The Non-Classical Mind: Cognitive Science and Non-Classical Computing

Colin G. Johnson

Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NF
England
C.G.Johnson@kent.ac.uk

Summary. This paper explores the ideas of *non-classical computation* (computing where one or more traditional assumptions about what defines computation have been dropped) in the context of cognitive science. A framework that classifies non-classical computing concepts is discussed, and the potential impact of each of these concepts on issues of cognition, mind and affect is analysed.

1 Introduction

Computational modelling of mental activity is one of the core activities in cognitive science. This is grounded in a *computational functionalist* view of mind. That is, the substrate in which mind is realised is irrelevant to being able to produce mentality, and a computer is a sufficient substrate in which to be able to carry out that realisation.

In recent years a discipline of *non-classical computing* has emerged. This aims to reappraise the foundations of computing, and discover what happens when we break in turn each of the assumptions that we make about what a computer is and how it functions. In particular, we would like to understand whether the system that remains when the assumption is removed can still be meaningfully understood as being a computer, and if it can solve computational problems that are difficult/impossible for traditional computers.

In this chapter we examine what the consequences of non-classical computing might be for cognitive science. If the notion of *computation* can be stretched, what does this imply for the *computational* side of computational functionalism? To do this, we take the taxonomy of classical computing assumptions devised by Stepney et al. [33], and look at each part of this taxonomy in turn.

The chapter is structured as follows: section 2 gives more detail on the background to computational functionalism and non-classical computation,

and section 3 gives a taxonomy of the assumptions within classical computation which can be relaxed or removed to create non-classical computing systems. The remaining substantive sections take one class of assumption from classical computing and examine the implications for cognitive science of non-classical computers that break those kinds of assumptions. Finally, there is a brief conclusion and some questions for further thought.

2 Computational Functionalism and Non-Classical Computing

One of the key ideas in cognitive science is the use of computational metaphors and models in understanding the mind. The justification for this is a set of assumptions that we will term *computational functionalism*.

This consists of two parts. The first part is the *functionalist* assumption. This is the notion that a mind equivalent in capability to a human (or animal) mind can be created in any medium that is sufficient to allow *processes* of the kind found within the brain to happen. Essentially the mind results from the processes in the brain, rather than from the stuff that the brain is made of. This assumption has been well studied (see [5, 19] for overviews). An important implication of this is that a mind having the same capabilities as a human/animal mind could, in theory, be realised on an alternative, non-neural, substrate.

The second assumption is the *computational* assumption (sometimes called *machine-state functionalism*). This states that a computer is a sufficient medium for the realisation of the activity that is realised in neurally-substrated minds [28]. This assumption has received less critical attention than the core functionalist assumption.

In order to take this computationalist stance towards mind, we need to have some notion of what a computer is. Traditionally, the definition of computation is given by giving an abstract, mathematical defintion of a sufficient computing device (a Turing machine or equivalent), then treating as a computer any physical device which is capable of processing information in a way which is (behaviourally) indistinguishable from a Turing machine. This is sometimes referred to as the *classical* notion of computation.

In recent years, a new approach to the foundations of computation has been created. This is called *non-classical*, *post-classical* or *reality-based* computing [33, 34]. This seeks to ground the notion of computation by assessing the computational properties of physical systems in the world, rather than beginning with a notion of a computational machine.

This stance has its origins in a 1982 paper by Feynman [13]. In this paper he discusses physical phenomena that are not efficiently realisable on (traditional) computing machines. The example given is drawn from quantum mechanics. He notes that there are some quantum processes that cannot be efficiently simulated on traditional computers. Then, significantly, inverts this

argument - if there are processes that cannot be simulated efficiently, then we can use these processes to *build* new kinds of computers that are, at least in that limited domain, capable of doing more than traditional computers:

> *"Now I explicitly go to the question of how we can simulate with a computer—a universal automaton or something—the quantum mechanical effects. [...] the full description of the quantum mechanics for a large system with R particles is given by a function* $\psi(x_1, x_2, \ldots, x_R, t)$*, which we call the amplitude to find the particles* $x_1, x_2, \ldots, x_R$*, and therefore, because it has too many variables, in* cannot be simulated *with a normal computer with a number of elements proportional to R or proportional to N. We had the same troubles with the probability in classical physics. And therefore, the problem is, how can we simulate the quantum mechanics? There are two ways we can go about it. We can give up on our rule about what the computer was, we can say: Let the computer itself be built out of quantum mechanical elements which obey quantum mechanical laws. [...]"*

Non-classical computing generalises this to other systems. Instead of defining computation "up front", we examine things in the world for properties that could lead to them being useful as computational devices. This is important for cognitive science because if such alternative computational capabilities exist in the world, it is likely that they will have been adopted by evolutionary processes as part of the "toolkit" of structures that could come together to create mind.

The remainder of this chapter explores the consequences of such reality-based computation for the study of mind. If we look at the assumptions that underly classical computation (and by consequence cognitive science), what happens if we break these assumptions?

3 A Taxonomy of Assumptions for Non-Classical Computation

In order to carry out this programme of work we first need to understand what assumptions underpin classical computation. For this purpose we will use the taxonomy developed by Stepney et al. [33]. This breaks down the classical computing assumptions into six categories:

The Turing Paradigm. The assumptions that are made in specifying the Turing machine as a canonical model of computation. In particular, the idea that a computer consists of processing discrete states that can be freely read and copied, that resources are freely available to extend the tape or carry out some process, and that the choice of substrate is an implementation detail rather than an important feature. More abstractly, the idea

of the *universal computer* which can be applied to all tasks is implicit in this set of assumptions.

The von Neumann Paradigm. The assumptions that computing machines contain a single core computational unit, and that information is brought to this unit for computation.

The Output Paradigm. The assumptions that suggest that a computational process has a well-defined output, and that what is interesting/productive about a computer is these outputs, rather than what happens during the computation.

The Algorithmic Paradigm. The assumptions that a computer executes a well-defined process, bounded in time, with input and output clearly understood. Also the idea that a computer is a deterministic (non-random) device.

The Refinement Paradigm. The assumptions that solving a problem on the computer consists of turning some description what a problem is into how to solve it, and the assumption that such a description exists, exists before the attempt to solve the problem starts, and does not change during this process.

The Computer-as-Artefact Paradigm. The assumptions that a computer is a device of fixed extent, that cannot extend itself and which does not interfere with the problem being solved during the solution. Also, the notion that solving a particular problem involves finding a way of processing that problem on a given computer system, rather than searching for something in the physical world which already does the computational process of interest.

In the remainder of this chapter we will examine each of these paradigms in turn, and consider the implications for cognitive science of breaking each of these paradigms.

4 Breaking Assumptions 1: The Turing Paradigm

The Turing paradigm is, at its broadest, the idea that computation is best defined by giving an abstract mathematical description of what a "computation" is, and then showing that real-world computations are equivalent to that description. More narrowly, this is the *Church-Turing thesis*, which asserts that a Turing machine (see e.g. [14]) is capable of acting as a universal computation device. By contrast we can start from a set of physical activities in the world, which appear all to be doing something that we can term "computation", and ask what is common to them (and not found in things/processes that are not computations).

In this section we explore both the consequences of going beyond the Turing machine model as a model, and more generally the consequences of computing embedded in the world.

4.1 More than Turing machines: Analogue, Quantum and Diffusive

One assumption that is part of the Turing paradigm is that representations of state in computer systems are discrete. This discreteness occurs at two levels: systems have a discrete set of *containers* for information (in the Turing machine model, there are discrete spaces on the tape to receive symbols), and those containers can contain one of a finite alphabet of symbols.

A system that can represent an analogue symbol of unbounded resolution could potentially act in a different fashion. For example, Siegelmann [32] has shown that neural networks with analogue weights are more powerful than those with digital weights (in the sense that there are processes that can be carried out efficiently with the analogue weights that cannot be carried out efficiently with integer or floating-point weights).

Another assumption is that symbols can be independently modified without interfering with other symbols, and where multiple states can exist simultaneously. Quantum systems, which have been controversially [29] hypothesised by Penrose [24, 25] as a way of carrying out cognitive processes that cannot be realised on a traditional neural structure. However, it has been argued [36] that the difference in timescales between the quantum and neural processes make this infeasible.

A final assumption is that the modelling of neurons and their connections alone is sufficient to reproduce cognition in a connectionist model. The traditional model of connectionist networks used in cognitive science abstracts from brain physiology at the level of the discrete neuron. However, recent models such as the GasNets model, devised for robot control and related problems [15, 30] include the non-local effects of neurotransmitters such as nitrous oxide, where the molecules are small enough to diffuse throughout the brain. This provides a level of non-synaptic communication within the brain.

4.2 Exploiting Non-Standard Features during Learning

One aspect of embedding computing in the real world, and then using a learning system to adapt that computational system to its environment, is that the learning process might exploit features of the medium in which it is embedded, rather than just features of the computing system which is embedded in the medium.

An example of this is illustrated in Thompson's work [37] on the evolution of digital circuits. The aim of this work is to evolve circuits within Field Programmable Gate Arrays (FPGAs), which are programmable logic devices that can be configured to a particular digital circuit via a program. Evolution can be carried out on these by evolving programs on a conventional computer, using a variant on genetic programming [17, 4], and testing fitness by downloading the programs to the FPGA and testing them on the chip.

During one of these experiments [38], an unusual behaviour was observed. The circuit worked very well during evolution, but when tested on a second

FPGA chip, the circuit failed to work at all. Upon closer inspection, its was noted that the evolved circuit was relying on some analogue property of the original digital circuit. As the FPGAs are not designed to behave consistently at the analogue level (just to be equivalent in terms of the digital process that they carry out), the evolved circuit could not be transferred to the second chip successfully.

Might we see equivalent phenomena during neural learning in the brain? Might learning exploit particular non-neural aspects of brain physiology in order to provide a different kind of computation from that which is easy/possible to implement in neurons? Or is a physiological system such as the brain changing too much in the off-substrate areas to make this kind of thing possible?

5 Breaking Assumptions 2: The von Neumann Paradigm

The assumption here is that there is a central, sequential processing unit, that fetches program code and data to it and writes its output to an addressed space.

5.1 Concurrency and Parallelism in Cognition

This is one of the more established non-classical computing notions in cognitive science. It has been long argued that the brain implements computation in a parallel fashion, and neural networks work in an inherently parallel way.

One recent argument [27] about concurrency in computer systems is that we should stop regarding concurrency as a special kind of computing, and instead regard serial operation as being unusual. This is of particular relevance in building models and simulations of real-world phenomena on the computer—a concurrent system (or a programming language where concurrency is the default) allows us to model such a system by reproducing the real-world concurrency. This raises questions for cognitive science about whether the concurrency in the brain is there in order to model the many concurrent actions in the real world.

Another way in which to break these assumptions is to consider systems where the data remains in its location and the computational device moves over that data, modifying the data or the device (or both) to produce an output, rather than the processing unit remaining in one place and bringing data to it. For example, reaction-diffusion computers [3] which move through a physical representation of a problem (such as a fluid diffusing through a maze [2]). There do not appear to be any immediate applications of these concepts in understanding cognition, beyond trivial examples such as viewing the flow of neural activity through fixed-position neurons as an example of this.

6 Breaking Assumptions 3: The Output Paradigm

The output paradigm is the set of assumptions that the output from a process is the only way of getting information out of a comptuational process, and that we have to wait until the program has finished its execution before we can get a meaningful answer.

6.1 Algorithms and Interruptibility

One assumption that is made in many computer learning systems, including many that are designed to emulate human or animal learning, is that we can run a carefully-designed schedule of learning trials leading to a final learned system.

However, in many learning situations that are embedded in the world, the process of learning is regularly interrupted. The time required for a learning process to work to completion is variable. Given that this will have been the case throughout the evolution of these learning systems themselves, it is likely that the kinds of learning algorithms that will evolve will be able to carry out an action at any time which makes maximal use of what has been learned up to the point of interruption.

Such algorithms have been termed *anytime algorithms* [41, 6] in computing. An anytime algorithm is one that can be disrupted at any point in its calculation, and which is able to report an intermediate result which makes use of some or all of the calculation that has occurred up to that point. These have been applied e.g. in planning systems [6] and robotics [42]. Some commonly used bio-inspired learning algorithms have this property. For example, a neural network trained using an algorithm such as backpropagation adjusts its weights when each example is presented. Therefore it would be possible to run an example "for real" after an arbitrary number of training runs, and the network would report an answer based on all of the training runs to date.

6.2 Physical Observations and Trajectories

In order for a computation to be carried out on a computer, the physical state of the machine must change, and energy taken into the system. These changes are usually seen as unimportant consequences of the physical grounding of computation. However, in a number of specialised areas these physical characteristics are important. One example is in the design of software for low-energy embedded devices such as environmental monitoring or satellite systems [35]. Another example can be found in the concept of *power analysis* attacks in computer security [7]. This is the idea that an attacker could get some information about the process going on inside a device (such as a smartcard) by analysing the power usage of the device in response to various inputs, and correlating this power usage profile with profiles from known computations. Overall, we can ask what we can learn by observing physical

properties of a computational device whilst it is computing, and how a computational device with multiple components might communicate between those components other than with explicit outputs.

How might this be of significance in cognitive science? In carrying out mental processes, the brain makes varying demands on the body—for example certain processes may require more blood flow to the brain. Might some components of the brain monitor these demands (perhaps indirectly, via the monitoring of other processes in the body), and adjust aspects of mental functioning appropriately? Might this represent a possible mechanism for the evolution of certain kinds of emotional processes. The mechanism would be that a certain mental process might monitor other mental processes not directly, but by monitoring the demand that the brain is making on the body, and learn the correlations between those demands and relevant mental responses. In certain circumstances this might prove more efficient than communicating directly with other brain-based processes.

7 Breaking Assumptions 4: The Algorithmic Paradigm

The assumptions here are that programs take an input, and then compute some output, ignoring the remainder of the world whilst the computation is taking place, and that randomness and noise are problems the be avoided.

7.1 Interactivity

Wegner [39] has argued that the major shift in computer technology from the 1970s to the 1990s has been that computer systems have moved from being algorithmic systems (with a fixed idea of when a particular computation starts and stops, and with no cumulative memory or history of the computation) to interactive systems, which run for long periods of time, have many interactions over that time, and learn or accumulate information as they run which influences future interactions.

His core argument is that interactive systems can achieve more than algorithmic systems - interaction cannot be reduced to algorithms. This is formalised [40] by the development of a model of an "interaction machine", consisting of a Turing machine with the addition of input and output actions that allow the external environment to influence the computation whilst it runs. It can be shown that interaction machines, and other such formalisms for interactive systems [20], cannot be reduced to a Turing machine model.

One example of particular interest in cognitive science is interactive systems that "outsource" some of their cognition to systems in the outside world. A (somewhat artificial) example of this is given by Wegner [39] in the context of computer chess. This is illustrated in figure 1. An *interactive agent*, largely ignorant of chess, plays against two master-level players. The agent lets one of the master players (say player A) play first, then copies that move on the

other board. The agent then waits for player B to make a move, and then copies that as its response to player A. This process then continues back and forth. In such a system the interactive "player" will typically win half of its games; by contrast a noninteractive player with minimal chess experience will lose both games.

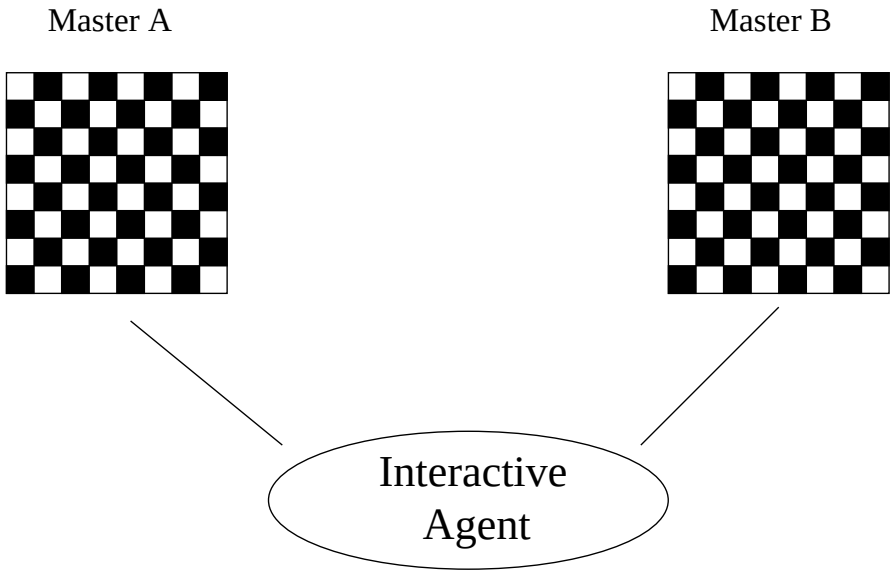

Fig. 1. Wegner's Interactive Chess-Playing Agent (after [39]).

Do humans delegate cognitive capacity out "into the world" in this way. Clearly we do this with memory - for example through making notes, looking up references in books, or using an internet search (which has, for some people, become almost an automated volitional action!). Clearly, too, we use devices to carry out cognitive processes that are too complicated or time lengthy to do on-substrate, such as the use of a calculator. A less obvious example of this phenomenon is flipping a coin, which delegates the task of generating a random event, something that we cannot readily do mentally, to an external device. Does the brain's "outsourcing" of cognition go beyond working with simple cognitive devices?

8 Breaking Assumptions 5: The Refinement Paradigm

The assumptions here are that programs are created by a process of refinement from a clearly defined specification, via a sequence of intermediate stages, to a concrete program. By contrast, many natural systems demonstrate emergent

properties [26] which are not "refinable" from an attempt to solve a particular well-specified problem.

8.1 Data-Driven Systems

Traditional computer programs are *specification defined*. That is, the best way of describing the desired behaviour of the program is to give a specification of that behaviour, that is, a description of *what* the system should do. Then, the process of computer programming consists of translating that specification into a working program, by an (informal or formal) process of *refinement*, where descriptions of *what* the system should do are replaced by descriptions of *how* the system should do it.

By contrast, some problems are best defined by data [21, 22, 23], for example a program designed for letter or word recognition. In order to describe what is meant by a letter "a", for example, the best approach is to give a large number of examples of such a letter, rather than giving a description of what is and isn't an "a". Along similar lines, some properties that we might want to realise on a computer system are best defined by the system interacting with a human user—for example, properties concerned with aesthetic judgements.

Algorithms for learning in neural networks typically work by calculating errors based on examples. This is a typically data-defined way of working—take data, calculate errors, update model based upon those errors. However, high-level cognition is able to operate on specification driven problems. A significant issue in cognitive science is how to reconcile these two theories? Are systems for learning specification-driven problems able to be built from generic neural architecture (for example, Lebière and Anderson [18] show how a variant on the ACT-R symbolic architecture can be built on a connectionist substrate)? By contrast, dual-process theories [12] suggest that evolutionarily-old parts of the brain are focused on learning from examples, whilst structured thinking of the kind required to solve specification-driven problems is a newcomer on an evolutionary timescale, and has its own distinct neural architecture.

9 Breaking Assumptions 6: The Computer-as-Artefact Paradigm

The assumption here is that computation is performed by specific computational artefacts built for the purpose (whereas, many natural objects "just do" some kind of computation as part of their normal existence) and that these artefacts remain unchanged throughout the computation.

9.1 Delegation

In traditional computing, it is assumed that a single kind of general-purpose computing device is available. As anything that can be computed can be

computed by such a device, there is no need to use different kinds of physical device to solve different kinds of problems. This is in contrast to most other kinds of problem-solving technologies, where different physical devices are used for different stages of a problem, rather than a single generic device being "programmed" for the different stages of the problem.

This is a contrast to other engineering disciplines, where machines are built from a number of different materials, rather than a single material which can be molded to the relevant task. However, in recent years a number of computational devices have been created that exploit features of particular material substrates to carry out particular computations in an efficient manner [3, 1]. For example Adamatzky has demonstrated chemistry-based computers that exploit reaction-diffusion systems to calculate Voronoï diagrams [10] or to find paths through complex environments by exploring many paths simultaneously [2].

Now consider the practical application of a non-classical computing technology such as quantum computing or computing with a liquid medium. It is likely that in preparing a problem for solution on such a computer, a number of (conventional) computational steps would have to be carried out before the non-classical computer could be applied, and other computational steps carried out afterwards. For example, in the application of Shor's algorithm for factoring numbers on a quantum computer [31], a pair of numbers need to be provided as a starting point for the algorithm, and, following the quantum processing, algorithms to extract the period of a sequence and calculate greatest common divisors need to be applied.

Therefore, when we talk about "non-classical" computing, we are rarely talking about entire problems that can be solved using solely a non-classical computer. Instead, the classical computer is *delegating* certain aspects of its computation to a special-purpose non-classical computation engine.

We can imagine that the mind might act in a similar way with delegation of tasks to special-purpose structures in the brain or body that deal with tasks in a non-neural fashion. One example is provided by the *somatic markers* hypothesised by Damasio [8, 11]. In this theory, it is suggested that the mind "uses" bodily state to facilitate rapid computation of certain situations (i.e. that the mental process is realised through a mixture of neurally-substrated processing and bodily state). This is carried out by neural processes that trigger non-neural changes in body state, which are subsequently re-perceived by the neural-substrate, in particular drawing attention towards some salient feature of the external world that might otherwise be missed.

As an example consider the falling sensation felt in the belly when reaching a cliff-edge unexpectedly, or the feeling of nausea felt when witnessing a violent act. In both of these cases, the feeling is generated by some complex perceptual process in the brain, and is subsequently reperceived (by a different brain component) for rapid action. The neurally-substrated mind has delegated the task of dealing with emergency situations to the body-state and proprioceptive systems.

In such scenarios, the task of dealing with attention-shift has been delegated to the somatic system. However, much of the remainder of the computation is carried out using the main neural substrate. This is reminiscent of the various ways in which non-classical computation uses specific, non-generic computing devices to carry out a particular aspect of a computation (a quantum factoring engine; a reaction-diffusion route finder).

9.2 Porous Boundaries

Somatic markers extend the "computation" that goes on in the mind beyond the neural substrate into the body. However, need we stop here? One important theme in non-classical computing is the extension of computing beyond the immediate computing artefact, perhaps including computations that are "unbounded" in the sense that we cannot decide before beginning the computation exactly which information will be needed to carry it out.

A similar idea has been explored in the biological context by Dawkins [9] in the concept of the *extended phenotype*. The core of this idea is that the phenotypic effect of a gene can extend beyond the body of the organism in which it is found"

> *"An animal's behaviour tends to maximize the survival of the genes 'for' that behaviour, whether or not those genes happen to be in the body of the particular animal performing it."*

As a simple example, the genes of a male bird that influences it to mate preferentially with female birds with blue feathers can be viewed as having a phenotypic effect within the body of the female birds. Another example are the dams and subsequent lakes built by beavers.

Can we see examples of this extension process in cognition? One example could be somatic markers which extend beyond the body [16]. We have described somatic markers as effects of the mind on the body that are subsequently perceived by the proprioceptive system, and subsequently acted upon by the mind. An *extended somatic marker* is some change that is carried out in the world outside the body, by a process that is not currently the focus of attention, and which is subsequently reperceived and acted on in an attentive way. For example, might inattentive scribbling on a piece of paper when anxious be a marker of that anxiety, later to be perceived and acted upon visually? Such a mechanism as this might have been evolved/learned from the somatic marker mechanism, exploiting a different perceptual route.

10 Conclusions and Questions

In this paper we have taken a model of non-classical computation which consists of outlining the various assumptions of classical computing and asking

what happens to computing when each of them is removed. These ideas have then been applied to questions about cognition and mentality.

A theme that recurs multiple times in this discussion is that of specialization and delegation: does (some component of) mind make use of specialised structures (within the brain, within the body, out in the world) to do particular tasks that cannot be readily done by the neural substrate? This dovetails well with many of the questions asked in non-classical computing, which move away from the notion of a universal computer and instead ask questions about the computational capabilities of particular kinds of materials and machines on particular kinds of problems.

References

1. A. Adamatzky. Computing in nonlinear media: make waves, study collisions. *Lecture Notes in Computer Science (vol 2159)*, pages 1–11, 2001.
2. A. Adamatzky, B. De Lacy Costello, C. Melhuish, and N. Ratcliffe. Experimental reaction-diffusion chemical processors for robot path planning. *Journal of Intelligent and Robotic Systems*, 37:233–249, 2003.
3. Andrew Adamatzky. *Computing in Nonlinear Media and Automata Collectives*. Institute of Physics Publishing, Bristol, 2001.
4. Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.
5. Ned Block. Functionalism. In *The Encyclopaedia of Philosophy (Supplement)*. Macmillan, 1996.
6. M. Boddy and T. Dean. Solving time dependent planning problems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit*, 1989.
7. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Power analysis: Attacks and countermeasures. In Annabelle McIver and Carroll Morgan, editors, *Programming Methodology*. Springer, 2003.
8. A.R. Damasio. *Descartes' Error : Emotion, Reason and the Human Brain*. Gosset/Putnam Press, 1994.
9. Richard Dawkins. *The Extended Phenotype*. Oxford University Press, 1982.
10. B. De Lacy Costello, A. Adamatzky, N. Ratcliffe, A.L. Zanin, A.W. Liehr, and H.-G. Purwins. The formation of Voronoï diagrams in chemical and physical systems:experimental findings and theoretical models. *International Journal of Bifurcation and Chaos*, 2004.
11. B.D. Dunn, T. Dalgleish, and A. Lawrence. The somatic marker hypothesis: A critical evaluation. *Neuroscience and Biobehavioural Reviews*, 30(2):239–271, 2006.
12. Jonathan St. B. T. Evans. In two minds: Dual process accounts of reasoning. *Trends in Cognitive Sciences*, 7:454–459, 2003.
13. Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.
14. John Hopcroft, Rajeev Motwani, and Jeffery Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, second edition, 2001.

15. P. Husbands, T. Smith, N. Jakobi, and M. O'Shea. Better living through chemistry: Evolving GasNets for robot control. *Connection Science*, 10(4):185–210, 1998.
16. Colin G. Johnson. Does a functioning mind need a functioning body? In Darryl N. Davis, editor, *Visions of Mind*, pages 307–321. Idea Group Publishing, January 2005.
17. John R. Koza. *Genetic Programming : On the Programming of Computers by means of Natural Selection*. Series in Complex Adaptive Systems. MIT Press, 1992.
18. C. Lebiére and J.R. Anderson. A connectionist implementation of the ACT-R production system. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 635–640, 1993.
19. Janet Levin. Functionalism. In E. Zalta, editor, *The Stanford Encyclopedia of Philosophy*, 2004.
20. Robin Milner. Elements of interaction. *Communications of the ACM*, 36(1):78–89, 1993.
21. D. Partridge and A. Galton. The specification of 'specification'. *Minds and Machines*, 5(2):243–255, 1995.
22. D. Partridge and W.B. Yates. Data-defined problems and multiversion neural-net systems. *Journal of Intelligent Systems*, 7(1–2):19–32, 1997.
23. Derek Partridge. The case for inductive programming. *IEEE Computer*, pages 36–41, January 1997.
24. Roger Penrose. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford, 1989.
25. Roger Penrose. *Shadows of the Mind*. Vintage, 1995.
26. Fiona Polack and Susan Stepney. Emergent properties do not refine. In *REFINE 2005 Workshop, Guildford, UK, April 2005*, volume 137(2) of *ENTCS*, pages 163–181. Elsevier, 2005.
27. Fiona Polack, Susan Stepney, Heather Turner, Peter Welch, and Fred Barnes. An architecture for modelling emergence in CA-like systems. In Mathieu S. Capcarrère, Alex Alves Freitas, Peter J. Bentley, Colin G. Johnson, and Jon Timmis, editors, *Advances in Artificial Life, 8th European Conference on Artificial Life (ECAL 2005)*, volume 3630 of *Lecture Notes in Computer Science*, pages 427–436. Springer, September 2005.
28. Hilary Putnam. *Mind, Language, and Reality*. Cambridge University Press, 1975.
29. Hilary Putnam. Book review: Shadows of the mind. *Bulletin of the American Mathematical Society*, 32(3):370–373, 1995.
30. C.L.R. Santos, P.P.B. de Oliveira, P. Husbands, and C.R. Souza. Three case studies on the GasNets model in discrete domains. *International Journal of Neural Systems*, 11(3):295–304, 2001.
31. P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. In *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science*. IEEE Computer Society Press, 1994.
32. Hava T. Siegelmann. *Neural Networks and Analog Computation : Beyond the Turing Limit*. Progress in Theoretical Computer Science. Birkhäuser, 1998.
33. S. Stepney et al. Journeys in non-classical computation I: A grand challenge for computing research. *Int. J. Parallel, Emergent and Distributed Systems*, 20(1):5–19, 2005.

34. S. Stepney et al. Journeys in non-classical computation II: Initial journeys and waypoints. *Int. J. Parallel, Emergent and Distributed Systems*, 21(5):97–125, 2006.
35. Walid Taha. Resource-aware programming. In *Proceedings of the 2004 International Conference on Embedded Software and Systems*, 2004.
36. Max Tegmark. The importance of quantum decoherence in brain processes. *Physical Review E*, 61:4194–4206, 2000.
37. A. Thompson. *Hardware Evolution: Automatic design of electronic circuits in reconfigurable hardware by artificial evolution*. Distinguished dissertation series. Springer-Verlag, 1998.
38. A. Thompson and P. Layzell. Analysis of unconventional evolved electronics. *Communications of the ACM*, 42(4):71–79, April 1999.
39. Peter Wegner. Why interaction is more powerful than algorithms. *Communications of the Association for Computing Machinery*, 40(5):80–91, May 1997.
40. Peter Wegner. Interactive foundations of computing. *Theoretical Computer Science*, 192(2):315–351, 1998.
41. Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, pages 73–83, 1996.
42. Shlomo Zilberstein and Stuart J. Russell. Anytime sensing, planning and action: A practical model for robot control. In *Proceedings of the 1993 International Joint Conference on Artificial Intelligence*, 1993.