital computer). The objection, in this case, is that, if asked to perform a computation, the differential analyzer would come out with more precise answers than those of the digital computer. The interrogator in the game would therefore be able to tell which is the digital computer and which the analyzer. According to Turing the objection is blocked by the observation that:

> It is true that a discrete machine must be different from a continuous machine. But if we adhere to the conditions of the imitation game, the interrogator will not be able to take any advantage of this difference.[10]

So, it appears, what grounds the notion of implementation, according to Turing, is behavioral indistinguishability. This is perfectly fine for defining a scientific programme, for whether the behaviors of two agents are distinguishable or not by a human interrogator is an empirical fact that needs no further specification. Moreover, it is consistent with the analysis put forward in this work. The notion of implementation of a machine that performs a computation, in fact, is thought to be relative to the judgement of the interrogator. The two behaviors must be semantically addressed, to be compared. Suppose, for example, that the machine competitor in the game output the results of its computations in some unknown, or unrecognizable symbolic format. This machine would not pass the test, and could therefore not be said to

[10]Turing [48], p. 47.

objectively implement the right machine table. But if the human competitor, as well as the interrogator, were taught the unknown symbolic code that the machine uses, then the machine would suddenly pass the test, and the human could be argued to implement that computational structure.

In sum, phycological states, per se, are just as abstract as Turing machine states. So the issue of implementation is logically totally uncorrelated with Turing's analysis (or, for that matter, with any other functional computational explanation). Of course the multiple realizability of computational structures is crucial for the allure of computationalism, but the latter, as I have argued throughout this thesis, dose not entail AH.

It should be mentioned that Putnam has indeed argued that his model theoretic v-argument could be applied just as well to any functionalist psychological explanation. As functionalist psychological explanations are structurally identical to their computational correlates, Putnam's argument can be applied, mutatis mutandis, also to them. If this was true, then, Putnam's argument could be applied also to Turing's analysis of computation. In this case, however, the unwanted models would not be unwanted implementing objects, but unwanted abstract psychological structures. Here, however, I am interested in exploring the consequences of v-arguments for the notion of implementation of computational structures. I am therefore not concerned (still less committed) to the validity of Putnam's v-argument when it is applied to generical functional psychological theories.[11]

[11]In fact, as I have argued in section 3.3.3, I believe that Putnam's argument cannot be so applied.

5.4.3 The functional view of implementation in the wild

Anyone who thinks that v-arguments fail to make a point, because they are not valid or (and this is the view that I am considering here), because they are irrelevant, must think that computers are a natural kind.[12] I have discussed in the previous sections what characterizes this natural kind, according to the relevant experts. The thesis that the mind is a virtual machine running on a hardware (presumably the brain, or, as we shall see, the brain and its environment), must be tested according to these criteria. To observe the functional account of implementation at work, here I will briefly discuss how the hypothesis that the brain implements a van Neumann architecture is tested against the relevant neurophysiological data.

Recall that the first thing to be checked is that the appropriate ontological furniture is in place. At the most basic level, we must check that memory cells exist that are apt for hosting information encoded in binary strings and that can be accessed in the way that was discussed in the previous sections. So, are there bistable devices in the brain? And if yes, is there a machine language that encodes instructions that can be executed upon them? If things were so smooth for the functional view as it is sometimes claimed, it should be relatively easy to answer this question, but, as it turns out:

> A definitive answer to this question is surprisingly hard to come by.[13]

The most natural place to look at, to start with, are single neurons. The "all or none" nature of the action potential of neurons, in fact, makes them prima facie good

[12]This is true under the assumption that computationalists take a physicalistic metaphysical stance.

[13]Wells [95] p. 199.

candidates to implement unit cells. The plausibility of this hypothesis seemed so uncontroversial that, in a ground braking work of 1943, McCulloch and Pitts claimed that:

> The response of any neuron [can be treated] as factually equivalent to a proposition which proposed its adequate stimulus.[14]

Now, although it is certainly possible to type-identify neurons according to whether they are activated or not, there are considerations that make the hypothesis that neurons (or collections of neurons) act as flip-flops implausible. 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 behavior of neighboring 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).

These and other considerations are now believed to weaken the hypothesis that neurons could be used to host a binary symbolic code.

At the time when McCullch and Pitts were writing, much less was known about the physiology of neurons than it is now, but similar considerations as the ones made above could be made also then. That some properties of neurons do not fit with the nice picture of them as potential memory cells is a fact that was well known also then.

[14]McCulloch and Pitts [56], p. 352, quoted in Wells [95] p. 199.

What counts is whether these "deviant" properties can be neglected or disregarded for the purpose of computational analysis. 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 bistable 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.[15]

Nowadays, instead, the most accepted opinion is that the above mentioned "deviant" properties block a plausible treatment of neurons as binary memory cells:

> The principles of computer memories can hardy be realized in biological organisms for the following reasons: i) All signals in computers are binary whereas the neural signals are usually trains of pulses with variable frequency. ii) Ideal bistable circuits which could act as reliable binary memory elements have not been found in the nervous systems.[16]

Now, on one side, these considerations appear to support the idea that the functional conceptual repertoire of implementation is in no trouble. If we were wondering whether the brain implements a van Neumann architecture, the functional account gives us a reliable criterion:

[15]McCulloch and Pitts [56], p. 352, quoted in Wells [95], p. 199.
[16]Wells [95] p. 200.

> If the brain does not contain bi-stable circuits as its basic information storage elements the fundamental idea that the brain is a universal computer becomes implausible. The primary reason for this is that if type identifiable symbols do not exist in the brain, it lacks an independent code in which the equivalent of its standard descriptions can be expressed.[17]

Notice that this (and similar) arguments make no reference to the truth conditions of a formal sentence like $Imp(S, A)$. So now we are in the position to take a closer look at the general objection that this chapter ought to address. If there is a (functional) account of implementation that allows to apply the relevant concepts in the wild, i.e. to a domain that is not a priori known to be "friendly" to implementation, why should we worry about v-arguments, and why should we waist time with model theoretic approaches to the issue?

The objection is sound and it should be taken as a plausibility argument against the standard conclusion drawn from v-arguments. Those who think that v-arguments show that the concept of implementation is vacuous should also explain where computational properties derive their explanatory power and why they appear to be safely applicable in most relevant domains. This, however needs only apply to some conclusions drawn from v-arguments. The general aim of this work, for example, was precisely that of providing a picture of implementation that would retain the explanatory power of computational explanations, while learning a lesson from v-arguments. The stance that I have taken, in fact, if valid, would only block unwanted implementations, not the wanted ones. Unwanted implementations are never seriously taken into account by the relevant community of experts. No one has ever tried to exploit

[17]Wells [95] p. 200.

the computational powers of a wall.

I think, on the other side, that it is hard to deny that the ontology of computer science has a peculiar status among other scientific conceptual repertoires. It is standard in all empirical sciences that the models are never neatly, perfectly observed to apply in the wild. There are always some aspects that must be neglected to claim that a model succeeds at describing the workings of a system. But usually there are strict shared criteria for selecting the relevant properties. Scientists manage to agree on how to interpret even the most indirect and complex observations (think of how we individuate elementary particles). They agree on what should be neglected, or considered as noise.

Computer scientists, instead, sometimes encounter serious difficulties in agreeing as to how to apply their conceptual repertoire to non-artificial mechanisms. We have seen how the same data about the nature of neurons elicit discordant conclusions about their functional status.

The comments quoted above are an example of the kind of physical facts that are considered relevant in deciding which computational architecture the brain implements. They are certainly part of a respectable scientific enterprize, but it is also clear that there is not (not even ideally) an experimentum crucis that could be devised to conclusively decide for one option or the other. As a matter of fact, the plausibility of one option over others is indirectly derived from the general adequacy of a model in explaining cognitive behavior. As the latter is the goal of cognitive science, no one could object that there is some methodological mistake in the way cognitive scientists apply their concepts.

However, it is also clear, to me, that cognitive science would greatly profit from a better understanding of implementation. In the early years of AI and computational cognitive science, not only were implementational details considered irrelevant, they were also thought to be trivially recognizable, once a given architecture was proved to be explanatory efficacious. We can now say that things are not so.

Consider the above argument against the idea that the brain implements a van Neumann architecture. The basic structure of it is:

1. van Neumann machines make use of type identifiable symbols to form the "code in which the equivalent of its standard descriptions can be expressed".

2. Real van Neumann machines use bi-stable circuits as their basic information storage elements. Hence,

3. If the brain does not contain bi-stable circuits as its basic information storage elements, the brain does not implement a van Neumann machine.

This is a perfect example of how Piccinini's functional account should be applied. Notice that what is required is not only that some (any) bistable device be found in the brain. Notoriously, in fact, bi-stable devices (at least in classical physics) are not natural kinds: whether something is or isn't a bi-stable device depends on what properties we decide to neglect. This, by itself, is not a problem. Most concepts in the empirical sciences, in fact, are of this kind. For example, there certainly is no strict physical criterium for characterizing a biological cell. This, however, does not pose a problem for biology, for there are other, more mundane, criteria to tell whether something is or isn't a cell.

The problem here is that the more mundane criteria, in the case of computational concepts, presuppose that abstract concepts such as "information storage element", or "symbol", can be safely applied in the wild. Notice, in fact, that what appears to be relevant for the argument is not whether there are elements whose states can be identified as belonging to two possible types, but whether these elements are used as basic information storage units. To conclude that an item is used as a basic storage unit, one must make sure that it contains a symbol or a string of symbols. So, unlike what is claimed by the functional account, it is not that certain physical items support, or implement a symbol code, by virtue of some specifiable monadic or relational physical properties. It is, I argue, the presence of the code and of its symbols that elicits the application of computational concepts. As at the moment there is no uncontroversial functional characterization of symbol, or of code, functional accounts of implementation are conceptually incomplete.

It is possible that one day some functional account will succeed. It will have to provide for a precise criterium of identification for the ontology of computer science. This may or may not require that a functional account of symbols is given. In any case, until such a functional picture is provided, it is interesting to explore other options.

5.5 Is it possible to apply the semantic implementational schema to universal Turing machines?

Another potential objection related to the adoption of a MTA to implementation is that my treatment appears to be applicable only to finite state automata. All the model theoretic v-arguments discussed in this thesis, in fact, are applied to the implementation of FSAs. Also the implementation schema that I have proposed at the end of the last chapter applies only to the implementation of FSAs. But most of the interesting applications of a computational conceptual repertoire, including the computationalist theory of the mind, make use of more complex architectures.

No one, for example, has ever suggested that the brain implements a FSA. The typical proposal is that the brain implements a finite universal Turing machine. It is easy to see how this could turn into a serious objection to my (and similar) treatments. Even setting aside, for the sake of the argument, the issue of the validity of v-arguments, one could reasonably observe that no v-argument has ever been applied to Turing machines, or to any other more complex computational architecture. Are there good reasons to think that they could be so applied? Has anyone shown, or argued that anything could be treated as the tape of a universal Turing machine? Similarly, one could ask how my implementation schema could be applied to these architectures. If it turns out that it cannot, the whole treatment could be argued to be of no use for the foundations of computational cognitive science.

Now, there are two reasons for having applied my schema only to FSAs. The first one is that the schema was devised to block v-arguments, and these are applied in the literature mainly to FSAs. The second reason (which also explains why v-arguments generally confine themselves to FSAs) is that FSAs are by far the simplest computational structures. If we were to apply my schema to universal Turing machines, for example, we would have to provide for an exact functional characterization of their realizations. Such characterization exists, and it is rather complex. Given the time framework of this work, I have preferred to concentrate on the simplest case. Here I shall briefly outline how my schema should be applied to more complex architectures.

Some considerations are in order. The general objection raised in this chapter to functional accounts of implementation is that they ultimately rest on an encodingist picture of information. The tacit assumption is that just like there is a physical correlate for the mathematical notion of magnitude (we have discussed this in chapter 2), there is a physical correlate for the mathematical notion of string of symbols, or for that of code. The inputs to a Turing machine (universal or not) must belong to an alphabet. The encodingist picture consists in assuming that the concept of *alphabet* can be safely applied in the wild. Of course real alphabets do exist, and the concept, therefore, has several realizations. There is no question that the Greek alphabet, for example, exists. What is questionable, however, is whether it is an instance of a natural kind. The point of view that I have defended in this work is that alphabets, and thereby strings of symbols, are indeed instances of a natural kind. What I have denied is that they would be instances of a natural kind even if there weren't representational properties fixing what is or isn't a string of symbols.

> On the one hand, we have a very elegant set of mathematical results ranging from Turings theorem to Churchs thesis to recursive function theory. On the other hand, we have an impressive set of electronic devices that we use every day. Since we have such advanced mathematics and such good electronics, we assume that somehow somebody must have done the basic philosophical work of connecting the mathematics to the electronics. But as far as I can tell, that is not the case. On the contrary, we are in a peculiar situation where there is little theoretical agreement among the practitioners on such absolutely fundamental questions as, What exactly is a digital computer? What exactly is a symbol? What exactly is an algorithm? What exactly is a computational process? Under what physical

conditions exactly are two systems implementing the same program?[18]

The labelling scheme that I have proposed ensures that the right conditions obtain for the presence of symbols. It dose not purport to replace or contradict the standard functional characterization of computational items altogether: on the contrary, it ensures that these functional characterizations can be applied in the wild in a non-vacuous way.

5.5.1 The requirements of a labelling scheme for universal computers

On the one side, we said, a universal Turing machine is just like any other Turing machine, in that it has a finite alphabet and finitely many functional states. On the other side, its peculiarity is that it can "simulate" any other machine. Although each machine has only a finite number of symbols, there are virtually infinitely many codes that could be deployed. The behavior of each machine to be simulated (target machine) is described in terms of its specific coding scheme. As it is impossible to provide our universal machine with infinitely many coding schemes, before a machine can be simulated its coding scheme will have to be "translated" into one that the universal machine can interact with.

Fortunately the names of the inputs, of the outputs and of the states of a machine have no effect on the functional characterization of their bearers. This allows to take a first step towards the wanted "translation": we simply have to rename each item in the target machine table into a conventionally chosen code. A blank, for example, is conventionally called $s0$. The other input and output symbols are named $s1$, $s2$, etc. Similarly, the states are renamed as $q1$ (in Turing's works, by stipulation,

[18]Searle [82], p. 205.

this is always the name of the starting state), $q2$, etc.. This allows to rewrite any machine table into a uniform coding format (the description of a target machine, once translated in to the universal format, is called its standard description).[19]

When the simulation begins, the tape of the universal machine contains only a string of symbols that encodes the standard description of the target machine. By convention, the target machine starts in state $q1$ on a blank square (#). The UM, governed by its complete configurations, executes cycles of functional processes and actions each of which corresponds to a step executed by the target machine. Each cycle consists of executing a flow of processes that belong to nine functional types. The first function executed at the beginning of each cycle, for example, is **b**. It prints the starting configuration of the target machine $(q1, \#)$ immediately on the right of the standard description. To do so, it uses a leftmost symbol finder to locate the symbol "*", which signals the end of the standard description.

Here I shall not get into the details of the cycle, as these are irrelevant for our discussion. What will be said of the **b** functional component will be applicable to all the other eight ones.

So, for a physical system to implement a UM the following conditions must be met:

1. There must exist a (the equivalent of a) tape that is scanned one (equivalent of a) square at the time.

2. There must exist a set of items that belong to finitely many types and that can

[19]As a matter of fact, for practical reasons, the decimal notation for naming states and inputs should be replaced by one that uses unary numerals. State q_i, for example, is usually written as a D followed by A repeated i times. As this has no conceptual relevance, however, in our discussion we will maintain the decimal format.

act as symbols.

3. The coding scheme provided by 2 must be suitable for encoding the standard description of any target Turing machine.

4. The functional organization of the physical structure must be such as to realize each of the nine functions described by Turing.

Let us start with the tape. What is the "equivalent of a tape"? That is, what is relevant (and what isn't) in establishing the relation of equivalence among all the tokens of the type "tape"? It appears that what makes the difference is the possibility to read (and write) symbols. A tape is anything where symbols can be read and written one at the time by a central processor. So the question "which is the tape" is meaningful (and has a definite answer) only if the question "where can the symbols be read and written?" has a definite answer. Of course the symbols need not be currently present on the tape (a tape whose squares are all currently blank is still a tape): what counts is the disposition to use a physical item as a tape.

5.5.2 What does it take to implement a universal coding scheme

Let us turn to the presence of a coding scheme. When do some physical items implement a coding scheme? One sometimes reads that what is necessary and sufficient is that there are tokens that can be classified as belonging to finitely many types. This, however, is clearly not sufficient, if we want to rule out unwanted arbitrary instantiations of "code". What is necessary is that the elements of the candidate "code" (the tokens) bear certain relations with the finitely many states of the central processor.

If the description of a machine is given, this requires that in the physical implementing medium there are: 1) type-identifiable items that can be named by the names of the symbols used for the description of the input architecture, and 2) groupings of states that can be each named by the names of the states in the description. The names must be assignable to the physical items in such a way that every instruction in the description is reliably respected by the named items: each state transaction or action prescribed must be reliably caused by the state of affairs named by the relevant complete configuration.

These "states of affairs" cannot be individual physical states of affairs, like the ones that MDSs rigidly describe. The latter, in fact, would never fulfill the requirements of distinguishability and multiple realizability. Each of these states of affairs will instead be a large disjunction of precise states of affairs. Notice that it is precisely the liberalism which must be allowed in the construction of these disjunctions that allegedly makes room for vacuousness arguments. For example, the implementation of a b-function introduced above requires, to start with, that a set of items play the role of a string of symbols. These symbols must be the potential names of the symbols input and output, and of the states of another Turing machine (the target). The string of symbols, in fact, must be the standard description of a target machine. Whatever these symbols are physically, the physical structure which supports them is, by convention, the tape. The physical system, once started, must "find" the last symbol in the string and write two names right after it. These two names must be the potential names of the starting configuration of the target machine, i.e. they must be systematically interpretable as such.

Let us consider more closely what are the requirements that these symbols must satisfy. There are two categories of requirements. On one side, the symbols must be such as to comply with the formal constraints placed by the machine table of the UM. They must, for example, interact with the central processor in such a way that as the machine is started, a b-function is implemented. On the other side they must be the potential names of a description of another machine. Once the mappings from physical to computational items have been proposed, the first requirement places physical constraints on the system. It prescribes what should be physically caused by what. Once the candidate physical correlates of the symbols have been chosen, the second requirement, instead, places syntactic constraints: it must be possible to establish a one-to-one correspondence between the elements of the string of symbols written on the tape and those of the target machine table.

Notice that the cycles of a universal machine are not dedicated to simulating one particular target machine. Once a universal code has been introduced, the description of any Turing machine can be written on the tape. Each cycle will simulate one step of whatever target machine (table) that is input. Of course, the universal code is relative to the universal machine table. The particular universal code adopted by Turing, for example, is such only relative to the specific universal architecture that he designed. More precisely, it is not sufficient that every target machine table can be "translated" into the same (universal) code. The code must further be readable by the universal processor. A code that is universal in the sense that every human can translate into it any machine table, but that is different from that that the universal machine uses, would be useless.

If we are to build a universal machine, these constraints are rather entangled. It would be rather odd if someone devised a universal code and then built a machine that used another one. The description of the workings of a universal machine, although complex, are perfectly suitable for building artificial devices. The code and the parts of the machine to be built, co-evolve from an engineering point of view. So, for example, if a bistable device does not allow to host the wanted binary encoding, it is replaced by another one. If, viceversa, the devices do properly encode the instructions that are to be executed, then, by definition, the code is a code, the tape is a tape, the processor is a processor, and the bi-stable device is a bi-stable device. Under these circumstances, we say that the object that was built implements the given computational structure.

The circumstances in which artificial devices are built are particularly apt to avoid vacuousness. The tasks, at each step of the construction of a machine, are rigidly constrained. The task that interests us most, in this context, is the task of ensuring that a given part of the real machine "encodes" the correct information. The high level code (i.e. the one that the programmer uses) in these circumstances, is given. Also the elementary parts are largely given. It is a question of putting everything together so that the right encodings are in place. If things don't work, at one stage, then new parts might be introduced or some parts might get replaced with others, or the code might be adapted to some specific practical need. What is relevant, however, is that the final wanted result, i.e. the I/O behavior of the machine, is determined in advance in a code specific way. It is predetermined what physical structure should (tentatively) implement the "tape". It is predetermined what physical structures should tentatively implement the symbol strings that are input or output.

In short, computer designers are never faced with the task of building any universal computer, using any code and any physical item for realizing any functional part. But our task, here, is to evaluate the idea that computational explanation are naturalistic explanations, i.e. whether computing devices are natural kinds. In particular, in this section I am evaluating the idea that real universal machines are a natural kind. The philosophical constraints, given this task, are very different from the practical constraints faced by a computer designer.

Things are more complicated when the empirical question is whether a natural object (such as the brain, or worse still, any object) implements a universal Turing architecture.

We said before that there are two kinds of constraints, the implementational constraints due to the requirement that all the functional components of a UM be implemented, and the syntactic constraint for the code, due to the requirement that the code adopted be "universal". These two constraints are logically uncorrelated, as the functional characterization of each functional component is not dependent on the code adopted. Of course, if a random (non universal) code is input into a machine that implements all the functional components of UM, the result will not be a simulation of a target machine. The cycles, in fact, might not halt. If no "$*$" is input, for example, the b-function will never halt, and it will never proceed to the other steps in the cycle. In any case, the behavior of the machine will in general be unintelligible (i.e. uniterpretable). This, however, doesn't make the two constraints logically correlated.

Now, the requirement that the code be a universal code, is perfectly intelligible from a mathematical point of view. But if, as we have argued, there is no physical counterpart to the mathematical notion of symbol string, or code, then there is not, a fortiori, a physical counterpart for the notion of universal code. Remind that under a functional account of implementation, symbols and codes need not be true representations: they only need to be systematically interpretable. The only viable option, for a functionalist account, appears to be the claim that a string of symbols, or a code, be such only relative to a machine that processes them. But this introduces a circularity in the functional notion of realization. The implementation of a functional component, in fact, is relative to a given input architecture. Would a b-function be implemented if no strings of symbols could be detected and type-identified? The implementation of a given input architecture, in its turn, is relative to the presence of a code. How can a given input architecture be identified if there are no symbols? Now, the circularity is apparent: the notion of code is dependent on that of implementation, and viceversa.

The difficulty in providing a natural counterpart for the notion of symbol (or code), is due to the fact that too many things can be type-identified as tokens of finitely many types, if any arbitrary grouping of physical states of affairs is allowed. Remind that, for example, the attempts to ground the notion of symbol by using the notion of digital scheme, have been argued to be unsuccessful. What is being said now, depends on the validity of these skeptical arguments. But these arguments are not dependent on a MTA to implementation. So, as this chapter is concerned with defending the general approach proposed in the thesis, it is reasonable to ask what conclusions we should expect to follow from the claim that the notion of symbol has no physical counterpart.

So, what is a physical universal code? One could require that the symbols used in building a universal code really refer to the Turing machine tables that the UM ought to simulate. This, provided that a suitable physicalistic account of representation is given, would be a sufficient physical constraint. No one could now argue that too many things could count as a universal code, for real representations are not ubiquitous. But this would be clearly an absurd constraint. It would violate the very idea that stands behind any computational explanation, i.e. that the symbols are meaningless physical tokens identified and manipulated solely on the base of their shape. It must be their "interpretability" (not their representational properties) that grounds computational explanations.

The labelling scheme proposed in this thesis concedes that the semantic properties of the symbol system need not really refer to their intended interpretations, but it denies that a coherent notion of realization of symbol system can be given without assuming that some structurally isomorphic semantic properties be instantiated by the symbols. Only then can a naturalized notion of "interpretability" be applied. I have argued, in fact, that something is "interpretable" as meaning something else only if it already means something. A morse code, for example, is a code only if one already knows what "S" is and what "..." is. Codes, in short, only establish connections between meaningful entities. A potential misunderstanding is that one could think that "meaningful" must mean interestingly meaningful. But this is clearly not the case. "S" means the same as "the 19th letter of the roman alphabet". A physical item is an "S" only if it refers to the 19th letter of the roman alphabet. The same goes for "...", which means the same as "three dots in a row". Three pulses encode an "S" only if, together, they mean (to the telegrapher, in this case) the same as "three dots in a row". Notice that the three pulses need not refer to an "S", this is why we need a code. But if three pulses did not refer to "three dots in a row", if they did not refer to anything, they would not be tokens of a symbol. A single pulse, for example, might be arbitrarily sectioned into three stages. A careful telegrapher might systematically identify which single pulses are so sectioned. Does this make a single pulse interpretable as "three dots in a row"? Only if it means "three dots in a row" to the telegrapher.

Similarly, the items that realize a universal Turing machine code, need not refer to machine tables (it is the code's responsibility to ensure that this is the case) , but they must refer to something. With respect to the representational apparatus of the computer designer, for example, a physical item that realizes the symbol referring to the starting state of the target machine, must refer to (in our encoding) "q_1", i.e. to a "q" followed by a small "1". Of course, this is not sufficient, for if the UM does not treat a token of "q_1" in the suitable functional way, the mere fact that the item means "q_1" to the designer, doesn't guarantee anything. However, if the physical item that is intended to refer to the starting state of the target machine doesn't mean anything to the designer (for example because it is a microscopic state that the designer cannot even individuate), then, even if its functional relations with the other states of the machine can be interpreted as the "right functional relations", the I/O behavior of the UM will not be the right one (to the designer): the real system will therefore not implement a universal machine. In sum, the representational apparatus of the designer must intentionally individuate a code that is appropriate to serve as a universal code. This, for example, can be Turing's universal code.

5.5.3 Requirements for implementing a given simulation cycle

Similar considerations apply to the functional constraints for the implementation of Turing's universal architecture. For a physical structure to implement a b-function, for example, the representational apparatus of the designer must be such as to: 1) individuate what counts as the symbols, 2) recognize what counts as a string built out of these symbols, 3) recognize what counts as "after the end of a string".

The simulation cycle as it was discussed by Turing comprises a flow of processes that is articulated into nine functional components (one of which is **b**). After the machine has processed the initial string of symbols through **b**, the information is passed to the second component, called **anf**. This function marks the configuration in the most recent complete configuration. **Anf** initiates a cycle through the other components (sometimes passing the activity back again to **anf** itself). A cycle terminates when the last function (**ov**) clears the tape and makes a final transition to **anf** where a new cycle begins. The result of a single cycle corresponds to one step of the target machine.

None of these functional components corresponds to a primitive operation. Each of them, to be executed, must be further functionally decomposed in a way that is similar to that discussed in section 5.3.1, down to the execution of a set of machine language instructions. We shall not here get into the details of these components, or of their ultimate functional decompositions. This means that it will be impossible to provide a complete labelling scheme like we have done in chapter 4. The degree of complexity of such a labelling scheme would require a dissertation of its own to be spelled out. Here I am simply concerned with providing an outline of how this should be done. In particular, I wish to address (in principle) the additional difficulties of universal computers.

The syntactic constraints discussed in the previous section for the universality of the code are relative to the representational apparatus as well as to the presence of a machine that goes through a simulation cycle (of some kind or other). The constraints discussed here, instead, are the requirements for the particular architecture that Turing suggested for implementing a UM. If the first set of constraints are met relative to the second set of constraints, then the physical system objectively implements Turing's UM.

5.5.4 What would an intentional labelling scheme for a UM look like?

There are two difficulties in applying my labelling scheme to universal computers. None of them, I believe, is insurmountable. One has to do with expanding the schema to Turing machines (of any kind). The second is the application of this extension to universal Turing machines. The crucial factor to notice, in addressing both difficulties, is that my labelling scheme does not propose a criterium of implementation that is entirely alternative to the standard functional one. The semantic picture is devised to conceptually ground the notion of functional realization, where this makes references to concepts that (in principle) can be instantiated too broadly. One such concept is the realization of a string of symbols. This feature of the semantic picture is illustrated by the labelling scheme proposed in chapter 4. Remind, in fact, that the notion of implementation, there, is grounded on the match between functional and semantic criteria of individuation. All standard computational explanations comply with those desiderata. The semantic picture, therefore, should be considered as an explanation of the efficacy of computational explanations, rather then as a criticism of them.

Having said this, the difficulties addressed in this section are of a technical nature. What is conceptually relevant, if one is asking how the labelling scheme would look like, when applied to universal machines, is that there will have to be an additional set of functional/semantic constraints. As I said, in fact, the realization of a UM requires both that a specific architecture be implemented (e.g. Turing's cyclic structure) and that the code used be a universal code (i.e. that it can be used to compose the standard descriptions of all possible Turing machines). Both sets of constraints will require that the relevant semantic relations of equivalence be deducible[20] from the correspondent functional relation of equivalence. Thus, for example, two strings of physical tokens "representing" the same machine table will have to be semantically equivalent. Two tokens that are functionally equivalent in that they both functionally "represent" the same simulated input, will have to be also semantically equivalent with respect to the machine's or to the designer's representational apparatus. In the latter case, the physical system will not, by itself, be objectively the implementation of a UM.

The requirement that the code be universal, moreover, will set syntactic constraints on the representational schema. Consider, for example, two tokens of, respectively, the type "$s1$" and "$s10$", input to a UM. The constraints of the semantic picture would predictably impose that they belong to the same syntactic categories. This requires that all "$s1$"s and "$s10$"s belong to two definite semantic types. The difficulty in applying the schema to universal machines, is that each of the two tokens belong to different syntactic types depending on whether one interprets the UM as a simulator of the target machine, or as a standard Turing machine (which it is). Both

[20]As I have already noted at the end of chapter 4, here the word "deduced" should not be interpreted in its standard logical sense. The relevant semantic relations of equivalence must be empirically observed to be in place in correspondence with the correlate functional relation of equivalence. The relations of implication between the two kinds of equivalence, in other words, must be understood as material implications.

interpretations are always possible for UM's. UM's, like any other Turing machine, in fact, are governed by their configurations. Under this interpretation, then, "$s1$" and "$s10$" are tokens of two strings of respectively 2 and 3 symbols. The machine, governed by its configurations, reads these symbols one at the time, and responds accordingly. Under this respect, therefore, the semantic picture would not require that "$s1$" and "$s10$" belong to the same semantic type. It would not even require that the compound symbols belong to any semantic type at all. No restriction should be placed on the aggregate semantic properties of the compound symbols "$s1$" and "$s10$", for the machine to be implemented. Only tokens of "s"s, "1"s and "0"s, should be respectively so constrained to be semantically equivalent.

But if one interprets the same machine as a simulator of the machine whose table is described in standard format by the string of symbols initially input, then one ought to require that also the compound symbols possess individual semantic properties. In short, it appears that the semantic picture is committed to the claim that the representational apparatus of the implementing system (or user) be concatenative with respect to the elements of the candidate universal code.

Now, one could raise the following objection. The semantic picture is committed to the claim that the notion of implementation is grounded once the appropriate semantic labelling scheme is provided. The configuration-governed universal Turing machine is implemented, according to the semantic picture, if and only if the relevant functional/semantic constraints are in place. Therefore, there appears to be no room for the further constraint of compositionality. In other words, if a set of constraints is sufficient for implementing the configuration-governed universal Turing machine, and this is nothing but the same rule-governed simulator machine (just interpreted differently), what space is left for another set of constraints?

Notice that this is a conceptual issue and not a practical one. All real universal computers, such as the one that I use to write these words, in fact, have inputs and outputs whose symbol structure is compositional in exactly the sense envisaged above. Consider for example the case of program execution presented in section 5.3.1. At the highest level of analysis, the programmer inputs an instruction like: $n = 0$ UNTIL $n = 10$ TRUE DO $n + 1$ ENDUNTIL PRINT n. Now, if the processing machine implemented a universal Turing machine (as opposed to, say, a van Neumann machine, which is usually the case with ordinary computers), it would process one symbol at the time. The programmer knows both what, for example, "U" is (it is a certain key on his or her keyboard and it is a letter of the alphabet) and what "UNTIL" is (it is a word and it means until). If the result of this input is not something that can be recognized as the symbol "10", then one does not believe that she is in front of a UM. Of course, if one is independently told that the machine is a UM, then one can think that she doesn't know the right code for it. But to learn the "right code" one would have to deploy a concatenative representational apparatus. This will be needed to compose words out of letters and instructions out of words. If, for example, the machine did print the symbol 10 after executing the input instruction, but did not print the symbol 11 after inputting the instruction $n = 0$ UNTIL $n = 11$ TRUE DO $n{+}1$ ENDUNTIL PRINT n, the programmer would think that she doesn't know the "right code" for that instruction, or, alternatively, that the machine doesn't implement a universal machine.

It might be objected that what is needed is not a representation of the symbols and of the words, but simply the capacity to type-identify them. This, however, as we have discussed, would beg the question of what counts as "type-identification". Just like the concept of discrimination, the concept of identification is (potentially) vacuous outside of the domain of cognitive processes (does a rounded whole in a metal plate type-identifies spheres of a certain size?). But to require that the information-bearing physical structures must provide the input to a cognitive process would circularly presuppose what we are supposed to explain by the notion of computation, i.e. what counts as a cognitive process.

Thus, in all artificial cases of implementation it is the user that deploys the relevant representational capacities. When the concept of universal computer is applied in the wild, instead, we must assume that these capacities are instantiated by the same system that allegedly implements the Turing machine. In the case of humans this is the brain or, as we have seen, the brain and its environment.

In sum, assuming that a double set of semantic/functional constraints is in place does not contradict the standard practise of computational individuation. These two sets of constraints ensure, respectively, that a UM specific architecture is implemented, and that the code used is a universal code. The relation between the two sets of semantic constraints, according to the semantic picture, will have to be such as to instantiate a concatenative representational system.[21]

5.6 Conclusions: model theoretic approaches and cognitive science

Should we take the above considerations about the tenability of the functional view to imply that the conceptual repertoire of computer science is in general inapplicable?

[21]This kind of representational system is discussed, for example, in Millikan [59], p. 224.

Can we use the difficulties encountered in applying these concepts as an argument against the very idea that computational explanations can be really explanatory? Of course not. Had all scientists always waited until the conceptual, foundational issues of their scientific practices were secured, before applying their concepts, we would never have had any scientific explanation. As a matter of fact, the same considerations about the functional view that I have made about the application of computational concepts to the brain, can equally well be applied to the original domain of computer science, i.e. to artificial digital computers. Should we conclude that computer designers should not use computational concepts for building computers?

The explanatory apparatus of computer science, when applied to artificial computers is exceptionally sound. There is no reason to think that future developments of computational neuroscience would not come by some criteria and standards of applicability that are equally unproblematic. Moreover, this needs not wait for the foundational difficulties to be eliminated. Even as things stand now, exploring what neural structures might implement what computational architectures, notwithstanding all the difficulties encountered in doing so, is a most fruitful scientific practice. The functional view, no doubt, appears more promising in this respect. On the other hand, MTAs have not yet (to my knowledge) helped at all in solving the technical problems discussed above.

However, the difficulties in applying computational concepts to brain structures are smaller than the ones that worry the proponents of v-arguments. Cognitive scientist testing the hypothesis that the brain implements a given computational architecture, in fact, have constraints in applying their conceptual repertoire. The field of neuro-computational cognitive science, in fact, has developed, and will continue to improve, a specific heuristics for applying the relevant concepts: there is some degree of agreement about "where to look for what". But what worries the proponents of v-arguments, instead, is the hypothesis that the notion of implementation can be rendered independent from specific physical domains. The claim that intelligence is the same as (any) implementation of a given computational structure, in fact, requires that total abstraction is made from any particular physical medium. This is not a problem for a scientist who is trying to test a computationalist hypothesis on some particular domain of application, such as the brain, or an artificial device. In these cases, in fact, we may assume that some heuristics will come to help to dissolve the ambiguities.

The philosopher of mind, in stead, would like to understand what metaphysical picture of the mind is entailed by the computationalist hypothesis. This requires that the notion of implementation in the wild be conceptually (and not only practically) sound. No empirical medium-specific heuristics can satisfy completely the philosopher under this respect.

In sum, I think that, although foundational issues sometimes help to clarify in what direction a scientific practice should evolve, philosophical and scientific problems usually remain substantially independent from one another. What computer scientists mean when they say that a concept is "applicable", is not the same as "philosophically applicable". Typically, the difference is that while scientists apply whatever best conceptual repertoire that is available, philosophers use (often in the same thoughtless way) their conceptual repertoire to argue that scientific conceptual repertoires are not perfect, hence that they are not applicable. In other words, philosophers, unlike scientists, are not trying to "save the phenomena": they are trying to save the integrity of their conceptual repertoire.

Different goals, predictably, require different means. MTAs were devised to render the debate over vacuousness arguments more philosophically salient and precise. Consider the difference between Searle's non model theoretic v-arguments and Putnam's argument. Many philosophers with a technical inclination or background often treated Searl's arguments as irrelevant, arguing that they are vague and hence inconclusive. The reaction of many cognitive scientists and computer designers to Putnam's model theoretic argument, instead, has been to respond with specific proposals for a notion of implementation that would block it. The intellectual confrontation between expert practitioners and philosophers is in itself valuable and usually profitable, and should therefore speak in favor of the model theoretic methodological stance.

On the other hand, and for very similar reasons, so long as MTAs to implementation do not help us solving some scientific problem, cognitive scientists and computer designers have all the rights to regard them as irrelevant. This work aims at evaluating what positive conclusions should be drawn from MTAs approaches. In sum, these considerations should be taken as pointing at the need to better understand the notion of implementation, not as a criticism of how cognitive scientists use their concepts. Quite on the contrary, what motivated my work was precisely to explain how and why computational concepts can be applied so profitably as they appear to be.

Notice in fact that if my proposal for a semantic individuation of computational properties is valid, this does not entail that computer designers or cognitive scientists should apply their concepts in a different way. The consequence of my view is simply that the validity of a computational explanation cannot be explained without taking into account the semantic properties of the modeler (in the cases where the implementing system does not possess its own semantic properties). Consider, for example, the case of a pocket calculator. A consequence of my view is that the calculator cannot in itself be claimed to belong to a natural kind. It is the semantic properties of the user that fix the relevant constraints, and only the coupled system comprising the user together with the calculator belongs to a natural computational kind. This, however, doesn't entail that the components of a calculator should be individuated differently, or that the calculator should be used differently, or, finally, that the program execution explanation of its behavior is false or vacuous.

Chapter 6

Computational externalism: applications and potential objections

6.1 Introduction

So far, in this treatment, I have concentrated on the proper understanding of computation (from the point of view of its physical realization). In this final chapter, instead, I intend to discuss what consequences my understanding of implementation has for a computationalist theory of the mind.

Because teleological theories proved to be the most suitable accounts of intentionality for my purposes, I call a computational theory of the mind that endorses computational externalism: *Teleological Computationalism.* The purpose of this chapter is then to test Teleological Computationalism against some recalcitrant shortcomings of orthodox computationalism, as well as against some apparent inconsistencies. As a consequence, the issues dealt with in the sections of this chapter will be relatively uncorrelated between each other.

I shall argue that computationalism, as it is construed in my externalist understanding, is free from standard objections. This further corroborates the soundness of my analysis, in the sense that it provides evidence for its consistency with other, independent domains of applicability than computational theory. These applications of my theory of implementation, however, should not be considered as further arguments for a semantic view of implementation (recall that a theory of computation, in fact, should not be answerable to cognitive science).

- I shall start by treating an apparent counterintuitive consequence of computational externalism as a model for the mind. In a nutshell, I shall dissolve (section 6.2) the worry that comes from the following paralogism: (1) computation is observer-relative (v-arguments), (2) mental properties are not observer-relative, hence (3) computation cannot explain mental phenomena. I argue that this apparent inconsistency rests on a misleading understanding of observer-relativity.

- An alleged shortcoming that has been hotly debated in the past couple of decades is the "Symbol Grounding Problem": the (alleged) incapacity of a computational system to ground the meaning of its symbols in a non derivative way. Teleological Computationalism is argued to be immune to it (section 6.3).

- John Searle has raised a now famous series of objections to the computationalist stance. I discuss how Teleological Computationalism can be used to defy these criticisms (section 6.4).

- A major attack to computationalism came from the proponents of an alternative paradigm for understanding the mind: connectionism. I provide a critical analysis of the fight for explanatory supremacy (section 6.5), with particular emphasis on the issue of syntactic constitutivity of thought. It is argued that part of this fight has been staged on the wrong battleground.

- Externalism as a theory of meaning has become increasingly popular over the past decades. Philosophers in the field of the foundations of AI and cognitive science have not been immune to this fashion (this whole work itself can be thought of as an example of this tendency). According to several proposals (including some from the camp of orthodox computationalism itself, as we shall see), symbols and, consequently, symbol manipulation should be conceived as happening "outside" of the cognitive system: in the environment. I discuss (section 6.6) how Computational Externalism can be used to integrate and improve on these proposals.

- Throughout this work, (mathematical) dynamical systems have been used in two different contexts: (1) as describing the physical systems that implement computations, and (2) as a basis for the contrastive analysis put forward in chapter 2. Here (section 6.7), instead, I shall discuss the thesis that dynamical system theory can provide an alternative conceptual repertoire for modelling the mind. It is argued that dynamical system theory, rather than an alternative means of description of cognitive phenomena, should be used to provide a precise description of the relational physical properties that instantiate computational ones.

6.2 Computational externalism and the observer-relativity of mental properties

The treatment of computation proposed assumes that observer-relativity arguments are sound: it was an attempt (hopefully successful) to draw less than dramatic conclusions (for the computationalist stance) from these arguments. So, what came of observer-relativity? Are teleo-computations, or externalist computations in general, observer-relative? Consider again the fictitious example of the XOR bacterium. It might be that it is possible to segment the inputs, outputs, and state-space of the XOR bacterium in the example so as to argue that under that interpretation it is implementing the Wordstar program. However, if there are no teleo-representations input and output to support the very meaning of that claim, or if (in the case there were such representations) the abstractions required to reach the computational level were such as to disrupt the status or the meaning of these representations (for example because the former require a minimum tolerance of error that is greater than that of the latter), the computational model would not be honest under our understanding.

Notice how, as I promised, teleological theories allow us to make our proposal consistent with the principle of multiple realizability. A completely different system than our bacterium (perhaps an artificial device), made of different stuff, or living in a different environment, and obeying different physical laws, or simply differing as to the time factors and errors tolerated, could well implement the very same function (XOR). This is achieved by requiring that the properties on which intentional ones supervene (as specified by teleological theories, according to the definition proposed in section 4.4.2) be relational properties of the tokens and of the respective environments (as opposed to intrinsic, encodingist or internalist properties).

A consequence of the treatment proposed is that the same physical system implements (or doesn't implement) a computational structure depending on the obtaining of certain external physical conditions: those conditions that must obtain for its inputs and outputs to be representations, and for these representations to be preserved under any variation of physical quantities that leaves the computational abstractions intact. This is in keeping with the intuitions behind v-arguments: being the implementation of a computational structure is not an intrinsic property of a physical system. So, is the property of implementing a computation observer-relative?

In order to get rid of an apparent counterintuitive consequence of this understanding we can devise the following thought experiment. Computers are paradigmatic cases of implementations of a computational structure. Imagine that the symbols input and output to and from a digital computer, although finite in number, were printed so small that no human being could read them. Or imagine that, although all the tokens of the symbols were large enough to be seen, we did not have the capacity to type identify them (recognize the tokens as belonging to a symbol-type). In either case, we wouldn't call that (whether human or machine) a "computer". Isn't this, then, an essential requirement for something to be a computer: that someone is able to identify an input-output behavior by fixing what should count as a representational item?

In my view, it should be clear, the microscopic computer is not a computer, for it doesn't interact like one with any representational system. Recall, in fact, that the digitality of a system has to do with the fact that there are "reliable" procedures for applying input and measuring output within the operating limits of the system and that there are finitely many distinct, discrete values given those limits. However, if something does interact like a computer with at least one representational system (the user) then not only would that something be a computer, but the finite automaton describing its behavior, relative to the representational labelling scheme, would be objectively realized by it. The behavior that the FSA would objectively describe, is not the "physical" behavior, whose description would require a complete set of magnitudes measurable with infinite precision, but the behavior that is observed using the observer's representational apparatus and limited epistemic capacities.

One might object as follows. There is a microscopic computer. No one has ever been able to "appreciate" its computational powers (how the device came to be made needs not interest us, it could be a random miracle, for what we know). Suppose now that someone invents a microscope that is so powerful as to display and use publicly the computational powers of the machine. Aha! Then it did have computational powers, it is just that we couldn't see them before! Isn't it rational to think that it had these capacities all along, rather than thinking that it acquired them all of a sudden when we first observed it with the microscope? After all, the observation (unlike what happens in quantum physics) didn't change the machine in the least: how can it have made any difference as to what determines its computational status?

It should be clear that a consequence of the proposed understanding of computational properties is that they (as well as semantic properties) are secondary properties. To say that the microscopic computer implemented the computational structure even before that someone interacted with it, is like saying that Mars was red even before someone looked at it. Sure, under physicalistic assumptions, we think that there are physical properties that ground the color of Mars, but we don't believe them to be instantiated by Mars alone: we think that they are instantiated by triadic, relational properties of Mars, of the environment, and of the representational system that observes it. So the computer was not computing before the microscope was invented, but it did have objective features congenial to being used as a computer by beings with the right epistemic access.

Conversely, if one day Searle will be able to use his wall as one does the Wordstar program, using some (as yet unknown) sophisticated technique and representational capacities: good for him! Teleo-computationalism does not rule that out as absurd (for digitality is relative to the discriminatory and epistemic capacities of the user), but simply as very, very unlikely.

In sum, can the class of all real physical systems (B) be partitioned into (1) the class of all systems that implement some computation (B_1) and (2) the class of all the systems that do not implement any computation (B_2)? The answer to this question is yes, even under the assumption that observer-relativity arguments are sound. At the end of chapter 2, I proposed that we distinguish between intrinsic and extrinsic restrictions. Well, observer-relativity arguments, if assumed to be sound, only entail that B_1, the class of all RDS's that implement some computation, cannot be an intrinsic restriction of B.

Recall that these arguments are usually suggested by their proponents as reductiones ad absurdum of computationalism. My externalist understanding of computation, I believe, makes room for a "third way": the property of being the implementation of a computation does allow us to restrict the class B to a subclass (B_1), but the class B_1 is an extrinsic restriction of B.

Contrast this picture with Searle's criticism:

The aim of natural science is to discover and characterize features that are intrinsic to the natural world. By its own definitions of computation and cognition, there is no way that computational cognitive science could ever be a natural science, because computation is not an intrinsic feature of the world. It is assigned relative to observers.[1]

Searle is right that computational properties are not intrinsic to their bearers, but from this he wrongly infers that they are not "intrinsic features of the world". By the same token, from the observation that colors are not intrinsic properties of their bearers, one should infer that the science of color will never be a natural science, because colors are not "intrinsic features of the world": which is certainly false, at least a-priori.

6.3 Teleological computationalism and the Symbol Grounding Problem

6.3.1 The Physical Symbol System Hypothesis

In 1956 Allen Newell, Cliff Shaw and Herbert Simon[2] showed how their creation (the Logic Theorist), successfully proved 38 of the first 52 theorems proved by Russell and Whitehead in their Principia Mathematica. In their work the authors deployed the following heuristic recipe. Use a language-like symbolic code to represent the world (the objects of the world and the workings that these objects exhibit). This constitutes the "knowledge base" of the machine. Use input devices to appropriately

[1]Searle [79], p. 843.
[2][1].

transduce the flux of environmental stimuli incoming from the environment into appropriate symbolic representations of them (these representations should deploy the same code as the one used to form the knowledge base). The machine is then to use both the knowledge base and the transduced input information to produce further symbol structures (according to algorithmic procedures). Some of these "newly formed" symbol structures should then be designated to serve as output. Finally, further transduction should "translate" these output symbol structures into the appropriate behavior.

The Symbol System Hypothesis (SSH), proposed by Newell and Simon in 1976[3], and since then held to be the hard core of the received view of cognition, can be thought as an answer to three fundamental questions about thought. 1) Can a machine think? 2) What is necessary for a machine to think? And 3) What is sufficient for a machine to think? The answer to the first question, if the SSH is true, is yes: machines can think. The answer to the second question is that symbol manipulation is necessary for thought to take place. Finally, the sufficiency requirement is that a machine is built along the lines suggested by the above recipe. The sufficiency hypothesis presupposes that 1) implementing a symbolic machine is an intrinsic property of some physical systems and 2) that it is possible to suitably connect the appropriate machine to the environment.

I have discussed some concerns over the first assumption in the previous chapters, here I concentrate on the second one. The above outlined recipe prescribes that we use a language-like symbolic code to represent the world. This presupposes that we know what it means to do so. Surely, it would be argued, we know what it means

[3][63].

to do so: we do it all the time. We continuously use a language-like symbolic code to represent the world, we use our language (as well as artificial formal languages such as those of logic and mathematics) to represent the world. However, we know that the symbols we use, words, mathematical entities, are meaningful because we ascribe meanings to them. Sure once we have ascribed meaning to some of them, we can concatenate them to build "new meaningful" symbols, sentences, but if we don't ascribe meaning to at least a reduced (atomic) set of symbols, then no amount of formal manipulation would produce meaning. Now, as I said, we ascribe meanings to the words we use.

Who ascribes meaning to the atomic symbols in a physical symbol system? The problem is that, as the machine is to think, we want the symbols it deploys to have meanings that are not parasitic on our arbitrary ascription: they must be intrinsically meaningful. To use an expression that has recently become part of the technical jargon, symbols must be grounded in the world. The recipe, moreover, prescribes that we use input devices to appropriately transduce the flux of environmental stimuli incoming from the environment into appropriate symbolic representations of them. This presupposes that we know what it takes for something to be a representation of environmental stimuli, and that we know what is an appropriate transduction of them. I mentioned how it is hard enough to explain what it is for something to be intrinsically meaningful. Another (related) problem, is to explain what counts as an appropriate transduction.

Acknowledging that it has proved extremely difficult to reproduce intelligent agent/environment interactions, many classically oriented authors have argued that what is missing, is "simply" knowledge about how computers should be connected to the environment. The lack of technical expertise as to how to engineer the appropriate interface with the environment has often been invoked as an scape goat for the failures of artificial intelligence. So, when following the PSS recipe didn't produce the expected results, instead of challenging its basic assumptions, it has often been suggested that all these problems will be washed away when we will be able to appropriately connect our computers to the environment. The job transduction is meant to do has thus grown over the years to become so hard that some started wondering whether it is reasonable to expect that a physical system could do it. These problems with the PSS hypothesis, and the solutions proposed, from our perspective, are as many manifestations of the criticized assumptions of syntactic internalism and semantic encodingism. The problems expressed above, from the perspective endorsed here, are therefore ill-posed. To further corroborate my view, however, here I deal with how these discontents have been perceived and voiced within the received view itself.

6.3.2 Where does transduction end?

Turing analysis provided cognitive science with an exceptionally powerful conceptual repertoire. A most perspicuous heritage, and one that is most relevant for the present discussion, is the distinction between a cognitive process and a physical process. A computer, every computer, is a real physical dynamical system. As such its workings (in principle), are describable by means of the laws of physics or, more generally, by means of a physically projectible vocabulary. However, it is assumed (wrongly, we have argued) that computers can be grouped into classes of equivalence that don't correspond to classes of equivalence of physical properties. The generating relation of equivalence is that of "functional equivalence".

A given functional architecture (a specific task architecture, for example) allegedly partitions the class of physical systems in those that implement it and those that don't. Moreover, given a particular physical system that implements our architecture, not all of its physical properties are "relevant" with respect to its being an implementation. A common digital computer is a complex physical system with virtually infinitely many physical states and processes. Only very few of these, however, are "relevant" for it's being a "computer". Any physical process is (in principle) eligible to being considered as the implementation of a (not given) computational process. Given the particular functional architectures that we are considering, however, physical processes and events happening "outside" a computing machine are never thought of as computational events and processes.

A large amount of evidence, mainly pertaining to the domains of psychophysics and experimental psychology, suggests that there exists a strong correlation (hence a strong interaction) between the behavior (including verbal reports) of cognitive agents and their environment. It is therefore very important that a cognitive theory endorsing the TM hypothesis accurately accounts for the details of such interaction. The process responsible for ensuring a causal interaction between a computing system and its environment is usually termed "transduction". In its most general meaning a transduction is any systematic transmission (possibly transformation) of incoming patterns of energy. However, if it is to be of any use to our case, transduction must be constrained in a number of ways. Otherwise the whole of cognition could be viewed as a transduction from sensory input to behavioral output.

Although transductive processes are very poorly understood, there is substantial agreement on what the general constraints should be. For example it is believed that transduction shouldn't be a cognitive process. This is to say that it should be a non computational, physical process. I will not discuss here the reasons for postulating these constraints.

Another (and indeed trivially necessary) requirement vastly agreed upon, is that transduction "*must preserve all distinctions that are relevant to the explanation of some behavioral regularity*".[4] Our problem is to understand what job is transduction exactly doing. In the causal chain connecting the environment to the computing machine, where do physical processes cease being mere physical processes and begin to be also implementations of functional relations (symbol processing)? The first requirement, that transduction is substantially stimulus bound (data driven), suggests a late entrance of symbol processing.

The second one, that transductors be "aware" of all cognitively relevant differences, on the contrary, suggests an early entrance of "cognitive penetrability". This is what is often called the "transduction problem". This is a problem because often "cognitive relevant differences" aren't (systematically) projectible onto physical differences. The above consideration might constitute an "evolutionary" difficulty, for the computing and the transductive process must be accurately co-designed so as to satisfy both constraints at the same time. But all attempts to "suitably" embed computing machines in the environment have consistently failed (to various extents), and this has led many to believe that the requirements cannot be met.

Among the constraints that transduction has been argued to be subject to, the following are the ones that most interest us in our discussion. They meet general consensus.

[4]Pylyshyn [74], p. 158.

> a) The transducer output is an atomic symbol [...] Because the output is to serve as basis for the only contact the cognitive system ever has with the environment, it should provide all (and only) cognitively effective information. [...] The output of a set of transducers to an organism must preserve all distinctions present in the environmental stimulation that are also relevant to the explanation of some behavioral regularity.
>
> b) No limit is placed on how "complex" a mapping from signal to symbol can be induced by a transducer [...], so long as the complexity is not of a certain kind, namely, a combinatorial complexity that makes "infinite use of finite means". In other words, the mapping must be realized without use of recursive or iterative rules [...]. This basic requirement must be adhered to if we are to maintain a principled interface between processes to be explained in terms of rules and representations and those to be explained in terms of intrinsic [...] properties of the mechanism carrying out the rules or implementing the algorithm. Transducer inputs must be stated in the language of physics[5]

The problem with these constraints is that it is questionable whether they can be met:

> Requiring that the output of transducers should respect some criterion of cognitive relevance appears to be requiring something that is beyond their

[5]Pylyshyn [74], p. 158.

> capacity in principle. It seems analogous to requiring that a computer keyboard filter its input according to some criterion of relevance. Pylyshyn is correct, I believe, to state the constraints on transducers as he does. [...] Where he errs is in thinking that the transducer functions he describes are possible.[6]

A consequence of my externalist understanding of computation is that the transduction problem, as it is expressed above, is certainly untractable. A solution to it, in fact, would confirm the claim that: (1) computational structures can be implemented even before they are suitably connected to the environment (thus forcing us to endorse syntactic internalism) and (2), that the symbols can be subsequently endowed with intentional properties, thus making the machine to display publicly its preexisting cognitive capacities (thus implicitly endorsing an encodingist picture of intentionality).

6.3.3 The symbol grounding problem

The physical symbol system hypothesis, in all the various variants that have been proposed for it, states that a physical system is a cognitive system if an only if it implements a certain symbol system.

> A symbol system is a set of arbitrary "physical tokens" [...] that are manipulated on the basis of "explicit rules" that are likewise physical tokens and strings of tokens. The rule-governed symbol-token manipulation is based purely on the shape of the symbol tokens (not their "meaning"),

[6]Wells [94], p. 279.

> i.e., it is purely syntactic, and consists of rulefully combining and recombining symbol tokens. There are primitive atomic symbol tokens and composite symbol-token strings. The entire system and all its parts - the atomic tokens, the composite tokens, the syntactic manipulations both actual and possible and the rules - are all "semantically interpretable": the syntax can be systematically assigned a meaning e.g., as standing for objects, as describing states of affairs.[7]

Thus, it is argued, for cognitive processes to take place there must exist a set of physical tokens (the symbols) that are manipulated by the use of explicit rules. These explicit rules are themselves physical tokens and operate on the symbols by rulefully combining and recombining them. The ruleful manipulation must be based on the shape of the symbols only (i.e. it is individuated independently of their meaning). Finally, and most importantly, the symbols and the syntax must be semantically interpretable: it must be possible to give a systematic interpretation of the syntax as "meaning" something that is external to the system itself. Not any assignment of meaning would do just as well, for there are constraints that have to do with the systematicity and consistency of the interpretation. These constraints, however, are external to the symbols themselves: they are met if it is possible to assign a system of meanings that meets them.

There are several arguments in the literature that suggest that this is actually a problem. I will mention only two: the Chinese-Chinese Dictionary argument and the Chinese Room argument.

[7]This is how Stephen Harnad ([45], p. 337) reconstructs the definition from Newell and Simon [63].

6.3.4 The Chinese-Chinese dictionary argument

Another argument that claims to reveal a deficiency in the sufficiency hypothesis is due to Steven Harnad. Suppose we were to learn Chinese with the only help of a Chinese-Chinese dictionary. Although very difficult, this task could, in principle, be managed. After all, cryptologists of ancient languages or of secret codes do something very similar. It is possible, but at the condition that one already possesses a first language. If we were faced with the task of learning Chinese as a first language with the only help of a Chinese-Chinese dictionary, then the task would clearly be impossible. According to Harnad, this example provides a faithful picture of the problem. If the mind is a physical symbol system, how does it exit the symbol-symbol merry-go-round? It should be clear that, under our analysis, the symbol-symbol merry-go-round is yet another expression of the failure of internalist views to fix their intended models. The fact that most computationalists acknowledge the problem, moreover, further exposes their commitment to internalism, syntactic and semantic.

The standard reply is that *the symbols are grounded by suitably connecting them to the environment.* This amounts to the hypothesis that, given the appropriate isolated symbol system, if we connect it to the environment in the right way it will see, talk and understand just how we do it. This response is, in the light of our analysis, either false or trivial. If it means that there exists a symbolic module that awaits to be attached (or interfaced) to the environment in a specified way, then the argument still applies to that module: in what way should that module await to be attached? What is the "right way"?

If the words "in the right way", instead, just mean "in such a way as to make the system see, talk and understand", then the statement is trivially true, but it then becomes an empirical question whether the requirement can be met. If our analysis is correct, the requirement cannot be met.

Turing Tests and Total Turing Tests

The line of criticism discussed in the previous paragraphs can be best appreciated if we consider its methodological premises. Turing Test (TT) indistinguishability constitutes the single most influential methodological criterion of classic artificial intelligence. The rationale behind it rests on a (pre-theoretical) relation of symmetry between the problem of what it is to have a mind and the problem of other minds (how do I know that other people also have a mind). If "pen-pal" indistinguishability is a sufficient condition for inferring the presence of a human mind (other than mine), there is no reason why the same criterion shouldn't apply also to machines. Considerations of biological make up don't play a role in the former kind of inference, so, coherently, they shouldn't play a role in the latter. The reason why Turing Test candidates are supposed to be out of sight is that the judge is thus sure to disregard irrelevant and biasing aspects of intelligence, concentrating on the appropriate level of analysis. There is little doubt that several features that we consistently observe in association with intelligent behavior are not essential to it. Methodological concerns about how to best screen these biasing aspects off are therefore necessary for a proper treatment of intelligence.

What some authors (like Harnad) object, however, is that a criterion based on pure symbol manipulation (pen-pal indistinguishability) might be too strict. It is certainly the case that intelligent behavior comprises much more than mere word-in-word-out behavior. We do seem to possess what can be called "symbolic" capacities, that is our capacities to identify, describe and respond coherently to descriptions of objects and states of affairs in our world. On top of these, and possibly responsible for these, are our "robotic capacities": the capacities to discriminate, identify and manipulate the objects and states of affairs of our world.

The most basic robotic capacity is the ability to discriminate pairs of stimuli: to tell whether two stimuli are different or the same, and in the case they are different, to tell how similar they are. This capacity can be (and most likely is) realized by analog mechanisms[8].

An analog of the sensory projection of one stimulus is superimposed on that of another, and a measure of congruence between these analogs is then deployed to effect discriminations. These analogs of sensory projections apt for discriminating incoming stimuli can be called "iconic representations". The capacity to identify objects and states of affairs, instead, is a matter of absolute judgements: a horse is a horse regardless of how different horses are from pigs or sheep. In a sense the capacity to identify stimuli is contrary to the ability to discriminate them: it is based on the capacity to selectively detect the invariant features of different proximal stimuli and react accordingly. The representations that are apt for performing identifications ("categorical representations") are thus selectively blind to within-category incongruencies and are oversensitive to incongruencies related to stimuli belonging to different categories.

Although the mere capacity to identify something doesn't by itself entail the presence of an "inner name" for it, categorical representations seem to be a necessary feature of any autonomously functioning intelligent symbol system. In fact categorical representations, as well as allowing to sort the world into categories, also allow

[8]As I have already mentioned, however, our capacity to discriminate and identify stimuli should not be conceived as coextensive with the presence of systematic factual correspondences.

us to assign a unique arbitrary name to each of them. These arbitrary symbols, together with strings of them that "can be interpreted as propositions about category membership", constitute "symbolic representations".

Harnad argues (wrongly, I believe) that a symbol system, thus grounded into non arbitrary sensory (iconic and categorical) representations, would be immune to Searle-like arguments. In the Chinese-room argument (to be discussed in the following) the notion of functional identity is characterized as identical to Turing-indistinguishability.

According to the analysis outlined above, Turing-indistinguishability is unnecessarily restricted to the indistinguishability of symbolic (word-in-word-out) capacities. Because symbols are not intrinsically meaningful, symbolic capacities indistinguishable from those of a Chinese speaker may fail (Searle's argument goes) to suffice for real understanding to take place. This failure would reveal a fatal shortcoming of computationalism (functionalism in general). Harnad argues that requiring robotic indistinguishability (indistinguishability of robotic capacities) as well as symbolic (Turing) indistinguishability would block the argument. If we replaced the standard Turing Test with a Total Turing Test we would be able to detect (and select away) unwanted unintelligent pen-pal simile-minds. The symbol system of the Chinese speaker, unlike that of Searle's, would be grounded in her iconic and categorical representations, so Searle would fail to pass the Total Turing Test.

6.3.5 Teleological computationalism and the symbol grounding problem

Harnad summarizes the symbol grounding problem in the following way:

> How can the semantic interpretation of a formal symbol system be made

> intrinsic to the system, rather than just parasitic on the meanings in our heads? How can the meanings of the meaningless symbol tokens, manipulated solely on the basis of their (arbitrary) shapes, be grounded in anything but other meaningless symbols? [...] Hence, if the meanings of symbols in a symbol system are extrinsic, rather than intrinsic like the meanings in our heads, then they are not a viable model for the meanings in our heads: cognition cannot be just symbol manipulation.[9]

The view of computationalism that has been put forward here, agrees with the above criticism, so far as it is applied against the orthodox, internalist treatment of computation (and symbol manipulation). Orthodox understandings, in fact, assume that whether a physical system manipulates symbols or not is a matter that can be assessed independently of how these symbols are interpret-ed or even from whether they are interpreted at all. Interpretation comes later, and separately, if ever: what counts is interpret-ability. This hope, I have argued, has been sustained by the encodingist preconception of the nature of intentionality. We have argued that such divorce of syntax from semantics is fatal, and not necessary (if not impossible, if we believe in the soundness of v-argument) to the computationalist stance.

Harnad, instead, assumes that such divorce is necessary, although he agrees that it is fatal to the computationalist stance as it stands. Coherently, he proposes to amend the computationalist hypothesis (rather than the notion of computation, as I have done) by requiring that cognition be not coextensive with symbol manipulation. The idea is to set constraints on the symbols that are being "manipulated", requiring that they be symbolic representations. In his own words:

[9]Harnad [45], p. 335.

> Symbolic representations must be grounded bottom-up in nonsymbolic representations of two kinds: (1) "iconic representations" , which are analogs of the proximal sensory projections of distal objects and events, and (2) "categorical representations" , which are learned and innate feature-detectors that pick out the invariant features of object and event categories from their sensory projections. Elementary symbols are the names of these object and event categories, assigned on the basis of their (nonsymbolic) categorical representations.[10]

Harnad, then, requires that "symbol manipulation" be sensitive not only to the "arbitrary" shape of the symbols, but also to the "non-arbitrary shape of the iconic and categorical representations connected to the grounded elementary symbols out of which the higher-order symbols are composed".

Here, the expression "sensitive to the non-arbitrary shape of the iconic and categorical representations" is meant to require that there be a factual correspondence between individuals in the external world, with the structure of differential features that they instantiate, and the responses of the system. I have argued that such factual correspondences do not carry any information as to what they are correspondences with, and that for this reason they are unsuitable, by themselves, for the task of endowing something with intentional properties. Harnad's proposal, in fact, is silent as to what feature of the system would allow it to "use" these factual correspondences in a semantically appropriate way.

The key difference between the view that is being promoted here and Harnad's analysis, is this. Harnad thinks that there can be (and should be) two distinct sets

[10]Harnad [45], p. 335.

of constraints on the symbol tokens: one "merely syntactic" and the other, we might say, "semantic" (grounded on the iconic and categorical representations). We, instead, have argued that the two sets of constraints are mutually, logically connected, as far as their supervenience on internal features of the system are concerned: there cannot be such a thing, according to computational externalism (or Teleo-computationalism, which is my preferred example of externalist theory of computation), as an independent "merely syntactic" set of constraints. We have argued, in fact, that such an independent set of syntactic constraints would not even be able to model the notion of "same symbol token", for the very notion of "sameness", as applied to symbols, can be argued to be observer-relative. In sum, while Harnad proposes to amend the sufficiency hypothesis by adding to it non-symbolic constraints, I propose to maintain the hypothesis at the price of a redefinition of the notion of real computation.

6.4 Teleological computationalism and Searle's criticism

6.4.1 Teleological computationalism and the Chinese Room

The necessity to tackle the difficulty exposed by the Chinese Room argument has been seen by some as nearly definitory of cognitive science. Refutations and strategies for combating it occupy thousands of pages, and I do not wish to add any. The aim here is simply to test the capacity of our treatment to address the issue.

The argument has some authoritative predecessors. Leibniz, for example, had to say:

> Moreover, we must confess that the perception, and what depends on it, is inexplicable in terms of mechanical reasons, that is, through shapes and

> motions. If we imagine that there is a machine whose structure makes it think, sense, and have perceptions, we could conceive it enlarged, keeping the same proportions, so that we could enter into it, as one enters into a mill. Assuming that, when inspecting its interior, we will only find parts that push one another, and we will never find anything to explain a perception. And so, we should seek perception in the simple substance and not in the composite or in the machine. [11]

The reductio ad absurdum, here, is directed against the idea that we could "construct" a machine that thinks and has perceptions. The contrast revealed by the thought experiment is between overt behavior (the apparently intelligent behavior) and the inner workings of the machine. Buying v-arguments, we might revive Leibniz thought experiment by saying that if we "constructed" an object following the "recipe" of the computational structure that allegedly explains intelligent behavior, and if we then looked into it, we would "find only parts which work one upon another", and never something that relates to the formal structure used to build the object. Thus, no room for explanatory efficacy is left to the recipe.

Notice that my treatment of computation, and the computational theory of the mind that follows from it, is sympathetic to the above thought experiment. The form of a causal structure is not something that can be "seen" by looking inside a physical object, as if it were a physical structure. The intentional properties of a structure, those that ground all syntactic properties, supervene on observable physical properties (if the naturalization programme can be carried out). But they cannot be expected to be "transferrable" to the observer: the sole fact that we observe that a physical

[11]Leibniz, *Monadology* (1714). Section 17. ([51], p. 216).

structure possesses the relevant intentional properties does not entail that, by this reason alone, we share theses properties with it.

Indeed Chalmers once commented ironically that if Searle was to be taken seriously, then the following syllogism should also have to be taken seriously. 1) Recipes are syntactic, 2) syntax is not sufficient for crumbliness, 3) cakes are crumbly, so 4) implementation of a recipe is not sufficient for making a cake.[12]

I would add, to reverse the parody in accordance with my view, that indeed recipes are not sufficient to make cakes if there isn't someone who type-identifies the symbols and understands the language in which they are written in: but this is not very funny.

The direct historical origins of Searle's argument lie in the idea of a Turing Test. Remember that Turing suggested that if a computer could pass for human in a chat under cover, it should be counted as intelligent in all relevant respects. Following such intuitions, computer scientist R. Schank proposed a schema for passing simple versions of the Turing Test called "conceptual representation"[13]. According to Schank's schema, "scripts" should be used to represent relations between concepts. Originally, Searle's argument was proposed as an objection against Schank's conceptual representations: it was aimed at showing that Schank's programmes could not be considered as literally understanding anything (even if they passed the Turing Test).

"*The point of the argument*", says Searle, "*is this: if the man in the room does not understand Chinese on the basis of implementing the appropriate program for understanding Chinese then neither does any other digital computer solely on that basis because no computer, qua computer, has anything the man does not have.*"[14]

[12]Chalmers [16].
[13]Schank [76].
[14]Searle [80].

The Robot Reply

Searle's example of what kind of thing the human operator inside the room does not know, is the meaning of the (Chinese) word for hamburger. One of the strategies for combatting the argument, known as the "Robot Reply" is based on the observation that we know what "hamburger" means because we have seen, tasted, perhaps made one. Or because we have heard someone talking about it, in which case we know what it means by reference to something that has been seen, or tasted. The "Robot Reply", in its many variants, consists in the claim that a digital computer, freed from the room and attached to a robot "in the appropriate way", so as to be able to "see" through sensors and manipulate the world, would be able to learn the meanings of the symbols it manipulates and eventually speak and understand a natural language. Among others, Margaret Boden, Tim Crane, Daniel Dennett, Jerry Fodor, Stevan Harnad, Hans Moravec and Georges Rey have all proposed some variant of the "Robot Reply".

The treatment developed in this work, I believe, has a lot in common with the robot reply. Let us start, however, with the most striking difference. All variants of the Robot Reply have in common one thing: they concede that the computer, confined in the room, implements the relevant computational structure. They also concede that the computer does not understand what "XZX" (a fictional Chinese word for hamburger) means. According to our view, instead, the thought experiment is ill-posed, for it assumes (coherently with the orthodox view that it aims to criticize), that a computational structure can be implemented without implementing (by definition, as we suggest) some intentional icons. Thus, according to our view, the argument would be blocked at its onset, when it supposes that the "right" program is being "implemented". We would require, to accept the premise, a specification of the intentional icons with respect to which the program can be said to be implemented, before we even wonder if it is the "right" program.

Notice that my proposal does not deny that (real) computations operate on "symbols" solely according to their shape, nor it denies that virtually anything (any stuff) could implement a computation: this much it has in common with the orthodox view. What it denies, I repeat, is that it make any sense to individuate a particular computation as the one that is being implemented without making reference to the intentional icons that realize its input architecture.

Having made this important distinction, what makes our proposal close to a Robot reply is, I think, that:

1) given my definition of implementation, even if the Chinese Room as a whole, or, through it, its human operator, implemented a computation, it could hardly be the right sort of computation, for it is highly unlikely that the external description of such computation is implemented by the right sort of intentional icons. In fact, the only symbols in the story, with respect to which alone the computation can be said to be implemented, would be intentional icons referring to patterns of ink (such as XZX). No other system in the story, in fact, can be reasonably assumed to instantiate natural semantic properties. As hamburgers and XZX's (the patterns of ink) are different sorts of entities, the computation implemented by the man inside the room is very unlikely to be the "right" one.

2) Moreover, the "right" computation will have to be one grounded on some

appropriate causal relation with real hamburgers (we have seen what sort of causal relation). This is compatible with the intuition that the missing bit is the role played by robotic capacities.

6.4.2 Syntax is not intrinsic to physics

Searle later divorced the Chinese Room argument from other versions of v-arguments which we set to combat. His reasons are articulated in various arguments that we now briefly comment on. The first one, and one we are familiar with, proceeds from a denial that syntax is a feature intrinsic to physics:

> [...] "syntax" is not the name of a physical feature, like mass or gravity. I think it is probably possible to block the result of universal realizability by tightening up our definition of computation. Certainly we ought to respect the fact that programmers and engineers regard it as a quirk of Turing's original definitions and not as a real feature of computation. Unpublished works by Brian Smith , Vinod Goel, and John Batali all suggest that a more realistic definition of computation will emphasize such features as the causal relations among program states, programmability and controllability of the mechanism, and situatedness in the real world. *But these further restrictions on the definition of computation are no help in the present discussion because the really deep problem is that syntax is essentially an observer relative notion. The multiple realizability of computationally equivalent processes in different physical media was not just a sign that the processes were abstract, but that they were not intrinsic to the system at all. They depended on an interpretation from outside.*

> *We were looking for some facts of the matter which would make brain processes computational; but given the way we have defined computation, there never could be any such facts of the matter. We can't, on the one hand, say that anything is a digital computer if we can assign a syntax to it and then suppose there is a factual question intrinsic to its physical operation whether or not a natural system such as the brain is a digital computer. [My emphasis].*[15]

It is apparent from the long passage quoted above, that Searle excludes that a restriction of the notion of computation could be used to make the very notion of syntax objective. It has been argued, instead, that this is not the case, and this has been the purpose of this treatment: to restrict the notion of computation so as to make it objective in physical terms. Further passages should make this claim clearer.

> There is no way you could discover that something is intrinsically a digital computer because the characterization of it as a digital computer is always relative to an observer who assigns a syntactical interpretation to the purely physical features of the system.[16]

We have seen how such observer-relativity should not be thought of as irreducible. On the contrary we suggested that the very same physical system that is said to implement the computation could play (and indeed should play), if the notion of implementation is to have any empirical content) the role of a naturalized observer.

The application of this criticism to the computationalist theory of the mind is straightforward:

[15]Searle [79], pp. 841-842. From Searle [81].
[16]Ibid., p. 842.

> As applied to the Language of Thought hypothesis, this [the fact that syntax is not intrinsic to physics] has the consequence that the thesis is incoherent. There is no way you could discover that there are, intrinsically, unknown sentences in your head because something is a sentence only relative to some agent or user who uses it as a sentence. As applied to the computational model generally, the characterization of a process as computational is a characterization of a physical system from outside; and the identification of the process as computational does not identify an intrinsic feature of the physics, it is essentially an observer relative characterization.[17]

According to my view, instead, the "characterization of a process as computational" is a matter of objective fact, and as such needs not be a "characterization from outside", if by this it is meant that it is a characterization that makes reference to "external", irreducible, semantic capacities. The characterization is, we have seen, irreducibly relational, but this by no means implies that it cannot be instantiated by a system alone in its environment.

6.4.3 Syntax has no causal power

Another, related line of argument proceeds from the claim that the programs that (allegedly) are being implemented do not exist if not in the eyes of the beholder; hence, per force, they cannot have any causal power, hence no explanatory role in a theory of the mind.

> The thesis is that there are a whole lot of symbols being manipulated in

[17]Ibid., pp. 842-843.

> the brain, 0's and 1's flashing through the brain at lightning speed and invisible not only to the naked eye but even to the most powerful electron microscope, and it is these which cause cognition. But the difficulty is that the 0's and 1's as such have no causal powers at all because they do not even exist except in the eyes of the beholder. The implemented program has no causal powers other than those of the implementing medium because the program has no real existence, no ontology, beyond that of the implementing medium.[18]

My claim is that the 0's and 1's do exist in the implementing structure. Moreover, they "must" exist, for the physical structure to be an implementation. The difference between a physical token and the symbol "1" is mirrored, and explained, by the difference between an intentional icon and its content. It remains true, under our analysis, that "[t]he implemented program has no causal powers other than those of the implementing medium because the program has no real existence, no ontology, beyond that of the implementing medium." But this is not different, in the proposed understanding of computation, than the claim that a mathematical structure (say a system of differential equations), has no causal powers other than those of its instantiating real dynamical system.

6.4.4 The brain does not do information processing

To conclude with Searle's criticism, we mention a last line of argument. It is argued that that there is no sense in which a brain can be significantly said to "process information", for, again, information processing needs interpretation.

[18]Searle [81], pp. 30-31.

> The mistake is to suppose that in the sense in which computers are used to process information, brains also process information. To see that that is a mistake contrast what goes on in the computer with what goes on in the brain. In the case of the computer, an outside agent encodes some information in a form that can be processed by the circuitry of the computer. That is, he or she provides a syntactical realization of the information that the computer can implement in, for example, different voltage levels. The computer then goes through a series of electrical stages that the outside agent can interpret both syntactically and semantically even though, of course, the hardware has no intrinsic syntax or semantics: It is all in the eye of the beholder.[19]

Recall that our preliminary diagnosis for the failure of naturalization proposals blamed the "encodingist paradigm". The above passage is an example of how such criticism can be applied to the computationalist stance. In sum, there is no intrinsic notion of encoding: it is an external agent that does the "encoding". But who is the external agent, when the notion of encoding is being used to explain the agency of agents? In the case of the naturalization of intentionality, we have seen, the problem can be tackled by the deployment of teleological concepts. This entails an abandonment of the encodingist paradigm. We have applied the same strategy to the naturalization of implementation. The result is that we cannot determine whether a system is or isn't an information processor unless we are told some further (physical, albeit relational) facts about the entities that "encode" the information.

[19]Searle [79], p. 844. From Searle [81].

Thus, for example, voltage levels are not sufficient to settle the matter, unless we are told something more as to how such voltage levels relate causally to the survival of the organism to which they pertain. Such information, we argue, would be sufficient to determine what syntactic and semantic properties they possess. So, I concede that "*it is all in the eye of the beholder*", but the beholder is naturalizable and it is part of the story since the beginning. It is not called in later, to interpret the structure: it is part of the structure. In a slogan: for a thing to be an implementation of a computational structure, or for it to be an information processor, that thing must be a beholder. And for something to be a beholder, that something must produce and use intentional icons or, equivalently, it must induce dynamical presuppositions.

6.5 Computational externalism and the connectionist challenge

The following is a discussion of some of the issues raised by the connectionist challenge, judged from the perspective of teleological computationalism.

6.5.1 Principles of Connectionist Modelling

In the middle 1980s the (re)introduction of cognitive models called "connectionist networks" set about (or coincided with) a major debate about how cognition should be construed. The conceptual import of the "new wave" of cognitive modelling has traditionally been understood in contrast with models constructed according to the Physical Symbols System recipe (see chapter IV). Classical computational architectures, as discussed in the previous chapters, are conceived as performing algorithmic operations upon discrete strings of symbols. Syntactic rules operate upon strings of symbols each of which possesses semantic properties. Possessing semantic properties, being "about something" or, minimally, representing something, is held to be a necessary condition for being computed. "No computation without representations", is the slogan of classical computationalists. Connectionist networks, on the contrary, compute through continuous functions on multidimensional vectors of activation. At the fundamental level, that which is directly implemented, the objects of computation do not possess semantic properties. The operands are activation values that don't correspond to anything in the world: they are not symbols. Nevertheless, they are in some views (Smolensky, for example) constituents of symbols, and are for this reason called subsymbols.

> Entities that are typically represented in the symbolic paradigm by symbols are typically represented in the subsymbolic paradigm by a large number of subsymbols. [...] they participate in numerical - not symbolic - computation. Operations in the symbolic paradigm that consist of a single discrete operation (e.g. a memory fetch) are often achieved in the subsymbolic paradigm as a large number of much finer-grained (numerical) operations.[20]

This attempt to conceptualize the "new way" of modelling cognition issued in a lively (ongoing) debate where a group of authors, primarily Smolensky, responded to the criticism of classically oriented Fodor, Pylyshyn and others. Soon after its onset the debate drifted away from the conceptual foundations of the two paradigms and concentrated on their explanatory advantages and disadvantages. In particular, the debate focussed on the alleged difficulty of connectionist models to explain the syntactic constitutivity that seems to characterize all higher cognitive phenomena.

[20]Smolensky [86], p.774.

Basic assumptions and properties of connectionist models

Connectionist models are attempts to describe some formal aspect of cognition through the deployment of artificial neural networks. Their basic assumption is that cognitive capacities are built out of a set of primitive, brain-like processes. At the present time about 50 different kinds of connectionist models are under investigation.

At a formal level (regardless of how they come to be implemented), they are all networks of simple units. Inputs and outputs are not information-bearing messages, but simple values of activation. Each unit, in fact, can take up a value of activation. Values of activation are intrinsic properties of the units which must be reliably associated with some physical state of the instantiating unit. The activity of units influences that of others through a net of weighted connections. The activity of all input units is summed to yield a total, weighted net activity. The activity of each recipient unit is then calculated as a function of such net input value alone.[21]

Thus, the activation of input units is spread through to the other units. A set of units of the network is then designated to be the output pool: their pattern of activation constitutes the output of the network.

There are two main families of connectionist networks: *feed-forward* networks and *recurrent* networks. Feed-forward networks are built out of layers of units. Each unit of a layer may connect to any unit of that layer or to any unit in the next layer (*downstream*). It may not connect to any unit upstream. The activation of units can

[21]Such function, termed *the activation function* is often taken to be the so called *logistic function*:

$$output = \frac{1}{1 + \exp -(netinput)}$$

A function of this kind is called a quasi-linear function and manages to bound the output while avoiding unwanted discontinuities.

thus spread only in one direction.

In recurrent networks, instead, any unit can connect to any other unit: therefore units need not be divided in layers.

The performance of a network is modified by changing the weights of the connections. This process is usually termed *learning.* Changes to the matrix of connection weights, in fact, are made dependent on some feature of the performance of the network. This kind of learning is often termed *supervised learning*, for changes to the weights are made to comply to some constraint on the (desired) performance. This kind of learning should be contrasted to an *unsupervised* kind, where changes of weights result from intrinsic characteristics of the connections themselves. Some details of these models will be discussed in the following paragraph.

6.5.2 The problem of syntactic constitutivity: symbolic vs/ connectionist representations

One of the most discussed issues about connectionist networks has to do with how they manage to store information. Before getting into the details of this discussion it is worth pointing out that networks have two main ways of storing information: through the pattern of activation and through the pattern of weights. the former means of storage relies on a very transient feature of the network. Patterns of activation, in fact, appear and disappear as the network is input different sets of values. Patterns of weights, by contrast, are more lasting and best suited to analyze how information is encoded, stored and decoded. In either case, however, the means of storage and manipulation of information stand in clear contrast to the way in which information is stored and manipulated by a symbolic system, such as a language. As we will discuss later, in fact, the most startling difference is that the elementary units and their processing properties are usually not singularly ascribed any particular meaning. Moreover, the performance of the network rarely depends heavily on the activity of a single unit.

This feature is responsible for a very much acclaimed feature of networks that is referred to as *graceful degradation.* Partial omission of input, or lesion of connections, or any kind of local damage to the network, doesn't result in complete collapse of its performance: performance may be partially nonfunctional, but still interpretable. This is often claimed to be a factor in favor of connectionist models of cognition, as damages or lesions on real brains, unlike those on computers, don't usually result in catastrophic bankruptcy of cognitive capacities. This same feature, however, has been argued to constitute a fatal shortcoming of connectionist models.

Classic account of syntactic constitutivity

There is a received view in cognitive science according to which there are two distinct categories of phenomena that call for explanation: on one side there is cognitive performance, on the other there is cognitive competence. Individuating the cerebral structures responsible for the production and understanding of language, for example, is certainly relevant for understanding how we manage to speak or why we make certain linguistic mistakes, but understanding what language and thought really are, and what their structure is, is something completely different. The two domains of enquiry, it is claimed, are so distinct as to require two methodologies and conceptual apparatuses to be described. The classic symbolic paradigm, with its distinction between symbolic-functional architecture and physical implementation is a clear example of this received view.

Consider the sentence: "*Mary and John Eat a Chicken*" (I shall denote this sentence with letter A). A expresses a thought that "contains" (in a way to be explained) the thoughts *Mary*, *John*, *Eat* and *Chicken.* Moreover, A expresses a thought that must also contain the separate thoughts that *Mary Eats a Chicken* and that *John Eats a Chicken.* In other words, if I understand the sentence A, the thought that it expresses must be compound. This property of some representation is called "constitutivity".

How must a representation be, in order for it to be complex? What do all complex representations have in common that makes them all complex? Is the property of being complex an intrinsic property of representations? The symbolic paradigm has an answer to these questions that, I have argued, pose unnecessary (and possibly unsatisfiable) constraints on representations.

There are symbols in the mind, the story goes, that refer to all terms in a proposition. When you think that A your mind concatenates these symbols. A "contains" the thought that *Mary Eats Chicken.* Why doesn't it also contain the thought that *Chicken Eats Mary*? In other words: once we claim that thoughts are complex when they are built out of simpler symbols, we must explain why the same simpler symbols can be concatenated to build different thoughts.

The symbolic paradigm claims to have a solution to this problem too. To concatenate symbols does not merely amount to put them together, in a bunch: it means to place them orderly into appropriate "boxes". In your mind there is a "subject box". a "verb box", etc. The same symbol can end up in different boxes, thus building different thoughts. The received view then requires that this apparatus be implemented, by a brain for example.

It is straightforward to see how this picture presupposes the kind of syntactic internalism that we have criticized. To be a "subject box", for example, is a property that is characterizable in terms of syntactic properties alone, independently of what word will end up filling it.

Towards the end of the eighties an old research programme, the connectionist programme, re-emerged from disrepute. The successes achieved by neural networks, particularly at modelling lower cognition (like perception) convinced many authors that a "paradigm shift" was to take place. No more mental symbols or syntactic boxes: no more language of thought. There are a thousand pages written about the opportunity of this "Kuhnian revolution" and I shall not review them here (personally I believe that the concept of scientific revolution is very much abused in the literature). Instead, I shall discuss 1) what is the minimal notion of syntactic constitutivity that we can afford (short of the orthodox symbolic paradigm) and 2) what are the (interesting) shortcomings of the proposals to solve the problem of constitutivity in the sub-symbolic paradigm.

What constitutivity do we really need?

In a connectionist model intentional states are represented by activation vectors. The composition of representations is rendered by the (vector-) sum of the compounds. This feature of connectionist models is appealing in many respects (it accounts well for aspects of cognition such as categorical perception, discretization of a sensory continuum, emergence of prototypical representations, etc.) but it seems to fail to satisfy the fundamental desideratum of syntactic constitutivity: that it be an intrinsic property of complex representations. Given a vector, sum of two components, it is in fact impossible to unambiguously trace the components back. This shortcoming of connectionist models is often called the "superposition catastrophe".

Very generally, one can distinguish two ways of composing entities: one, similar to the building of a house out of bricks, keeps "memory" of the elements that form the compound; the other, more similar to the way in which two raindrops that come together to from one, "forgets" about the components. I shall call the first kind of composition "architectonic", the second "superpositional". Well, according to orthodox computationalists, the constitutivity that we need to describe complex representations must be "architectonic". Why is that? Let's consider the desiderata of our theory of representation.

1. We want thought, like language, to be productive and generative. It seems that we are able to produce a (virtually) infinite number of thoughts out of a finite number of thoughts. It follows that there must be rules for generating these thoughts. Hence thoughts must have a recognizable internal structure.

2. We want thought, like language, to be systematic. The comprehension of certain thoughts implies the possibility to comprehend other (related) thoughts. Cognitive systems that understand the sentence "*Mary Eats a Chicken*", are observed to be also able to understand the sentence "*A Chicken Eats Mary*". It follows that there must be grammatical rules that determine appropriate relations of equivalence.

3. We want thought, like language, to be compositional. Given a (complex) thought, it is possible to individuate components that have semantic properties that are independent from the context. It is always possible to assign a syntactic-semantic structure to complex thoughts in a non-arbitrary fashion.

In order to satisfy these desiderata, representations, whatever they are, must have a constitutive structure. This structure must be non-arbitrary and persistent.

6.5.3 The Variable Binding Problem

One easy way to touch the above mentioned problem with a concrete example is to consider the task of binding variables to their values. From *Mary Gave John Book1* one can infer that *John Ows Book1.* This inference can be formalized by use of first-order predicate calculus (a paradigmatic case of symbol system). Let us introduce the three-place predicate *give(x,y,z)* so that its three arguments correspond to the three semantic roles: giver, recipient and given object. Let moreover the two-place predicate *own(x,y)* stand for the fact that x owns y. We can now express the above inference by the following formula:

$$\forall x, y, z[give(x, y, z) \Rightarrow own(y, z)]$$

The use of variables allows us to make the specific inference a special instance of the more general inference that tells us that, in the act of giving, whoever is the recipient must also be the owner. Notice that here, the words "recipient" and "owner" refer to variables, abstract "places" in a syntactic structure: they do not possess any specific meaning. Thus, for example, a token of a name does not satisfy the property of being the "recipient" by virtue of its meaning, but by virtue of the place it occupies in the overall structure. To apply the above general rule to the case of *Mary* and *John* one must solve the variable binding problem: the names must be assigned to the variables in the formula in a consistent way (for example the first occurrence of z must be bound to the same object as the second one, and that object must be book1). The result of binding, in this case, produces the (correct) inference:

$$give(Mary, John, Book1) \Rightarrow own(John, Book1)$$

In structured representations, such as the triplet $< Mary, John, Book1 >$, the problem of multiple binding for multiple variables is straightforward. Fodor and Pylyshyn argue that the mind must deploy structured representations in a system that is at least as powerful as predicate calculus if it is to perform such inferences.

6.5.4 First Connectionist Response: Dynamical Binding

Functional and neurophysiological analysis of a brain processing a visual image shows that many distinct areas are simultaneously active and singularly responsible for the processing of various features of the image (shape, color, etc.). Lokendra Shastri and Verkat Ajjanagadde designed a connectionist model[22] inspired by these empirical findings. The idea is that units in the network have activation values that oscillate creating patterns in time. The binding feature is then taken to be the synchronicity of such oscillations. A network that must assign to the variable *owner* the argument *Mary*, has the units relative to the role *owner* oscillating synchronously to those relative to the name *Mary*. The model, known as *dynamical binding*, was capable of significant logical work and became even more promising as evidence of the brain actually deploying synchronous activity to bind information emerged in the 1990's.

The main concern raised by Fodor and Pylyshyn[23], however, remains untouched: even if such a model adequately matched the performance of a symbolic machine, nothing would make of it more than an implementation of a computational structure. The relevant psychological evidence would be explained at the higher, computational

[22]Shastri and Ajjanagadde [83].
[23]Fodor and Pylyshyn [40].

level, regardless of how the system is implemented. The attempts of connectionist theorist to meet the symbolic challenge obscured the fact that the notion of being the argument of a variable, in the case of symbolic structures, is far from trivial. We have seen, in fact, that in the case of dynamical systems measurements assign a number to variables, exploiting the fact that the variable is identified (independently of other variables) by one and only magnitude that is measured at different times.

The case, for symbolic structures, is not as simple. The abstraction from real time, and the identification of variables on internal relational properties alone, makes it impossible to say that an argument is the value of a certain variable. According to computational externalism, thus, the orthodox symbolic explanation of constitutivity is a pseudo-explanation, for it presupposes syntactic internalism. This is not to say that connectionist and symbolic systems are on a par at solving the binding problem, considered as a technical difficulty. Sure symbolic systems are more suitable to build physical systems that comply (once supplied with our intentional capacities) with the desired syntactic constraints. But this very fact, as we have seen, has obscured a grave deficiency in their fundamental conception.

It is interesting to consider how connectionist modelers attempted to meet the computationalist challenge. Where they appeared to have succeeded to some extent, their computational opponents argued that, to that extent, their models were nothing but "implementation" of standard computational systems. I shall apply my understanding of implementation to comment on some of these responses.

6.5.5 Second Connectionist Responce: Tensor Products

Smolensky's Proposal

In 1990 Smolensky[24] proposed a solution to the superposition catastrophe: the use of tensor products for describing complex representations. As we shall see, whether this is really a solution is questionable. I shall argue that, however, some of its detractors (notably Fodor), miss the point.

The solution proposed by Smolensky is this. There are three kinds of entities we wish to describe:

semantic roles r_i

Contents f_j

Relations binding contents to roles

Semantic roles and contents are respectively represented by two distinct neural nets (R and F). The r_i's are vectors in a space V_R and the f_j's are vectors in a space V_F. Let u_ρ be the units of the net R and v_ϕ those of net V. To link the two nets we use hebbian connections: $w_{\rho,\phi} = \sum r_{i,\rho} f_{i,\phi}$.

Now (this is Smolensky's trick), we introduce new units $b_{\rho,\phi}$ connected by unitary weights to u_ρ and to v_ϕ (the activity of $b_{\rho,\phi}$ is then $w_{\rho,\phi} = \sum r_{i,\rho} f_{i,\phi}$).

As we have: $\sum w_{\rho,\phi} b_{\rho,\phi} = \sum_{\rho,\phi} \sum_i r_{i,\rho} f_{i,\phi} u_\rho \bigotimes v_\phi = \sum_i r_i \bigotimes f_i$, the expedient represents an implementation of the tensor product $V_R \bigotimes V_F$. Consider the following (elementary) example. Let the space of contents V_F (for the sake of simplicity I shall only use localist representations) be 3-dimensional. The three vectors $\overline{f_0}$, $\overline{f_1}$, and $\overline{f_2}$ represent respectively the contents "Mary", "Eats" and "Chicken".

[24]Smolensky [85].

The three vectors $\overline{r_0}$, $\overline{r_1}$ and $\overline{r_2}$ represent respectively the roles *subject*, *verb* and *object*.

The net implementing the tensor product $V_R \bigotimes V_F$ describes complex representations (obtained by superimposing terms belonging to the 9-dimensional vector space). For example, the vector $\overline{f_0} \bigotimes \overline{r_0} + \overline{f_1} \bigotimes \overline{r_1} + \overline{f_2} \bigotimes \overline{r_2}$ represents the proposition: Mary⨂Subject+Eats⨂Verb+Chicken⨂Object.

6.5.6 Problems with the connectionist solution

Fodor and McLaughlin[25] argue that: (1) The decomposition of a vector is arbitrary. (2) The constitutivity of a vector representation is therefore also arbitrary. But (3) we need the constitutivity of representations to be causally efficacious, so (4) Smolensky's solution cannot account for the intrinsic constitutivity of mental representations.

The conclusion of their argument may well be correct (indeed, if our analysis is sound, nothing could account for intrinsic syntactic properties), but the argument itself is certainly wrong. In point (1) the authors (presumably) refer to the fact that there always is an infinite number of possible choices for the basis of a vector space. Given a vector space V, in fact, there is no canonical basis: there always exists a non trivial symmetry group (GL(V)) whose elements are the invertible linear transformations. The plot thickens when we ask what is the relevance of this fact for the causal structure of the physical system that is being described. I argue that Fodor's argument rests on a mistaken answer to this question. In order to make my point clear, I have dissected Fodor's argument in the following chain of syllogisms (I have also included the necessary, although not expressed assumptions):

[25]See Fodor and McLaughlin [39].

1.1 In a mathematical model only canonical properties can have a factual value with respect to the physical system they describe.

1.2 The decomposition of a vector (i.e. the choice of a basis) is not canonical, hence:

1.3 The decomposition of vectors cannot have any factual value.

2.1 The choice of the basis of a vector space is not factual.

2.2 In Smolensky's model the individuation of a syntactic structure depends on the choice of a basis, hence:

2.3 The structures individuated by the model cannot be factual.

3.1 The structures individuated by Smolensky's model cannot be factual.

3.2 The constitutivity of mental representations must be such that syntactic structures are causally efficacious (cognitive processes are sensitive to syntax), hence:

3.3 Smolensky's model is inadequate to explain the constitutivity of mental representations.

Criticism of the Argument

The argument, I think, rests on a false premise. Proposition 1.1 is false: it is only true if we assume that syntactic properties are intrinsic to their bearers. Once again, I shall argue, a reductio ad absurdum proceeds from the false premise that syntactic properties adhere to their bearers.

The following example shows, by contrast, what is needed to prove that a mathematical structure is not factual with respect to the entity that it purports to describe. As we shall see, short of further information, the isomorphism between a mathematical description and the represented field is insufficient to fix an intended model.

In classical physics the positions and velocities of particles are described by vectors and, as we have seen, the decomposition of these vectors is not canonical (it depends on the arbitrary choice of the frame of reference). Do decompositions have a factual value? In other words, assuming that our theory of motion is correct, can we "deduce" that the choice of a frame of reference has no factual (i.e. metaphysical) correlate? As a matter of fact, the answer to this question is yes: the choice of a frame of reference has no physical relevance. However, this conclusion is far from trivial and cannot be derived from the mere fact that decompositions are not canonical. The correct argument is this:

4.1 Given the field equations of Newtonian Physics (defined, independently of co-ordinate systems, by abstract geometrical objects), and given the (metaphysical) hypothesis that the objects corresponding to the affine connection, time and space are absolute (galilean relativity), the symmetry (invariance) group is isomorphic to the covariance group of the standard formulation of the equations of motion.

4.2 The covariance group of the standard formulation of the equations of motion is isomorphic to the group of linear transformation of co-ordinates. Hence:

4.3 The (mathematical)fact that there is no canonical basis for the space describing the positions of particles suitably describes the fact that the choice of a basis is not causally relevant.

The conclusion rests on the metaphysical choice of absolute objects, from the field equations and from the standard formulation of the equations of motion. Notice that, for example, if we follow Newton and add three-dimensional space among the absolute objects of the theory, in spite of the symmetry of the group of linear transformations, and of the isomorphism with the covariance group of the standard formulation of the equations of motions, the choice of a basis at rest with respect to absolute space would not be "causally irrelevant". Once again, we are faced with the fact that isomorphism, short of further information, bears no factual relevance as to what it is an isomorphism with.

What lesson should we learn from this example? Mathematical structures are neither factual nor causally irrelevant, per se. The arbitrariety of the choice of a basis in Smolensky's model, similarly, doesn't entail anything (a priori). Notice, for example, that there is only one basis whose elements correspond to the activation values of the units of the net. Can a structure defined on this basis be causally efficacious? Again, no a-priori argument could settle this issue.

There is a sense, however, in which Smolensky's proposal falls short of accounting for the causal role of syntactic structures. The semantic roles, in Smolensky's model, are "imposed" from above. How can we be sure, then, that the model is not (like Fodor thinks) the model of an implementation of a symbolic model? Structures do not emerge as a result of specified causal happenings in the net. Both proposals (Fodor's and Smolensky's) are silent on this issue. However, Fodor's symbolic approach is at an advantage on this point. The symbolic paradigm, in fact, doesn't claim to explain the nature of its implementations: the computationalist's challenge is to guarantee the explanatory power of formal structures without dirtying his hands by making explicit the causal structures that implement them. The computationalist assumes a genuine under-determination of the causal mechanisms that implement the symbolic structure of cognition.

Understanding the mechanisms that, at the neural level, implement this structure (even if this were possible) wouldn't explain cognition, as it would give us an idiosyncratic picture of what cognition really is: it would prevent us from understanding cognition at the appropriate level of abstraction.

The connectionist proposals, on the contrary, aim at instantiating (not implementing) cognitive principles through formal principles derived from the mathematics of dynamical systems. In this sense Smolensky's proposal (the tensor product model) falls short of satisfying the desiderata of the connectionist hypothesis. In fact the proposal to deploy tensor products to mental representations only aimed at solving a hard problem of connectionist models: binding variables to their values. Does this solve the problem of rendering the constitutivity of representations? It depends on what we demand from constitutivity.

I have argued that the constitutivity of mental representations presents the theorist with the necessity to account for the causal status of syntactic structures. However, the entities realizing these syntactic structures need not be described, as isolated entities, by a mathematical apparatus that presents a canonical decomposition that can be mapped onto it. If my analysis is correct, in fact, we should expect no such thing. Even if there were a mathematical description of symbolic complexes that presented a canonical decomposition that mapped onto their alleged syntactic structure, this fact alone would not speak in favor of the model: for we would still further require that real semantic properties (that match such decompositions) be instantiated, in order to fix the wanted model.

6.5.7 Third Connectionist Response: Functional vs/ Concatenative Compositionality

More in keeping with our analysis, is another line of response to Fodor's criticism. This view grants that a cognitive system must operate on compositionally structured representations, but stresses that it is not necessary that such structures be *explicit.* Timothy van Gelder[26] suggested that we distinguish between *concatenative* and *functional* compositionality. In the paradigmatic case of compositional structure, linguistic structures, the composition of elements to build a complex compound is performed by putting together and ordering tokens of the same elements, without altering them. Van Gelder calls this kind of composition *concatenative.*

As a matter of fact, however, we don't need this kind of compositionality, for the minimal requirement is that structure be retrievable by some operation: this second, less demanding, kind of composition is called *functional.* Smolensky's tensor product

[26]Van Gelder[91].

networks and Pollak's recursive autoassociative memory networks (RAAM) are examples of networks that exhibit functional, albeit non concatenative, compositionality. I have briefly discussed the tensor product strategy in the previous section. I will now introduce some relevant aspects of Pollak's model of language structures.

Pollak's RAAM networks

In his 1990 study[27], Pollak has proven two interesting results. By training a Recurrent Auto Associative Memory (RAAM) network on explicitly compositional structured inputs he showed that concatenative, compositional trees can be coded in a distributed non-symbolic representation and then decoded and recovered as output, unchanged. Moreover, he showed that such distributed representations could be used by another network to perform inferences that required sensitivity to compositional structures. This, in Van Gelder's terminology, shows evidence of functional compositionality being in place.

The goal of Pollack's simulation was to represent recursive structures of various lengths (linguistic trees) using representations of fixed length (patterns of activation of input units in a connectionist network). So, for example, a sentence like "*Pat new John loved Mary*" was to be recast in the nested proposition: *(Knew Pat(Loved John Mary))*. The components of this nested structures are the triplets $P_2 =< Loved, John, Mary >$ and $P_1 =< Knew, Pat, P_2 >$. P_1 is the whole tree and P_2 is embedded in P_1.

The input pattern of the network was distributed over 3 sets of 16 units. In each set a distribution of activity represents a word (A noun or a verb). Although each unit can take any value between 0 and 1, input units are always assigned either value

[27]Pollak [70].

0 or value 1. Semantically similar words are represented by vectors of activation that are close to each other. A distribution of activity representing one of the four names presented to the network, for example, has value 1 on unit 5, and a different, individuating pair of values (various combinations of 0 and 1) on units 6 and 7; all other units of that set have value 0. When a sentence was composed of more than three words, recurrent connections were used to present it to the network (the same three 16-unit input sets were used to process all sentences, no matter how long).

The activity of the three input sets was passed on to a set of 16 hidden units through 3x16x16 weighted connections. The sentence presented to the network was therefore represented in a compressed way (encoded) by these connections.

The encoded representation activity of these hidden units was finally passed on to three 16-unit output sets, again through 3x16x16 weighted connections. The target was for these latter connections to decode the compressed information and reproduce as output the same sentence presented as input. In the case of embedded sentences it was necessary to go through successive recurrent encodings and decodings.

When the network was trained, it successfully processed (reproduced decoded outputs identical to the inputs) sentences up to four levels of embedding. When presented with sentences of higher levels of embedding, the network performance is expected do degrade (gracefully). As humans show a similar pattern of degradation in their performance, when presented with long, complex sentences, rather than constituting a shortcoming of the model, this feature is taken to be evidence of its soundness.

This is a rather common reason for debating over the explanatory capacity of connectionist models. Classical, symbolic models, where explicit (concatenative) structured representations are used, don't naturally perform this way. It is however possible to design a mechanism that operates on explicitly structured representations so as to mimic human performance. What does this tell us about the two ways of modelling cognitive performances? As they can both achieve similar positive simulations, it is impossible to rule one of them out. However, connectionists usually argue that it is preferable that degradation of performance happens as a natural (unintended) result of the workings of the model, rather than being imposed on the system as an additional feature.

The two parts of the network, once it has been trained, can then be detached and used as encoder and decoder of structured information.

The analysis of the performance of Pollak's RAAM networks showed some interesting features. Cluster analysis of the activity of the hidden units[28] showed that the network had attained relevant generalizations about semantic roles. Verb phrases and prepositional phrases, for example, gathered in separate clusters. But had the network achieved the right kind of productivity and systematicity? Pollak argues that the model was able to show some degree of productivity, as it was able to encode and decode patterns that were structurally similar to those used in the training. The capacity to correctly process new sentences, however, didn't achieve full productivity.

Again it can be argued that humans also show limitations in their capacity to produce and understand new sentences when they become too complex. A preliminary study also showed some degree of systematicity in the performance of the network,

[28]Cluster analysis displays the distribution of hidden unit vectors of activation after training, making it possible to observe what implicit *knowledge* the network has attained.

as the model was able to generalize correctly over sentences that had not appeared in the training but that made use of the same words.

Again, however, the amount of systematicity that could be observed was limited. A more accurate analysis of what kind of productivity and systematicity the network had achieved could only be assessed by studying what could be done using the encoded information.

It is clear that Pollak's analysis is carried out under the assumption that factual correspondences (between the syntactic structure of the input and the cluster distribution in the activity of hidden units) is sufficient for encoding. We have criticized this encodingist understanding of cognition. Again, the distribution of vectors of activation of hidden units after training does not bear any information as to what it is isomorphic with. Thus we may speak of encodings only in a loose, uninteresting way: i.e. from the intentional perspective of a cognitive system (the scientist's cognitive system) that provides both (1) the inputs with a syntactic structure and (2) the vectors of activation with the relevant semantics, so that, relatively to this semantic ascriptions, the two structures are isomorphic.

The relevant issue that would allow us to decide whether the model is or isn't an implementation of a computational structure (i.e. what physical properties instantiate the intentional capacity of the theorist), is not addressed by the model.

Pollak's proposal, however, is (potentially) suitable to describe the environmental differentiator that could explain higher cognition: the contact-making part of the system, as understood by teleological theories of intentionality. What is missing, therefore, is the content-making part of the system: the part with respect to which alone the system could be said to produce the relevant representations (for example by inducing the relevant dynamical presuppositions). If these presuppositions (or intentional icons, depending on what teleological theory of semantics is adopted) systematically matched the supposed syntactic structure of the input, then the network would be implementing a symbolic structure. If not, then the model proposed would not be computational, but it would be impossible, in this case, to claim that it instantiates any form of compositionality.

Operations on implicit representations

A number of studies issued from Pollak's simulation, mainly trying to process inferences or transformations over compressed representations that responded to Fodor and Pylyshyn desiderata. Pollak himself trained an additional 16-8-16[29] feedforward network to perform inferences of the kind: *Love X Y* implies *Loved Y X*. He then used the four names to built the 16 simple sentences, and the resulting 16 inferences. He succeeded in training the network to make the correct transformations over 12 of these sentences (compressed as discussed above). Moreover he showed that the model could correctly generalize over the 4 untrained sentences.

Other studies, like the ones of Blank, Meeden and Marshall[30], or Chalmers[31], managed to further improve on Pollak's results, showing that indeed it was possible to operate (albeit to a limited extent) on non symbolic representations in a structure-sensitive way.[32]

[29]This notation indicates that the network was built of 16 input units, 8 hidden units and 16 output units.

[30]Blank, Meeden and Marshall [22].

[31]Chalmers [14].

[32]Criticism of these results can be found in Hadley [43] or in Haselager and van Rappard [46].

6.5.8 Compositionality and computational externalism

I have argued that, even if we accept the validity of observer-relativity arguments, the notion of implementation can be seen as inheriting a "safe" observer dependence from its representational labelling scheme, i.e. one that can be projected onto a physicalistic language. So, I have argued, observer-relativity arguments do not necessarily jeopardize the explanatory power of computationalist theories of the mind. In commenting the nature of such observer relativity, I likened computational properties to secondary properties, such as colors.

The analogy is apt for expressing the advantage of externalist theories of implementation in treating the problem of constitutivity. Colors, as I have already mentioned, have for centuries been thought of as properties that adhered to their bearers. Our languages are still fully reminiscent of this (quite natural) understanding. We say that the sky *is* blue, just like when we say that the acceleration of a body towards the center of our planet *is* 9.8 m/s^2. We know, however, that it would be more appropriate to say that the sky *looks* blue, for blueness does not belong to the sky.

Our internalist intuitions about colors are so rooted that we are surprised when a beautiful light blue lagoon turns out, at closer inspection, to be made of nothing but a lot of transparent water. Understandably, we tend to ascribe intrinsic compositional properties to multicolored objects. Now, syntactic properties of representations (such as "being made of three components"), are not grounded on intrinsic properties of them, just like being a flag made of three colors is not a property that is grounded on intrinsic properties of the flag itself. I argued, in fact, that syntactic properties are fixed only relative to semantic ones, and these are not intrinsic properties of representations.

I think there is no reason to think that the syntactic properties of our mental states (or of a physical system, for that matter) should be grounded in a different way. Requiring that representations display *functional compositionality* is a step in the right direction. Think of the example of the three-sector flag. Rather than claiming that the flag is, intrinsically, divided into three sectors, it is more appropriate to claim that it possesses functional compositionality, by which it is meant that our (human) cognitive systems are able to systematically ascribe to it (and to similar flags) the property of being three-sectored.

Teleo-computationalism requires that the label bearers of the input architectures of computations be intentional icons. When do these representation possess compositional structure?

According to Millikan's teleological theory of intentionality:

> represented conditions are conditions that vary [...], in accordance with specifiable correspondence rules that give the semantics for the relevant system of representations. More precisely, representations always admit of significant transformations [...], which accord with transformations of their correspondent representeds, thus displaying significant articulation into variant and invariant aspects. If an item considered as compounded of certain variant and invariant aspects can be said to be "composed" of these, than we can also say that every representation is, as such, a member of a representational system having a "compositional semantics."[33]

Notice that, coherently with our analysis, this articulation of representations into

[33]Millikan [59], p. 224.

parts, supervenes on relational, externalist properties of the system using these representations. Consequently, being sensitive to such syntactic properties (which, we have seen, is definitory of symbol systems), cannot be a capacity that supervenes on intrinsic properties of their implementing system alone.

According to the teleological computational sufficiency hypothesis, the computational properties of a system partly supervene on the representational properties of the input label bearers. These, and their syntactic properties, in their turn, supervene on teleological properties as specified above. Mental states, i.e. teleo-computational states, according to this view, inherit the syntactic articulation of the input label bearers.

Applying my analysis to the debate over compositionality has, I believe, the advantage of highlighting a shortcoming of orthodox computational models that has so far been overlooked. The kind of complexity that a system of symbols must possess for it to be a model for cognition, we have seen, is *a combinatorial complexity that makes infinite use of finite means.* A result of my analysis is that no system can make infinite use of finite means, without being endowed with autonomous intentional properties.[34]

Let me turn back to the connectionist strategies for combatting Fodor-like objections. From the standpoint of computational externalism, connectionist models implement classical computational ones only if articulated representations of the kind mentioned above can be found (relative also to the environment) bearing the labels of the inputs and outputs to the implementing system. A consequence of my analysis

[34]Notice that Skölem's theorem can be proved precisely because our theories only feature a finite number of symbols. The infinite use that can be made of these finite means, however, is insufficient to fix the intended model, unless these finite means are additionally endowed with intentional properties.

is that no amount of a-priori reasoning can settle the issue of compositionality.

6.6 External symbols: varieties of externalist proposals

6.6.1 Fourth Connectionist Response: External Symbols

Elman's model: Learning Grammatical Categories

Another possible connectionist response is based on the observation that the processing of sentences happens sequentially in time: we read, or hear, or write or utter sentences one word at the time (one sound or sign at the time, for that matter). In spite of this, we are able to be sensitive to long-distance dependencies: a noun may be in accordance with a verb that occurs very far from it (in the sequential occurrence of words). This implies that a cognitive system must withhold information about previous occurrences of words.

The RAAM model described above was capable (through the deployment of recurrent connections) to hold information about parts of a sentence previously input (but no longer present). As we shall see, Simple Recurrent Networks (SRN) have been used to process sentences one word at the time.

In 1990 Jeffery Elman, following a study by Micheal Jordan[35], developed a model[36] that "memorized" previous occurrences of words. At any given time the activity of hidden units was copied back to some additional input units (called *context units*). So at every time the network's input consisted of the novel pattern of activation plus the activity of hidden units at the previous time step. Such activity is of course itself the result of the input pattern at the previous time step plus that at the time before

[35]Jordan [50].
[36]Elman [31].

that. The system can thus retain (to a gradually degrading extent) information about various previous input patterns. The task of Elman's study was to figure out if such a model could show sensitivity to grammatical structures without these being explicitly encoded anywhere.

The model had 31 regular input units, 150 context input units, 150 hidden units and 31 output units. Unlike Pollak's RAAM model, words were represented in such a way that their activation vectors where all mutually orthogonal (this ensured that no semantic information was surreptitiously fed into the system). He used these representations to form a 29-word vocabulary, and these 29 words to form 10,000 simple sentences. He finally concatenated these sentences to form a 27,354-word complex with no indication as to where a sentence begins or ends. Feeding the network one word at the time, he trained it to predict the next word. Obviously, not all words have the same chance of being the "next" word at a given time.

Grammatical constraints, as well as semantic constraints, in fact, forbid or elicit the use of one or another word. The network succeeded (predictably) in learning the correct statistical dependencies. Although this doesn't seem like a very interesting result (and as far as performance is concerned, indeed, it is not), what was interesting was how the network achieved its results.

Elman applied cluster analysis to the hidden units of the trained network. Interestingly, various grammatical and semantically correct generalizations had been made by the network. The analysis, for example, showed that verbs and nouns, in their hidden-unit implicit representations, gathered into two distinct clusters. Moreover, at a smaller scale, the vectors "representing" the words in hidden-unit space distinguished animate from inanimate nouns, and even domestic from aggressive animals.

So not only did the network succeeded in detecting grammatical structures from a corpus of purely statistical information, but it also managed to infer some semantic information. Again, this analysis would at best individuate some smart way to differentiate environments so as to support adaptive dynamical presuppositions. This would indeed be a relevant result. But, again, no model for cognition would be supplied.

In 1990 Elman[37] developed a similar network that was able to process information (similarly input one word at the time in an uninterrupted chain of sentences) that pertained to a rather more complex grammar. This time complex, embedded sentences (allowing for relative, subordinate clauses) were input and the network was again trained to predict the next word. Again, the network learned to make predictions consistent with the statistics of the training set. It showed to be sensitive to long-distance relations: for example it correctly predicted a plural verb in accordance to plural nouns even when the two where distanced by complex subordinate clauses.

In 1994 Christiansen[38] extended these results to even more complex grammars and he compared the errors made by his network with relevant human performances.

These studies, I argue, far from providing reasons to reject computationalism as a theory of the mind, I argue, constitute the basis for a teleo-computationalist or, more generically, for an externalist computationalist research programme.

External Symbols

What interests me in the above connectionist strategy to meet the challenge of constitutivity, is that it hypothesizes that we learn to process structured information

[37]Elman [30].
[38]Christiansen [17].

without internalizing its compositionality. Classical interpretations have it that the semantic and syntactic features of languages are parasitic (derivative) on those of thought. The language of thought is then assumed to be the primal source of compositionality and meaning. So, for example, the analysis of the semiotic vocabulary that I presented in section 3.3.4 is thought not to apply to mental representations: whence the thesis that syntactic and semantic properties of mental states, unlike those of semiotic items, supervene on some internal property of the cognitive system. The kind of connectionist response sketched above, instead, assumes that the only real compositional and semantic structures are external symbol systems, such as natural languages.

While these proposals are considered by both their proponents and by computationalist opponents as potential alternatives to the orthodox symbolic understanding of cognition, computational externalism allows us to consider them as (possible) improvements on a computational theory of the mind. Attempts to accommodate the difficulties encountered by classical computationalist models, led some to view symbol systems as being external to the computing system.[39]

In a book originally published in 1934, Vygotsky[40] had suggested that problem solving could be characterized as the manipulation of external symbols at very early stages of development, by means of what he termed *egocentric speech*; at later stages of development, according to this view, these procedures can be internalized to some extent by the use of an *inner speech*. Similarly, Smolensky suggested[41] that the capacity to manipulate symbols, originally formulated externally, is then internalized

[39]See Wells' proposal in the next section.
[40]Vygotsky [93].
[41]In Rumelhart [23].

and used by a *conscious rule interpreter.*

Most connectionist models of higher cognition try to implement a rule governed system. I have argued that, even where they are successful, these models leave the hardest foundational problem aside. It is very unlikely that a system that mimics the symbolic behavior of cognitive systems as described from the high level of the rules that it implements, would enlighten us as to what cognition really is, or on what computations it supervenes on. It is however well possible that when we will have a proper understanding of the physics of intentionality, cognition will turn out to supervene on computation. To this end, the last connectionist perspective envisaged above appears to be more promising.

Instead of trying to implement both the rules and the symbols by an isolated physical symbol system (a programme that I have argued to be not viable), it attempts to teach a connectionist network to manipulate external symbols. As William Bechtel and Adele Abrahamsen put it:

> The suggestion we are developing here is rather different from the approach of directly designing networks to perform symbolic processing. Rather than trying to implement a rule system, we are proposing to teach a network to use a system (language) in which information, including rules, can be encoded symbolically. In encountering these symbols, however, the network behaves in the same basic manner as it always does: it recognizes patterns and responds to them as it has been trained.[42]

In their seminal work, Smolensky, Rumelhart and Hinton[43] pictured a similar

[42]Bechtel [4], p. 191. For an attempt to apply this strategy see Allen [2].
[43]In [23].

circumstance (performing a multiplication) in the following way:

> We are good at "perceiving" answers to problems... However... few (if any) of us can look at a three-digit multiplication problem (such as 343 times 822) and see the answer. Solving such problems cannot be done by our pattern-matching apparatus, parallel processing alone will not do the trick; we need a kind of serial processing mechanism to solve such a problem. Here is where our ability to manipulate our environment becomes critical. We can, quite readily, write down the two number in a certain format when given such a problem
>
> ```
> 343
> 822
> ---
> ```
>
> Moreover, we can learn to see the first step of such a multiplication problem. (Namely, we can see that we should enter a 6 below the 3 and 2.)
>
> ```
> 343
> 822
> ---
>   6
> ```
>
> We can then use our ability to pattern match again to see what to do next. Each cycle of this operation involves first creating a representation through manipulation of the environment, then a processing of this (actual physical) representation by means of our well-tuned perceptual apparatus leading to a further modification of this representation [...] These dual skills of manipulating the environment and processing the environment

> we have created allow us to reduce very complex problems to a series of very simple ones. [...] This is real symbol processing and, we are beginning to think, the primary symbol processing that we are able to do.[44]

These proposals are intended as alternatives to computationalist models. I argue, instead, that they need not be considered as such. They should be viewed as part of a larger philosophical programme: the description of the workings of the contact-making part of a teleo-computational system. If the correct intentional items were proved to be in place, in fact, these systems would be implementations of externalist computational structures. Indeed, proposals that appear to go in this direction (i.e. suggesting that symbols must be expected to be found outside the cognitive system) can be found even within the orthodox computationalist camp, as the following example shows.

6.6.2 A tribute to Turing

Turing's analysis of routine computation constitutes undoubtedly a major source of inspiration, if not the foundation of contemporary computer and cognitive science. Much of this tribute to Turing's work is due to the development of the technology of digital computers, and to the intensive application of it to the understanding of human cognition. If the direct relevance of Turing's analysis for the building of the first digital computers is questionable, there is little doubt that when cognitive or computer scientists think of a digital computer, they think of a particular implementation of a universal Turing machine.

In fact, although digital computers are not (strictly speaking), implementations of a universal machine as it was understood by Turing, the best way of understanding

[44]Rumelhart [23], p. 85.

their functioning at an abstract level, is as universal machines subject, for practical reasons, to a number of constraints (such as a finite memory).

On the one hand mediating the conceptual import of Turing analysis via the architecture of digital computers has proved to be a most fruitful and powerful tool for understanding human cognition. On the other hand, Wells[45] argues, such mediation has obscured some conceptually relevant differences. As a consequence, realizing that classical (symbolic) AI finds itself at an impasse (for reasons to be discussed), many authors conclude (through a false reductio ad absurdum) that the false premise is Turing's understanding of cognition.

So, while the functioning of digital computers takes (justly) the credits for the outstanding successes of classical AI, the architecture of Turing machines takes (unjustly) the blame for its alleged failures. At the most abstract level a TM is made of a finite control machine, a (infinite) tape and a read/write head connecting them. This is the "structural architecture".

Each TM is characterized by the set of its states, by the symbol alphabet it uses (together with the syntactic constraints on the strings it receives as inputs) and by a specific systematic interaction between these two. Such characterization constitutes the "task architecture". The internal states of the TM constitute the "control architecture", while the system of symbols with the syntactic constraints constitute the "input architecture". If we call $Q = \{q_i | 0 < i < m + 1\}$ the set of internal states, the control architecture, $S = \{s_j | 0 < j < n + 1\}$ the alphabet of symbols, and $D = \{L, R, N\}$ the set of possible movements, we can characterize the interaction between control and input architecture with a function F from $Q \times S$ to

[45]Wells [94].

$S \times Q \times D$. This constitutes the "machine table" for a specific task.

This architecture cuts across the distinction between universal and non-universal machines (this depends on whether what the machine takes as input can be interpreted as Turing-machine-tables or not). As described above, the structural architecture of a TM comprises a finite control machine, a (infinite) tape and a read/write head connecting them. In the von Neumann architecture the infinite tape is realized by a finite (although large) electronic memory. The control architecture is realized by the CPU.

In the abstract Turing architecture the read/write head moves about (according to the instructions output by the function F) "scanning" the tape for the next symbolic input (operand). In the von Neumann architecture the read/write head is realized by a "hard wired" connection between the CPU and the electronic memory. This means that each "location" in the electronic memory (whose position is symbolically represented by the use of address arithmetic) is physically connected to the CPU, and can be "addressed" by it when required by the current output instructions.

The only conceptually relevant difference between a physically realized von Neumann architecture and an ideal implementation of a universal Turing machine is the finiteness of electronic memory. The fact that there is no read/write head scanning a tape bears no conceptual relevance whatsoever.

Before turning to the interactive approach it is worth remarking another difference (again of a practical and not conceptual kind) between universal Turing machines and von Neuman machines. Turing machines were designed to model ("imitate", as Turing would say) a real human computer facing the task of routine computation with the help of (infinitely many) sheets of paper and a (never ending) pen. Having said this it is not hard to imagine where the idea of an infinite tape (paper) and of a read/write head (pen) came from. This fact seems totally irrelevant to the Turing/von Neumann "trade off". It sounds like more of a historical curiosity than a perspicuous aspect of the structural architecture of Turing machines. Although to a large extent it is a curiosity, Wells argues that there is a sense in which it isn't.

For the moment it suffices to notice how in the von Neumann architecture the "pen/paper" analog is irremediably lost: as I mentioned, in a VN machine there is no external tape on which symbols are "read" and "written". Admittedly the words "read" and "write", especially after the cognitive revolution, don't seem to be much more than a metaphor even in the case of a TM. They cease to have any descriptive efficacy when applied to a VN computer.

Before considering whether the read/write device of a TM bears any conceptual relevance with respect to cognition, one can observe that it certainly has a practical relevance. The VN machine as I described it, in fact, doesn't have any practical use whatsoever: it is a closed box. Actual computers are always provided with peripheral devices, such as screens, keyboards and mouses that ensure that a user can interact with the machine. So reading and writing is done (non metaphorically) by a human user.

However, these (keyboards, screens and human users) are not part of the architecture of a VN machine. Contemporary main stream theories of cognitive architecture pay a large tribute to Turing's analysis of routine computation. The hypothesis that, in different forms, underlies most of the work done in the field is that the architecture of human brains implements the architecture of a universal TM.

This hypothesis (TM hypothesis) can be understood in various ways. Wells argues that, because of the particular way in which TM's have been implemented, the paradigmatic way of understanding the hypothesis owes a lot to the peculiarities of VN architectures. As we have seen, the architecture of digital computers has the full structural architecture of a TM encapsulated in one unit that is then embedded in the environment through peripheral devices. As a consequence, the TM hypothesis is commonly understood as saying that the brain implements the full structural (input and control) architecture of a TM. Sensory organs and motor outputs provide then the appropriate connection with the environment.

6.6.3 The interactive approach: Turing computation with external symbols

The proposal is to amend the above outlined received understanding of the TM hypothesis by restoring (or rather re-emphasizing) the original distinction between infinite tape + symbol alphabet (input architecture) and finite state control (control architecture). In Wells' understanding, if Turing's analysis has anything to say about cognition, the TM hypothesis should be stated as: the architecture of the brain and that of its environment (together) implement the architecture of a TM.

The need for such an amendment stems from acknowledging a number of problems of the received view. Predictably, the class of problems we are looking at has to do with how we can appropriately embed the full structural architecture of a TM in the environment. Here "appropriately" means in such a way as to account for the large number of behavioral regularities that we observe in agent/environment interactions.

As a consequence of these problems (and others that I have not mentioned) many authors opted for a thorough reconsideration of the theoretical premises. They, as

I said, blamed Turing. Wells, so to speak, blames von Neumann. The proposal is to restore Turing's original analysis where, playing the imitation game in the shoes of the human computer, there was the finite state control, the role of the sheets of paper being played by the infinite tape. Turing's example represents a rudimental example of interactive cognitive architecture. The following step is to emancipate the architecture from the particular input architecture Turing had in mind.

The tape, in fact, was only needed to ensure that the control architecture scanned a "square" at the time, detecting one out of a finite set of alphanumerical symbols. Emancipating from the alphabet of alphanumerical symbols is probably the easiest step. Turing machines do not "know" what they are detecting, any object or event would do just as well (provided it can be suitably type-identified). Non-symbolic processing is certainly capable of type-identifying events and objects in the world (so long as this claim is not taken to simply mean that non-symbolic processes can establish factual correspondences with disjunctions of physical properties of the environment).

So let's suppose that we have an "alphabet" of (primitive) symbols (in a sense to be defined). Attentive, emotional or affective processes might then be called into cause to compensate for the lack of a tape that is scanned one square at the time. The tape is the environment itself, and when the sensory ports, the "doors of perception", are open, the tape/environment is constantly scanned. Non symbolic processing is then responsible for selecting and type-identifying the symbol (or symbols) that are being input at any given time. Only now does the computational machine (the real self) enter the game, producing the output (or the outputs) that the machine table prescribes (these can also involve a mere change of internal state, and don't necessarily involve a physical action to take place).

At the end of chapter 2 (section 2.6.5), I discussed a potential a-priori objection to my view of implementation, that consisted in pointing out that there are computations whose input architecture does not appear to be systematically interpretable in a meaningful way. As an example of such computations I mentioned Buntrock and Marxen's algorithm: the simple five-state machine that is proved to halt after 23,554,764 steps. Piccinini, we have seen, uses the algorithm to argue against a semantic view of computation.[46]

Wells, instead, uses it to corroborate his interactivist view of computationalism: its surprisingly complex behavior, in fact, "*is not predictable from an examination of the internal structure of the machine. It is an emergent property of the interaction between the machine and the sequence of symbols that it leaves on its tape. [...] Buntrock and Marxen's machine*", Wells argues, "*shows that behavior cannot be understood through an analysis of either the internal structure of the cognitive system or the structure of the environment alone. There can be no substitute for studying behavior interactively, as it unfolds.*"[47]

The above outlined picture of computationalism is arguably a brand of computational externalism. It is interesting to investigate what this picture has in common with the view advocated in this thesis.

Computational externalism and the interactive approach

Does the expedient of having the symbols (re-)placed outside the implementing RDS allow us to block v-arguments? No. If we grant the validity of v-arguments (as I do ex hypothesis), Well's understanding is also subject to the threat of observer-relativity.

[46]Piccinini [68] p. 6.
[47]Wells [94], p. 288.

The "computational machine", in fact, is assumed to implement the relevant computational structure on the basis of its physical properties alone.

Similar considerations apply to the external symbols, if no restriction is added to what could count as a symbol (other than the vacuously obtaining requirement that the symbols be "interpretable"). Put plainly, an interactivist, "long arm" individuation of computational properties is equally arguable to be incapable to rule out unwanted instantiations (if v-arguments are sound).

This, however, is not forced upon the approach. Indeed, the interactivist approach is better suited for exposing the virtues of an externalist understanding of computation, and the shortcoming of an orthodox view. We have argued, in fact, that the internalist construal of computation has been sustained by an implicit encodingist preconception of semantics. Keeping Well's analysis in mind, we can further argue that the persistence of the encodingist paradigm, in its turn, has been allowed, if not determined, by the the peculiarities of VN architectures.

If the symbols, as well as the inner computational structure, are thought to be compactly implemented by the same system, it is tempting to adopt an internalist construal of implementation and semantics. The internal causal structure of the system, in fact, would be responsible for both its states having the computational status they allegedly have, and for its symbols encoding the features of the environment that they allegedly encode.

It would then be "only" a matter of connecting this box to the environment in the appropriate way through peripheral devices. In ordinary digital computers, such connections are realized by the human users: the peripheral devices (keyboards and screens) do not do any transductive work by themselves. They do not, in other words, produce symbols from non-symbols: they simply encode already existing symbols (as, we have argued, must always be the case with encodings).

If we adopt the interactivist approach, the difficulties with this preconception become apparent. As the symbols lie outside, in the environment, it is no longer tempting to say that they possess meanings in themselves. I believe that this non-sensical, overtly encodingist picture of intentionality has never been advocated by aneyone.

The only reasonable option, for someone who wants to hold onto the internalist picture of implementation[48] while adopting an internalist theory of semantics, would be to claim that these symbols acquire their intentional status by virtue, and only by virtue, of the causal (conceptual) role of the states they induce in the computational machine: in fact, as the symbols may legitimately be assumed to be causally uncorrelated between each other, they cannot be assumed to have any causal role independently of the computing machine.

It should be clear that syntactic internalism, thus presented as an individual thesis (i.e. not accompanied by some encodingist thesis about intentionality), is not viable. It is in fact exposed, when applied to a theory of the mind, to the criticism of conceptual role theories of semantics that I have discussed in section 3.2.

The alternative route, for those who do not accept the validity of v-arguments,

[48]Notice that I use the expression *internalist picture* as meaning something different from Wells. In the cited work, the expression "internalist picture" refers to the orthodox construal of the sufficiency hypothesis as claiming that cognitive systems compactly implement both the control and the input architectures. In my work, instead, the "*internalist picture*" is to be contrasted with my externalist construal of syntactic properties as grounded on teleological semantic properties. As I have already pointed out (section 4.4.3), while my claim that computational properties should be individuated relative to the teleological intentional properties of the input architecture, entails that they should be individuated "broadly", like Wells urges, the opposite implication does not hold: Wells' interactive approach need not be committed to a semantic picture of implementation.

is to suppose that the symbols acquire their meanings by virtue of some fact that pertains to the cognitive economy of the implementing organism. Wells, for instance, hypothesizes that there might be "natural symbols", whose interpretations are "intrinsic". As an example of how these "intrinsic" interpretation may come about, consider the fear of snakes.

> It is not hard to see how evolution might have selected for a generalized avoidance/fear reaction to snakes based on the "natural syntax" of their shape and movement which type-identified them as creatures to which a particular class of response is appropriate. [..] Thus we arrive at the idea of a natural symbol as a type which has a built-in grounding arising from its adaptive significance.[49]

This picture of computationalism joins a broad construal of syntax (albeit of the non-semantic brand), with a broad construal of semantics. It is therefore precisely the same computationalist picture of the mind that is affordable by my theory of implementation (Computational Externalism).

The key difference is that while Wells maintains the syntax/semantics division of conceptual labor, I have the broadly individuated semantic properties entering the picture at an earlier stage, as a necessary condition for the applicability of computation to a theory of the mind.

6.6.4 Computational externalism and evolutionary theory

Among the items that interact with the implementing system in a systematic way (i.e. complying to some behavioral regularity), some will also comply with the conditions

[49]Wells [94], pp. 286-287.

for being representations (intentional icons, if we buy Millikan's theory, for example). These will be all and only the computations that the system implements.

The functions computed by the system will then be the ones that it implements (in the externalist sense of the word), when functioning properly (in the sense specified by teleological theories). In other words, the system will function properly only if the relevant equivalence relations obtain (those discussed in section 4.4.2).

One of the advantages of having the symbols laying outside of the computing system, as the interactivist picture prescribes, is that they can be manipulated so as to comply with the relevant constraints. One of the alleged shortcomings of the orthodox view of computation, in fact, is that it is not particularly friendly to evolutionary theory, when applied to a theory of the mind.

The rigidity of computational structures, conceived as implemented by the intrinsic properties of the implementing systems, with the consequent fragility that this entails, leaves one wondering how the random variations selected by evolutionary pressure might have brought these computational properties about.

Moreover, supposing that a computational structure was achieved by sheer luck, it is not clear how subsequent random variations would not fatally disrupt such architecture. This unwanted feature has often been mentioned by the detractors of computationalism as a major shortcoming. Wells dubbed it: the *evolutionary problem.*

To better contrast this difficulty with the relative advantage of the externalist stance, consider the following picture. Consider the space of maximal internal configurations of biological systems (each point of this space represents the precise physical description of a possible biological system at a given time). All biological systems, as described at the physical level, spend their lives moving about this space. If we assume an internalist stance, the configurations (the points) that comply with the conditions for being implementations of computational structures, will be scattered about the space, but isolated. Let us call these points: *computational points.*

Under internalist assumptions, even the slightest displacement from a computational point (caused by gene mutation or crossing through different generations, or by natural causes within the same organism), would determine the abrupt fall of the system (or species of systems) from the "computational heaven". So, even assuming that a system (or its species) managed somehow randomly to reach an adaptive computational point, it is reasonable to assume that this would not be a stable equilibrium.

As a consequence, a "computational strategy", i.e. the pursuit of adaptive advantage on the basis of the capacity to implement a certain series of computations, would soon be selected away. The traits that get selected must be (almost by definition) stable points of evolutionary equilibrium in the motions of biological systems through their configuration space. Another way of putting it, is that the amount of information needed for computational points to be stable evolutionary equilibria would be unreasonably high (if available at all) for biological systems as we know them.

Contrast this with computationalism as understood from an externalist perspective. As the conditions of applicability of the notion of implementation are externalist (relational), computational points need not be isolated and scattered in the space of internal configurations of systems. The obtaining of the relevant equivalence relations, in fact, is compatible with large portions of configuration space: as it does not supervene on intrinsic properties of the system alone (because points in configuration space now represent properties of both the system and of its environment), the computational status can be maintained in spite of ontogenetic or filogenetic variations. The additional degree of freedom is provided by the possibility to actively intervene on the environment so as to preserve a certain computational status.

Under Well's interactivist approach this advantage is particularly obvious: it is reasonable to suppose that both (1) the internal causal makeup of a system (its position and movements in internal configuration space) and (2) the properties and representational status (relative to it) of the items of its environment, would have co-evolved.

It is now no longer unreasonable to suppose that computational points be stable evolutionary equilibria: it suffices to suppose that a system (and its siblings) be actively engaged in the attempt to preserve their computational status by manipulating their representational environment. The attentive, emotional or affective processes that Wells hypothesizes to supply the lack of a tape that is scanned one square at the time, from the standpoint of external computationalism, need not perform the daunting task of filtering everything that is input to the system: they only need to "scan" the inputs that correspond to intentional icons.

This is not to say that these items must be fixed once and for all in the environment of the system: we can imagine that an original set of fixed items constitutes the basis on which the representational repertoire of the cognitive system (and consequently its repertoire of computations implemented) is then built through learning. When or whether this is the case now only depends on empirical investigation.

6.7 Teleological Computationalism and the proper description of cognitive systems

6.7.1 The thesis that all cognitive systems are dynamical systems

It is obvious that only RDS's really undergo change in space-time and that MDS's describe some (usually not all) aspects of this change. Given a certain RDS, then, several MDS's can be said to be instantiated by it. Given the two possible meanings of the expression "dynamical system" (real dynamical system and mathematical dynamical system), to say that a given real entity is a dynamical system is either trivial or absurd. In fact, if the claim is intended to say that the entity is a RDS, then this doesn't add anything to the notion of a real entity: any real entity is a real dynamical system. If, on the other hand, the claim is intended to say that the real entity is a MDS, then this is an absurdity, for it identifies a real entity with a timeless mathematical description.

However, there is a third way to interpret the claim that a real entity is a dynamical system. Any real entity, I repeat, is a dynamical system. Nevertheless not all entities are, as a matter of fact, described by means of the mathematical dynamical systems that they instantiate. Sure a school is a (real) dynamical system, but no policy about education is decided by means of a mathematical description of its change in time. There is then a methodological understanding of the claim that an entity is a dynamical system. According to this understanding the claim is to be interpreted as the claim that a certain (real) entity can be and should be understood by means of a mathematical description that pertains to the theory of dynamical systems.

Giunti[50] has argued that "all cognitive systems are dynamical systems". The thesis is to be understood as a methodological claim. The premise of Giunti's argument is that cognitive systems (to our best knowledge) belong to at most three categories: symbolic processors, neural networks and other continuous systems described by a set of differential (or difference) equation. The argument consists in showing how a system that belongs to any of these three categories is a dynamical system (in the methodological sense specified above).

Now, systems belonging to the third type are obviously dynamical systems. Neural networks also are clearly dynamical systems: a complete state of them can be identified with the activation level of its units and the change they undergo can be specified by the differential equations that regulate the updating process. The burden of the proof is then left to the argument that symbolic processors are dynamical systems. The strategy is to consider a specific kind of symbolic processor: Turing Machines.

The future behavior 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. We can therefore take the set of all triplets $< state, headposition, tapecontent >$ as the state space M of the system. The set of non-negative integers can be taken as the time set T. The set of quadruples of the machines is then used to form the set $\{g^t\}$ of state transitions. This shows that all Turing machines are in fact dynamical systems, for they can be described by a mathematical structure $\langle T, M, \{g^t\}\rangle$ with T being discrete. In other words any Turing machine can be described by a cascade.

[50]Giunti [41].

Research in cognitive science is characterized by various attempts to provide an explanation of cognitive phenomena based on the study of models that reproduce some aspect of the change of the system. These models are (so far) symbol processors, neural networks or systems governed by differential (or difference) equations. All of them are MDS therefore, the argument concludes, all cognitive systems are dynamical systems. The thesis should probably be rephrased in a less ambiguous way along one of the following lines. Cognitive scientists believe that cognitive systems are dynamical systems. Or: the object of cognitive science has so far been the study of a particular kind of dynamical system. Or: all attempts to explain cognition at a fundamental level within a scientific framework have been characterized by various attempts to determine the dynamical system that appropriately describes some aspects of it.

The argument, I believe, has the effect of conferring a substantial degree of methodological unity to the research programme of cognitive science. The following paragraph will be instead devoted to the discussion of some significant methodological distinctions.

6.7.2 The two main explanatory styles

The above considerations seem to narrow the gap between the theoretical stances that can be found in the literature. One could say that scientists are just arguing about what mathematical model we should use to describe cognition, but that they all agree on a main point: an explanation of cognition should be in the form of a (mathematical) dynamical system that describes it. At the methodological level, however, the different approaches lead to very different strategies. These strategies inevitably bring with them different conceptual apparatuses, and this cannot but produce different pictures of what cognition is, at least conceptually and most likely also metaphysically.

If the models adopted by cognitive scientists, as we have seen, can be sorted into three categories, explanatory styles (or methodological strategies) can be sorted into two main families: computational and dynamical. Computational strategies are characteristically conceptually grounded in computability theory. Dynamical strategies in dynamical system theory. Traditionally symbolic processors have been studied by means of the conceptual repertoire of computability theory, while neural networks and other "dynamical" models by means of that of dynamical system theory.

Here I want to discuss whether this is a mere historical happenstance or whether it can be justified by showing that there are dynamical systems (RDS) whose behavior can only be explained by using one or the other conceptual repertoire. I shall discuss, that is to say, if there exists a RDS that cannot be described using concepts drawn from computability theory and if there exists a RDS that cannot be described by using dynamical system theory. To answer these questions we need a more precise characterization of "symbol processors".

There are two kinds of symbol processors: automata (like Turing machines) and systems of rules for symbol manipulation (like Post canonical systems or tag systems). As we have seen, these systems can be described mathematically by discrete-time dynamical systems (i.e. cascades). It is legitimate to ask whether the concept of cascade is coexstensive with that of symbol processor. It turns out that the concept of cascade is too broad. In fact not all cascades are symbol processors, for the latter additionally require that the cascade be effectively describable.

Notoriously the notion of effective procedures is an intuitive one, and one can only hope to capture it appropriately by a formal concept. Traditionally this has been achieved (through a very corroborated conjecture) by means of the concept of computable (recursive) function. The intuition behind the requirement that the cascade be effectively describable is that its workings must be "mechanical", or that they must be describable by an idealized machine. Giunti[51] provided a formal characterization of "effective describability" of a (mathematical) dynamical system.

[Description of a Dynamical System] A "description" of a dynamical system $S = \langle T, M, \{g^t\} \rangle$ is a second dynamical system that is isomorphic to it.

An effective description is naturally one for which the state space M_1 is a decidable set and for which every state transition h^t is a computable function (there exists a Turing machine that computes it). The notion of an effective cascade (a computational system), can thus be formalized in the following way:

[Computational System] S is a computational system if and only if $S = \langle T, M, \{g^t\} \rangle$ is a cascade and there exists another cascade $S_1 = \langle T_1, M_1, \{h^t\} \rangle$ such that:

(1) S is isomorphic to S_1

(2) If $P(A)$ is the set of all finite strings built from a finite alphabet $A, M_1 \subseteq P(A)$, and there is a Turing machine which decides if a finite string is a member of M_1.

(3) For all $t \in T_1$ there is a Turing machine which computes h^t.

[51]Giunti [41], p. 560.

It is clear from the above definition that if the description of a (mathematical) system S is such that either T_1 or M_1 is non denumerable then the system cannot be a computational system.

We can now answer the question asked at the beginning of this section. All (mathematical) systems whose time set or whose state space are non denumerable (such as neural networks that make use of continuous activation values or systems described by differential or difference equations) cannot be described within the conceptual framework of computability theory.

This result should not be confused with the (empirical) claim that there are (real) systems that cannot be described by computability theory. A (real) computer, for example, is a physical system that is describable by both a dynamical system and a computational system, depending on the preferred level of description. This does not contradict the above result: the claim, in fact, must be understood as referring to the impossibility that a dynamical description of a (real) computer be "translated" into a computational description.[52]

I now turn to the other question: is it possible to base a computational description on a dynamical description? Computational systems (effective cascades) are a special kind of dynamical system, so concepts drawn from dynamical system theory can be applied to computational systems. We can, for example, treat the processes of a symbolic system as motions within an orbit in a state space. We can distinguish periodic, aperiodic and eventually periodic orbits. Attractors and basins of attraction make also sense. These systems, however, usually lack a natural topology in their

[52]It is nevertheless possible that a computational description approximate a dynamical description and if the real numbers involved are computable the approximation can be carried out at an arbitrary degree of precision: this, however, has no conceptual consequences.

state space, and this precludes the use of many concepts that are commonly used in dynamical system theory (chaos theory, for example, falls beyond the scope of computational systems). It is natural to ask if forcing the mathematical apparatus of dynamical system theory to the restricted scope of effective cascades fatally reduces the explanatory power afforded by the full conceptual repertoire.

6.7.3 Dynamic system treatment of computational systems

It is an open question whether it is possible to recover the full explanatory power of the computational strategy if we take a dynamicist stance in treating computational systems. "Infiltrations" of dynamical concepts in the treatment of symbolic processors, however, are becoming very common in the literature. Here I consider two examples: one is Giunti's treatment of the halting problem in Turing machines and the other is Morris' analysis of Sosic and Gu's algorithm for solving the N-Queen problem. Notoriously the halting problem for Turing machines is undecidable: there is no universal (non specific to a particular machine) algorithm that can decide whether an arbitrary Turing machine halts.

Giunti provides a dynamical characterization of a machine halting. When a machine halts, from a dynamical point of view, it enters a cycle of period 1 in state space. There are two ways in which this can happen: either the machine enters the cycle immediately, or it does so after a number of steps (in this case we say that the machine has an eventually periodic orbit). Under what conditions can we be sure that a system does not have an eventually periodic orbit?

A theorem in dynamical system theory ensures that a system with an eventually periodic orbit has at least one transition g^t that is not injective. From this it follows that any logically reversible Turing machine (one such that all its state transitions are injective) cannot have an eventually periodic orbit. The only way in which such a machine can halt is if it halts in its first step. Thus, if we select the class of logically reversible Turing machines the halting problem becomes decidable.

An analogous case of application of dynamical system theory to computational systems is the treatment of Sosic and Gu's algorithm for solving the N-Queens problem. The problem is that of placing N queens on a $N \times N$ board in such a way that no two queens are on the same row, column or diagonal. The problem has at least two solutions (two different arrangements of the N queens) for every $N \leq 4$. The problem here is that of finding one (any one) solution. The most straightforward solution to the problem deploys an algorithm suggested by Sosic and Gu. In a first phase the system is to distribute the queens in such a way that no two queens are found in the same row or column. The problem is then reduced to replacing the queens in such a way that all diagonal collisions are also eliminated. The algorithm now prescribes that two columns should be swapped if and only if the result of doing so reduces the total number of collisions. The procedure is continued until when it is no longer possible to perform any further swapping: i.e. until when the total number of collisions has reached a minimum. In a generic programming language the algorithm has been described as:

```
1. Repeat
2. $Swaps_{index} = 0$
3. For i in [1...n] do
4. For j in [(i+1)...n] do
```

```
5. If $queen_i$ is attacked or $queen_j$ is attacked then
6. If $swap(queen_i, queen_j)$ reduces collisions then
7. Perform swap $(queen_i, queen_j)$;
8. $Swaps_{index} := Swaps_{index} + 1$;
9. Until $swaps_{index} = 0$
```

As a matter of empirical fact the algorithm has been proved to find a solution in 99% of cases. Why? There is no available explanation for the success of the algorithm that makes use of typical computational terms. Morris has explained the result by proving that the solutions are dense for the swap-the-column heuristics. The algorithm creates a space in which the density of solutions among equilibrium points approaches 1 as N increases: i.e. the ratio of the probabilities of relaxing in a non-solution (a configuration from which it is no longer possible to perform any swap but for which the number of collisions is not 0) and the probabilities of relaxing in an equilibrium point (0 collisions) rapidly approaches 0 as N increases.

It turned out that although the system is clearly algorithmic (it is described by an algorithm!) the explanation of its behavior is only afforded by referring to terms of dynamical system theory. The analogy between this argument and a typical dynamical description of the behavior of a system is more then superficial, and this can be better appreciated by confronting Sosic and Gu's algorithm with a typical "dynamical" approach.

The most used strategy for solving such constraint satisfaction problems is to deploy Attractor Neural Networks (ANN). These are connectionist networks whose state changes in a space according to a dynamics that "attracts" the system to "solution" points. The desired states are points (or areas) in state space that "attract" the trajectories of the system so that regardless of the initial state position the system (under certain conditions) is guaranteed to reach a stable relaxation point (a solution).

If an ANN is such that its weights are fixed, symmetric and non-reflexive and the activation values are updated one at the time then it is guaranteed to reach a solution in a finite amount of time. In solving the N-queens problem the board is represented by a (fully recurrent) network of N^2 units, where each unit represents the state of one cell of the board (the unit is activated if and only if there is a queen on it). The dynamics of the system is traditionally described by assigning an energy function (a function that assigns a value to each point in state space) and imposing that the system evolves in such a way as to minimize the function. The minima of the function (i.e. the equilibrium points) represent the solutions to the problem. Technically, an appropriate energy function for solving the N-queen problem has been proved to be:

$$E = 1/2A \sum_x \sum_k \sum_{j \neq k} a_{xk} \cdot axj + 1/2B \sum_k \sum_x \sum_{y \neq x} a_{xk} \cdot a_{yk} + 1/2C \sum_x \sum_k (a_{xk} - n)^2 + \ldots$$

$$\ldots +1/2\mathrm{D} \textstyle\sum_x \sum_k \sum_{m \neq 0} a_{xk} \cdot a_{x+m,k+m} + 1/2F \sum_x \sum_k \sum_{m \neq 0} a_{xk} \cdot a_{x+m,k-m}$$

When the constant values are positive numbers the function E has a global minimum (value 0) that is reached if and only if all the sums are zero. The first two triple sums are zero respectively iff there are no two queen on any given row and no two queen on any given column. The third sum is 0 iff there are exactly N queen on the board. The fourth and fifth triple sums, finally, are 0 iff there are no two queen on any given diagonal line.

It has been proved that the density of solutions (global minima) among all minima, for a system that uses the above energy function and is allowed to move in directions that increase it (to allow escape routs from local minima) is $\frac{N^2}{N^2+1}$.

This result is obtained from considerations that make reference only to mathematical relations between the energy function and other dynamical magnitudes of the system. Given an ANN like the one described above and set to solve the N-queen problem, the explanation of its behavior is, as we have seen, analogous to that of the behavior of the (algorithmic) system considered at the beginning of this section.

6.7.4 Computational externalism and the debate over cognitive architecture

What we have done is, first, provide a computational description of a procedure for the completion of a task; then, provide a dynamical system description of the computational one, that explains that the procedure achieves its goal, and why it does. What is this dynamical description a description of? As we have stressed, it is not meant to be directly the description of the behavior of a RDS, but rather the dynamical (mathematical) description of the computational description of the behavior of the RDS. So the order of instantiation is the following. First comes the real dynamical system. Its behavior is then interpreted as implementing a certain algorithm, or a certain connectionist network: these are the computational and connectionist descriptions of its behavior. Then dynamical system theory is used to describe the computational, algorithmic or connectionist description.

The fact that the genuine solutions are showed to be dense in the set of all solutions is deduced by using mathematical features of the model alone, and is used to conclude that the computational system described will achieve its goal, if implemented.

Now, the same mathematical dynamical system can be used to describe directly the connectionist model that performs the same task by minimizing the energy function described above.

Imagine that we came across an agent (not necessarily human) who we observe as she solves the N-queen problem. Let us call her Katherine. We observe, also, that Katherine's strategy for solving the problem systematically complies with the Sosic and Gu's algorithm presented above. We may ask what accounts for her behavior. Is she really implementing Sosic and Gu's algorithm, or does she just appear to be doing so? Is she instead implementing the neural network that minimizes the energy function described above? Are the two options compatible (for example because any instantiation of a connectionist network like that is, ipso facto, the implementation of Sosic and Gu's algorithm), or are they mutually excluding models?

Let us analyze how orthodox computationalists and connectionists would argue. The orthodox computationalist would argue that the physical object that is relevant for explaining the behavior of Katherine (her brain, if she is human, but any other object, as far as we know), must be implementing the algorithm.

Her sensory apparatus must be such as to transduce the signals that depart from the wooden pieces (the queens) and from the squared patterns on the board, into the appropriate symbols of the computation. For example, her sensory apparatus must be such as to transduce the signals that depart from Queen1 and Queen2 on the board into the symbols "$queen_1$" and "$queen_2$" respectively. And when the physical object that implements the computational description of her activity, tokens the string of symbols: "Perform swap: $(queen_1, queen_2)$", her physical output must be such as to result in the physical swapping of Queen1 and Queen2 on the real board. It doesn't really matter how the physical object manages to do this.

Is it a neural network that minimizes its energy function? Fine! What counts is that anything else (even if it is not a neural network like that), would do just as well, provided that the relevant mappings (the ones that ensure that the physical system implements Sosic and Gu's algorithm) are realized. So, according to the orthodox internalist view, the connectionist model is at best an implementation of the algorithm: it might explain how Katherine's nervous system manages to solve the problem, but it doesn't tell us what it means, in general, for a cognitive system, to solve the problem the way she does.

The orthodox connectionist would contest that if, as he thinks, Katherine is implementing the connectionist model, then she cannot be also implementing Sosic and Gu's algorithm. For example, he argues, there is nothing in the description of the workings of the network that corresponds, explicitly, to the rule: "*two columns should be swapped if and only if the result of doing so reduces the total number of collisions*". All the network ever does is minimize the energy function. The activity of the nodes, moreover, need not be interpreted as representing the presence of a queen on a cell, for the input to an analogous, more complex, but equally successful network, could well be distributed over several nodes, with effects on the overall activity of the system that could not be systematically interpreted (mapped) as referring to operations on configurations of queens on the board.

Who is right, the computationalist or the connectionist? A standard connectionist attitude is to resort to empirical evidence on the performance of real cognitive systems in completing the task. For example, our connectionist could inflict some damages to Katherine's brain and observe her subsequent performance at the task. He would claim that, as (say) Katherine now succeeds in finding a solution some 80% of the times, rather then 99% of the times (as she did before), this entails that she could not possibly be implementing Sosic and Gu's algorithm. If she were, in fact, any damage to the system would totally disrupt her capacity to complete the task by complying with the rules prescribed. So, her score should either be unchanged (if the damage did not involve any relevant implementing parts), or suddenly reduced to 0.

The computationalist would reply that he can patch things up by adding features to his computational model that account for the graceful degradation of Katherine's cognitive capacities. But the connectionist, he would continue, is unable to explain why Katherine can combine together, for example, the same symbols that represent Queen1 and Queen2, to produce two different representational complexes, corresponding, say, to two different actions like: "Perform replace: $(queen_1, queen_2)$" and "Perform replace: $(queen_2, queen_1)$", which correspond to the actions of replacing Queen1 (respectively Queen2) with Queen2 (Queen1). Katherine's cognitive system, that is, shows compositional capacities that the connectionist cannot account for. The debate would continue along these lines...

Contrast the above imaginary clash of paradigms with the analysis that is afforded by my externalist treatment of implementation.

1. The mere fact that we can individuate, inside Katherine's brain, a causal structure that can be mirrored (mapped) onto a Turing machine that (virtually) implements Sosic and Gu's algorithm, does not speak in favor or against either theories. If we accept the criticism discussed throughout this work, in fact, one can always find such a causal arrangement, inside any physical system, and given any algorithm.

2. In order to argue conclusively that Katherine is implementing the algorithm,

and that the connectionist model represents, at best, an implementation of it, it must be proved that: (a) Katherine's brain, in those circumstances (i.e. when she is solving the problem), does indeed have states that represent (in the sense specified by some physicalistic theory of intentionality) states of affair that comply with the intended interpretation of the input architecture of the putative Turing machine; and (b) it must be further proved that the transformations of these physical states that are allowed to sustain the relevant computational abstractions are such as to preserve these intentional properties (at least throughout the completion of the task). This amounts to the requirement that the conditions set forth at the end of the last chapter for the matching of the equivalences obtain.

3. The judgement on whether such intentional states are instantiated by Katherine's brain or not should be based on our preferred physicalistic theory of intentionality, and not on our independent judgement as to what cognitive architecture she is implementing. The cognitive architecture implemented supervenes, in part, on the intentional properties instantiated, and not viceversa.

4. If such intentional states are not instantiated, or if they do not relate to each other in compliance with the algorithm, then the computationalist is wrong, and the connectionist model is a genuine alternative.

5. It might well be that the damage inflicted to Katherine's brain has no effect on her behavioral capacities, but has the effect of disrupting her intentional properties. So, it is well possible that: (a) we can observe no difference in her ability and ways to complete the task, i.e. she still does it 99% of the times, seemingly complying, relative to our representational capacities, with Sosic and Gu's algorithm; but, nevertheless, (b) now she is merely instantiating the connectionist model, even though before being

damaged she was (really) implementing the algorithm.

6.8 Potential future developments of computational externalism

The debate over cognitive architecture has so far concentrated on explaining/predicting cognitive behavior and, to this end, on the capacity to build artificial systems that simulate it, relative to our representational capacities.

If my treatment proves to be correct, more attention should be payed, instead, to systems that are autonomously endowed with representational capacities, along (and improving on-) the lines suggested by the now enormous literature about the naturalization of intentionality. The capacity that our artificial devices should attempt to simulate, I argue, is the capacity to manipulate the environment in such a way as to systematically preserve the equivalence relations that sustain the implementation of a given computation.

My treatment of implementation, I believe, has three main concrete consequences that need to be further explored. For one, the implicit internalist assumption about syntactic properties, might have thrown scientific investigation off the track, unduly excluding from its scrutiny all potentially relevant syntactic generalizations that could not be seen as being instantiated by a physical system alone.

Secondly, the unnecessary division of conceptual labor between syntactic and semantic machines, that comes with the orthodox internalist construal of computation, put, in many cases, an unsatisfiable explanatory burden on both of them, at the expense of the credibility of the sufficiency hypothesis.

Finally, the syntactic/semantic division of labor contributed to the maintenance of a correspondent isolation of the mathematical tools that are used to describe them. The theory of computation, and the theory of dynamical systems, as a consequence, have not so far perspicuously interacted with each other.

A prospect for the future of my analysis of computation is the possibility to develop a mathematical theory of the emergence of adaptive syntactic properties out of the interaction of dynamical systems. A condition for this to happen is the possibility that our philosophical understanding of intentional properties develop into a full blown empirical science. I believe it has the resources for doing so. Although I have not tackled directly these potential developments, I hope my theory of implementation will help to lift the explanatory burden that has so far prevented us from foreseeing such an empirical investigation.

An intuitive upshot of my treatment is that if we wish to uphold the thesis that thought and cognition are coextensive with the implementation of certain computations, then we must investigate (a) how nature instantiates cross-semantic properties[53] and (b) how it manages to operate transformations in the domain of these representations that can be systematically subsumed under syntactic generalizations.

[53]Recall that I have called a property (or a set of properties) *cross-semantical*, if it instantiates the truth conditions of a relational statement whose terms refer to entities that belong to different semantic levels (i.e. one term refers to an entity that belongs to the domain of representations, and another refers to an entity that belongs to the correspondent represented domain).

Conclusions

The main argument that has been put forward in this thesis is the following. For the sake of clarity, I have omitted to mention explicitly when a claim depends on the assumption that vacuousness arguments are sound. Each claim that so depends on the assumption is indicated by an asterisk on top of its correspondent number. I also presupposed a physicalistic metaphysics, according to which a property is *real* if an only if it can be described (at least in principle) by reference to physical properties only[54].

1* No amount of information on the intrinsic properties of a physical system suffices to ascribe to it computational properties in an objective (non observer-relative) way. It follows that:

2* Either computational properties are not real, or they supervene on extrinsic properties of the implementing system.

3 Either (a) these extrinsic properties are essentially (hence always) sufficient to instantiate semantic properties, or (b) there are extrinsic properties that suffice for instantiating computational ones but do not suffice for instantiating semantic ones.

[54]The expression "physical property" should be understood openly as coextensive with "any property that features in some explanation provided by the physical sciences".

4* If (b) is the case, then vacuousness arguments can be applied to show that the candidate extrinsic implementing properties are insufficient to rule out unwanted models. It follows that:

5* If computational properties are real, then they are partly grounded on semantic properties.

6 From 2 and 5 it follows that the semantic properties on which computational ones are grounded must be instantiated by extrinsic properties of the implementing system.

7 The only class of remotely specific externalist theories of content are Teleological Theories of content.

8* We can conclude that, to our best knowledge, if computational properties are real, a necessary condition for their implementation is that the candidate implementing system instantiate teleological intentional properties.

9* A necessary and sufficient condition for implementation is that the candidate system instantiate teleological representations whose transformations can be systematically subsumed under true syntactic generalizations.

Bibliography

[1] J. Shaw A. Newell and A. Simon, *Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics*, Computers and Thought (J. Feldman E. Feigembaum, ed.), New York: McGraw-Hill, 1957.

[2] R. Allen, *Sequential Connectionist Networks for Answering Simple Questions about a Microworld*, Proceedings of the Tenth Annual Conference of the Cognitive Science Society (L. Erlbaum, ed.), Oxford University Press, Hillsday, NJ, 1988.

[3] T. Bays, *On Putnam and his Models*, Journal of Philosophy **48** (2001), 331–350.

[4] W. Bechtel and A. Abrahamsen, *Connectionism and the Mind*, Blackwell, Oxford, UK, 1991.

[5] A. Bianchi, *Naturalizing Semantics and Putnam's Model-theoretic Argument*, Episteme **22** (2002), 1–19.

[6] M. H. Bickhard, *How to Build a Machine with Emergent Representational Content*, CogSci News **4(1)** (1991), 1–8.

[7] ———, *How does the Environment Affect the Person?*, Children's Development within Social Contexts: Metatheory and Theory (J. Valsiner L. T. Winegar, ed.), Erlbaum, 1992, pp. 63–92.

[8] ———, *Emergence*, Downward Causation (P.B. Andersen, ed.), University of Aarhus Press, Aarhus, Denmark, 2000, pp. 322–348.

[9] ______, *The Dynamic Emergence of Representation*, Representation in Mind: New Approaches to Mental Representation (P. Slezak H. Clapin, P. Staines, ed.), Elsevier, 2004, pp. 71–90.

[10] N. Block, *Advertisement for a Semantics for Psychology*, Midwest Studuies in Philosophy **10** (1986), 615–678.

[11] ______, *The Computer Model of the Mind*, Readings in Philosophy and Cognitive Science (A. I. Goldman, ed.), MIT, Cambridge, MA., 1993, pp. 819–831.

[12] N. Block and J. Fodor, *Cognitivism and the Analog/Digital Distinction*, quoted in: O. Shagrir, Minds and Machines (7), 1987, 321–344, 1972.

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

[14] D. Chalmers, *Syntactic Transformations on Distributed Representations*, Connection Science **2** (1990), 53–62.

[15] ______, *On Implementing a Computation*, Minds and Machines **4** (1994), 391–402.

[16] ______, *Does a Rock Implement Every Finite-State Automaton?*, Synthese **108** (1996), 309–33.

[17] M. Christiansen and N. Chater, *Generalization and Connectionist Language Learning*, Mind and Language **9** (1994), 273–87.

[18] A. Clark, *Being There*, Mit Press, Cambridge, MA, 1997.

[19] J. Copeland, *What is Computation?*, Synthese **108** (1996), 335–359.

[20] R. Cummins, *Meaning and Mental Representation*, MIT Press, Cambridge, MA., 1989.

[21] ______, *Representations, Targets, and Attitudes*, MIT Press, Cambridge, MA, 1996.

[22] L. Meeden D. Blank and J. Marshall, *Exploring the Symbolic/subsymbolic Continuum: A Case Study of RAAM*, Closing the gap: Symbolism vs. Connectionism (J. Dinsmore, ed.), L. Erlbaum. Hillsdale, NJ, 1992.

[23] J. McClelland D. Rumelhart and the PDP research group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Mit Press, Cambridge, MA, 1986.

[24] F. Dretske, *Knowledge and the Flow of Information*, Mit Press, Cambridge, Ma, 1981.

[25] ______, *Precis on Knowledge and the Flow of Information*, Behavioral and Brain Sciences **6** (1983), 55–63.

[26] ______, *Misrepresentation*, Belief (R. Bodgan, ed.), Oxford University Press, Oxford, 1986.

[27] ______, *Explaining Baehavior*, Mit Press, Cambridge, Ma, 1991.

[28] ______, *Naturalizing the Mind*, Mit Press, Cambridge, Ma, 1995.

[29] C. Eliasmith, *How Neurons Mean*, Ph.D. thesis, University of Waterloo, 2000.

[30] J. Elman, *Distributed Representations, Simple Recurrent Networks, and Grammatical Structure*, Machine Learning **7** (1990), 195–225.

[31] ______, *Finding Structure in Time*, Cognitive Science **14** (1990), 179–212.

[32] H. Field, *Logic, Meaning, and Conceptual Role*, Journal of Philosophy **74** (1977), 379–409.

[33] J. Fodor, *The Language of Thought*, Thomas Crowell, New York, 1975.

[34] ______, *Psychosemantics*, Bradford Books, MIT Press, Cambridge Ma, 1987.

[35] ______, *A Theory of Content and other Essays*, Bradford Books, MIT Press, Cambridge Ma, 1990.

[36] ______, *A Theory of Content*, Mind and Cognition (G. Lycan, ed.), Blackwell, second ed., 1999, pp. 230–249.

[37] ______, *The Mind doesn't Work that Way*, Bradford Books, MIT Press, Cambridge Ma, 2000.

[38] J. Fodor and E. Lepore, *Holism: a Shopper's Guide*, Blackwell, Oxford, 1992.

[39] J. Fodor and B. McLaughlin, *Connectionism and the Problem of Systematicity: Why Smolensky's Solution doesn't Work*, Cognition **35** (1990), 183–204.

[40] J. Fodor and Z. Pylyshyn, *Connectionism and Cognitive Architecture: A Critical Analysis*, Cognition **28** (1988), 3–71.

[41] M. Giunti, *Computers, Dynamical Systems, Phenomena, and the Mind*, Ph.D. thesis, Indiana University, Department of History & Philosophy of Science, 1992.

[42] N. Goodman, *Languages of Art*, Bobs-Merrill, Indianapolis, 1968.

[43] R. Hadley, *Systematicity in Connectionist Language Learning*, Mind and Language **9** (1994), 247–72.

[44] G. Harman, *Thought*, Princeton University Press, Princeton, 1973.

[45] S. Harnad, *The Symbol Grounding Problem*, Physica **D 42** (1990), 335–346.

[46] W. Haselager and J. van Rappard, *Connectionism, the Frame Problem and Systematicity*, Minds and Machines **8** (1998), 161–79.

[47] J. Haugeland, *Mind Design II*, MIT Press/Bradford Books, Cambridge, MA, 1981.

[48] ———, Mind Design II, MIT Press, Cambridge, MA, second ed., 1997.

[49] T. Hobbes, *Leviathan: Or the Matter, forme and Power of a Commonwealth Ecclesiastical or Civil*, Collier Books, London, 1651/1962.

[50] M. Jordan, *Attractor Dynamics and Parallelism in a Connectionist Sequential Machine*, Proceedings of the Eith Annual Conference of the Cognitive Science Society (L. Erlbaum, ed.), Oxford University Press, Hillsday, NJ, 1986.

[51] G. Leibniz, *Monadology*, Philosophical Essays, Hackett, 1714/1989, transl. Roger Ariew and Daniel Garber, pp. 213–24.

[52] B. Loewer, *From Information to Intentionality*, Synthese **70** (1987), 287–317.

[53] ———, *A Guide to Naturalizing Semantics*, A Companion to the Philosophy of Language (B. Hale and editors Wright, C., eds.), Blackwell, Oxford, Oxford, 1996, pp. 108–126.

[54] W. G. Lycan, *Form, Function, and Feel*, Journal of Philosophy **78** (1981), 24–50.

[55] ———, *Logical Form in Natural Language*, MIT Press, Cambridge, MA, 1984.

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

[57] R. Millikan, *Naturalizing Intentionality*, url: http://www.ucc.uconn.edu/~wwwphil/xxwcong.pdf (Retrieved 12-03-2005).

[58] ———, *Language, Thought and other Biological Categories*, MIT Press, Cambridge, MA, 1984.

[59] ———, *Biosemantics*, Mind and Cognition (W. G. Lycan, ed.), Blackwell, second ed., 1999, pp. 221–230.

[60] ———, *Teleological Theories of Mental Content*, Stanford Encyclopedia of Cognitive Science, Summer 2004 Edition, url: http://plato.stanford.edu/archives/sum2004/entries/content-teleological/.

[61] K. Neander, *Functions as Selected Effects: The Conceptual Analyst's Defense*, Philosophy of Science **58** (1991), 168–184.

[62] ———, *Teleological Theories of Mental Content*, The Stanford Encyclopedia of Philosophy (Edward N. Zalta, ed.), Summer 2004 Edition, url = http://plato.stanford.edu/archives/sum2004/entries/content-teleological/.

[63] A. Newell and A. Simon, *Computer Science as Empirical Enquiry: Symbols and Search*, Communications of the Association for Computing Machinery **19(3)** (1976), 113–126.

[64] M. H. A. Newman, *Mr. Russell's Causal Theory of Perception*, Mind **37** (1928), 137–148.

[65] C. Peacocke, *Content, Computation, and Externalism*, Mind and Language **9** (1994), 303–335.

[66] G. Piccinini, *Computational Modelling vs. Computational Explanation: Is Everything a Turing Machine, and Does it Matter to the Philosophy of Mind?*, Forthcoming in Australasian Journal of Philosophy, url = http://www.umsl.edu/~piccininig/ (Retrieved 10-09-2006).

[67] ———, *Computations and Computers in the Sciences of Mind and Brain*, Doctoral dissertation, University of Pittsburgh, Pittsburgh, PA, 2003, URL = http://etd.library.pitt.edu/ETD/available/etd-08132003-155121/.

[68] ———, *Computation Without Representation*, PhilSci-Archive (2004).

[69] A. Pitts, Cybernetics: Proceedings of the 7th Macy Conference (H von Foerster, ed.), New York, NY, Macy Foundation, 1951.

[70] J. Pollak, *Recursive Distributed Representation*, Artificial Intelligence **46** (1990), 77–105.

[71] H. Putnam, *Meaning and the Moral Sciences*, Routledge and Kegan Paul, London, 1978.

[72] ———, *Models and Reality*, Realism and Reason (H von Foerster, ed.), Cambridge UP, 1983, pp. 1–25.

[73] ———, *Representation and Reality*, MIT Press, Cambridge, MA, 1988.

[74] Z. W. Pylyshyn, *Computation and Cognition*, second ed., MIT/Bradford, Cambridge MA, 1984.

[75] B. Russell, *The Analysis of Matter*, Kegan Paul, Trench, Trubner, London, 1927.

[76] R. Schank and P. Childers, *The Cognitive Computer: On Language, Learning, and Artificial Intelligence*, Addison-Wesley, New York, 1985.

[77] M. Scheutz, *When Physical Systems Realize Functions*, Minds and Machines **9** (1999), no. 2, 161–196.

[78] ———, *Causal vs. Computational Complexity*, Minds and Machines **11** (2001), 534–566.

[79] J. Searle, *A Critique of Cognitive Reason*, Readings in Philosophy and Cognitive Science (A. I. Goldman, ed.), MIT, Cambridge, MA., 1993, pp. 833–847.

[80] ———, *The Chinese Room*, The MIT Encyclopedia of the Cognitive Sciences (R.A. Wilson and F. Keil, eds.), MIT Press, Cambridge, MA, 1999.

[81] J. R. Searle, *Is the Brain a Digital Computer?*, Proceedings and Addresses of the American Philosophical Association **64** (1990), 21–37.

[82] ______, *The Rediscovery of Mind*, MIT Press, Cambridge, Massachusetts, 1992.

[83] L. Shastri and V. Ajjanagadde, *From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables and Dynamic Bindings Using Temporal Synchrony*, Behavioral and Brain Sciences **16** (1993), 417–94.

[84] B. Smith, *On the Origin of Objects*, MIT Press, Cambridge, MA., 1996.

[85] P. Smolenky, *Tensor Product Variable Binding and the Representation of Symbolic Structures in Conncetionist Systems*, Artificial Intelligence **46** (1990), 159–216.

[86] P. Smolensky, *On the Proper Treatment of Connectionism*, Readings in Philosophy and Cognitive Science (A. I. Goldman, ed.), MIT, Cambridge, MA., 1993, pp. 770–799.

[87] D. Stampe, *Toward a Causal Theory of Linguistic Representation*, Midwest Studies in Philosophy: Studies in the Philosophy of Language (Jr. P. A. French, T. E. Uehling and H. K. Wettstein, eds.), vol. 2, University of Minnesota Press, Minneapolis, 1977, pp. 81–102.

[88] P. Stich, *Narrow Content Meets Fat Syntax*, Mind and Cognition (W. G. Lycan, ed.), Blackwell, second ed., 1999, pp. 221–230.

[89] E. Trizio, *Reflexions Husserliennes sur la Mathmatisation de la Nature*, Nature, sciences, philosophies, ENS-Editions, Paris, (forthcoming).

[90] A. Turing, *On Computable Numbers, with an Application to the Entscheidungsproblem*, Proceedings of the London Mathematical Society **45(2)** (1936), 230–65.

[91] T. van Gelder, *Compositionality: A Connectionist Variation on a Classical Theme*, Cognitive Science **14** (1990), 355–84.

[92] T. van Gelder and R. Port, *It's about time: An overview of the Dynamical Approach to Cognition*, Mind as Motion: Explorations in the dynamics of cognition (R. Port and MA T. van Gelder, Cambridge, eds.), MIT Press, 1995.

[93] L. Vygotsky, *Thought and Language*, Mit Press, Cambridge, MA, 1962.

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

[95] ———, *Rethinking Cognitive Computation: Turing and the Science of the Mind*, Palmgrave Macmillan, Houndmills, Basingstoke, Hampshire, 2006.

[96] R. Wilson, *Boundaries of the Mind: The Individual in the Fragile Sciences*, Cambridge University Press, Cambridge, 2004.

[97] L. Wright, *Functions*, The Philosophical Review **82** (1973), 139–168.