Representation in Analog Computation

Gerard O'Brien and Jon Opie
Philosophy
School of Humanities
University of Adelaide
South Australia 5005

Final Draft @ May 2010. Forthcoming in Newen, A., Bartels, A. & Jung, E. (eds.) *Knowledge and Representation*, CSLI Publications

Not so long ago there was something of a consensus regarding the conceptual foundations of cognitive science. Participants in this multidisciplinary enterprise were united in the conviction that natural intelligence is to be explained in *computational* terms. Furthermore, since "there is no computation without representation" (Fodor 1975, p.34), it seemed obvious that cognitive explanation requires *mental representation*. Computation and representation were the twin foundation stones on which the whole business was based (see, e.g., Thagard 2008).

These days things are very different. Representation is under attack from several quarters. There have always been sceptics about the prospects of naturalizing the mysterious "aboutness" of mental states, but attention has recently turned to concerns about the explanatory role of mental representation in cognitive science. Two lines of argument stand out. The first stems from the worry that semantic properties are, by nature, incapable of shaping cognitive processes. It was precisely this worry that prompted Stephen Stich to advocate replacing the computational theory of mind with the *syntactic* theory of mind (1983). The second derives from the perceived failure of the computational theory of mind to deliver on its promise of explaining intelligence, and correlatively, the failure of traditional approaches in AI to construct deeply intelligent systems. Some theorists regard this problem as so severe that cognitive science should abandon its flirtation with *representation*: "Representation is the wrong unit of abstraction in building the bulkiest parts of intelligent systems" (Brooks 1991, p.140).

These arguments have begun to marginalise representation in cognitive science. There is talk of a Kuhnian-style paradigm shift in which the computational theory of mind will give way to *dynamical systems theory*, or some other approach that emphasises the *embodied* and *embedded* nature of cognition.[1] What unites these positions is scepticism about explaining intelligence in computational terms. Instead, cognition is conceived as a coupling between organism and environment, such that the world acts as its own representation (e.g., Brooks 1991, O'Regan & Noë 2001). This is an extraordinary situation, given the pivotal role of the concepts of representation and computation in the development of cognitive science.

We think this anti-representationalist movement marks a wrong turn in cognitive science. Without representation, cognitive science is bereft of its principal tool for explaining natural intelligence. It is thus incumbent on admirers of cognitive science to rescue representation from its threatened extinction. This chapter is written with that aim in mind.

Our diagnosis is that anti-representationalism has its roots in an overly restrictive take on mental representation that arose in the 1960s and 1970s. It was during this period that digital computers superseded analog computers, and in the process profoundly influenced our thinking about the mind. We will examine how digital computation has shaped our understanding of mental representation, and consider what cognitive science might have looked like had it arisen in the 1930s and 1940s, during the heyday of analog computation. Specifically, we will flesh out an account of representation based on careful analysis of the *Differential Analyzer*: the world's first general-purpose computer, and an analog device to boot. Our account undermines anti-representationalism, which turns out to be an attack on digital forms of representation, rather than representation *tout court*.

[1] See, e.g., Beer 1995; Brooks 1991; Clark 1997a, 1997b; Keijzer 2002; Port & Van Gelder 1995; Van Gelder 1995; Wallace et al 2007; Wheeler 2005.

1. Revisiting the Foundations of Cognitive Science

The birth of cognitive science (and its technology-focused fellow traveller, the field of artificial intelligence) is often identified with the Dartmouth conference of 1956.[2] Whether or not such a specific dating is possible, what was ultimately responsible for establishing the field was a series of intellectual and technical achievements during the 1950s and 1960s. Noam Chomsky's (1959) lacerating critique of B. F. Skinner's *Verbal Behavior* is among the former, because it sounded the death knell for the behaviourism that had dominated psychology in the first half of the twentieth century. And chief among the latter was the rapid development of the electronic digital computer. A number of important innovations took place at this time, including the development of the von Neumann, stored-program architecture. But the efflorescence of digital computers was largely the result of a technical breakthrough—the introduction of the transistor as the basic computing element. As a result, digital computers were immediately smaller, faster, cheaper to produce, and more reliable.

So when cognitive scientists first proposed that cognition is computation, it's no surprise they turned to the theory of digital computers to flesh out this conjecture. To solve a problem by digital means one first develops a *formal description* of the problem domain, and then *physically implements* that description. A formal description comprises a set of symbols that represent the elements of the target domain, together with syntactic rules that describe the behaviour of those elements. A physical implementation of a formal description consists of a device whose internal states realize the symbols in the description, and whose state-transitions satisfy the syntactic rules.[3]

When cognitive science looked to computer science for a model of cognition, it was this familiar picture of rule-governed symbol manipulation that emerged. And despite various developments in cognitive science since that time, including the appearance of connectionism in the 1980s, this conception of cognition has stuck.

However, the history of computer science has *two* strands: digital and analog. Digital computing has its origin in the methods of arithmetic. The first digital computers were devices such as the abacus, which appeared in Babylonia around 300 BC. Analog computation originated with *non-symbolic* graphical and geometric methods. The earliest known analog computer — the Antikythera mechanism, used to calculate astronomical positions — was built in about 100 BC (Bromley 1990). During the 1930s and 1940s, computer science was dominated by mechanical analog computers. And contrary to the popular view that analog computers are all "special-purpose" devices, the first general-purpose, automatic computer was actually an analog machine: the Differential Analyzer developed by Vannevar Bush at MIT in the 1930s.

The rapid advance of electronic technology during the Second World War enabled digital computers to perform numerical calculations at speeds sufficient to rival analog devices. But it wasn't until the 1960s, with the dramatic reduction in costs that accompanied the development of transistorised circuits, that digital computers largely supplanted analog devices. Indeed, prior to this time, analog computers were often preferred, particularly in applications where accurate real-time calculations were required (Small 1993).

Given that the emergence of digital computation is best explained as a result of a technical innovation, rather than a conceptual revolution, it is worth considering what cognitive science might have looked like had it been founded on an analog conception of cognition. In particular, what account of representation might have emerged from the analog computational framework? In the next section we will start to explore this question by revisiting the distinction between analog and digital computation.

2. Distinguishing between Digital and Analog Computation

The relationship between digital and analog computation is subject to some confusion. It's often claimed that the essential difference between the two is that digital computers use *discrete* variables to represent their targets (e.g., high and low voltage states), whereas analog computers use *continuous* variables. This way of dividing things up appears to be based on the view that a physical system performs a computation just in case its operation can be interpreted as implementing some function. Churchland et al express the idea as follows:

---

[2] See, e.g., Copeland 1993, p.8.

[3] Haugeland 1985 is the classic philosophical account of formal descriptions and their physical realization.

> [W]e can consider a physical system as a computational system just in case there is an appropriate (revealing) mapping between some algorithm and associated physical variables. More exactly, a physical system computes a function f(x) when there is (1) a mapping between the system's physical inputs and x, (2) a mapping between the system's physical outputs and y, such that (3) f(x) = y. (1993, p.48)

This view of computation suggests the following way of distinguishing between digital and analog computers: a physical system is a digital computer if its state variables map onto a discrete function, an analog computer if its state variables map onto a continuous function. Despite the tidiness of this scheme, and a certain degree of acceptance within the computer science community, we believe this way of proceeding is deeply mistaken. It fails on two grounds.

First, there is good reason to reject the idea that computation is merely a matter of implementing a function. Since all law-governed physical systems can be interpreted as implementing some function or other, this view leads to the conclusion that *all* physical systems are computational. But the concept of computation was originally introduced into cognitive science as a way of distinguishing two classes of causal processes: those characteristic of systems (such as ovens, cups of tea, and cyclones) that show no signs of intelligence, and those exhibited by intelligent systems alone. Computational processes are supposed to be *special* in some way—in a way, moreover, that provides us with some explanatory purchase on the problem of intelligent behaviour. Implementing a function is a ubiquitous feature of nature, so characterizing computation in this way undermines the motivation for introducing the concept in the first place.[4]

Secondly, recourse to the divide between discrete and continuous variables is at odds with the way computer scientists themselves have generally drawn the analog/digital distinction. Consider, for example, the familiar tactic of representing a physical variable as a curve on the plane. If we plot the velocity of a moving object against time, it is possible to compute the distance it travels by measuring the area under the curve, or its acceleration at some instant by constructing a tangent to the curve (Figure 1).

Figure 1 about here.

These computations employ an *analog* representing vehicle, a 2-d curve plotted against a pair of linear axes. What makes this representation analog is the existence of a relation-preserving mapping between the curve and its target. Velocity is represented as the projection of the curve onto the y-axis, such that relations among velocities correspond to relations among those points. If the velocity at some time t_3 is greater than the velocity at t_1, then its representative point v_3 will be further along the vertical axis than v_1; if the velocity at t_2 is mid-way between the first two velocities, then v_2 will lie between v_1 and v_3, and so on (Figure 1). In other words, there is a simple *physical analogy* between the curve and the variable it represents. Such analogies are responsible for the effectiveness of both special-purpose analog devices, such as scale models, and general purpose analog computers such as the Differential Analyzer (as we will demonstrate).

Classic texts in engineering and computer science such as *Analog Computation* (Jackson 1960), *Basics of Analog Computers* (Truit & Rogers 1960) and *Analog Methods in Computation and Simulation* (Soroka 1954) make precisely this point.

> Devices that rely…on the analogous relationships that subsist between the physical quantities associated with a computer and the quantities associated with a problem under study are called *analog* computers. (Jackson 1960, p.1)
>
> All [analog computers] have one characteristic in common — that the components of each computer…are assembled to permit the computer to perform as a model, or in a manner analogous to some other physical system. (Truit & Rogers 1960, p.3)
>
> The term *analog* means similarity of properties or relations without identity. When analogous systems are found to exist, measurements or other observations made on one of these systems may be used to predict the behaviour of the others. (Soroka 1954, p.v)

[4] For further discussion see O'Brien & Opie 2006. We there suggest an alternative characterization of computation that avoids this criticism.

What these authors are suggesting is that an analog computer is a device designed to exploit an analogy between the physical properties of a system of representing vehicles and some target system. Digital computers don't work that way, instead deploying symbolic vehicles whose physical properties stand in an arbitrary relationship to the objects they represent.[5] The distinction between continuous and discrete variables is not fundamental to the relationship between analog and digital computation, because analog representation need not involve continuous variables. Modern analog clocks, for example, represent time via the angular position of hands on a dial. But such clocks typically represent time to a resolution of no better than a second, because their hands move in *discrete* steps around the dial.[6]

3. The Differential Analyzer

Most early analog computers were special-purpose devices. They were designed to perform computations related to a particular problem or system. For example, during the 1870s Lord Kelvin invented an analog device that determined the components of tidal variation by performing a Fourier analysis of tide height records. Kelvin's *Harmonic Analyzer* used a system of disk-and-ball integrators developed by his brother James Thomson (Bromley 1990, pp.172-4). In the 1910s and 1920s, Hannibal Ford produced a number of very accurate analog computers for determining the range of moving targets. These were used extensively in ship-to-ship naval gunnery and later for anti-aircraft fire control (Clymer 1993). But these sophisticated mechanical computers were designed with a narrow range of applications in mind. Scientists and engineers, who sought to develop mathematical models unhindered by the limits of analytic techniques, were sorely in need of general-purpose computers.

In 1876, Kelvin discovered that by creating a feedback connection between two integrators he could solve linear second-order differential equations. He also saw how to generalize the feedback principle to differential equations of any order, but technical difficulties prevented him from constructing a working model (Bromley 1990, pp.174-7). Fifty years later, Vannevar Bush and his students independently developed an electromechanical device capable of solving second-order equations. Again, the original impetus had been to solve equations connected with specific physical systems[7], but the result was a general-purpose device. Like Kelvin, Bush was aware of the need to generalize to higher-order systems of equations. The key to Bush's success was the introduction of mechanical torque amplifiers (a suggestion of his student Harold Hazen) and highly accurate disk-and-wheel integrators. Torque amplifiers permit a series of integrators to drive one another despite their inherently limited output — the very problem that had hindered Kelvin's work (Owens 1986, pp.66-70). With these innovations in place, Bush and his colleagues were able to build the first *Differential Analyzer*, a mechanical analog computer capable of quickly and accurately solving differential equations of any desired order.

Figure 2 about here.

Bush's computer consisted of a long table criss-crossed by a series of rotating shafts (Figure 2). On one side of the table were six disk-and-wheel integrators interlinked by torque amplifiers. On the opposite side were a set of boards, some used to graph output variables, others to generate input via an operator who traced a graph of the input function, thereby governing the rotation of a shaft. Each shaft was associated with a variable, and by using geared connections between them it was possible to add, subtract, multiply, divide and integrate variables.

Figure 3 about here.

The heart of a Differential Analyzer is the disk-and-wheel integrator. This device takes any variable as input and produces its integral as output. It consists of a metal disk, and a small wheel oriented at right angles to the disk such that rotation of the disk causes the wheel to rotate. The wheel is also free to move along the radius of the disk, its position determined by the lateral movement of a connecting shaft (Figure 3). Suppose the disk turns through the angle Δx. If the radius of the wheel is r and its distance from the centre of the disk is f, then:

[5] This implies that symbols don't carry their meaning intrinsically, but acquire meaning by virtue of how they are manipulated. For example, numerals represent numbers not because of their physical form, but because we operate on them according to syntactic rules that support a numerical interpretation. See Section 4 for further details.

[6] See Lewis (1971) for further examples of discrete analog representation.

[7] Among other things, Bush and his students investigated long-distance transmission lines and vacuum-tube circuits. See Owens 1986, pp.66-70, for further details.

$$f\Delta x = r\Delta y$$

This expresses the fact that the length of the path traced out on the disk by the rotating wheel must equal the arc on the wheel itself. In the Differential Analyzer the rotation of the disk indirectly governs the lateral movement of the wheel (see below for details), so f is actually a function of x. Rearranging and taking the limit for small Δx we get:

$$f = r\frac{dy}{dx}$$

That is, f is proportional to the derivative of y with respect to x. For a unit wheel, this implies:

$$y(t) - y(0) = \int_0^t f(x)dx$$

In other words, the angular displacement Δy of the wheel is the integral of the function that describes the linear motion of the upper shaft. Depending on one's interests one might therefore describe the disk-and-wheel system as a *model* of the integration process, or a *means of calculating* the integral of a function.

To illustrate how a Differential Analyzer works, consider the problem of modelling an object in free fall near the surface of the earth. Such an object is subject to two forces: the force of gravity, and an opposing force caused by air resistance which is proportional to the object's velocity. The acceleration of the object is given by:

$$\frac{d^2y}{dt^2} = g - k\frac{dy}{dt} \qquad \text{Equation 1}$$

Here g is the acceleration due to gravity and k is a drag coefficient. To solve Equation 1, the Differential Analyzer is set up as shown in Figure 4. Each shaft represents a variable or constant in the equation. The first lengthwise shaft is driven at a constant rate, so its angular position is a suitable representation of time. It acts like a metronome, tapping out a beat that is followed by the rest of the system. The next three shafts represent acceleration d^2y/dt^2, velocity dy/dt, and position y, respectively. Both integrators are driven by the time shaft. The acceleration shaft (shaft 2) determines the linear motion of the wheel on the first integrator, so its output is:

$$\frac{dy}{dt} = \int\frac{d^2y}{dt^2}dt \qquad \text{Equation 2}$$

The velocity shaft (shaft 3) determines the linear motion of the wheel on the second integrator. Its output is position, because:

$$y = \int\frac{dy}{dt}dt \qquad \text{Equation 3}$$

The last two shafts (5 and 6) represent the product of velocity and the drag coefficient, and the acceleration due to gravity, respectively. They are added by a crucial set of connections which provide input to the acceleration shaft, as per Equation 1. To solve Equation 1 one initializes the shafts and turns on the driving motor. A graph of position versus time will immediately begin to appear on the plotting table, and it is this graph that constitutes a solution to the equation.

Figure 4 about here.

It is noteworthy that the operation of this system is *completely parallel,* in the sense that every variable is generated *simultaneously*. The instant the time shaft begins to rotate, this motion is communicated to the integrators, the first of which calculates velocity, the second, position, taking velocity as its input. Because the components of the system are rigidly connected, there is virtually no delay between these events and the moment when the sum of the final two shafts starts to provide feedback to the acceleration shaft. Although the free-fall equation can be solved using formal methods, the Differential Analyzer produces a solution very rapidly, and can with equal facility solve equations for which no exact formal solution exists.

So how and in what sense does this device compute a solution to the free-fall equation? Certainly not in the manner of a digital computer, which instantiates a set of formal rules designed to govern the transformation of numerical symbols. Here there are no formal rules, and no symbols. Instead, the Differential Analyzer is "an elegant, dynamical, *mechanical model* of the differential equation" (Owens 1986, p.75, emphasis added). Describing the analyzer in this way is appropriate when our target is a mathematical problem. But it can equally be described as a model of an object in free fall. Under either description, the analyzer computes a solution by producing an output that represents the variable of interest to us.[8] And it does so because its own operations are described by Equation 1. The classic texts in computer science are again clear on this point:

> …the analog-computer family consists of all those devices in which measurable physical quantities are made to obey mathematical relationships comparable with those existing in a particular problem. (Smith & Wood 1959, p.1)
>
> Analog models are devices that behave in a fashion analogous to others simply because they obey the same or similar fundamental laws of nature. (Truit & Rogers 1960, p.7)

The analyzer models an equation by mechanically acting it out. This does not apply to a digital computer, whose governing equations are, in most cases, those that apply to the components of a digital electronic device. A digital computer models a target system by "running the maths", that is, by numerically calculating the values of a function that describes the system. It does not itself obey those equations.

4. Representation in Analog Computation

The Differential Analyzer was the first general purpose analog computer. It was followed in the 1950s and 1960s by a series of increasingly sophisticated electronic computers, which employed electronic counterparts of the analyzer's adders, multipliers, integrators, and so on (Small 1993). Such computers were initially less accurate than their mechanical brethren, but cheaper to build and easier to program (Bromley 1990). Like the Differential Analyzer, they were general purpose analog devices that could be used to model a wide variety of systems by combining components as required (Truit & Rogers 1960).

Analog computers of all stripes operate in conformity with the equations they are designed to solve. We claimed in Section 2 that such systems exploit a physical analogy between a system of representing vehicles and the objects they represent. In this section we will consider what this implies about analog representation, using the Differential Analyzer as our touchstone.

Scientists who worked with Differential Analyzers took it for granted that they could be used to represent physical or mathematical objects. Representations are *content-bearers*: things that carry meaning. Although a completely general account of representation still eludes us, there is some consensus that such an account must address two main questions:

- *The Role Question.* What systemic role must something fill to *be* a representation, i.e., what makes some state or object a content-bearer in the first place.
- *The Content Question.* What determines the content of a representation?

It is commonplace these days for theorists to insist that any answer to the role question must emphasize the place of "interpretation". Bits of the world only become representations when there is some *user* for whom they operate *as* content bearers.[9] Digital and analog computers are alike in this regard: both digital and analog representation is always representation *for* or *to* some further system or process. But despite this common ground, digital and analog computers differ crucially with respect to the content question.

Symbols stand in an *arbitrary* physical relationship to the things they represent. Consequently, the content of the representations in a digital computer cannot be determined by their intrinsic properties. The standard story is that this content emerges as a result of the computational processes in which such representations are implicated.[10] In a modern electronic calculator, for example, the internal circuitry is so arranged that transitions among select groups of voltage states conform to the rules that regulate arithmetic. And it is this internal causal organization that licenses the interpretation of the voltage states as representing specific numbers.

[8] Note that the analyzer of Figure 4 can easily be modified to output a graph of velocity or acceleration.

[9] Peirce (1955) argued long ago that representation is a *triadic* relation between a representing vehicle, a represented object, and an "interpretant". Others who have emphasised the triadic nature of representation are Millikan (1984), whose "representation-consumers" correspond to Peirce's "interpretants", Bechtel (1998, 2009), and O'Brien & Opie (2004).

[10] The standard story is disputed by some theorists, most famously by Jerry Fodor (see, e.g., his 1987).

Analog computers, on the other hand, are designed to exploit an analogy between the physical properties of a system of representing vehicles and some target system. This entails that analog representations, by contrast with symbols, stand in a *non-arbitrary* physical relationship to the things they represent. The content of analog representations is determined by certain of their intrinsic properties—those that ground the physical analogy between the computer and the system it represents.

Consider the Differential Analyzer we described in Section 3. The physical analogy between this analyzer and an object in free fall is captured formally by Equation 1, but to fully appreciate this relationship one needs to understand the dynamics of the two systems. When first released, an object falling near the surface of the earth has an acceleration g = 9.8 ms^{-2}. Such an object quickly picks up speed, but the faster it falls, the more drag it experiences. Eventually the object's acceleration will reduce to zero, at which point it reaches its terminal velocity. Equation 1 describes the relationship between linear acceleration and velocity that results from these forces. But equally, Equation 1 describes the behaviour of our Differential Analyzer. In this case, the variable "y" corresponds to the angular position of shaft 4. This shaft rotates in a manner that is analogous to the behaviour of an object in free fall. It begins at rest, but then begins to rotate with an angular acceleration equal to the constant g (set by the position of shaft 6), and a steadily increasing angular velocity. Its acceleration gradually reduces as a result of feedback from shaft 5, which mimics the effect of drag, and hence its angular velocity will reach a "terminal" value. The mechanical principles that give rise to this behaviour are those of a device incorporating two integrators (as described in Section 3) and the set of gearings and connections portrayed in Figure 4.

So our Differential Analyzer can be used to model an object in free fall precisely because the rotational dynamics of its "position" shaft (shaft 4) is physically analogous to the linear dynamics of an object in free fall. The angular position of shaft 4 and the displacement of a falling object are in fact *isomorphic.* Formally, this means that there is a relation-preserving mapping from angular position onto displacement. Informally, it means that the graphs of these two variables have the same shape, and overlap if placed one on top of the other. Thus, the graph produced by our Differential Analyzer is at once a record of the behaviour of shaft four, and a representation of the motion of an object falling near the surface of the earth.

Physical analogy is a *resemblance relation*. Two systems resemble each other if they share first-order properties, such as mass or conductivity, or second-order properties, such as layout or organization. A nice example of second-order resemblance is the relationship between the width of a tree's growth rings and seasonal rainfall. Plentiful seasons produce wide rings, whereas drought years produce comparatively narrow rings. For this reason, if we map the set of growth rings (indexed by year) into the set of seasons, variations in rainfall are reflected in the relative width of the rings.[11] When a second-order resemblance is grounded in properties or relations that are *intrinsic* to some material system, as in this case, it is a physical analogy. Other physical analogies include the relationships between: i) a land map and the terrain it portrays, ii) the mercury levels in a thermometer and ambient temperature, and iii) the movements of the dial on a kitchen scale and the masses it registers. In each of these cases we harness a second-order physical analogy for representational purposes. A thermometer can be used to represent temperature because one of its intrinsic properties (the height of the mercury) bears a non-arbitrary relationship to that variable.

What the foregoing makes clear is that the representational content of a Differential Analyzer, and our free-fall analyzer in particular, depends on a physical analogy between the analyzer and its target. In the next and final section we consider what implications this might have for cognitive science.

5. Representation Redux

The current disquiet about mental representation depends on two main lines of argument. The first begins with the premise that the specifically semantic properties of representing vehicles are by nature incapable of shaping cognition. The second, with the claim that cognitive science has by and large failed to explain the kind of fluid, context-sensitive intelligence exhibited by human beings and many other organisms. Both conclude that mental representation fails to earn its explanatory keep in cognitive science, and is best abandoned.

[11] This mapping is unlikely to be an isomorphism, because the width of tree rings only approximately corresponds to variations in the weather. But even when the mapping between the parts and relations of two systems is patchy and inexact, we may still legitimately speak of them resembling each other.

However, as we observed in Section 1, cognitive science grew up in an era when digital computers were in the ascendant. The heyday of analog computation was already over, and it was the theory of digital computation that shaped the thinking of the first cognitive scientists. Consequently, the dominant picture of mental representation ever since has been the one suggested by digital computation: mental representations are *symbols*. In our view, it is this picture of mental representation that has led to the marginalisation of representation in contemporary cognitive science.

A symbol, like any representation, has both intrinsic properties and semantic properties. The representational content of a symbol depends on the causal organization of the digital computer in which it is embedded. Machine states become symbols, as opposed to meaningless bits of syntax, if their behaviour is regulated by physically implemented transition rules that conform to an intended interpretation. But the intrinsic properties of a symbol have no direct bearing on its meaning. Consequently, the causal impact of a symbol on computational processes is not determined by its representational content.[12] This feature of digital computation has led to considerable controversy about the significance of representation for cognition, and to the view that representational content is at best merely "explanatorily relevant" to cognition.[13] Little wonder then, that Stich (1983) and others countenance dispensing with the computational perspective altogether.

As for deep intelligence, here the so-called "knowledge problem" looms large—the problem of equipping a cognitive system with the informational resources to choose appropriate courses of action in real time, in response to open-ended internal goals (Dennett 1984). This problem has stymied the best attempts of classical AI to produce fluid, human-like intelligence. One plausible analysis of this failure is that it represents a limitation inherent in classical AI's commitment to symbolic representation. We noted in Section 1 that to solve a problem by digital means one must physically implement a formal description of the problem domain. Because the symbols composing the formal description bear arbitrary relations to the elements of the modelled domain, the former can capture the latter only when their behaviour is disciplined by carefully crafted syntactic rules. As a consequence, representation in a digital computer inevitably has a "micro-managed" character, in the sense that each and every semantic relation among its symbols must be regulated by proprietary syntactic machinery. Moreover, whenever descriptive components are added to the formalism or existing components are updated (as is necessary in modelling the real world), a myriad of further rules must be installed to ensure continued semantic coherence. As the formalism grows richer in content, the number of rules, and with it the number of individual processing steps, increases dramatically. In a digital system operating in real-time, the representational demands quickly outstrip the processing capacities of even the fastest hardware.

How would cognitive science look if our conception of mental representation had been shaped by experience with analog computers? The answer, we contend, is that the field's recent flirtation with anti-representationalism would not have occurred. This is because the two principal arguments that have led to the marginalisation of representation in contemporary cognitive science gain no traction when viewed from the perspective of analog computation.

First, we've seen that analog computation, unlike its digital counterpart, is directly shaped by the content-determining properties of its vehicles. Since the representational content of a symbol is determined by factors extrinsic to that vehicle, its content can have no bearing on what it does. The representing vehicles in an analog computer, by contrast, acquire content by virtue of their intrinsic physical properties and the resemblance relations these support. Hence, the computational processes that occur in an analog computer are driven by the very properties that determine the content of its vehicles. In this sense, an analog computer is not a mere semblance of a semantic engine—it's the real thing. Any organism whose inner processes are analog in nature is causally indebted to the semantic properties of its inner states.

Second, the constraint that makes the knowledge problem so acute for digital computers—viz., that syntactic rules are required to discipline the causal commerce between symbols—doesn't apply to analog computers. An analog computer is powered by a physical analogy between its representing vehicles and its task domain. As a consequence, representation in an analog computer has an "autonomous" character, in

[12] The relationship between content and causation is in fact the reverse. It is the causal role of a symbol that determines its content.

[13] See, for example, Baker 1993, Fodor 1989, Jackson & Pettit 1990, and LePore & Loewer 1989.

the sense that the semantic relations among its representing vehicles are inherently aligned with their physical relations. So when the content of an analog computer is augmented, via the enrichment of the physical analogy between its representational medium and its target domain, the semantic relations among its constituent vehicles take care of themselves. Prima facie, analog computers have the potential to capture the real-world, real-time character of deep intelligence.

From this perspective, contemporary anti-representationalism is best understood as a reaction to the *symbolic* form of representation implicated in digital computation, rather than *representation* per se. Cognitive science grew up at a time when digital computers, largely on the back of technical innovations, had supplanted their analog brethren. Digital computers have dominated and conditioned theorising in the field ever since. But digital computers have limitations—limitations owing to the fact that they are not genuine semantic engines, but ingeniously crafted syntactic devices that behave *as if* they were semantic engines. Instead of responding to these limitations by ditching representation and entertaining alternatives to computation, as many in the field are doing, the analysis we have presented suggests it would be more fruitful for cognitive science to explore the alternative *form* of computation.

References

Baker, L. R. (1993) Metaphysics and mental causation. In J. Heil & A. Mele, eds., *Mental Causation*. Oxford University Press.

Bechtel, W. (1998) Representations and cognitive explanations: Assessing the dynamicist's challenge in cognitive science. *Cognitive Science* **22**: 295-318.

Bechtel, W. (2009) Constructing a philosophy of science of cognitive science. *Topics in Cognitive Science* **1**: 548-569.

Beer, R. D. (1995) A Dynamical Systems Perspective on Agent-Environment Interaction*. Artificial Intelligence* **72**(1-2): 173-215.

Brooks, R. (1991) Intelligence without representation. *Artificial Intelligence* **47**: 139-59.

Bromley, A. G. (1990) Analog computing devices. In M. Aspray, ed., *Computing Before Computers*. Iowa Sate Press.

Chomsky, N. (1959) Review of *Verbal Behavior* by B.F. Skinner. *Language* **35**: 26-58.

Churchland, P.S., Koch, C. & Sejnowski, T. (1993) What is computational neuroscience? In E. Schwartz, ed., *Computational Neuroscience*. Cambridge MA: MIT Press.

Clark, A. (1997a) The dynamical challenge. *Cognitive Science* **21**: 461-81.

Clark, A. (1997b) *Being There: Putting Brain, Body and World Together Again*. MIT Press.

Clymer, A. B. (1993) The mechanical analog computers of Hannibal Ford and William Newell. *IEEE Annals of the History of Computing* **15**(2): 9-34.

Copeland, B. J. (1993) *Artificial Intelligence: A Philosophical Introduction*. Wiley-Blackwell.

Dennett, D. C. (1984) Cognitive wheels: The frame problem of AI. In C. Hookway, ed*.,* *Minds, Machines and Evolution*, Cambridge University Press.

Fodor, J. A. (1975) *The Language of Thought*. Harvester Press.

Fodor J.A. (1987) *Psychosemantics: The Problem of Meaning in the Philosophy of Mind*. MIT Press.

Fodor, J. A. (1989) Making mind matter more. *Philosophical Topics* **17**:59-79.

Haugeland, J. (1985) *Artificial Intelligence: The Very Idea*. MIT Press.

Jackson, A. S. (1960) *Analog Computation*. McGraw-Hill.

Jackson, F. & Pettit, P. (1990) Program explanation: A general perspective. *Analysis* **50**:107-17.

Keijzer, F. (2002) Representation in dynamical and embodied cognition. *Cognitive Systems Research* **3**: 275-88.

LePore, E. & Loewer, B. (1989) More on making mind matter. *Philosophical Topics* **17**:175-91.

Lewis, D. (1971) Analog and digital. *Noûs* **5**: 321-7.

Millikan, R. G. (1984) *Language, thought, and other biological categories*. MIT Press.

O'Brien, G. & Opie, J. (2004) Notes towards a structuralist theory of mental representation. In H. Clapin, P. Staines & P. Slezak, eds., *Representation in Mind: New Approaches to Mental Representation*. Elsevier.

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

O'Regan, J. K. & Noë, A. (2001) A sensorimotor account of vision and visual consciousness. *Behavioral and Brain Sciences* **24**: 939-1031.

Owens, L. (1986) Vannevar Bush and the Differential Analyzer: The Text and Context of an Early Computer. *Technology and Culture* **27**(1): 63-95.

Peirce, C. S. (1955) Logic as semiotic: The theory of signs. In J. Buchler, ed., *Philosophical Writings of Peirce*. Dover Publications.

Port, R. & Van Gelder, T. J. (1995) *Mind as Motion: Explorations in the Dynamics of Cognition*. MIT Press.

Soroka, W. (1954) *Analog Methods in Computation and Simulation*. McGraw-Hill.

Small, J. S. (1993) General-Purpose Electronic Analog Computing: 1945-1965. *IEEE Annals of the History of Computing* **15**(2): 8-18.

Smith, G.W. & Wood, R.C. (1959) *Principles of Analog Computation*. McGraw-Hill.

Stich, S. (1983) *From Folk Psychology to Cognitive Science: The Case Against Belief*. MIT Press.

Thagard, P. (2008) Cognitive Science. In E. N. Zalta, ed., *The Stanford Encyclopedia of Philosophy*. URL = <http://plato.stanford.edu/archives/fall2008/entries/cognitive-science/>

Truitt, T.D. & Rogers, A.E. (1960) *Basics of Analog Computers*. John F. Rider.

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

Wallace, B., Ross, A., Davies, J. & Anderson, T., eds., (2007) *The Mind, the Body and the World. Psychology after Cognitivism?* Imprint Academic.

Wheeler, M. (2005) *Reconstructing the Cognitive World: The Next Step*. MIT Press.

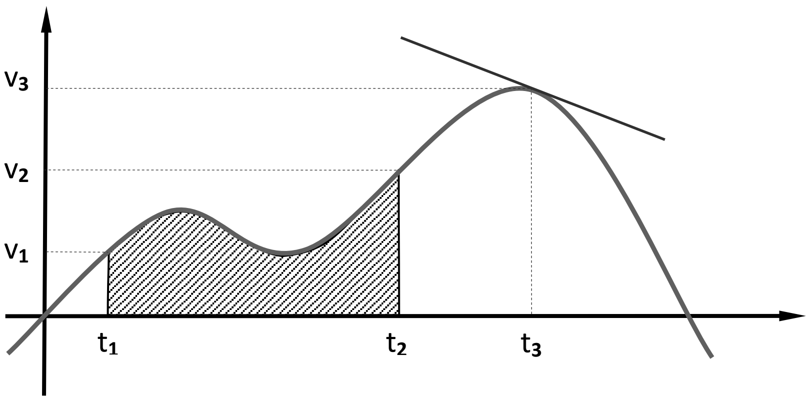

Figure 1: A graph of velocity versus time. The distance travelled between t_1 and t_2 can be computed by measuring the area of the shaded region. The acceleration at t_3 can be computed by measuring the slope of the tangent to the curve at that point.

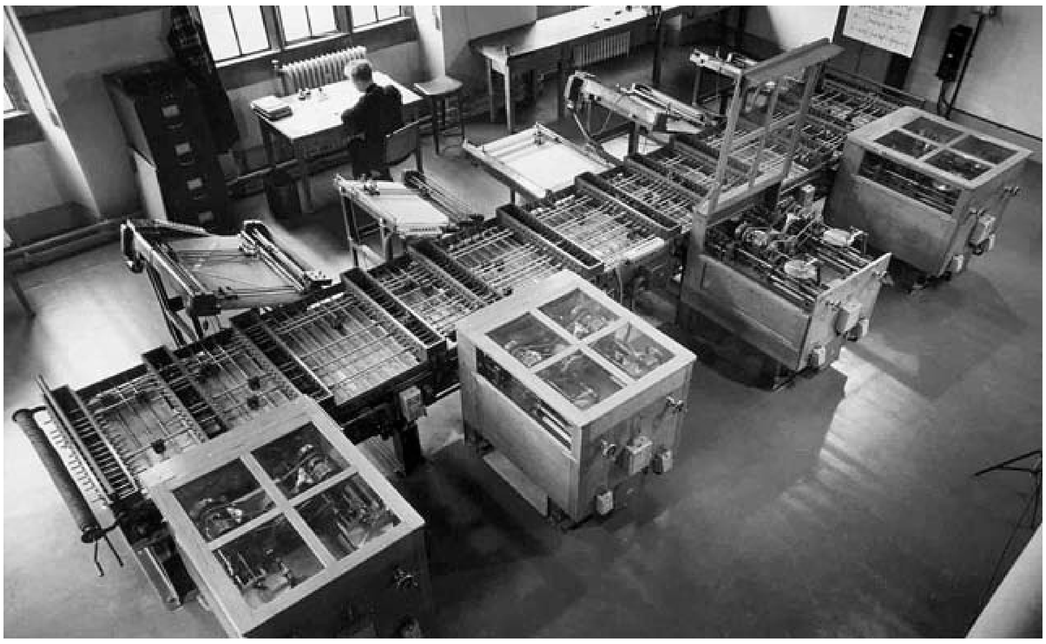

Figure 2: The Cambridge Differential Analyzer, which is similar to Bush's design. The disk-and-wheel integrators are on the right; the input and output tables on the left.

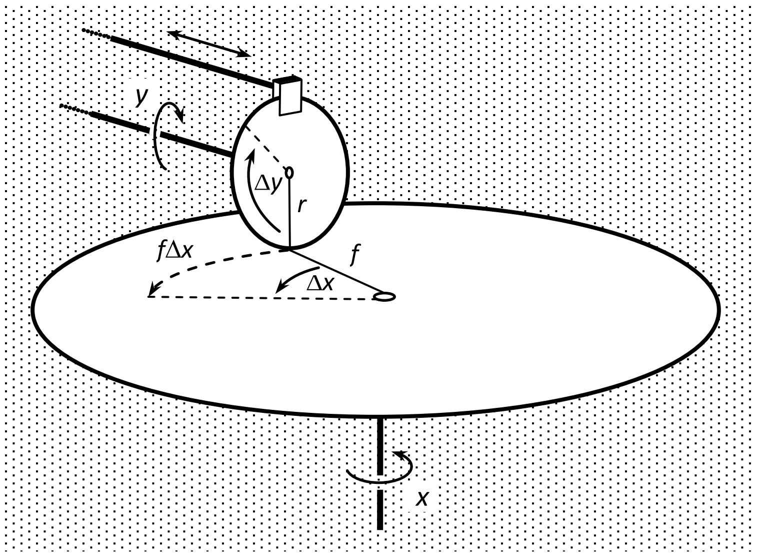

Figure 3: A disk-and-wheel integrator. The small wheel rotates in response to rotation of the disk, and is free to move along the radius of the disk as a result of linear movements of the upper shaft.

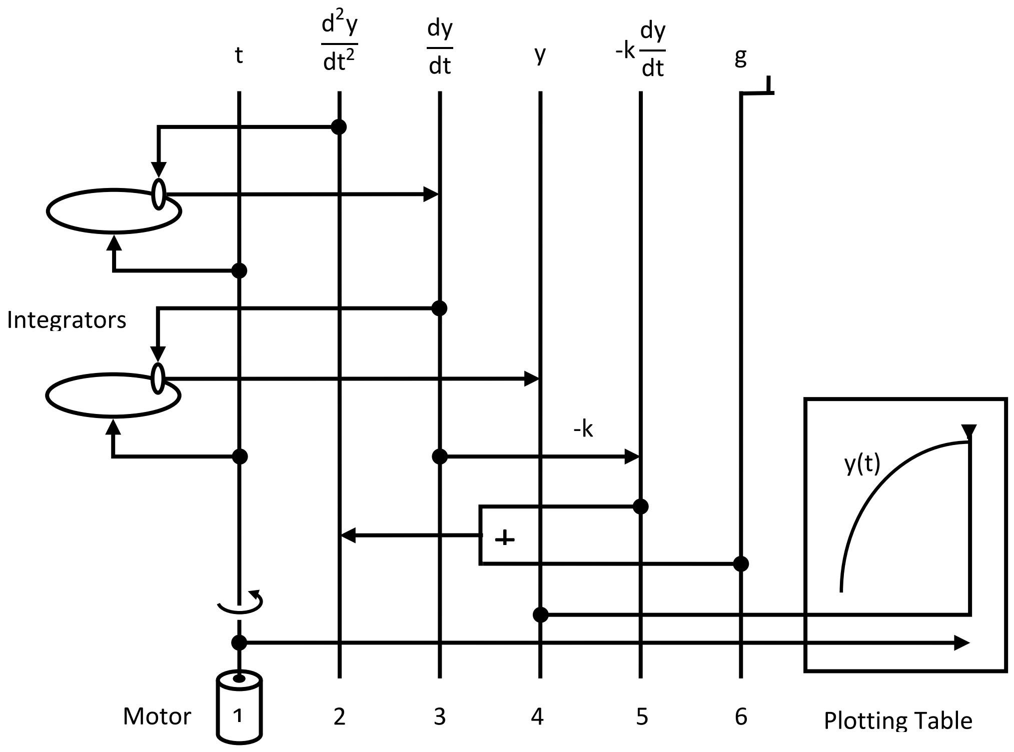

Figure 4: A Differential Analyzer set up to solve the free-fall equation. Each shaft represents a variable or a constant in the equation. The analyzer requires two integrators because it is designed to solve a second-order differential equation. The first lengthwise shaft is driven by a motor at a constant speed and represents the independent variable t.