Explaining computation without semantics: keeping it simple

Nir Fresco
School of History and Philosophy, The University of New South Wales, Sydney, Australia
Fresco.Nir@gmail.com

Abstract. This paper deals with the question how computation is best individuated.

1. The semantic view of computation: computation is best individuated by its semantic properties. On this view, there's no computation without representation, because computation is individuated in the same way that mental states are.
2. The causal view of computation: computation is best individuated by its causal properties. On this view, the relevant formal structure of computation is mirrored by the causal structure of its implementation without appealing to any semantic properties.
3. The functional view of computation: computation is best individuated by its functional properties. On this view, functional properties are insensitive to content and needn't presuppose semantics.

Some scientific theories explain the capacities of brains and computers by appealing to the computations they perform. The reason for that is usually that computation is individuated by its semantic properties. I criticize the reasons in support of this view and its presupposition of representation and semantics. Furthermore, I argue that any representational view of computation (e.g. the semantic view) can be no more than first order representation (i.e. limited to the availability of representational states to first order cognitive processes, like belief-forming, practical reasoning, etc.).

Introduction

Some psychologists, neuroscientist and cognitive scientists share the view that minds can be explained computationally. The underlying assumption of their view is that computation can't be explained without appealing to external semantics[1] and representation. The immediate outcome of this approach is that theories of computation and theories of mind become closely coupled. The purpose of this paper is to criticize the semantic view of computation, which presupposes representation and external semantics, and to demonstrate that any representational view of computation can be no more than first order representation of computation.

It is worthwhile noting a concise caveat concerning the distinction between internal and external semantics in the present context. While internal semantics may well be relevant for theories of computation, I maintain that external semantics is not. Internal semantics in computing mechanisms doesn't require an independent knower, but external semantics does[2]. An output of a complex computation only makes sense when an independent knower interprets the result as representing a particular state of affairs. In a similar manner the computer processes certain information while being completely unaware of the fact that this information expresses a complex system of beliefs.

Internal semantics however remains intrinsic to the process of computation itself and to the platform that enables the computation process (i.e. the computer's hardware). A computing function stores its output in a particular segment in the computing mechanism’s memory. A subsequent function then refers to this particular memory address to retrieve the content, which was previously stored there. In a similar manner, a particular internal computational state has an internal semantics in the context of a given computation. A computational process, which proceeds through a particular state, may yield a different result going through another state. For reasons of simplicity whenever the term 'semantics' is used, it used to indicate *external semantics*.

The three predominant views on how computation is best individuated[3] are the semantic view of computation, the causal view of computation and the functional view of computation. In contrast to the semantic view of computation, proponents of the other two views don't presuppose representation to be essential for individuating computation. As mentioned above, I maintain that representational theories of computation are no more than *first order representations*. But this obviously requires some clarification.

First Order Representation theories

The fundamental idea underlying first order representational theories (FOR theories in short) is that a system represents a certain property iff that system has the function of providing information about the property of a certain domain of objects. Thoughts, beliefs and judgements are understood as internal representations of external states of affairs. On this view, all *mental facts* are *representational* facts, facts about what the system is designed to do and what information it's supposed to carry. Understanding of the manipulation and use of representations is essential to the understanding of the mind. The FOR theories' approach establishes a framework within which cognition can supposedly be studied objectively (Carruthers 2000: p. 114; Dretske 1988: pp. 80-81, 1995: pp. xiii-xvi, 1-3).

FOR proponents view conscious experience as the output of the various perceptual systems. This output is presented as input to the central cognitive systems responsible for generating beliefs, problems solving capacities, practical reasoning etc. Representational perceptual states have a functional role in the sense of being available to the subject's first order cognitive processes (e.g. belief-forming processes, thinking, practical reasoning, etc.) in a way that guides behaviour. On this view, all that one needs for being capable of phenomenally conscious experience is the cognitive capacity to form beliefs and desires, and the availability of representational states with the right content for the control of behaviour (Carruthers 2000: pp. 24, 114-15, 123; Dretske 1988: pp. 79-107, 1995: pp. 128-134).

The semantic view of computation

This shared view can be best summarized in Fodor's words "… computation is a causal chain of computer states and the links in the chain are operations on semantically interpreted formulas in a machine code… There is no computation without representation." (Fodor 1981: p. 122). Proponents of this view attribute semantic properties to *internal* computations that go on inside an artefact that is a computing mechanism (Fodor 1981; Pylyshyn 1984; Pylyshyn 1989;

Shagrir 2001; Shagrir 2006; Smith 1996; Smith 2002). Smith (1996: pp. 9-11; 2002: pp. 32, 56) sees computation as a symbolic or representation based phenomenon. In his view, the states of a computer can model or *represent* other states in the world. Computation is an intentional phenomenon and that is partly why so many intentional words in English are used as technical terms in computer science (e.g. representation, data etc.). This is analogous to physics use of 'energy', 'force' and the likes that objectively represent states of affairs in the world. So not only does a computer program represent 'something' for independent knowers, but it can represent states of affairs, facts or whatnot in their absence too.

Shagrir (2001: pp. 370, 375; 2006) maintains that content plays an individuative role in computation by way of determining which of the *various syntactic structures* implemented by the computing mechanism constitutes its *computational structure* (i.e. the one that defines the underlying computational task). This is the source of the explanatory force of applying computational explanations to account for the way complex systems carry out semantic tasks. The crucial point here is identifying the correspondence relation between a computational structure of a system and certain logical properties of the states and objects that are being *represented*. So semantics plays an individuative role here in terms of what is being represented. And these are the objects, properties and relations assigned by interpretation to the computational states of the computing mechanism. Different computing mechanisms can implement different syntactic structures simultaneously. Thus, in order to determine which syntactic structure *constitutes* the mechanism's computational structure a *semantic constraint* has to be applied.

Fodor and Pylyshyn (Fodor 1975: p. 73; Fodor 1998: pp. 10-11; Pylyshyn 1989: pp. 9, 11, 55-57) add a new factor into the 'semantic computation' equation: the truth-preserving rules. The symbolic representations are manipulated, while the meaning of the symbols processed is coherently maintained by the *transformation rules*. These rules are meant to ensure that the generated symbolic expressions continue to make sense when semantically interpreted in a consistent way. The semantic properties are preserved in a similar manner to truth preserving in logic, but if one wishes to ensure that the algorithm always gives the correct answer, one *has to* refer to facts and states of affairs, i.e. to the *semantics of the symbols*.

The semantic view of computation is essentially the view that computation has content, and that resorting to computational explanation entails content ascription to the explanandum (Piccinini 2007). Proponents of the semantic view of computation are hoping to ascribe content to computational mechanisms as a way of explaining minds computationally. The motivation is clear, if computation is semantically individuated, then minds by possessing mental contents can be explained computationally. But then the whole endeavour is weakened on pain of circularity. The common assumption is that computational states are best individuated in the same way that mental states are, i.e. by their contents. However, this simply begs the question whether mental states ought to be individuated in the same way that computational states are.

The causal view of computation

Proponents of the causal view of computation offer an alternative analysis of computation according to which computation is best individuated by its causal properties, rather than by any semantic properties (Chalmers 1994; Copeland 1996; Scheutz 1999). Chalmers (1994: pp. 391-393) asserts that the crucial bridge between abstract computations and physical computing mechanisms is a theory of implementation. Thus, implementation is the relation that holds between the abstract computation and the physical mechanism that executes this computation. The formal analysis provided by Chalmers is as follows. A computing mechanism implements a given computation when there exists a group of physical states (of the mechanism), which is mapped by a one-to-one relation into the abstract computational states. The formal computational states are mapped into physical state-types related to each other by a corresponding causal state-transition relation. The formal state-transitional structure of the computation mirrors the causal state-transitional structure of the physical computing mechanism. Hence, according to Chalmers' (ibid: p. 401) account of computation it is individuated by its causal organization: a computational description of a computing mechanism is an abstract specification of its causal organization.

Copeland (1996: pp. 337-338, 350-353) offers a somewhat different account, which is based on an extended notion of causal interpretation of computation. A mechanism M may be deemed to be computing a function F iff there exists a labelling scheme L and a formal specification SPEC of an architecture and an algorithm specific to that architecture such that (M, L) is an honest model of SPEC[4]. The labelling scheme is the means of bridging the gap between the SPEC, which is the computing mechanism's formal description, and the computing mechanism itself. The labels constitute a 'code' such that sequences of labels have semantical interpretations. Copeland himself says that an analysis of computation in terms of causation only is "intolerably narrow" (ibid: p. 353). In order to overcome this difficulty, he suggests that the computational states are chosen for the labelling scheme in such a way that the associated labels are causally related in the required ways, so that output labels are a causal outcome of the input labels and the appropriate labels of the computing mechanism's structure.

Scheutz (1999, 2001) claims that the standard concept of correspondence between physical states and computational states is inadequate and does not give a clear account of how computations mirror the causal structure of physical systems. Instead he suggests that: "the state transitional structure of the computation must be at most as complex as the causal transitional structure of the implementing physical system with respect to computational sequences" (2001: p. 560).

The functional view of computation

Another alternative to the semantic view of computation is in a form of a functional view of computation (Piccinini 2007; Popper 2006; Stich 1983). Popper (2006: pp. 79-81) held a functional view that borders with the causal view of computation. He maintained that the individuating properties of a computing mechanism aren't just physical. Thus, two computers, which may differ physically, may still both operate according to the same "standards of logic". These abstract properties make the computer operate in conformance to logical standards and have physical effects.

Stich (1983: pp. 149-183) while building his argument against the possibility of a scientific intentional psychology implicitly rejects the semantic view of computation. His main claim with regard to folk psychology is that we needn't refer to the content of cognitive states. Cognitive states can be explained in terms of syntactic properties and relations of the abstract objects to which the cognitive states are mapped. A consequence of his syntactic theory of mind is that the same applies to computational states too. The processes governing computational processes are sensitive to the syntax of the symbolic structures involved in these computations. Explanation of computation needn't appeal to any semantic properties (ibid: pp. 192-193). There is no role for contents in theories of computation and no need to presuppose any semantics.

Piccinini (2004, 2006, 2007) claims that an appropriate kind of functional explanation is sufficient to individuate computation without appealing to semantic properties. On his view, computational states are individuated by their functional properties. These functional properties are specified by a *mechanistic* explanation in a way that doesn't presuppose any semantic properties. In his view of a functionally – mechanistic explanation[5], a mechanism has certain components, which have certain *functions* (e.g. the heart's function to pump blood) and are *organized* in a certain way. And the mechanism's capacities are constituted by the relevant components, their functions and their particular organization. The functional properties that are relevant to computational individuation are the computers' particular components (e.g. memories, processors, etc.) as well as the interrelations between them (e.g. signalling), the states of those components and the functions of these components.

Discussion

Cognitive science is based by and large on the premise that minds and computers have a great deal in common. Proponents of the semantic view of computation seem to think that semantics is the necessary platform to bridge the apparent gap between computers and minds. They believe that the capacities of brains can be thus explained computationally. They assert that individuating computation semantically makes it possible.

The main reasons in favour of the semantic view of computation can be summarized in the form of the following two arguments[6]. The first argument: 'computational identity' is based on three premises:

a. Computation is individuated by the functions being computed.
b. Functions are defined by syntactic structures generating output *F* (*I*) for input *I*.
c. But, the underlying function being computed *in a given context* is (at least partly) semantically individuated.
d. Conclusion: Computation is (at least partly) individuated semantically (Dietrich 1989: pp. 119, 126-128, 131; Shagrir 1999: pp. 137-143; Smith 1996: pp. 6-9).

There's a prima facie conflict emerging from accepting the first and the second premises: a different conclusion follows: computation is individuated syntactically (or functionally). The problem is then: which conclusion should we adopt? There's an obvious tension between the first and the third premise. If we choose to discard the third premise, then the conclusion d doesn't follow. As a result, we get two different answers to the question how computation is best individuated: the first being syntactically, and the second answer is semantically. The tension between the first and the third premises can be resolved in the following way.

A computing mechanism may implement more than one function simultaneously. This claim, combined with the second premise, entails that computation is individuated syntactically. But the computational identity of a computing mechanism is context dependent (Shagrir 1999: p. 142). The dependency is on *a specific* function being computed (i.e. the underlying function of premise c). This introduces the need for a semantic constraint, which individuates computation relative to that given context. Since other syntactic structures may be extracted, an *external* constraint is needed to determine which of these structures define the *computational identity* of a computing mechanism (ibid: p. 143). This constraint supposedly must have something to do with the contents of the computation being executed.

Be that as it may, why should an underlying function be semantically individuated? As

Egan (1995: pp. 185-187) and Piccinini (2006) explain there's no necessity that the underlying function should be individuated semantically. The functions relevant to computation are defined logically, and generally, formal specifications of functions facilitate the articulation of the logical function into an algorithm, which in turn can be implemented in any given programming language[7].

Computations operate on data structures and in the process of computation those data structures are indirectly transformed by means of underlying physical processes (MacLennan 1994: p. 434). Programs written in high level programming languages (like C++ or Java) are only abstraction layers, which are eventually compiled on any given computing mechanism into the relevant machine code. And the machine code instructions are no more than binary operations running at the machine level. Semantics only enters the picture when the knower interprets these logical operations as representing physical quantities, external states of affairs, objects etc.

The second argument: 'identity of mental states' is based on two basic premises:

a. Cognitive science resorts to computational explanations in trying to explain the nature of mental states (and cognitive processes).
b. Mental states are individuated semantically.
c. Conclusion: computation should be individuated semantically. (Burge 1986: pp. 28-29; Fodor 1975: p. 27; Pylyshyn 1984: pp. 57-58, 1989: pp. 55-57)

The main flaw with this argument is that a third premise has to be assumed so that the conclusion logically follows from the preceding premises (Egan 1995: p. 183; Piccinini 2006). A sufficient third premise could be that the explanans has to be individuated in the same way that the explanadum is. So in order that a computational explanatory framework is successful in explaining mental states (and cognitive processes) *computation* has to be individuated semantically too. But this simply assumes that computational explanation is the appropriate framework for explaining mental states and cognition to start with[8]. Another line of argument is thus called for.

Furthermore, in my opinion any representational view of computation can be no more than first order representation of computation. Computing mechanisms are artefacts, which get their meaning, purpose and intentionality from us. Their information providing function originates in human agents as their designers, makers and users. We 'play the role' of natural selection in selecting the appropriate internal process to produce the necessary external change. (The exception to this would be biological computers, which may someday evolve by natural selection[9]. Far-fetched as it sounds!) The reason that a computing mechanism operates in the way it does (i.e. its behaviour) is determined by its design and the way we program it. A computing mechanism's triggered operation would be in response to an internal condition satisfied in some way (Dretske 1988: pp. 48-50, 1995: pp. 6-8).

In the case of human agents, thoughts, beliefs and judgements are understood as internal representations of external states of affairs. An appropriate belief may be formed in response to an internal representational state (e.g. the belief that glasses break when falling off tables formed as a result of the perceptual state of a shattered glass on the floor). A computational outcome of an internal 'representational' state in computing mechanisms, in contrast, can always be logically predicted[10]. An internal 'representational' state of a certain memory segment may be available to another consumer function, which retrieves the stored contents used for a particular computation, in a given context. However, as I said above, an independent knower is required to interpret any computational state as *representing* external states of affairs.

Representational states of computing mechanisms are not available to any computational processes in a manner that is analogous to human agents. Arguably the availability of representational states to first order cognitive processes has an evolutionary advantage in *guiding behaviour*. There is no advantage in the case of computing mechanisms. As long as input can be fed into computations performed and appropriate devices are available to generate output, no further interaction with the surrounding environment is necessary to properly compute[11].

As Dretske (1988: pp. 81-82) explains computational models of mental processes are sometimes overrated with regards to their explanatory power. Computing mechanisms manipulate symbols, and the dispute is whether those symbols have intrinsic meanings, i.e. semantics. If we agree that those symbols have meanings and that computations preserve the semantic relations while manipulating symbols (Fodor 1998: pp. 10-11; Pylyshyn 1989: pp. 55-57), then we can conclude that computing mechanisms may be capable of first order representations. Not only are computations individuated semantically, but computing mechanisms may also have dispositions for thoughts, beliefs etc. But, this is merely an illusion! (Dretske 1995: p. 82) The reason for questions being prompted by computing mechanisms in response to certain inputs (e.g. did you mean flying cargo?) isn't that they inferred what the user meant[12]. It is rather that the semantic characterization of internal computations may be predictively useful, due to rational design and *meaning attribution by human agents*.

The causal view and the functional view of computation overlap in some respects (namely by appealing to causal properties to explain what computation is). Proponents of the latter view maintain that appealing to causal properties only is insufficient for the individuation of computation. To properly individuate computations further constraints are needed to determine the relevant causal properties. Everything has causal properties, so in a way everything may be included by causal criteria of individuation. The additional constraints are best defined in functional terms. (Piccinini, personal correspondence). Regardless of whether computation is best individuated by causal properties or functional properties, both the respective views are to be preferred over the semantic view of computation, since they don't presuppose representations or semantics.

It may very well be the case that a particular implementation of a given computation may be attributed some sort of semantic content. But this alone is not a good enough reason to *semantically individuate* computation. Proponents of representational views of computation can claim no more than low-level computational representations. For any prospects of establishing a solid framework to explain minds computationally, semantics can't be presupposed.

References:

Bedau, M. (2003). Artificial Life. (In Floridi, L. (Ed.) Blackwell Guide to the Philosophy of Information and Computing. pp. 197-211). Malden, MA: Blackwell Publishing

Carruthers, P. (2000). Phenomenal Consciousness: A naturalistic theory. Cambridge: University Press.

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

Copeland, B. J. (1996). What is computation? Synthese, 108, 335-359.

Dietrich, E. (1989). Semantics and the computational paradigm in cognitive psychology. Synthese, 79, 119-141.

Dretske, F. (1995). Naturalizing the mind. Bradford: The MIT press.

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

Fodor, J. A. (1975). The Language of Thought. Cambridge, MA: Harvard University Press.

Fodor, J. A. (1981). The mind-body problem. Scientific American, 244, 114-123.

Fodor, J. A. (1998). Concepts. Oxford, Clarendon Press.

MacLennan, B, (1994). Words Lie in Our Way. Minds and Machines, 4, 421-437.

Piccinini, G. (2004). Functionalism, Computationalism, and Mental Contents. Canadian Journal of Philosophy, 34, 375-410.

Piccinini, G. (2006). Computation without representation. Philosophical Studies.

Piccinini, G. (2007). Computing Mechanisms. Philosophy of Science. (forthcoming)

Popper, K. R., Eccles, J. C. (2006). The Self and Its Brain: An Argument for Interactionism. Routledge.

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

Scheutz, M. (2001). Computational versus Causal Complexity. Minds and Machines, 11, 543-566.

Shagrir, O. (1999). What is computer science about? The Monist, 82, 131-149.

Shagrir, O. (2001). Content, Computation and Externalism. Mind, 110, 369-400

Shagrir, O. (2006). Why we view the brain as a computer. Synthese, 153, 393-416

Smith, B. C. (1996). On the Origin of Objects. Cambridge: The MIT Press.

Smith, B. C. (2002). The foundations of computing. (In Scheutz, M. (Ed.) Computationalism: new directions. pp. 23-58). Cambridge: The MIT Press.

Pylyshyn, Z. W. (1984). Computation and Cognition. Cambridge: The MIT Press.

Pylyshyn, Z. W. (1989). Computing in cognitive science. (In Posner, M. (Ed.) Foundations of Cognitive Science. pp. 49-92). Cambridge: MIT Press.

[1] The distinction between external semantics and internal semantics has to be explicit in the context of theories of computation. An internal semantics in computing systems is assigned by the system's referring to its own internal sub components and processes. e.g. a function's output data is stored into the computer's memory to be later used as input for another function. In this case, internal semantics is assigned by storing the internal contents of different functions in the system. External semantics on the other hand implies that the computing system assigns contents, which is external to it, i.e. in its environment. These external contents can be events in the environment, properties, objects etc.

[2] My supervisor suggested avoiding the use of the term external in this context. An 'independent' knower here escapes the effects of discouraging objections. One might argue for instance that a knower can be engaged in a form of virtual reality, in which case it isn't clear cut whether he is in fact *external* to the computing mechanism.

[3] In the context of this paper, 'individuated' may be interpreted as 'distinguished'. Thus, the main question can be rephrased to read: "how is computation best distinguished from non-computation phenomena?". I don't explicitly discuss the criteria for distinguishing computation from non-computation in this paper. For a discussion about the criteria, which an adequate theory of computation has to meet, see my previous paper "On the need to better understand our computers" (presented in the AAP conference 2007).

[4] Copeland (1996: pp. 350-351) defines two necessary conditions for the honesty feature. First, the labelling scheme L must not be ex post facto. Second, the interpretation associated with the model must secure the truth of appropriate counterfactuals concerning the machine's behaviour.

[5] It may be worthwhile noting that Piccinini's mechanistic account of computation doesn't reduce computing mechanisms to nothing more than its sub-components. It is rather meant to support an account that appeals to the mechanism's sub-components, their functions and the particular way that they are organized together.

[6] The 'multiplicity of computation' argument, which was put forward by Shagrir (2001), is omitted due to space limitation. The argument can be crudely summarized as follows:

a. Any given computing mechanism may simultaneously implement different computations.
b. In any given context, the computation performed by a computing mechanism is determined by a single syntactic structure, which is the underlying task
c. The underlying task is individuated semantically
d. Conclusion: computation is individuated semantically relative to one particular task that the computing mechanism performs

Shagrir is obviously right about the multiplicity of computations that can be performed by any given computing mechanism. He also accurately points out that these computations are determined by syntactic structures, and thus another constraint needs to be employed to individuate the relevant computation. But the strong claim that this constraint must be semantical is uncalled for. The criticism of the third premise of the 'computational identity' argument offered above applies here too.

[7] A formal specification of a function can also be given as a non empty set of ordered pairs, for instance. These ordered pairs needn't even contain numbers or variables. For example, a particular function may map individuals in a group of people to their matching coloured hats, in such a way that every person is mapped to exactly one particular hat. In these cases, the formal specification doesn't facilitate the articulation of a corresponding algorithm.

[8] See Egan (1995: pp. 183-185) and Piccinini (2006) for a complete discussion of the weakness of this argument.

[9] In some ways biological computers are analogous to connectionist systems. They are both distributed systems by nature, in which a population of autonomous agents follows simple local rules. Connectionists systems operate by employing learning algorithms, but receive their information preprogrammed by a human designer and generate output, which requires human interpretation. However, in the case of biological computers they get their input from the environment in which it is situated. Accordingly, their output is dependent on an open-ended evolutionary dynamic process. (Bedau 2003: pp. 198-199). Nevertheless, this is yet to be proven. This research is still in its infancy stages, and only time will tell if researchers succeed in harnessing natural selection to take its course on biological computers. And the harnessing of natural selection to modify computing mechanisms is where I think it departs from computation proper. This exceeds the scope of the discussion at hand.

[10] This is obviously a strong claim and there are exceptions to this rule. Some edge cases may cause an unexpected outcome, thus resulting in an unpredicted output or behaviour. But these can be debugged and analysed with the appropriate development and testing tools, which exist for this particular purpose.

[11] Clearly, a power source, for instance, is also required for the operation of the computing mechanism, but this doesn't pose any significant environmental constraint.

[12] Anyone, who uses the famous Google search engine, may be prompted by Google to 'respond' to a question of some form (such as 'did you mean flying cargo?'). This doesn't entail that the underlying computation that occurred behind the scenes demonstrated an *understanding* per se of the searched term. It is rather a process built into the search engine that allows it to 'guess' what the user might have meant, when little or no 'hits' were found.