On the need to better understand our computers...

Nir Fresco

School of Philosophy, The University of New South Wales, Sydney

This discussion deals with the question: What are the criteria that an adequate theory of computation has to meet?

1. Smith's answer: an adequate theory of computation has to meet the empirical criterion – it has to do justice to computational practice, the conceptual criterion – it has to explain all the underlying concepts and the cognitive criterion – it has to provide solid grounds for computationalism.

2. Fodor & Pylyshyn's answer: an adequate theory of computation has to meet the semantic level criterion – it has to explain the semantics of computation, the symbol level criterion – it has to explain the information processing aspect and the physical level criterion – it has to explain the underlying physical realization.

3. Piccinini's answer: an adequate theory of computation has to meet the objectivity criterion – it has to identify computation as a matter of fact, the explanation criterion – it has to explain the computer's behaviour, the right things compute criterion, the miscomputation criterion – it has to account for malfunctions, the taxonomy criterion – it has to distinguish between different classes of computers and the empirical criterion.

4. Von Neumann's answer: an adequate theory of computation has to meet the precision and reliability of computers criterion, the single error criterion – it has to address the impacts of errors to computation and the distinction between analogue & digital computers criterion.

5. "Everything" computes answer: an adequate theory of computation has to meet the implementation theory criterion – it has to properly explain the notion of implementation.

There's a widespread tendency to compare minds to computers, but a better understanding of computation is required beforehand. I outline some of the competing answers and argue that Smith's criteria are inadequate and over demanding. My aim is to show why he's eventually concluded that an adequate theory of computation is unlikely.

1. Smith's answer

According to this answer an adequate theory of computation has to meet the empirical criterion, the conceptual criterion and the cognitive criterion (Smith 1996, pp. 14-17; 2002). In his view, questions like what computers are or what computation is, require tackling other questions of metaphysical nature. He asserts that not only must an adequate account meet the three criteria above and include a theory of semantics, it must also include a theory of ontology. It is not just intentionality that is at stake, in his view, but so is metaphysics.

The empirical criterion

The empirical criterion dictates the need to be compliant with extant computational practice. It means that any such a theory should be capable of explaining a program like the Open Office Writer. It should account for its construction, maintenance and everyday use (Smith 1996, p. 5; 2002, p. 24). The empirical criterion "does justice" to computational practice by keeping the analysis grounded in real world examples of computers. The computer and the Internet revolutions demonstrate again and again computers' ability to evolve, expand and adjust beyond the alleged constraints of any computational theory. This criterion serves to question the legitimacy of all extant theoretical perspectives. In this context, *Silicon Valley* is nominated as the gatekeeper to decide whether in practice something may be deemed computational. An adequate theory of computation must make a substantive *empirical* claim about what Smith calls *computation in the wild*, which is the body of practices, techniques, machines, networks etc. that revolutionized the last decades (Smith 1996: pp. 5-6; 2002: pp. 24-25).

The conceptual criterion

The conceptual criterion dictates the need to repay all intellectual debts, in the sense that any such theory ought to clearly explain underlying concepts like: compiler, interpreter, algorithm, semantics etc. With that in mind, we should understand what the theory says, its origins and its implications (Smith 2002: p. 24). The conceptual criterion, which is no more than a meta-theoretical constraint on *any* theory, is especially crucial in the computational case for two main reasons. The first reason is that many candidate theories of computation rely on important notions such as interpretation, representation and semantics with no proper explanation of these notions. The second is that there's a widespread tendency to resort to computation as a possible theory of exactly those very "disobedient" notions. The end result is thus a conceptual circularity that deprives candidate theories of their explanatory power (ibid).

The cognitive criterion

The cognitive criterion dictates the need to provide solid grounds for the computational theory of mind, often known as *computationalism,* the thesis that regulates the traditional fields of artificial intelligence and cognitive science (ibid). The cognitive criterion is also a meta-theoretical constraint on the form of any candidate theory of computation. In the present context, computationalism has potential epistemological consequences depending on the theory of computation one chooses to endorse. If the computational theory of mind were true, then a theory of computation would apply not only to computing in general, but also at the meta level to the *process of theorizing*. In other words the theory's claims about the nature of computation would apply to the theory *itself*. So if computationalism was true, then upon judging a candidate theory of computation and finding it to be adequate or not, there will be supposedly no reason to trust the conclusion. The reason for that is that the presumed meta-theory is conceptually inadequate.

Asking too much and too little

Smith rightly asks that a candidate theory does justice to computational practices or to what he calls *computation in the wild*. Elsewhere he claims that there's a big gap between the theory and the practice, which the theory won't be able to overcome (Smith forthcoming). Smith calls it the explanatory gap. True, any such theory has to account to some extent for the computer practices that have penetrated every aspect of our lives in recent years. Indeed, a candidate theory shouldn't be fully abstract and detached from computation in the wild. The technological gap, which is boosted by the computer and the Internet revolutions, doesn't force us to give up in advance just because the theory is supposedly left far behind. An adequate account needn't capture and explain all the aspects of *every* existing technological breakthrough in computer science, artificial intelligence, molecular computers etc.

Computation is a highly diversified and fluid concept, which may not be fully explained by a strict and definite theory. Instead a more flexible – context based account should be sought. Rather than looking for one essential core to account for all the computational practices and defining a clear boundary, a disjunctive account can be given. A candidate theory can account for Turing machines, desktop computers, the Open Office Writer program and a compiler. This theory can be *limited* to do justice to the basic computing devices only or be *extended* to account for more complex devices such as parallel and distributed computers, high speed network elements, expert systems, molecular computers etc.. This however isn't to say that such a theory would be so loose as to accept *any* device, since this may result in an account that attributes computation to any physical system.

Eventually, Smith (2002, p. 51) makes the strong claim that it's not only that we don't currently have any satisfactory theory of computation, but also that we'll always fail to provide an adequate theory. I believe that this assertion is a bit hasty. The empirical criterion may be indeed hard to meet, but insufficient to dismiss any attempt to provide an adequate theory. There's nothing that prevents us from refuting a particular theory of computation, and providing a new account, which addresses the weaknesses of its predecessor. When an existing theory of computation is falsified, its weakness should be addressed by a new candidate theory.

Smith's cognitive criterion is no more convincing than his empirical criterion. Regardless of the very critical debate regarding the legitimacy of computationalism, its claims needn't dictate any meta level constraints on the characteristics of the theory of computation. If anything it should rather be the other way around. Any proponent of computationalism has to show why minds work the same way that computers do. Smith accurately claims that if computationalism were found to be true, there would be significant epistemological consequences for the process of theorizing. But if we take his view seriously, then computationalism *itself* can't be an adequate theory. To put it simply, according to computationalism, the mind is a kind of computer. Smith (2002, p. 51) argues that we'll always fail to provide an adequate theory of computers. From these 2 premises it follows that proponents of computationalism will always fail to provide an adequate computational theory of the mind.

Smith argues that if computationalism were true, then a theory of computation would apply not only to computing, but also to the process of theorizing. So unless one nails down the reflexive implications of the candidate theory of computation on theorizing itself, and examines this theory from a reflexively consistent standpoint, one will be incapable of judging whether it's adequate. If computationalism were false, then this criterion would become irrelevant anyway. But if it was true, then the cognitive criterion would a-priori pre-empt any attempt to produce a theory of computation. The result would be that a theory, which deals with computers, is reflexive and deals with minds as well. The cognitive process of theorizing itself may thus be said to be computational. The best approach to deal with Smith's argument will be to simply avoid the trap. The burden of addressing the supposedly reflexive characteristic of the theory lies on *computationalism* rather on any *theory of computation*.

It appears to me that Smith is asking *too much and too little* with regard to potential theories of computation. The foregoing criteria constrain any candidate theory and leave very slim chances of providing an adequate theory. At the same time, I believe that he has overlooked other essential criteria, which should be seriously considered. For instance, the miscomputation criterion and what I call the “dichotomy criterion”. The former criterion dictates that any theory of computation has to explain miscomputation as an inevitable feature of computation. I shall elaborate more on this criterion in the answers that follow. The latter criterion dictates that things that compute should be clearly distinguished from things that don't.

It may be argued that the dichotomy criterion is too strong since computation is a graded concept. According to this line of argument, different devices and mechanisms range at different points on the scale. Some paradigmatic examples like UTMs, digital computers, multiple-processing computers etc. clearly perform computations and are located at one extreme end of the scale. Other examples like digestive systems, buckets of water, walls, toasters etc. don't perform computations, and are located at the opposite end of the scale. Whereas in the middle ground one may find mechanisms such as lookup tables, Ethernet cards etc. that aren't always clear cut cases. However, this criterion has to be methodically followed by any candidate theory of computation. Setting the goal high enough regarding what constitutes "performing computation" and what doesn't may achieve better results. A theory of computation, which clarifies a larger number of computing mechanisms, is Ceteris Paribus better than one that accounts for fewer. This approach has a better chance of producing a broader version of a theory of computation.

Smith concentrates so much on "doing justice" to computation in the wild and to computationalism, that he neglects essential meta-theoretical constraints. And these latter constraints are the ones, which are necessary to providing an adequate theory of computation. The following answers can be examined as alternatives to Smith's criteria, and illuminate what he has overlooked.

2. Fodor & Pylyshyn's answer

The view advocated by Fodor (1975) and Pylyshyn (1984; 1989) is applicable to both computers and cognition. According to the second answer an adequate theory of computation has to meet the semantic level criterion, the symbol level criterion and the physical level criterion. These criteria are essentially distinct levels of organization in what Pylyshyn (1989: pp. 58-59) calls "the classical view of computing and cognition". This explanation framework of complex systems can be traced back to the influential analysis of David Marr, who proposed three levels of explanation in his work on a computational theory of vision: the computational, the algorithmic and the physical (Staines et al 2001: pp. 209-212).

The semantic level criterion

At the semantic level we explain why appropriately programmed computers, do certain things by saying what their goals are, and by showing that these are connected in certain meaningful ways. Symbolic expressions are transformed in the computer in a way that is very important. This is because these expressions mean something. Hence, the transformations of these expressions are designed to coherently maintain their meaning and to ensure that the expressions continue to "make sense" when semantically interpreted (1989: pp. 58-60).

The symbol level criterion

The symbol level is required to account for questions like why some tasks take longer to process than others, or why some tasks result in more errors than others. Computers can operate at the semantic level only because of the underlying *symbol level*. At this level the system operates in terms of representations and information processing operations on these representations. The semantic content of "knowledge" and goals is encoded by symbolic expressions. If one wishes to explain why it takes more time to compute answers to some problems, one has to refer to the way “objects” are symbolically encoded and to the sequences of transformation of these symbolic expressions (1989: pp. 59, 63).

The physical level criterion

For the entire system to run, it has to be realized in some physical form. The structure and the principles by which the physical object functions correspond to the *physical level*. Due to our experience with a narrow range of architectures, we are accustomed to associate computation and algorithms with a limited class of algorithms that can be executed on specific architectures. But this leads to an inevitable mistake since different architectures permit different algorithms to be executed (1989: pp. 59, 63).

3. Piccinini's answer

Piccinini (forthcoming) offers an account of computation without representation. In contrast to the semantic view of computation advocated by Fodor and Pylyshyn, he maintains that computation has to be explained in functional terns in a way that is analogous to engineering and biology. He proposes a mechanistic account of computation, which doesn't presuppose semantic content of computational states and processes. Rather, it states that the capacities of a computing mechanism are attributed to the organization of its sub components and their corresponding functions. This account appeals to functional explanations, and endorses the distinction between successful computations and miscomputations.

The objectivity criterion

The objectivity criterion dictates that an adequate theory of computation ought to identify computations as *a matter of fact*. Piccinini asserts that some philosophers like Searle and Putnam have suggested that computational descriptions are vacuous, because any system may be described as performing any computation. So allegedly there is no further fact of the matter as to whether one computational description is more accurate than another (ibid).

Computer practitioners appeal to empirical facts about the systems they study, design and implement to determine which computations are performed by which mechanisms. They apply computational descriptions to concrete mechanisms in a way entirely analogous to other credible scientific descriptions (ibid).

The explanation criterion

The explanation criterion dictates that an adequate theory of computation should explain the behaviour of computing mechanisms. It ought to explain how program execution relates to the general notion of computation. Inner computations may explain outer behaviours of computers. Traditionally, computational explanations have been translated or reduced to explanations by program execution. Piccinni (ibid) however resists this one to one translation and gives music boxes as examples of mechanisms, which operate by executing programs, but do not perform computations.

The right things compute criterion

The right things compute criterion dictates that a candidate theory of computation need to only encapsulate the mechanisms and devices that actually *compute* (ibid). Such a theory should entail that paradigmatic examples like digital computers, Turing machines, and finite state automata, compute. At the same time, an adequate theory of computation ought to *exclude* non-computing mechanisms and systems like planetary systems, digestive systems, Hinck's pail and Searle's wall.

The miscomputation criterion

The miscomputation criterion dictates the requirement that an account of computation addresses the fact that a mechanism can miscompute, i.e. a computation may go wrong. A mechanism M is said to be miscomputing in case computing a function F on input I, where $F(I) = O^1$, but M outputs O^2, where $O^1 \neq O^2$. An adequate theory of computation should explain how it's possible for a physical system to miscompute. This requirement plays an important role in computer science and in computation in the wild. Computer science practitioners devote a large portion of their time and efforts to avoid miscomputations and coming up with the appropriate ways to prevent them (ibid).

The taxonomy criterion

The taxonomy criterion dictates the requirement that any adequate theory of computation distinguishes between capacities of different classes of computing mechanisms. For instance, logic gates, which are a very low level component in computers, can only perform trivial operations on pairs of bits (ibid). Ordinary digital computers can in-principle compute any function on any input until they run out of memory.

The empirical criterion

The empirical criterion dictates the need to account for computational practice and existing computational systems and applications. Piccinini emphasizes the importance that computational practice plays in an adequate account of computation, is a way similar to Smith's empirical criterion. However, Smith asserts that this criterion questions the legitimacy of all the theoretical perspectives and nominates Silicon Valley to decide whether in practice something can be deemed computational. Piccinini is only implicitly committed to a narrower conception of doing justice to the body of practices. He claims that the existing computational practice, computing applications, computing systems etc. need to be properly taken into account (ibid).

An adequate alternative

Though some of Piccinini's criteria require some fine-tuning, I believe that they serve as an adequate alternative to those argued by Smith. The criteria above deal with essential characteristics of computation and address the need to account for miscomputation as well. Performing computation is an empirical fact similar to the functional role of the heart to pump blood or the photosynthesis process in plants. It is likewise crucial for such a candidate theory to show why certain systems perform computations whereas others simply don't. Some philosophers assert that almost any system, which is complicated enough, realizes a function etc., can be deemed computational. In my opinion, views like these trivialize the notion of computation and consequently theories of computation become trivial. If everything can be deemed a computer, then computational explanations become pointless and lose any philosophical interest. It is sometimes the case that computational *models* are confused with computational *explanations*.

The miscomputation criterion is yet another important feature of any adequate theory of computation. Computational systems are susceptible to miscomputations that result in an abnormal behaviour, which generally speaking may end up in one of two ways. The system can “handle” the miscomputation and resume its normal functioning (while may be losing some output in the process), in best-case scenario. Or it can malfunction and stop functioning completely, in worstcase scenario.

4. Von Neumann's answer

Back in the late 40's Von Neumann (1948) claimed that we were very far from possessing a proper logical – mathematical theory of automata. He was correct then and to some extent his claim is still resonating today.

The precision and reliability criterion

The result of complex computation performed by computing mechanisms may depend on a sequence of a billion steps and has the characteristic that every step actually matters or, at least, may matter with a considerable probability. This is the most specific and most difficult characteristic of computing mechanisms (ibid, pp. 291-292). In dealing with modern logic the important thing is whether a result can be achieved in a finite number of elementary steps or not. The *number* of steps, which are required, is hardly ever a concern. But, when it comes to computing mechanisms the thing, which matters, is not only whether it can reach a certain result in a finite number of steps at all, but also how many such steps are needed.

The single error criterion

Von Neumann compares the error handling of computing mechanisms to that of living organisms. He asserts that any malfunction, which occurs in an organism, is corrected by the organism itself without any significant external intervention. Error handling in computing mechanisms on the other hand is treated entirely different. In actual practice every effort is made to detect any error as soon as it occurs. An attempt is then made to isolate the erroneous component as fast as possible.

A computing mechanism could be designed so that it's able to operate almost normally in spite of a limited number of errors. However, as soon as the mechanism has begun to malfunction it will most likely go from bad to worse and only rarely restore itself. The error-diagnosing techniques, which are employed in practice, are based on the assumption that the computing mechanism contains only one faulty component (ibid, pp. 305-306).

The analogue – digital distinction criterion

All computing mechanisms fall into two main classes in a way, which is immediately obvious. This classification is into analogue and digital machines. An analogue computing mechanism is based on the principle that numbers are represented by continuous physical quantities. A digital computing mechanism is based on the method of representing numbers as aggregates of digits. Digital computers represent quantities by discrete states, operate serially, and have a better accuracy (ibid, pp. 292-293).

5. "Everything" computes answer

Views like those of Putnam and Searle (1990) imply a very loose notion of computation so that almost everything can be deemed to be computing. Searle's notorious wall and Putnam's realization theorem of finite automata suggest that even a *rock* can be claimed to be computational (Chalmers 1996). The fifth answer advocated by Scheutz (1999), Putnam (1992) and others states that a satisfactory account of computation is underpinned by an adequate theory of implementation. However, existing accounts of computation are inadequate due to lack of a satisfactory theory of implementation. Scheutz (1999) asserts that the notion of implementation should be construed as *realization of functions*, rather then the standard concept of physical state to computational state correspondence.

Scheutz suggests tackling computation from a practical point of view, i.e. by looking at existing applications and systems that are designed, implemented and used by people. Rather then asking how abstract computations relate to physical systems, it should be the other way around.

His theory of implementation implicitly presupposes the *empirical criterion*. Scheutz (ibid) maintains that computation should be defined in terms of an abstraction over the physical properties determining the functionality of a physical mechanism.

6. Conclusions

At the outset of his paper: "The Foundations of Computing", Smith (2002: p. 24) asks and I quote "Will computers ever be conscious? How will computing affect science? ... For most of my life I have been unable to answer these questions, because I have not known what computation is. This uncertainty led me to undertake a long-term investigation of the foundations of computer science". What had started as a long endeavour to provide a satisfactory theory of computation ended up with a negative conclusion: *Computation is not a subject matter*. It is not only that we *currently* have no satisfying intellectually theory of computation, but we will never have such a theory (ibid: p. 51).

Smith claims that computation is intrinsically intentional, which also prompts him to formulate the *cognitive criterion*. He maintains that the misconception of computation as being entirely abstract and formal hides the semantic character of computation. Thus, any adequate theory of computation has to rely on a solid *theory of semantics and intentionality* (ibid: pp. 46-50).

Computation doesn't constitute a *distinct ontological* category. What qualifies as computers according to Silicon Valley doesn't form a coherent delimited class. By showing that extant theories of computation fail to meet the three criteria, Smith claims that any candidate theories are condemned to failure.

He takes the empirical criterion to the extreme. Not only does a theory of computation has to do justice to real life computation by *explaining* programs like the Open Office Writer, but it should give rise to *reconstructing* computational practice. This doesn't preclude adopting a subtler version of the empirical criterion. Piccinini's (forthcoming) implicit empirical criterion implies that existing practices, computing applications and other real life examples need to be properly *taken into account*. Scheutz's (1999) implementation theory criterion reflects the computational practice by revisiting the standard correlation between computation and implementation. Rather then appealing to a top down approach that begins in the abstract level and progresses down to the concrete, Scheutz's starting point is real life examples of computers and applications.

Smith's cognitive criterion is also too difficult to meet. Whilst proponents of computationalism should obviously heed the theory of computation, he believes it should rather be the other way around. An adequate theory of computation should apparently be an intelligible foundation for the formulation of the computational theory of mind. Moreover, when considering this criterion in conjunction with his claim that we will never have an adequate theory of computation, the result is surprising. If we take the main claim of computationalism being that minds are *computational* systems, and given that there will never be an adequate *theory of computation*, it follows that there will never be an adequate computational theory mind.

Acknowledging the paramount role of theories of computation in computationalism is by no means unique to Smith. Fodor and Pylyshyn's answer is applicable both to computers and cognition as complex systems. Piccinini (forthcoming) explains that the difference between computing mechanisms that execute programs and those that don't is important not only to computer science, but also to theories of mind. Even the earlier view of Von Neumann (1948) emphasizes the similarities of computing mechanisms to the human central nervous system. The last answer states that a lack of a tenable notion of implementation renders computationalism meaningless.

Though susceptible to some legitimate criticism, there are adequate alternatives to that of Smith's and there's no compelling reason to renounce any attempts to provide a satisfactory theory of computation.

Acknowledgments

This draft of the paper was presented in the AAP conference held in Armidale, July 2007. A more detailed draft has been submitted to *Minds and Machines* for review. Special thanks go to

Phillip Staines and Joseph Agassi, whose trenchant comments and insights made this paper appreciably better.

References:

Chalmers, D. J. (1996). Does a Rock Implement Every Finite-State Automaton? Synthese, 108, 309-333.

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

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

Putnam, H. (1992). Representation and Reality. 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. I. (Ed.) Foundations of Cognitive Science. (pp. 51-91)). Cambridge: The MIT Press.

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

Searle, J. R. (1990). Is the brain a digital computer? Proceedings and Addresses of the American Philosophical Association, 64, 21-37.

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

Smith, B. C. (forthcoming). The Age of Significance. Volume I.

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

Staines P., Bell, P., Mitchell, J. (2001). Evaluating, Doing and Writing Research in Psychology: A Step-By-Step Guide. Sage publication.

Von Neumann, J. (1948). The General and Logical Theory of Automata. (In A. H. Taub (1961) Design of computers, theory of automata, and numerical analysis. (pp. 288-328)). Pergamon Press.