Defending the Semantic Conception of Computation 
in Cognitive Science

Gerard O’Brien

Philosophy, School of Humanities, University of Adelaide
gerard.obrien@adelaide.edu.au

Cognitive science is founded on the conjecture that natural intelligence can be explained in terms of computation. Yet, notoriously, there is no consensus among philosophers of cognitive science as to how computation should be characterised. While there are subtle differences between the various accounts of computation found in the literature, the largest fracture exists between those that unpack computation in semantic terms (and hence view computation as the processing of representations) and those, such as that defended by Chalmers (2011), that cleave towards a purely syntactic formulation (and hence view computation in terms of abstract functional organisation). It will be the main contention of this paper that this dispute arises because contemporary computer science is an amalgam of two different historical traditions, each of which has developed its own proprietary conception of computation. Once these historical trajectories have been properly delineated, and the motivations behind the associated conceptions of computation revealed, it becomes a little clearer which should form the foundation for cognitive science.

Key words: cognition, computation, representation, cognitive science, computer science

1. Discord at the Foundation of Cognitive Science

Cognitive Science is a discipline founded on the conjecture that natural intelligence can be explained in terms of computations implemented in the biological substrate of evolved creatures. The concept of computation, therefore, lies at the very foundation of cognitive science. Indeed, many theorists would argue that without computation, there is no cognitive (as

Journal of Cognitive Science 12: 381-399, 2011
©2011 Institute for Cognitive Science, Seoul National University
distinct from behavioural, psychological, biological, or just plain physical) science in the first place. And yet, notoriously, despite its foundation role, there is no consensus among philosophers of cognitive science as to how computation should be characterised in the first place.

While there are numerous subtle differences between the various accounts of computation found in the literature, the largest fracture lies between those that unpack computation in *semantic* terms and those that cleave towards a purely *syntactic* formulation. Contrast, for example, the account offered by Chalmers (2011) with that developed by Cantwell Smith (2002).\(^1\) What unites these two discussions is they seek to derive characterisations of computation that will do justice to its foundational role in cognitive science. What divides them is they come to diametrically opposed conclusions about the place of semantic properties. According to Chalmers, “a computation is simply an *abstract specification of causal organisation*” (2011, p.5). He then observes that “nothing in my account of computation and implementation invokes any semantic considerations, such as the representational content of internal states. This is precisely as it should be: computations are specified syntactically, not semantically” (p.7). According to Cantwell Smith, on the other hand, “the only compelling reason to suppose that we (or minds or intelligence) might be computers stems from the fact that we, too, deal with representations, symbols, meaning, information, and the like. For someone with cognitivist leanings, therefore, it is natural to expect that a comprehensive theory of computation will have to focus on its semantical aspects” (2002, p.31).

There are thus two very different conceptions of computation abroad in the philosophy of cognitive science: the semantic and the syntactic.\(^2\) According to the former, a computational process is one that in some way

---

\(^1\) Although Cantwell Smith’s chapter was not published until 2002, it was written and widely circulated in 1996. Chalmers’ article was written at roughly the same time.

\(^2\) This distinction is intended to capture the conceptual discord at the highest level of generality. The literature hereabouts contains more fine-grained distinctions on each side of this divide. In a recent survey article, for example, Piccinini (2011) distinguishes between *causal* and *mechanistic* accounts of computation, both of which he contrasts with the *semantic* account.
implicates information.\textsuperscript{3} Since information must be encoded for it to play a role, the semantic conception is normally formulated in terms of the processing of representations.\textsuperscript{4} The syntactic conception, on the other hand, eschews representation in favour of characterising computation in terms of the abstract functional organisation of the physical system in which it is implemented. Computations are understood as purely syntactic operations, as casual processes sensitive to certain abstract properties of the entities over which they are defined.\textsuperscript{5}

Both conceptions have advantages and disadvantages. The great merit of the semantic conception is that it offers a robust explanation of why the computational theory of mind represents a substantive doctrine in cognitive science. Computation is a process in which information gets to play a role, to throw its weight around. In doing so it can account for how intelligence is possible in a merely material world. The main drawback of this conception, of course, is that it is saddled with explaining just how information can play this role. How does the brain encode information, and moreover how does it do this in such a way that the information represented is casually efficacious?

It is the principal advantage of the syntactic conception that it does not seem to be as heavily burdened with this explanatory task. The proponent of this conception has the option of either invoking computation to explain representation or simply remaining sceptical about the role of representation in computation and hence cognition. But the syntactic conception is not without its own problems. Chief among these is the observation that

\textsuperscript{3} Furthermore, the notion of information here is taken to be stronger than the quantitative sense in which it is employed in communication theory (Shannon and Weaver, 1949).

\textsuperscript{4} See Piccinini (2011) for a more detailed exposition of the semantic conception of computation. This conception has been defended, for example, by Cummins & Schwarz (1991), Dietrich (1989), Fodor (1975), Pylyshyn (1984), O’Brien & Opie (2006), and Von Eckardt (1993).

\textsuperscript{5} Again see Piccinini (2011) for an exposition of (two different versions of) the syntactic conception of computation. This conception has been defended, for example, by Chalmers (2011), Churchland, Koch, & Sejnowski (1990), Piccinini (2008), and Stich (1983).
all physical systems have an abstract functional organisation, and hence implement some kind of computation. This motivates the charge that the characterisation of computation provided by the syntactic conception is so promiscuous that it threatens to trivialise the computational theory of mind, and with it the whole discipline of cognitive science.

The dispute between the semantic and syntactic conceptions of computation represents a yawning rupture at the conceptual foundations of cognitive science. It is a disagreement that has profound implications for the computational explanation of our cognitive capacities. The resolution of this matter is thus of upmost importance to the advancement of cognitive science. Yet it would seem that precious little progress has been made towards this goal over the many years since this disagreement became apparent.6

It would be naïve to think, therefore, that this matter can be resolved in relatively short compass. Nonetheless, it is the aim of this paper to take a few steps down this path. But rather than pitting these two conceptions directly against one another in an effort to determine which does more justice to the explanatory role of computation in cognitive science, I will pursue a different strategy. It will be the main contention of this paper that this dispute does not arise from different interpretations of the one phenomenon of computation as developed and deployed by computer science; it arises because contemporary computer science is actually an amalgam of two different historical traditions, each of which has developed its own proprietary conception of computation. Once these historical trajectories have been properly delineated, and the motivations behind the associated conceptions revealed, it will be a little clearer which conception should form the foundation for cognitive science.

2. Mathematics and the Syntactic Conception of Computation

Chalmers’ primary aim in the target article is to develop and defend an account of computation that can do justice to its foundational role in cognitive science. He begins his discussion by framing the issue in the following way:

---

6 See, for instance, the disparity of views presented in Sprevak (2010).
Defending the Semantic Conception of Computation in Cognitive Science

Why should computation, rather than some other technical notion, play this foundational role [in cognitive science]? … In order for the foundation to be stable, the notion of computation itself has to be clarified. The mathematical theory of computation in the abstract is well understood, but cognitive science and artificial intelligence ultimately deal with physical systems. A bridge between these systems and the abstract theory of computation is required. (2012, p.2)

It is the mathematical theory of computation, according to Chalmers, that provides the starting point for a discussion of the computational foundations of cognitive science. In a more recent article, Piccinini is even more explicit about the role this mathematical theory plays in the history of cognitive science:

Computationalism is the view that cognitive capacities have a computational explanation or, somewhat more strongly, that cognition is (a kind of) computation…. The view that thinking has something to do with computation may be found in the works of some modern materialists, such as Thomas Hobbes (Boden 2006, p.79). But computationalism properly so-called could not begin in earnest until a number of logicians (most notably Alonzo Church, Kurt Gödel, Stephen Kleene, Emil Post, and especially Alan Turing) laid the foundations for the mathematical theory of computation. (2011, pp.222-224)

Most contemporary philosophers of cognitive science would agree with Chalmers and Piccinini about the significance of the mathematical theory of computation. Much of the philosophical discussion about the computational nature of mind and cognition over the last 50 years has been conditioned by this theory, especially as incarnated in the operations of the fabled Turing machine. This is not surprising given that the mathematical theory of computation forms the basis of one of the two historical traditions that gave rise to modern computer science. But where did the mathematical theory of computation come from? What problem did it seek to address? And what kind of conception of computation does it entail?

Most expositions of the mathematical theory of computation start with the
formalist program proposed by the German mathematician David Hilbert in the early part of the 20th century. Hilbert, like many of his contemporaries, was deeply worried about the philosophical foundations of mathematics, especially in the face of the inconsistencies and paradoxes that had arisen in all the attempts hitherto to provide such a grounding. His proposal was to treat mathematics like a formal game in which all the pieces are manipulated according to precise rules. Such a formalization required a precise formal language (comprising a finite set of symbols) in which all mathematical statements could be written, together with a precise set of (syntactically-specified) rules for altering these statements. Under such a formalization, mathematics becomes an autonomous, self-contained system, governed by its own set of precisely-applied rules, and hence there is no need for intuition or subjective judgement to determine whether a particular mathematical statement is “true.” All that matters for mathematical truth is whether the symbol string that represents that statement can be generated in the formalism by a proper application of the syntactically-specified rules.

Hilbert’s proposal was that such a formalization would secure a firm foundation for mathematics. But he thought that completing this program required three additional tasks to be completed:

1. A proof that all true mathematical statements could be generated in the formalism (“completeness”);
2. A proof that no contradiction could be generated in the formalism (“consistency”); and
3. A proof that any mathematical statement could be shown to be true or false within the formalism (“decidability”).

Hilbert thought all three of these proofs would eventually be forthcoming. Unfortunately, as everyone knows, history showed quite the opposite. The first two nails in the coffin, were supplied by Kurt Gödel, when he demonstrated, first, that for any sufficiently powerful formalism there are true statements that cannot be generated by an application of its rules (i.e, it is incomplete), and second, that such a formalism was not powerful enough to prove its own consistency. The final nail in the coffin was hammered home by Alan Turing. And this is where this story gets more interesting, insofar as
the mathematical theory of computation is concerned.

The decidability of mathematics, as Hilbert and his contemporaries conceived of it, concerned the question of whether there was, at least in principle, a “definite method” for determining whether any mathematical statement was true, or at least (after Gödel) provable. The problem they faced was that their understanding of what such a method or procedure consisted in was intuitive, rather than rigorous. This is what Turing set out to change. Taking his cue from a characterisation of decidability provided by Max Newman in some lectures presented in 1935 at Cambridge University (in which Newman stated that the central question was whether there was a “mechanical process” which could determine whether a mathematical statement was provable), Turing set about conceiving of a machine that could perform this task. The famous result, of course, is the Turing machine, replete with its read/write head shuttling backwards and forwards along its infinite tape reading and writing symbols in a purely mechanical fashion (Turing 1936).7

The Turing machine, for Turing, represented the mechanisation of formal mathematics. It prescribed the limits of what mathematical operations were possible within the kind of formalism that Hilbert proposed. In doing so it provided a rigorous characterisation of what a “definite method” within this formalism consisted in. And the most important result for Turing, at least when he first developed this abstract machine, was that it demonstrated that there were mathematical statements that could not be generated by a Turing machine (e.g., statements about irrational numbers) and hence were not provable by a “definite method” within any formalism. The third proof required for Hilbert’s program, like the first two, would never be forthcoming.

Hilbert’s formalist program in mathematics may have been well and truly dead, but the Turing machine and what it symbolised took on a life of its own. It has become rightly immortalised in the history of computer science, not because it was the means by which the ailing body of Hilbert’s program was put out of its misery, but because it represented a bridge between

---

7 See Hodges (1983, pp.96-110) for a fascinating discussion of the background behind Turing’s development of the Turing machine.
the abstract world of mathematics and the physical world of machines. In showing that mathematical operations can be mechanised, Turing was showing how *computation* could be performed by a physical machine.

Turing explicitly modelled the Turing machine on the kinds of arithmetical operations that we perform when, in accordance with Hilbert’s formalist picture, we manipulate symbols according to syntactically-specified rules. It was natural, therefore, to interpret Turing’s work as providing a rigorous characterisation of the notion of computation. Computation could now be unpacked in terms of the operations of a Turing machine. Furthermore, since the Turing machine prescribes the limits of what kinds of mathematical operations are possible in a formalism, the Turing machine was widely interpreted as prescribing the limits of computation. And this result in turn became enshrined in the Church-Turing thesis: the claim that any “well-defined” computation can be performed by a Turing machine.8

Given its role in forming the basis for the mathematical theory of computation, it is not surprising that theorists seek to define this in terms of the operations of a Turing machine. And, furthermore, given that the Turing machine forms a bridge between the abstract world of mathematics and the physical world of machines, it is not surprising that many theorists think it is appropriate to employ the mathematical theory of computation when seeking a computational foundation of cognitive science. The final element of this story, therefore, concerns how the mathematical theory of computation transmutes into the syntactic conception of computation that

---

8 Given its derivation, the perceived connection between Turing machines and the limits of computation needs to be carefully qualified. The Turing machine was conceived of by Turing as a means of mechanising *formal* mathematics—the kind of mathematics that is performed by manipulating symbols according to syntactically-specified rules. Consequently, the Turing machine prescribes the limits of mathematical operations within a formalism. It certainly does not prescribe the limits of mathematical operations more generally (something that formed the basis of Turing’s demonstration that Hilbert’s final proof was impossible), and it certainly does not prescribe the limits of computation, where the latter is taken to have a more liberal definition than that of syntactic operations over symbols (more about which below). See, for example, Copeland (1997).
was described in the introductory section.

The key here is the program in mathematics from which it issued—Hilbert’s formalist program. The whole point of this program was to reconceptualise mathematics as discipline that was self-contained and hence autonomous, both of human thought (as intuitionists claimed) and the actual world (as Platonists thought). According to formalism, mathematical statements are treated as syntactic objects which are manipulable according to rules. That such statements can be given an interpretation in terms of numbers, set, points, lines and so forth is, strictly speaking, extraneous to mathematics. As Hilbert was wont to say, such symbols could be interpreted as being about “tables, chairs and beer-mugs” so long as the formalism made sense in itself (Hodges 1983, p.82). In short, formalism takes the semantics out of mathematics, and reconceptualises it as a syntactic game.

The mathematical theory of computation, and especially its unpacking in terms of Turing machines, has formalist fingerprints all over it. It is this tradition in mathematics, for example, that motivates Chalmers’ dismissal of semantic considerations in any characterisation of computation:

The original account of Turing machines by Turing (1936) certainly had no semantic constraints built in. A Turing machine is defined purely in terms of the mechanisms involved, that is, in terms of syntactic patterns and the way they are transformed…. To implement a Turing machine, we need only ensure that this formal structure is reflected in the causal structure of the implementation…. [W]hen computer designers ensure that their machines implement the programs that they are supposed to, they do this by ensuring that the mechanisms have the right causal organization; they are not concerned with semantic content. In the words of Haugeland (1985), if you take care of the syntax, the semantics will take care of itself. (2011, p.7)

It is formalism, therefore, that begets the mathematical theory of computa-

---

9 In fact, Hilbert himself was not a strict formalist, in that he thought there was a way of understanding the meaning and truth of mathematical statements that took one beyond the formalism.
tion, which in turn begets the syntactic conception of computation. Just as mathematics is to be bleached of semantics, so too is computation. And computation without semantics is purely syntactic.

The story about the evolution of the syntactic conception of computation is now complete. It is embodied in one of the two historical traditions that gave rise to contemporary computer science. And once this conception is applied to cognition, the output is a certain way of construing the computational theory of mind. But as we noted in the introductory section, there is a quite different way of unpacking computation in the philosophical literature. Is the semantic conception just another way of interpreting the mathematical theory of computation? Is it just a different way of understanding what Turing achieved? A number of the proponents of the semantic conception would appear to take this line.10 In my view, however, we must look in a completely different direction for the primary source of this alternative conception of computation.

10 Fodor, for example, is fond of remarking that Turing came up with the only important idea about how the mind works that anybody has ever had:

[Given the methodological commitment to materialism, the question arises, how a machine could be rational?...Forty years or so ago, the great logician Alan Turing proposed an answer to this question...Turing noticed that it isn’t strictly true that states of mind are the only semantically evaluable material things. The other kind of material thing that is semantically evaluable is symbols... Having noticed this parallelism between thoughts and symbols, Turing went on to have the following perfectly stunning idea. “I’ll bet”, Turing (more or less) said, “that one could build a symbol manipulating machine whose changes of state are driven by the material properties of the symbols on which they operate (for example, by their weight, or their shape, or their electrical conductivity). And I’ll bet one could so arrange things that these state changes are rational in the sense that, given a true symbol to play with, the machine will reliably covert it into other symbols that also true. (1992, p.6)

Fodor’s description of the operation of the Turing machine, unlike Chalmers’, is not independent of semantic considerations. But Fodor’s construal, given the historical tradition in which Turing was working, is perhaps a little tendentious. It is Turing seen through the spectacles of the semantic conception of computation.
3. Engineering and the Semantic Conception of Computation

The development of the mathematical theory of computation over the early years of the 20th century is one of the historical traditions that gave rise to contemporary computer science. But it is arguable that even more important to the evolution of modern computer science is the long history of technological innovation that enabled the engineering of increasingly sophisticated computing devices.

The engineering tradition of computer science stretches back many hundreds of years. It centres on the construction of what Sloman calls “control mechanisms”—machines designed to regulate physical processes of various kinds:

Physical control mechanisms go back many centuries, and include many kinds of devices, including clocks, musical-boxes, piano-roll mechanisms, steam engine governors, weaving machines, sorting machines, printing machines, toys of various kinds, and many kinds of machines used in automated and semi-automated assembly plants. The need to control the weaving of cloth, especially the need to produce a machine that could weave cloth with different patterns at different times, was one of the major driving forces for the development of such machines. Looms, like calculators and clocks, go back thousands of years and were apparently invented several times over in different cultures. (Sloman, 2002, p.89)

Control mechanisms enabled the automation of all manner of tasks that hitherto could be performed only by humans. And the key to their success was that rather than merely supplying energy to drive certain physical operations, they also used information to shape the manner in which those physical operations unfolded. The construction of devices driven by information thus marks an important juncture in the history of technology. Yet, as Sloman observes, the significance of this development has not always been sufficiently appreciated:
Throughout the history of technology we can see (at least) two requirements for the operation of machines: energy and information. When a machine operates, it needs *energy* to enable it to create, change or preserve motion, or to produce, change or preserve other physical states of the objects on which it operates. It also needs *information* to determine which changes to produce, or which states to maintain. Major steps in the development of machines concerned different ways of providing either energy or information.

The idea of an energy requirement is very old and well understood. The idea of an information requirement is more subtle and less well understood. I am here not referring to information in the mathematical sense (of Shannon and Weaver) but to an older and more intuitive notion of information which could be called *control information* since information is generally useful in constraining what is done. (Sloman 2002, pp.93-4)

A simple example of the distinction between energy and information at work is the manner in which the (now philosophically familiar) Watt governor maintains a constant speed of a flywheel driven by a steam engine. In this case, it is the steam engine that supplies the energy that powers the motion of the flywheel. But it is the Watt governor (the spindle mechanism attached to the flywheel which continuously adjusts the steam pressure) that employs information about the rate at which the flywheel is spinning to maintain a constant speed.

This example is bound to be controversial in some quarters, however. In order for information to be used by a control mechanism, it must be *represented* in some way. And Van Gelder (1995, pp. 351-4) famously argues that the Watt governor is a dynamical, rather than computational, system principally on the grounds that it does not implicate representation in any meaningful way. But in more recent years a number of philosophers have challenged this contention. Bechtel (1998, 2011), for example, argues that information about the rate at which the flywheel is spinning is encoded by the angular displacement of spindle’s two arms, and that it is precisely this represented information that is used by the governor to regulate the pressure generated by the steam engine (see also Nielson 2010). The point
Defending the Semantic Conception of Computation in Cognitive Science

here, of course, is that in the absence of a universally accepted theory of representation, there will be philosophical disagreements about how and what information is represented by devices of this kind (and, indeed, whether it is even legitimate to speak of them representing information in the first place). But this philosophical uncertainty has not retarded the progress of control system engineering, the practitioners of which routinely invoke the concept of information to explain how their devices work.

Once the trick of harnessing information to control physical processes had been mastered, it was a relatively small step to appropriate these same technological innovations to engineer devices in which the regulated physical processes implemented abstract operations (such as arithmetical calculations). And this engineering transition from the physical to the abstract occurred long before Turing started to ruminate about machines that could manipulate symbols. A particularly vivid example of this occurred with the commandeering of the control mechanism employed in looms. At the beginning of the 19th century Jacquard designed a loom whose specific weaving patterns could be controlled by “instructions” encoded in the form of punched cards. The huge advantage of such a device was that its operator could generate different patterns without having to physically alter the actual weaving machinery. By the end of the 19th century, Hollerith used essentially the same technology to construct a “tabulating machine” capable of analysing census data. Instead of encoding information about different weaving patterns, Hollerith used the punch cards to encode numerical information about specific individuals. The machine then automated the laborious chore of compiling statistical information about the population as a whole. In doing so, it reduced a task that normally took years to just a few months.\footnote{Hollerith went on to develop a number of similar devices, adding innovations such as automatic card-feed mechanisms and key punches. He even constructed a programmable version of his tabulating machine by incorporating a modifiable wiring panel that enabled different arithmetical operations to be performed, depending on the task domain.}

It was the efflorescence in the late 19th century of control mechanisms constructed to implement abstract operations that led to the appropriation
of the term “computer” to refer to machines that automated the mundane business of calculating. And with this development the computing industry had well and truly come of age. But it is important to bear in mind that this transition from the physical to the abstract marks a change in the tasks performed by control mechanisms, not a change in their basic operating principles. All control mechanisms employ information to regulate physical processes, whether or not these physical processes are then appropriated to implement abstract operations.

From the perspective of the engineering tradition, therefore, computers are control mechanisms: devices that regulate the behaviour of physical processes. Control mechanisms are special, are set apart from other physical machines, because rather than merely supplying energy they represent and use information to shape the manner in which physical operations unfold. This explains why talk of the representation and processing of information is ubiquitous in computer science. It also explains the primary source of the semantic conception of computation. This conception is embodied in the long history of engineering practice that has seen the construction of progressively more sophisticated physical devices whose behaviour is controlled by information.

4. Minds as Sensorimotor Control Mechanisms

Modern computer science is an amalgam of two different historical traditions—a theoretical tradition that emerged from mathematics and a more practical tradition anchored in the evolution of control system engineering.

---

12 There is an oft repeated canard in the literature that prior to Turing’s development of his abstract machine in 1936, the term “computer” referred only to a person who performed mathematical calculations. However, according to the Oxford English Dictionary (Second Edition), while the use of this term to denote a person who does calculations does have a long history (dating back to 1613), from 1869 it was also used to refer to mechanical calculating devices.

13 For example, in 1896 Hollerith established the Tabulating Machine Company, which eventually merged with three other companies in 1911 to form the Computing Tabulating Recording Corporation, which in turn was renamed International Business Machines Corporation (IBM) in 1924.
technology. Paralleling the distinction between these two traditions is one between two different conceptions of computation—the syntactic and the semantic. Contemporary computer science has been enriched by the fusion of these two traditions. For example, even though the engineering tradition significantly antedates the mathematical tradition, the latter has facilitated a number of significant technological innovations. In this sense, therefore, it is neither appropriate nor fruitful to adjudicate as to which of these traditions embodies the “correct” conception of computation as far as computer science is concerned. Nonetheless, it is appropriate and fruitful to consider which of these traditions is better placed to capture the role that computation plays in cognitive science.

The birth of cognitive science (and its technology-focused fellow traveller, the field of artificial intelligence) is often identified with the famous Dartmouth conference of 1956. Whether or not such a specific dating is possible, what was ultimately responsible for establishing the field was a series of theoretical and technological achievements during the 1950s and 1960s. Among the former was the recognition that the behaviourist program that had so dominated psychology in the first half of the twentieth century was bankrupt; that it was impossible to explain intelligent behaviour without recourse to the “cognitive contribution” made by the neural mechanisms connecting stimuli with responses. Chief among the latter was the rapid development of computer technology, which was enabling the construction of machines displaying increasingly advanced forms of intelligence. In this climate it was natural to bring these theoretical and technological achievements together and hypothesise that natural intelligence, just like its artificial counterpart, was the product of computation, in this case implemented in the neural hardware of the brain.

14 Turing’s achievements represent an example of the fruitful intersection of these two traditions. While he is mainly remembered as the brilliant mathematician who devised the abstract Turing machine, he was also a talented engineer who made a number of important contributions to the design of working electronic computers (Hodges, 1983).

15 See, for example, Copeland (1993, p.8).

16 Chomsky’s (1959) lacerating critique of Skinner’s Verbal Behavior played a pivotal role in convincing theorists of this point.
And with this conjecture the field of cognitive science was up and running. The key point in this sociology of cognitive science is the connection between computation and intelligent behaviour. This discipline was founded on the conviction that computation is the solution to the problem of how intelligence evolved in the material world. In this context it is not surprising that many philosophers turned to the mathematical tradition of computer science for guidance in unpacking the computational character of cognition. Mathematical competence is the acme of intelligent behaviour; and the kind of abstract symbolic operations on which such competence is based seem ripe for explaining more general cognitive abilities, such as those associated with language use and rational deliberation. These high-level cognitive skills were the focus of cognitive science (and AI) in its early years, and the Turing machine was the model of mechanised rationality. Small wonder, then, that the syntactic conception of computation took root in certain parts of the field and conditioned philosophical discussion about the computational theory of mind.

But what got lost in the early years of cognitive science, in the understandable enthusiasm and optimism with which theorists embraced the computational paradigm, was the recognition that while we often associate intelligence with such high-level cognitive achievements as language use and rational deliberation, natural intelligence at base consists in the capacity of biological organisms to successfully navigate their environments—to behave in ways appropriate to the ambient conditions that obtain, given their goals of obtaining nutrients, avoiding predators, and finding mates. Natural intelligence, in other words, is grounded in the capacity for sensorimotor control: the capacity to gather information about the environment and use it to drive appropriate behaviour. Even the remarkable cognitive capacities found in human minds are anchored in neural mechanisms originally designed by evolution to harness information for the purpose of regulating behavioural responses. And the overarching goal of cognitive science is to invoke computation to explain how this neural machinery accomplishes this task.

From this naturalistic perspective, it seems clear that cognitive science is more closely affiliated with the engineering tradition of computer science than with the theoretical tradition that emerged from mathematics. The
engineering tradition, as we saw in the last section, has sought to design and construct control mechanisms—devices that regulate the behaviour of various kinds of physical processes. Cognitive science, as we have just seen, seeks to “reverse engineer” the sensorimotor control mechanisms housed in nervous systems—devices designed by evolution to regulate the behaviour of biological agents. Given this common focus on control system engineering, it is not surprising that these two endeavours share an increasingly fruitful collaborative relationship, whereby practitioners in each look to the other for ideas and inspiration. More important for our purposes, however, is that this common focus suggests it is the conception of computation embodied in the engineering tradition, not its mathematical counterpart, which is of more relevance for understanding the character of the cognitive processes that issue in the intelligent behaviour of biological creatures. At the core of the engineering tradition is the idea that control mechanisms operate by representing and processing information. It seems reasonable to propose, therefore, that this semantic conception of computation should also form the foundation of cognitive science.

Human minds are Nature’s most sophisticated solution to the problem of sensorimotor control. They represent an astonishing achievement of hardware and software engineering. They are the product of an incremental, trial and error design process over millions of years. In the course of this natural tinkering, they have undergone a similar transition from the physical to the abstract that occurred over the history of the engineering tradition in computer science. Control mechanisms originally designed by evolution to gather information from the environment and drive bodily behaviour, were appropriated and modified to regulate abstract operations. In addition to sensing and responding to the world around us, these mechanisms enable us to engage in all manner of abstract thought and deliberation. They even allow us to engage in mathematics. But it would be a mistake to model a conception of computation on the abstract operations that implement these highest-level cognitive achievements. Instead, our cognitive machinery is best characterised in terms of an account of computation that applies to control mechanisms quite generally.

This conclusion entails that the philosophy of cognitive science cannot avoid the explanatory burden briefly adumbrated in the introductory
section—that of explaining how information is represented by the brain, and how the information so represented is casually efficacious. This, of course, is the familiar project of naturalising mental representation. This project has become a little unfashionable in recent years. But if the argument developed in this paper is on the right track, this explanatory task is of the utmost importance to shoring up the computational foundations of cognitive science. It is time to bring mental representation back from the philosophical wilderness and once again place it on centre stage.

References

Bechtel, W. 1998. Representations and cognitive explanations: Assessing the dynamicist’s challenge in cognitive science. *Cognitive science* 22, 295-318.
Bechtel, W. 2011. Representing time of day in circadian clocks. In A. Newen, A. Bartels, & E. Jung (Eds.), *Knowledge and representation* (pp.129-62). Stanford: CSLI Publications.
Boden, M. 2006. *Mind as machine: A history of cognitive science*. Oxford: Clarendon Press.
Cantwell Smith, B. 2002. The foundations of computing. In M. Scheutz (Ed.) *Computationalism: New directions* (pp.23-58). Cambridge: MIT Press.
Chalmers, D. J. 2011. A computational foundation for the study of cognition. *Journal of Cognitive Science* 12(4), 1-21.
Chomsky, N. 1959. Review of Verbal Behavior by B.F. Skinner. *Language* 35, 26-58.
Churchland, P.S., Koch, C., & Sejnowski, T. 1990. What is computational neuroscience? In E. Schwartz (Ed.), *Computational neuroscience*. Cambridge: MIT Press.
Copeland, J. 1997. The broad conception of computation. *American Behavioral Scientist* 40, 690-716.
Cummins, R. & Schwarz, G. 1991. Connectionism, computation and cognition. In T. Horgan & J. Tienson (Eds.), *Connectionism and the philosophy of mind*. Dordrecht: Kluwer.
Dietrich, E. 1989. Semantics and the computational paradigm in cognitive psychology. *Synthese* 79, 119-141.
Fodor, J. 1975. *The language of thought*. New York: Harvester Press.
Fodor, J. 1992. The big idea: Can there be a science of the mind? *Times Literary Supplement* *Supplement July* 3, 5-7.
Haugeland, J. 1985. *Artificial intelligence: The very idea*. Cambridge: MIT Press.
Defending the Semantic Conception of Computation in Cognitive Science

Hodges, A. 1983. *The enigma of intelligence*. London: Unwin.

Nielsen, K.S. 2010. Representation and dynamics. *Philosophical Psychology* 23, 759-73.

O’Brien, G. & Opie, J. 2006. How do connectionist networks compute? *Cognitive Processing* 7, 30-41.

Piccinini, G. 2008. Computation without representation. *Philosophical Studies* 137, 205-41.

Piccinini, G. 2011. Computationalism. In E. Margolis, R. Samuels, & S. Stich (Eds.), *Oxford handbook of philosophy of cognitive science* (pp.222-224). New York: Oxford University Press.

Pylyshyn, Z. 1984. *Computation and cognition: Toward a foundation for cognitive science*. Cambridge: MIT Press.

Sloman, A. 2002. The irrelevance of Turing machines to AI. In M. Scheutz (Ed.), *Computationalism: new directions* (pp.87-128). Cambridge: MIT Press.

Shannon, C. & Weaver, W. 1949. *The mathematical theory of communication*. Urbana: University of Illinois Press.

Sprevak, M. (Ed.) 2010. *Computation and cognitive science*. Studies in History and Philosophy of Science 41, 223-334.

Stich, S. 1983. *From folk psychology to cognitive science: The case against belief*. Cambridge: MIT Press.

Turing, A. 1936. On computable numbers, with an application to the entscheidungs problem. *Proceedings of the London Mathematical Society* (Series 242), 230-65.

Van Gelder, T. 1995. What might cognition be, if not computation? *Journal of Philosophy* 92, 345-81.

Von Eckardt, B. 1993. *What is cognitive science?* Cambridge: MIT Press.
