Matthias Scheutz

THE COGNITIVE-COMPUTATIONAL STORY

Summary. Computationalism is not a new program, but rather the product of the mechanist views in the 17th century, which has been under attack by philosophers and others every since. However, recently new attacks were launched from adherents of dynamical systems within cognitive science suggesting that the computationalist program fails in explaining mind because computation necessarily neglects the real-time, embodied, real-world constraints with which cognitive systems intrinsically cope. As a consequence, many researchers turned their attention away from computationalist models. Yet, an increasing number of researchers are not willing to give up what has been *in nuce* a quite successful approach. Rather they are motivated to reconsider the very notion of computation as well as the whole enterprise of computing and computational modeling. To get an idea of where this "new computationalism" might be heading, I will sketch very briefly the "old paradigm" and some of its historical roots, paying tribute to two of the most influential computationalists: Leibniz and Turing. I will then offer but a glance at some of the shortcomings for which computationalism has been criticized, interspersed with issues that a successor notion of computation will have to address.

Zusammenfassung. Der Computationalismus ist kein neues Programm, sondern ein Produkt der mechanistischen Sichtweise des 17. Jahrhunderts und wird seither von Philosophen und anderen attackiert. In jüngster Zeit kommen neue Angriffe vonseiten der Anhänger dynamischer Systeme innerhalb der Cognitive Science, welche behaupten, daß der Computationalismus deshalb an der Erklärung des Geistes scheitert, weil Berechnung notwendigerweise Echtzeit, Körperhaftigkeit und andere Beschränkungen der wirklichen Welt, mit denen kognitive Systeme fertigwerden müssen, vernachläßigt. Folglich wandten sich viele Forscher von computationalen Modellen ab. Andererseits ist eine zunehmende Anzahl von Forschern nicht gewillt, das im Grunde erfolgreiche Program einfach aufzugeben, sondern vielmehr motiviert, den Begriff der Berechnung sowie das ganze Unternehmen der Computerei und der Computermodellierung neu zu überdenken. Als Vorgeschmack auf die zu erwartenden Entwicklungen des "neuen Computationalismus" skizziere ich zunächst das "alte Paradigma" sowie einige seiner historischen Wurzlen, wobei ich zwei der einflußreichsten Computationalisten, Leibniz und Turing, Tribut zolle. Danach stelle ich kurz einige Kritikpunkte des Computationalismus vor und deute auf Fragestellungen hin, die ein Nachfolgerbegriff der Berechnung ansprechen müssen wird.

***

The notion of computation, upon which the whole research program called computationalism is built, is undoubtedly one of the central notions of the last century. Its history traces back to Leibniz and before, when daring philosophers pondered mechanical systems that could aid humans in performing calculations, and possibly even calculate by themselves without any human assistance. The first functioning mechanical calculators were built in the 17th century, a century that showed interest for all kinds of mechanical devices. These calculators were composed of various mechanical parts (such as of gears, cogs, etc.) with the technology originally developed by clockmakers for making watches and were used in the construction of various mathematical tables (such as the table of logarithms). Leibniz himself constructed various mechanical calculators which could perform additions and multiplications, one of which is still in working order today (Williams, 1997, pp. 130). He was convinced that mechanical calculators would be of great commercial utility, but he also envisioned yet another much bolder application of calculators, that of “mechanical reasoners”.

While I will not attempt to explicate Leibniz’ ideas on logic and reasoning, I would still like to mention two major ideas that, I believe, led to the concept of a mechanical reasoner:

1. that logic can be viewed as a formal, deductive system in which reasoning takes the form of deductions (which themselves proceed according to rules), and
2. that reasoning involves the manipulation of symbols, that is, representations.

The first idea is intrinsically connected to Leibniz’ view on concepts and language, maintaining that there are *unanalyzable* so-called “first terms”, simple representations out of which all other complex representations are built. In modern terminology, these first terms are sometimes called “grounded terms” and it is one of the challenges in cognitive science to account for how these can be acquired (it seems clear that this cannot happen through language directly, rather language learning seems to presuppose the acquisition of such terms). According to Leibniz, words in ordinary language usually disguise the relationship between the various representations. For this reason he set out to devise a universal language (called “characteristica universalis”), which would formally reflect the relationship between “simple” and “complex” representations. As an aside, I might add that Leibniz believed that such a universal language would resolve any philosophical disagreement, as once a statement has been formalized, its validity could be checked *mechanically*: if controversies were to arise, he said:

> *"…there would be no more need for disputation between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, to sit down to their slates, and say to each other (with a friend to witness, if they liked): calculemus—let us calculate".* (Leibniz, 1875-90, p. 200)

Leibniz' idea of a mechanical reasoner—an instance of what today we might call an "automatic formal system" (Haugeland, 1985)—predates early efforts in artificial intelligence research. In particular, I would suggest, it is his view of calculating which already hints at the modern proposal that semantics will take care of itself as long as the syntax is right.

The second idea, that reasoning in particular, and thinking in general, involves representations, is also a product of the 17th century (due to Descartes, Hobbes, Locke, and others) and was supported by the rise of modern mathematics. After Vieta had introduced the concept of "representatives" suggesting that marks should *stand in* for numbers in calculations (a method stemming from lawyers' practice at court), calculations were done with signs rather than through physical manipulations. This mathematical practice of using marks and signs as representations in calculations quickly spread and soon became a paradigm for thought in general (Pratt, 1987). It led to claims such as Hobbes' famous dictum that "everything done by our mind is a computation" (Hobbes, 1994, p. 30).

The outline of the historical roots of the computation I have compiled so far is no more than a sketch. Yet, it should suffice to introduce what I would like to suggest: that the core ideas of present day computationalism had already been conceived and to some extent formulated in the 17th century, though not with today's terminology and not with today's understanding of computation. Because of its unique role of being intrinsically connected to mechanical calculators on the one hand and to cognitive processes such as calculating on the other, the notion of computation effected a link between the mental and the physical. And it was this link that eventually gave rise to the hypothesis that mind, in general, might be "mechanizable".

Computation at Leibniz's time was already tied very much to the idea of manipulating representations, and prototypical manipulators were found in the mechanical calculators of those days. While many attempts were made at building mechanical calculators up to the end of the 19th century (some of which were quite successful, e.g., see Pratt, 1987, Williams, 1997, or

Augarten, 1985), the computing capabilities of these machines were very modest. It was only in the 20th century, when we finally witnessed a quantum leap in the construction of computers and our conception of computing. This was largely due to two quite independent developments:

1. the thorough logical analysis of the notions "formal system" and "demonstrability" (i.e., proof by finite means) of formulas in formal systems, leading to further studies of notions such as "recursive function", "effectively computable function", "algorithm", "finite state automaton", and others, and
2. the rapid progression in the engineering of electronic components (from vacuum tubes, to transistors, to integrated circuits, and beyond)[1]

It is worth pointing out that the notion of computation took off in two different directions at this point, if not earlier. Each of these routes, in turn, led to a particular view on and interest in computation, one of which could be called the "logical" or "theoretical" route, the other the "practical" or "*techno*logical" route of computation. The logical route examines the theoretical limitations of computation, while the technological route explores issues that arise in constructing actual computers. While both approaches are concerned with important aspects of computation (and are by no means incompatible), their use of the term "computation" is subject and interest-specific. And, moreover, since research can be conducted quite independently in both disciplines, it is not surprising that these two routes did not cross very often. Only in recent times do we witness a mutual interest, as logic became more sensitive to real-world constraints (considering complexity theory, for example). Alternative conceptions of computations such interactive Turing machines, games, etc. are thought to overcome the entfremdung of the classical logical models from worldly concerns.

Computation from a Logical Perspective

The logical side of the history of computation, which started in the Thirties with different attempts to make the intuitive notion of computation (then called "effective calculability") formally precise, was solely concerned with

[1] The timeline available at IEEE's web site demonstrates graphically the enormous gain in momentum of computer development ever since the first transistor was constructed in 1947. See also Williams, 1997.

what could be computed in principle. This, in turn, required a thorough analysis of the intuitive notion of computation. The most crucial insight of the Thirties with respect to the meaning of this intuitive notion of computation was most likely the fact that three different attempts to characterize it formally could be proven equivalent: the class of recursive functions equals the class of λ-computable functions equals the class of Turing machine computable functions. These equivalence results are possible, because what "computing" means with respect to any of the suggested formalisms is expressed in terms of functions from inputs to outputs; and using functions as mediators, the different computational formalisms can be compared according to the class of functions they compute.

Later, other formalisms such as Markov algorithms, Post systems, universal grammars, PASCAL programs, as well as various kinds of automata were also shown to "compute" the same class of functions, referred to as *recursive functions* (e.g., see Hopcroft and Ullman, 1979). The extensional identity of all these formalisms supports a famous thesis formulated by Church as a definition:

> *"We now define the notion [...] of an effectively calculable function of positive integers by identifying it with the notion of a recursive function on positive integers (or of a λ–definable function of positive integers). This definition is thought to be justified by the considerations which follow, so far as positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion".* (Church, 1936, p. 356, also in Davis, 1965, p. 100)

Using the various equivalent results it follows from "Church's Thesis" that any of the above mentioned formalisms captures our intuitive notion of computation, that is, *what it means to compute*. Although this thesis cannot be proved in principle as mentioned by Church himself, it became more and more plausible as newly conceived computational formalisms were shown to give rise to the same class of "computable" functions.

What is common to all these computational formalisms besides their attempts to specify formally our intuitive notion of "computation", is their property of being independent from the physical. In other words, computations in any of these formalisms are defined *without* recourse to the nature of physical systems that (potentially) realize them. Even Turing's machine model, the so-called "Turing machine", which is considered the prototype of a "mechanical device", does not incorporate physical descriptions of its inner workings, but abstracts over the mechanical details of a physical realization.

Interestingly enough, Turing (1936) invented his machine model of "computation" in order to capture the human activity of "computing", i.e., the processes a person (the "computer") goes through while performing a calculation or computation using paper and pencil. He was not concerned with digital computers at all or the foundations of computing.[1] Rather, he focused on the problem of analyzing and modeling what the possible processes are that people go through when they "blindly" follow rules to do computations. In his analysis of the limitations of the human sensory and mental apparatus five major constraints for doing "automatic computations" crystallize (I follow the presentation in Gandy, 1988):[2]

1. Only a finite number of symbols can be written down and used in any computation
2. There is a fixed bound on the amount of scratch paper (and the symbols on it) that a human can "take in" at a time in order to decide what to do next[3]
3. At any time a symbol can be written down or erased (in a certain area on the scratch paper called "cell")
4. There is an upper limit to the distance between cells that can be considered in two consecutive computational steps
5. There is an upper bound to the number of "states of mind" a human can be in and the current state of mind together with the last symbol written or erased determine what to do next

Although there are certainly some steps in Turing's analysis of an abstract human being performing calculations that seem rather quick and not well supported, one can summarize the above in Gandy's words as follows:

"The computation proceeds by discrete steps and produces a record consisting of

[1] Although Turing used the term "computer" to refer to whatever is doing the computations, he did not intend it for a digital computer, but for a human person doing computations—at that time digital computers did not yet exist.

[2] Note that "Turing's account of the limitations of our sensory and mental apparatus is concerned with perceptions and thoughts, not with neural mechanisms. And there is no suggestion that our brains act like Turing machines." (Gandy, 1988, p. 87)

[3] This requirement does not exclude an arbitrary amount of scratch paper. It just delimits the range of perception, i.e., the amount of information the human "computer" can use at any given time to determine the next step in the computation.

a finite (but unbounded) number of cells, each of which is blank or contains a symbol from a finite alphabet. At each step the action is local and is locally determined, according to a finite table of instructions." (Gandy, 1988, p. 81)

In other words, by "abstracting away" from persons, scratch paper, etc., Turing (1939) claimed that all "computational steps" a human could possibly perform (only by following rules and making notes) could also be performed by his machine. In this way the Turing machine became a model of human computing, an *idealized* model to be precise, since it could process and store arbitrarily long, finite strings of characters. It is worth pointing out that Turing, as opposed to Church, did not only state a "thesis" regarding the intuitive notion of computation, but he actually proved a theorem (see also Gandy, 1988, p. 83, who restates Church's Thesis as Turing's Theorem): "Any function that can be computed by a human being following fixed rules, can be computed by a Turing machine".

Turing also believed the converse, that every function computed by a Turing machine could also be computed by a human computer (although this, again, does not take time and space restrictions seriously, but rather assumes an abstract human computer not subject to such worldly limitations). In particular, Turing was convinced that "the discrete-state-machine model is the relevant description of one aspect of the material world —namely the operation of brains". (Hodges, 1988, p. 9, see also Jack Copeland's chapter). The origins of Turing's claim can be found in the intrinsic connection between the notion of "computability" and Gödel's notion of "demonstrability" (of a proof in a formal system): that which can be "demonstrated" using "definite methods" amounts to what can be done by a Turing machine (see Turing, 1936). By relating the limitations of formal systems as pointed out by Gödel to the limitations of his machine model, Turing

"[…] perceived a link between what to anyone else would have appeared the quite unrelated questions of the foundations of mathematics, and the physical description of mind. The link was a scientific, rather than philosophical view; what he arrived at was a new materialism, a new level of description based on the idea of discrete states, and an argument that this level (rather than that of atoms and electrons, or indeed that of the physiology of brain tissue) was the correct one in which to couch the description of mental phenomena" (Hodges, 1988, p.6)

The independence of computations from their physical realizers was one major source of attraction for some psychologists in the late Fifties. Another was the potential of computers to process information—an ability thought to underlie human cognition. The information processing capabilities of computers and the possibility of specifying by virtue of programs exactly *how* information is processed, opened doors for an interesting thought: maybe cognition viewed as information processing could be understood in terms of computations? Maybe cognitive functions *are* computations. If so, explanations of mental processes in terms of programs would be scientifically justifiable without having to take neurological mechanisms into account. The wetware would simply be viewed as a computer on which the software "mind" is running (or if not mind itself, then at least all the cognitive functions that constitute it). Consequently, this position would go a long way in establishing cognitive psychology as a serious area of scientific research.

The analogy underlying the usage of the terms "computer" and "program" in cognitive psychology, the so-called "computer metaphor", can be succinctly summarized by saying that "the mind is to the brain as the program is to the hardware" (Searle, 1980, or Johson-Laird, 1988). The guiding ideas of this metaphor became so prominent, originally in psychology, later in artificial intelligence, as to establish "computationalism" (also called "the computational claim on mind") as a genuine research paradigm. This paradigm was the midwife assisting in the birth of what we nowadays know as cognitive science (e.g., see Gardner 1985).

To pinpoint the various positions and claims subsumed under the notion computationalism would be a research project in its own right. Just to give you an idea of what can be found in the literature, slogan-like phrases such as "the brain is a computer", "the mind is the program of the brain", "cognition is computation", or "the mind is a computer", are not uncommon, and these are only a few. Note that in a phrase like "cognition is computation" the interpretation of every single word matters, "is" included —do we interpret "is" as "extensional identity" or "extensional inclusion"? Or do we read it intensionally, etc.? Such statements are necessarily condensed and cannot be taken at face value; for if they were read together, they would equivocate essentially distinct notions (such as program and process, mind and cognition, etc.).

There are other descriptions of computationalism that emphasize the information processing capabilities of computers. For example, computationalism has been characterized as the conjunction of the following three theses "thinking is information processing", "information processing is computing (is symbol manipulation)", and "the semantics of those symbols connect mind and world".

In any case, computationalists have been content with notions of computation as provided by formal logic, thus respecting the delimiting results of the 30ies and 40ies (such as Turing machines, Post correspondence systems, Markov Algorithms, λ-computable functions, recursive functions, PASCAL programs, etc.), and in particular, the Church-Turing thesis. By taking computationalism as "the hypothesis that cognition is the computation of functions" (as Dietrich, 1990, for example, suggests in his computational manifesto entitled "computationalism"), this dependence of computationalism on classical notions of computation, that is, on what it means to compute a function, becomes apparent.

Two main assumptions are buried in the computer metaphor: 1) that the mind can somehow be "understood as a computation" or be "described by a program" (this will require the adoption of a notion of computation or program, respectively), and 2) that the same kind of relation that obtains between programs and computer hardware (i.e., the implementation relation) obtains between minds and brains too. Assumption 1) has led to fertile research in psychology as well as artificial intelligence collecting evidence for its truth, while assumption 2) by and large remained at the level of an assumption.[1]

Expanding a bit on assumption 1), to get a better idea of how to think of the mind as being described by a program, computation can be viewed as the rule-governed manipulation of representations. After all, this is what computers do: they manipulate symbol tokens (e.g., strings of bits), which themselves are representations of the subject matter the computation is about (compare this to Newell's notion of "physical symbol system"). This way of

[1] This is so, presumably, because neither AI researchers nor psychologists need to pay attention to it. AI researchers, who build computational models, implicitly deal with the implementation relation of software on computer hardware on a daily basis, but are not concerned with the implementation relation of minds on "brain hardware"; nor are psychological studies, which remain at the level of "program description". Neuroscience would probably be the closest discipline concerned with implementation issues of brains. Yet, neuroscientists do not attempt to relate program-like descriptions to brain areas, rather they attempt to study the functional role of these areas (with respect to the rest of the brain) directly by virtue of their physiological functions.

thinking of computation as rule-governed manipulation of representations is typical of philosophers (e.g., see Fodor, 1981, or Haugeland, 1985). The representations, on which computations operate, have both formal and semantic properties; the formal properties are the ones being manipulated by the computations, as the semantic properties are not causally efficacious.

The representational capacity of computations is only one of the many reasons, why computation is such an attractive candidate for explaining cognition (others include the potential of computational processes to have semantics, the causal efficacy of computations, the algorithmic specifiability of processes, etc.). Another important reason is that the notion of computation is a logically heavily studied, well-worked out notion, in particular, the notion of "Turing machine computability" (whence the reliance of most computationalists on these "classical" notions of computation). Yet another major plus of computations is that we know how to implement them, i.e., how to connect an abstract description of processes (in terms of programs) to a concrete physical system (i.e., the computer implementing the program)—assumption 2) underlying computationalism from above.[2] Finally, as already mentioned in the beginning, it is the crucial idea that programs, or more generally computations, somehow "mirror" the causal structure of computers (=hardware) in particular, and physical systems in general, which plays a central role in any computationalist account.

Recent Attacks on Computationalism

Computationalism has always been under attack from various directions even before it was officially recognized as such. As I pointed out before, the idea that mind could be at least in principle mechanizable, is not a product of this century, but goes back at least to the time of Leibniz. Yet, the possibility of using formal methods to characterize the notion of computation, especially the notion of "effective procedure", enabled critics of computationalist views to utilize results from formal logic (Gödel's incompleteness theorems being the most commonly quoted) in order to make their points more rigorously (see Lucas, 1961). Of course, there were other

[2] Cp. this to Brigdeman's statement about the subject matter of artificial intelligence: "Artificial intelligence is about programs rather than machines only because the process of organizing information and inputs and outputs into an information system has been largely solved by digital computers. Therefore, the program is the only step in the process left to worry about." (Bridgeman, 1980).

objections, not based on formal logic (e.g., Searle's Chinese room thought experiment), but none of them seemed to endanger the computationalist program as much as recent attacks coming from a completely different angle, and most importantly, from within cognitive science itself.

In particular, connectionists and dynamicists have tried to replace the notion of computation with alternatives arguing that the symbolic/computational level of description so crucial to computationalism cannot be taken for granted. While some connectionists believe that symbolic activity should emerge from a "sub-symbolic level" (e.g., Smolensky, 1988), most dynamicists find the symbolic level of description superfluous altogether and argue instead for an explanation of cognition in terms of dynamic systems (e.g., Port and van Gelder, 1995). Others—biologists and neuroscientists, for example—are trying to "go in under" the computational level to understand the mind directly at the level of the brain. Diversely, some social theorists and roboticists have argued that the essence of intelligence is to be found in situated interaction with the external world, rather than in a purely internal world of symbol manipulation.

The new objections supplementing the old ones are based on a broad spectrum of different reasons such as problems with the notion of representation (including inappropriate applications of the concept to various cognitive phenomena), wide notions of content and supervenience, universal realization results, etc. just to name a few. Yet, one of the major lines of attack on computationalism is advanced by the so-called dynamicists, who criticize the non-dynamical nature of computational systems. For example, van Gelder (1998) argued that what is essential to computation is the notion of an effective procedure, and essential to that is the notion of discrete steps in an algorithm. He claims that this discreteness, in both its temporal and non-temporal aspects, prevents computation from explaining many aspects of cognition, which he considers to be a fundamentally dynamical phenomenon.

Hence, instead of using any of the standard computational formalisms, one ought to use *dynamical systems* in describing cognitive functions, the idea being that every real-world system (and thus cognitive systems as well!) involving change can potentially be modeled by a dynamical system—this is what dynamic systems have been designed to do. According to the respective system, this will happen at different levels of description, at the very low level of fields (take Maxwell's equations), or the very high level of

Dynamical systems are collections of mathematical equations, either differential or difference equations, depending on whether time is taken to be discrete or real-valued. Independent of the assumptions on time, the values of the variables for each relevant physical dimension can also either be discrete or real-valued. So there is a total of four combinations, all of which are possible: all variables can be either discrete or continuous, but it is also possible to have a system with discrete variables for physical dimensions, yet a real variable for time. And the fourth option is to have time discrete, but all variables for physical dimensions continuous. Which option is used depends solely on the kind of physical system, the behavior of which one wants to describe as a dynamical system as well as pragmatic assumptions and constraints (such as "exactness of measurements", "required degree of precision", etc.). Dynamic systems are not committed to any particular physical quality either (the same way they are not committed to the discrete-continuous distinction). They are not committed to particular notions of physical states, nor are they committed to a realist or instrumentalist interpretation of a theory's entities. Whatever changes over time, can be modeled, and it does not even have to be time, because in case of difference equations all that matters is order! For that very reason, dynamical systems are sometimes viewed as encompassing computational systems, although exactly how they "encompass" them is what all the theoretical weight hinges upon (e.g., see Scheutz, 1999).

human decision making (take Busemeyer and Townsend's decision field theory).[1]

Another attack, also advanced by dynamicists, challenges the role of representation in cognitive science in general, and *a forteriori* can be seen as a challenge to the role of computation in cognitive science. Especially psychologists have argued that certain allegedly "cognitive" tasks have nothing to do with cognition proper, but are really motor control tasks that can be explained and modeled in the language of dynamical system without resorting to manipulations of representations (e.g., see Thelen and Smith, 1994). As a consequence, the following question arises: to what extent does the notion of representation have to be involved in explaining cognitive

[1] To describe a physical system, one needs to introduce a variable for each relevant physical dimension and consider it as a function of time. The simplest way to specify the behavior of the physical system would be to provide graphs of each such variable over time, that is, to have a set of functions $X_1(t)$, $X_2(t)$, ... , $X_n(t)$ where $X_i(t)$ yields the "state" of the relevant physical dimension X_i at time t. This set of functions will determine the behavior of the system for all times. However, it does not reveal the possible dependencies of the X_i on each other. This is where differential equations come in handy. They provide a way of specifying the various interdependencies of different variables in such a way that graphs of each variable can be obtained from them, yet the interdependencies are also brought to the open. The nature of these interdependencies will become a crucial factor in an explanation of the behavior of the system, and the mathematical theory of dynamical systems seems well-suited to describe quantitatively systems that exhibit such interdependencies, or in Clark's terms (1997) "continuous reciprocal causation".

abilities; and furthermore: is it possible to invoke the concept of representation within dynamical system theory itself when needed, thus bypassing the "classical representational=computational" level of description (see the various articles in Port and van Gelder, 1995)?

Quite a few attacks against computationalism have also been launched from a philosophical direction. For example, it has been argued that some mental states are relationally individuated (Putnam, 1975), while computational states are not (Fodor, 1981). Moreover, philosophers have argued that traditional notions of computation are conceptually inadequate and at best incomplete (Smith, forthcoming).

Some, for example, claim that various questions about the nature of computing as we see it in computational practice will not receive a satisfying answer, if computation is viewed as computation of an input-output function. For example, there are questions about the implementation of a computation, that is, *how* a function is computed: does computing a function imply "following rules" or "executing an algorithm"? Does the computation of a function have to be productive or would non-algorithmic methods, for instance, that arrive at the same results count as well? Other more general concerns regarding the reductive approach of viewing computation as the computation of a function itself have been voiced: is it true that every computation can be explained as the computation of a function? Consider, for example, Arcade video games. What input-output functions do they compute? Or take operating systems. They are specified by programs, which are designed to never halt. Contrast this with the classical approach, where such so-called "divergent" functions are neglected.

Another worry is with the notion of implementation: how do we establish that a computational description does indeed describe the processes in a physical system? Motivated by computational practice, it is widely held that this mirroring is established by setting up a functional correspondence between computational and physical states. For example, there is a tight correspondence between parts of the architecture of a von Neumann CPU and expressions of the assembly language for that CPU. Another example would be a logic gate (e.g., an AND gate), whose "computational capacity" is described by a Boolean function, the values of which in turn can be related to physical magnitudes in the physically realized circuit.

While we can more or less easily establish such a functional correspondence between physical and computational states for certain artifacts (i.e., devices we have designed to allow for computational descriptions), it is unclear that this can be done for natural kinds (such as brains) too. In particular, one has to be aware that any such correspondence depends crucially on "adequate" physical states:[2] computations "mirror" the causal structure of a system under a given correspondence function between computational and physical states only relative to the choice of the physical states. Computational explanations of the behavior of a given physical system, therefore, depend essentially on those physical states of the system that can be set in correspondence to some computation. This dependence, per se, is not problematic as long as one can assume "appropriate physical states" of a system (e.g., as in the case of electronic devices). If, however, it could be shown that for any computational description one can find "physical states" of a given system, which can be set in correspondence with the computational ones and are, furthermore, appropriate (in a certain sense of "appropriate" which will depend on the underlying physical theory), then computational explanations would be in danger: every system could then be seen to compute! In other words, computationalism would be vacuous if every physical system could be viewed as implementing every computation. And, indeed, it has been argued that any (open) physical system, for example, can be seen to implement any finite state automaton, or that walls implement the Wordstar program (as has been suggested by Putnam, 1988, and Searle, 1992, respectively) under an intuitive notion of implementation. If that is so, then something clearly must have gone wrong with our intuitions, since such a view of computation and implementation is not tenable, neither from the theoretician's nor the practitioner's perspectives.

While all of the above-mentioned critiques of computationalism vary, they share a common theme: computation fails as an explanatory notion for mind, because computation necessarily neglects the real-time, embodied,

[2] In the case of electronic devices the appropriate physical states can be determined either by asking the engineers who designed the devices or by looking at the blueprint and comparing it to the computational description of the device. In the case of biological systems, however, such states are not clearly defined. Consider, for example, physical states of a pyramidal cell: which of those states could correspond to computational states such that the respective computation captures essential parts of the causal structure of these cells? It has been suggested that "firing" vs. "not firing" would be "natural" candidates (e.g., by McCulloch and Pitts, 1943), but it turns out that this computational model is too reductive as essential temporal processes (such as temporal integration of signals, maximal firing rates, etc.) are completely neglected. Hence, more factors about pyramidal cells need to be taken into account, yielding more physical states that have to correspond to computational ones, etc. Artificial neural networks seem to be promising candidates for such computational descriptions, but the issue has not to my knowledge been resolved. It might well be that in the end the complex behavior of pyramidal cells defies a computational description, but this is obviously an empirical issue.

real-world constraints with which cognitive systems intrinsically cope. This is a consequence of assuming computation to be defined solely in abstract syntactic terms (abstracting over physical realization, real-world interaction, and semantics).

How can we continue to hold on to computationalism, it is asked, when it seems, for example, that digitality is restrictive, formal symbol manipulation is not sufficiently world-involving, and Turing machines are universally realizable to the point of vacuity? It should not come as a surprise that these issues together with recent progress made by dynamicists (while the classical approach *prima facie* appears stagnant), led many cognitive scientists to abandon computationalism altogether.

A New Computationalism?

Despite the dire prospects for computationalism currently envisioned by some dynamicists, an increasing number of researchers are not willing to give up what has been *in nuce* a quite successful approach. Rather they are motivated to reconsider the very notion of computation as well as the whole enterprise of computing and computational modeling. These researchers are inspired by their recognition of the fact that real-world computers, like minds, also deal with issues of embodiment, interaction, physical implementation, and semantics. This recognition allows them to consider the possibility of classical computationalism failing not because computing is irrelevant to mind, but because purely "logical" or "abstract" theories of computation do not to address real-world aspects that are vital to both (real-world) computers and minds. Perhaps, so it is speculated, the problem lies not so much in computing *per se*, but in our present understanding of computing, in which case the situation could be remedied. If such a successor notion of computation can be defined (respecting the real-world nature of computers and computing), the resultant rehabilitated computationalism may still be the best bet for explaining cognition.

Note that if contrary to common (mis)perception computationalism and "dynamicism" are not mutually exclusive, it might be possible to combine the strengths of each. Some attempts have been made to combine both approaches, in particular, with respect to the notion of representation (e.g., based on results from neuroscience, the idea of "emulators", i.e., dynamical circuits that serve as models of parts of the outside world and thus "stand in" for these parts, plays a prominent role in finding a dynamic substitute for the classical, symbol oriented notion of representation; e.g., see Clark and

Grush, 1999). While some of the "hybrid models" (i.e., models that include dynamical as well as computational components) successfully integrate both formalisms, they are silent about the general relationship between computations and dynamic systems. Rather, they simply demonstrate that in this very specific instance, computational and dynamical systems methods and explanations can be combined in a particular way. This leaves unanswered, however, what the general relationship of both modeling paradigms is. What is lacking in these individual symbiotic attempts is a thorough analysis of the advantages and disadvantages of each approach together with their interplay on a conceptual as well as an applied level. A new notion of computation more closely tied to the real-world constraints imposed by the implementing systems might be able to elucidate the relation between computational and dynamical descriptions.

Other issues that will have to be addressed by a successor notion of computation are the program/process distinction, the notion of implementation and questions of physical realization, real-time constraints and real-world interactions, the use and limitations of models, relations between concrete and abstract, the proper interpretation of complexity results, the relation between computation and intentionality, "wide" notions of computation, computational practice, and so on. I believe that all of these aspects of computation will eventually play a role in a notion of computation that can serve as a firm foundation for cognitive science in the new century.

References

Augarten, S. (1985) *Bit by Bit—An Illustrated History of Computers*. George Allen & Unwin: London.

Bridgeman, B. (1980), "Brains + Programs = Minds", reply to Searle, *Brain and Behavioral Sciences,* **3**.

Clark, A. and Grush, R. (1999) "Towards a Cognitive Robotics". *Adaptive Behavior*, Vol 7:1.

Church, A. (1936): *An Unsolvable Problem of Elementary Number Theory.* In: *American Journal of Mathematics* 58: 345 - 363 (Nachdruck in Davis, Ed., 1965: 88 - 107).

Davis, M., Ed.. (1965): *The Undecidable.* New York.

Dietrich, E. (1990) "Computationalism", *Social Epistemology*, vol 4, no 2, 135-154.

Fodor, Jerry A. (1981) *RePresentations. Philosophical Essays on the Foundations of Cognitive Science*. Cambridge, MA, MIT Press.

Gandy, R. (1980) "Church's Thesis and Principles for Mechanism". Proceedings of the Kleene Symposium (J. Barwise, H. J. Keisler and K. Kunen, eds.). New York: North-Holland Publishing Company.

Gandy, R. (1988) "The Confluence of Ideas in 1936", in *The Universal Turing*

Machine: A Half-Century Survey. Kammerer & Unverzagt: Berlin.

Gardner, H. (1985) The Mind's New Science: A History of the Cognitive Revolution. Basic Books Publishers: New York.

Gerhard, C.J. (1875-90) *Die philosophischen Schriften von Leibniz*. 7 vols. Berlin: Weisman. Reprinted (1965) Hildesheim: Olms.

Haugeland, J. (1985) *Mind, Design I*. Cambridge, Massachusetts: MIT Press.

Haugeland, J. (1996) "What is Mind Design?" *Mind Design II*. Cambridge, Massachusetts: MIT Press.

Herken, R. (1988) *The Universal Turing Machine: A Half-Century Survey*. Kammerer & Unverzagt: Berlin.

Hobbes, T. (1994) *Levithan*. In *The Collected Works of Thomas Hobbes*. Routledge.

Hodges (1988), "", in *The Universal Turing Machine: A Half-Century Survey*. Kammerer & Unverzagt: Berlin.

Hopcroft and, J. E. and Ullman, J. D. (1979) *Introduction to Automata Theory, Languages, and Computation*. Massachusetts: Addison-Wesley Publishing Company.

Leibniz, G.W. (1875) *De Arte Combinatoria*. In *Die philosophischen Schriften von Leibniz*. 7 vols. Berlin: Weisman.

Lucas, J.R. (1961) "Minds, Machines, and Gödel". *Philosophy* **36**, 122-127.

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

McCulloch, S. W. and Pitts, W. H. (1943) "A Logical Calculus of the Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*, Vol. 5, Chicago: University Press, 115—133.

Molesworth, W. (1994) *The Collected Works of Thomas Hobbes*. Routledge.

Pratt, V. (1987) *Thinking Machines – The Evolution of Artificial Intelligence*. Oxford: Basil Blackwell.

Port, R. and van Gelder, T. (1995) *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press, Cambridge.

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

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

Searle, J (1980) "Minds, Brains and Programs", *The Behavioral and Brain Sciences* 3, 417-424.

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

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

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

Turing, A. M. (1936) "On Computable Numbers, with an Application to the Entscheidungsproblem". *Proceedings of the London Mathematical Society, Series 2* 42, p. 230 –265.

Turing, A.M. (1939) "Systems of Logic Based on Ordinals" *Proceedings of the London Mathematical Society* 45, 161-228.

Van Gelder, T. (1995) "What Might Cognition Be, If Not Computation?". *Journal of Philosophy* 91, 345-381.

Van Gelder, T. J. (1998) "The Dynamical Hypothesis in Cognitive Science", *The Behavioral and Brain Science*s, **21**, 615-665.

Williams, M.R. (1997) *A History of Computing Technology.* 2nd ed. IEEE Computer Society Press: Los Alamitos, CA.

Matthias Scheutz
mscheutz@cse.nd.edu

Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
USA