

# Symbols, Neurons, Soap-Bubbles and the Neural Computation Underlying Cognition

ROBERT W. KENTRIDGE\*

*Department of Psychology, University of Durham, DH1 3LE, U.K.*

**Abstract.** A wide range of systems appear to perform computation: what common features do they share? I consider three examples, a digital computer, a neural network and an analogue route finding system based on soap-bubbles. The common feature of these systems is that they have autonomous dynamics – their states will change over time without additional external influence. We can take advantage of these dynamics if we understand them well enough to map a problem we want to solve onto them. Programming consists of arranging the starting state of a system so that the effects of the system's dynamics on some of its variables corresponds to the effects of the equations which describe the problem to be solved on their variables. The measured dynamics of a system, and hence the computation it may be performing, depend on the variables of the system we choose to attend to. Although we cannot determine which are the appropriate variables to measure in a system whose computation basis is unknown to us I go on to discuss how grammatical classifications of computational tasks and symbolic machine reconstruction techniques may allow us to rule out some measurements of a system from contributing to computation of particular tasks. Finally I suggest that these arguments and techniques imply that symbolic descriptions of the computation underlying cognition should be stochastic and that symbols in these descriptions may not be atomic but may have contents in alternative descriptions.

**Key words.** Computation, dynamics, symbolic-dynamics, cognition, neural-networks.

## 1. Introduction

In this paper I attempt to relate the nature of computation in continuous systems to cognition. Initially I discuss how any dynamical system can be seen as performing computation. Drawing on examples from three very different systems, all of which can be seen as performing *useful* computation, I go on to discuss how the dynamics of these systems can be harnessed for computation. It becomes clear that, although we can characterize any dynamical system in computational terms, unless we have a prior understanding of how a system's dynamics are to be used computationally we may not be able to determine which aspects of a system's behavior constitute that computation. I then go on to describe a method developed by Crutchfield and Young [5] [6] which allows one to produce symbolic descriptions of continuous dynamical systems. *If* one can establish which variables in a system it is appropriate to measure then, in principle, this method allows one to characterize continuous computation in terms of the grammar and complexity of the symbolic algorithm effectively being implemented by the system. How can this framework help us address the nature of computation underlying cognition?

\* This research was supported by DRA Fort Halstead, U.K. Contract number 2051/047/RARDE.

It is obviously impractical to treat the brain directly as a dynamical system whose computation we wish to characterize by application of the Crutchfield and Young algorithm. We may, however, make some inferences about computation in the brain more indirectly. Evidence from a variety of sources indicates that dynamical systems can only implement computation of sufficient grammatical power to underly cognition in a critical regime bordering on chaos. Moreover, there is evidence to show that the brain's dynamics, in terms of individual neural action potentials, are maintained in this regime. As reconstruction of symbolic descriptions of behavior impose finite inaccuracies on measurements of a system's state and those inaccuracies grow as a power law of time in the critical regime we can infer that symbolic descriptions of the neural computation underlying cognition must be stochastic.

## **2. Why Is a Unified Approach to Computation Useful?**

In this section I attempt to show how systems which can be used to solve problems computationally can, for all their diversity, nevertheless be characterized and classified on a common basis. The value of this approach is that our understanding of the complexity of problems and the capabilities of 'conventional' symbolic computational systems can be brought to bear on systems such as neural networks which initially appear to be so different as to defy this type of analysis. Such analyses cannot tell us how to build a better neural network, but they can tell us whether a particular type of neural network is or is not capable of solving a particular type of problem. In particular we would like to be able to characterize the algorithms which are effectively implemented by continuous computational systems in terms of their grammatical classes in Chomsky's hierarchy [4] and their algorithmic complexity (in terms similar to those of Chaitin and Kolmogorov see e.g. [3]). The first of these characterizations allows us to determine whether a computational system is, in theory, capable of solving a particular problem. The second complexity measure allows us to see whether a computational system which can solve a small-scale version of a problem is likely to produce practical solutions to large-scale versions of the same problem.

## **3. Three Computational Systems**

Throughout this paper I will use three examples of computational systems, one digital and symbolic, one continuous and physical and one, lying somewhere between the first two, which is analog and parallel but is comprised of discrete components. The first example could be any digital computer running a symbolic program – perhaps a PDP-11 running a FORTRAN program to find the first hundred prime numbers and print them out. The second example is the analog model. Consider the problem of finding the shortest route for a road linking a number of towns and avoiding obstacles such as lakes or the sea. We could solve

this problem with a FORTRAN program, but a simple alternative method is to make models of the problem geography on two surfaces linked by pins in positions corresponding to the towns and cut away in areas corresponding to the obstacles and then to create a soap-bubble between the model surfaces. The soap-bubble will be anchored by the pins and cannot pass over the cut-away areas. The shape of the bubble formed with these constraints will show the shortest path for the road (see e.g. [11]). The third example is a neural network which has been trained to discriminate patterns of activity on sensors connected to its input units produced in the presence of a particular target substance – perhaps SEMTEX explosive. One particular *detector* unit fires when this pattern appears in the input and is quiescent otherwise. The details of the implementation and training of the neural-network are irrelevant here, but we can imagine for example that this is a fully interconnected network which has been trained using the Boltzmann machine learning algorithm [1] and is now implemented in special-purpose hardware.

#### **4. What Is Computation?**

It is very difficult to define computation in general, however, it is possible to characterize the general features of systems which perform useful computation allowing us to solve problems. The three examples above differ tremendously, however, they share basic features in the way they can be made to perform useful computation. All these systems rely on our ability to interpret some of their characteristics as having meaning – be they numbers on a screen, the path of a soap-bubble on a perspex model or the firing of a unit in a network.

#### **5. What Does the Interpretability of Computation Depend Upon?**

Our confidence in the validity of our interpretation of the outputs of a computational system is dependent on two factors. First, we must have a model of the underlying behavior of the system in general which describes how the system produces some predictable behavior. This underlying model is based on a physical implementation of aspects of propositional logic and mathematics in the case of FORTRAN, the relationship between the surface-tension of bubbles, their area, and their intersection with surfaces in the case of our analog model and the relationship between the form of inter-neural interactions and the strength of those interaction through which certain patterns of activity become attractors in the collective dynamics of networks in the case of the neural network. We use our knowledge of these underlying characteristics to produce initial states in these systems which will map problems we want to solve onto the underlying behavior of the systems. The production of these initial states is not computation – it allows computation to be useful. For the digital computer, production of the initial state involves writing and compiling the program and invoking it under the appropriate

operating system. For the analog computation, it involves making a physical model of the problem's geography and anchoring the soap-bubble with the representation of the towns on that model. For the neural network, it involves determining the connection strengths between neurons which produce the desired behavior using a training algorithm and then setting up a network (which may have a different implementation from that used in training) with those weights. The second factor affecting our confidence in the computation is the validity of this mapping. Computation then occurs as the systems' underlying behavior transforms these initial states, together perhaps with inputs to the systems, into new states which are interpretable in terms of the problem to be solved.

We can summarize *useful* computational systems as follows:

- Useful computation is in the eye of the beholder.
- It requires an underlying system of whose autonomous dynamics we have a predictive model.
- To solve a problem computational we need to map the problem to be solved onto the underlying behavior of the system and hence produce a starting state from which the autonomous dynamics of the system will produce a solution.

In more formal terms the underlying predictive system  $S$  can be defined in terms of a set  $v$  of  $m$  state variables  $\{v: v \in R\}$  (or a vector  $\{v \in R^m\}$ ) and a set of functions  $f$  in time  $\{f: f \in R^m \rightarrow R^m\}$  describing the evolution of these variables, either as difference or differential equations. We can also represent these functions in time autonomously as a set of functions  $\{f': f' \in R^{m+1} \rightarrow R^{m+1}\}$ . The problem to be computed  $P$  is defined by  $n$  state variables  $\{u: u \in R\}$  (or a vector  $\{u \in R^n\}$ ) and a set of functions  $\{g: g \in R^n \rightarrow R^n\}$ . Our aim is to produce an initial condition for  $v$  so that there is some mapping  $h$  from a subset of  $v$  onto  $u$  such that  $h(f'(v)) = g(u)$ . Notice that by treating time simply as another variable which may be involved in the mapping from  $S$  onto  $P$  there is no requirement that the problem is dynamic, although it is through dynamics that we aim to compute its solution. 'Programming' involves finding an acceptable inverse of the mapping  $h$  (which, of course, may not be possible).

This view of computation is made clearer by considering the correspondence between the underlying predictive systems  $S$  and the starting states  $v$  in our examples. The FORTRAN system is now the most complicated to describe (because it is the most sophisticated in divorcing the problem space from the underlying system). In this system we might initially think of  $S$  as an abstract machine which can be derived from the syntax of FORTRAN (e.g. a stack machine) whose dynamics can be described reasonably simply. The starting state  $v$  is a representative of the FORTRAN program in this machine, the functions  $f$  describe the transition rules between states in this simple machine. This is, however, only an intermediate representation. At base the underlying system the functions  $f$  in  $S$  comprise of laws of solid-state physics and the starting state  $v$  is a PDP-11 computer and operating system together with a translation of the FORTRAN program appropriate to that environment.

The view of computation we have been taking applies much more directly to the soap-bubble example. Here the function  $f$  in  $S$  describe the physics of surface tension and  $v$  is the geographic model with a soap-bubble passing through the pins at points on the model corresponding to towns. At the instant when the bubble first passes through the points its intersection with the model surface may not be the shortest path between the two points. Computation occurs as surface tension minimizes the area of the bubble and, as the distance between the two surfaces in the model is constant, its side length on the perspex surfaces. As the bubble is anchored to the towns and cannot form where the model is cut away it must form a path of minimum distance avoiding obstacles between the towns as it minimizes its surface area. The underlying behavior of the neural network depends on its architecture and on the learning algorithm used to train the network. The property of the neural network which determines its computational abilities in our example system is its tendency to minimize a collective measure of its activity which can be referred to as *energy* (this *energy* analogy can fruitfully be applied to a range of computational systems). In a Boltzmann machine information is stored at global, as opposed to local, minima of *energy*. As the network minimizes *energy* locally an additional process which prevents the network getting 'stuck' in local minima is necessary if the global minimum of *energy* is to be found. Simulated annealing [19], the addition of a gradually decreasing amount of noise to the activity of neurons in the network, allows the network to 'jump out of' progressively smaller and smaller local minima and hence to eventually find the global minimum (at least on a statistical basis). The combination of the process through which the network performs local gradient descent in *energy* and the simulated annealing process which allows this local gradient descent to find global *energy minima* constitute the functions  $f$  in the underlying system  $S$ . As *energy* is a function of both the activity of units and the strength of connection between them a learning algorithm can be devised which adjusts these interconnection strengths so that a selected pattern of activity corresponds to a minimum of *energy*. In the present example the pattern of input activity to be detected is combined with a freely varying pattern of activity in other units of the network so that patterns of activity corresponding to the target substance plus the output unit firing fall at a minimum of *energy* as do non-target inputs plus a quiescent output unit. In the presence of a novel input the network will tend to minimize *energy*, the similarity between the novel input and the set of inputs trained as corresponding to the target will determine whether a pattern of activity in which the output units is firing is of lower energy than one in which it is not. As the network minimizes *energy* with this new input it will settle into a state in which the activity of the output units indicate whether the inputs should be treated as target or non-target. The set of interconnection strengths produced by the learning algorithm together with the architecture of the network constitute the starting state  $v$  of the system  $S$ .

In passing we should note that the neural network could also have been

implemented as a virtual machine running on top of another system, just as the FORTRAN program was. It is, however, much easier to conceive of a direct physical implementation of the neural network than it is of the FORTRAN program.

## 6. Computation in the Eye of the Beholder

We have described an approach to the definition of computation which allows us to distinguish between 'programming' and 'computation' in non-symbolic systems. It must be emphasized that the two components of this approach cannot be separated. In all of the examples the autonomous dynamical system that performs computation could be described by many more variables than the  $m$  variables used to describe the system state. For example, in the soap-bubble system we could have measured the changing width of the soap film or the vibration modes of the bubble which are not relevant to the performance of the soap-bubble system as a path length minimizer. When we attempt to characterize a computation in terms of the dynamics of an underlying system we need to be clear which aspects of the system's dynamic behaviors contribute to the computation and which do not. In the case of the soap-bubble system the variables effecting the system's performance as a model are the positions of the towns, obstacles and the soap bubble on the surfaces of the model. The process which leads to effective computation is the minimization of the surface area of the bubble between the model's surfaces by surface tension. The crucial message we should take home from these examples is that any direct attempt to characterize the power of computation in systems which we perform computation by some unknown method is very likely to be flawed. The functions  $f$  and variables  $v$  of the underlying computational system  $S$  must be known even though the system of equations itself cannot be solved.

At this stage the situation looks bleak. Although we have a framework for understanding how the dynamics of various systems might be harnessed in order to perform useful computation, when we are faced with a system whose computational basis is unknown we appear to have little chance of discovering which aspect of the system to measure. In the next section I discuss how qualitative characterization of the computational power of systems and the tasks we expect them to perform may provide a route around this impasse.

## 7. Characterizing Computation in Dynamical Systems

We have described an approach in which all computation is regarded as potentially continuous dynamics. We would like, however, to produce descriptions to which we could apply common symbolic characterizations of computation such as algorithmic complexity [3] or classes of grammars in a Chomsky's [4] formal

language hierarchy. Crutchfield and Young [5] describe a method in which the symbolic dynamics of continuous systems can be reconstructed as a non-deterministic automaton to which the symbolic characterizations of computation mentioned could be applied. Crutchfield and Young's method depends to a certain extent on the assumption that the dynamical system under investigation has had some noise added to it, however, this is a perfectly valid assumption in any physical implementation of a computational system.

The goal of symbolic dynamics is to find the most compact predictive description of the behavior of a system. For example, if a series of states always follow one another they can be collapsed into a single state for the purposes of symbolic description. In Crutchfield and Young's procedure that state space of the system is divided into a large number of labeled cells, at discrete time intervals the state of the system is noted, the label of the cell it currently occupies being appended to a string a symbols describing its discretized dynamics. The string thus produced is a symbolic description of the system's dynamics, it is, however, far from the most compact description possible. Rather than replacing common strings of labels with individual symbols all the sequences of labels up to a given length are arranged in a tree in which the edges are marked with the probabilities of transitions between labels and all subtrees of a given smaller depth are assigned symbols. Subtrees are assigned identical symbols if their component labels and their structures are identical and if the probabilities marked on their edges differ by less than a chosen criterion. A directed graph which describes the allowable transitions between symbols is then produced by identifying transitions between labels in the original tree which are members of particular subtrees corresponding to symbols. This procedure will produce different graphs depending on the depth of subtrees used to generate symbols, the criterion used in assigning common symbols to subtrees whose edge probabilities differ and, of course, the original spatial and temporal discretizations used to construct the string of labels from the system's continuous dynamics. As the edges in the reconstructed directed graphs are labeled with transition probabilities it is possible to measure the degree of indeterminacy in these graphs. The best symbolic description of a system's dynamics minimizes this measure allowing us to use this criterion to choose a single symbolic description from the range of graphs which can be produced.

The result then is a representation of the system's symbolic dynamics as a directed graph with probabilistically labeled edges—a stochastic finite state automaton. If the grammar which the system is obeying is higher than a finite state grammar in the Chomsky hierarchy (e.g. it is context-free, context-sensitive or unrestricted) then the automaton produced by the Crutchfield and Young procedure will be infinite. Regularities in the infinite automaton produced allow one to infer which higher level of grammar is required in order to produce an equivalent finite machine description of the system's symbolic dynamics. Moreover, we can apply complexity measures to the machine description produced and hence characterize the algorithms effectively being implemented by continuous

dynamical systems both in qualitative (grammar type) and quantitative (machine complexity) terms (see, e.g. [6]).

## 8. Cognition and Neural Computation

In the previous section we saw that it is possible to characterize computation in continuous systems in familiar symbolic terms of grammar and complexity. Suppose that we have a description of a task to which we believe a certain system is computing a solution. If we can characterize the task in terms of its grammatical requirements then we can at least be sure that any measurement of the system which produces computation falling into a lower class in the Chomsky hierarchy than the task cannot form the basis of the task's solution. On this basis we could at least rule out some aspects of a system's dynamics as producing computation which was too weak to solve the task at hand.

The task I now wish to consider is cognition, and the system in which I expect to discover its computational basis is the brain. It is, of course, impractical to apply the Crutchfield and Young algorithm to the brain. We can, nevertheless, still make some inferences about neural computation and cognition from our understanding of the grammatical requirements of language, the types of systems which can implement such grammars and the type of symbolic descriptions of them which would be produced by the Crutchfield and Young algorithm were it practical too apply it.

As language is part of cognition, and we can easily understand sentences which can only be parsed by higher level grammars [24] then it seems reasonable to assume that cognition depends on computation beyond the level of finite state automata (it has even been suggested by Hintikka [10] that natural language abilities lie quite outside the Chomsky hierarchy (i.e. natural language grammars are not recursively enumerable), however Manaster-Ramer and Yu [21] argue that this might be a quality of language but not of linguistic performance).

Crutchfield and Young [6] have shown that certain simple non-linear dynamical systems (non-linear maps with negative Schwartzian derivatives) can only produce computation at a level beyond the base of the Chomsky hierarchy at transitions to chaos. It is, however, something of a leap of faith to infer from this that the implementation of cognition in our brains also requires dynamics at the transitions to chaos (brains are, after all, more complex dynamical systems than 1-dimensional maps). There are, nevertheless, a number of other lines of evidence which suggest that non-trivial computational properties only arise in more complex systems, at phase transitions, for example, Packard's work on complex behavior and adaptation in cellular automata rule space [23] [20] (but see also [22] which indicates that Packard's task may have been an inappropriate test of his thesis) or Kauffman's on random boolean networks [12] [13] [14]. Of course, we can also look directly to the brain for evidence that it operates at the boundaries of chaos. Recently Kelso's group [15] reported that magneto-encephalographic measure-

ments of human brain activity during a rhythm tracking task showed meta-stable spatio-temporal patterns of activity around a phase transition. Less direct evidence for critical brain dynamics can also be gleaned from electro-encephalographic power-spectra from a variety of sites [7] [25] [8] which exhibit the  $1/f$  pattern typical of systems with critical dynamics [2]. I have shown [17] [16] that a critical state can be produced in physiologically realistic neural network models through a process of self-organizing criticality.

The evidence cited above suggests that we could profitably consider what the implications are for symbolic models of cognition of an underlying implementation in a brain operating at a phase-transition. Although the brain is composed of discrete components we will treat it as if it were continuous system as we might expect the scale of symbols which describe cognition to be much greater than that of individual synapses.

## 9. Qualitative Effects of Approximation

Crutchfield and Young's algorithm allows us to produce a description which is a discrete approximation of a continuous system. In some cases this approximation may be perfect, however, often discretization leads to an unavoidable loss of information in the system's representation. This loss of information can produce qualitative differences between the continuous and discrete descriptions of the system. In particular, a chaotic continuous non-linear dynamical system *or one at the edge of chaos* is deterministic, however, its symbolic representation is inevitably probabilistic. The initial discretization of its state space imposes a finite precision on our knowledge of the system's state which grows exponentially as the system evolves. At a theoretical level this difference is unsurmountable, no symbolic computational system can recreate chaotic or critical dynamics with perfect accuracy. If, however, our concern is with physically realizable systems then we may argue that these too face the same problem. We cannot observe the state of a physical system at any given time without uncertainty (assuming that measuring state involves measuring the position of part of the system, e.g. electrons in the digital computer, soap-film in the analog system or electrons again in the physical implementation of the neural network, see e.g. [9]). In both cases the underlying systems, be they abstract or physical, are performing deterministic dynamics, however, our interaction with them through symbols or measurements is probabilistic. The first conclusion we can draw about symbolic models of cognition therefore is that it should be stochastic (given a brain exhibiting critical dynamics).

A more interesting difference between symbolic and continuous descriptions in these cases is that the continuous description of a system is unique while many different probabilistic symbolic descriptions can be produced depending on the 'grain-size' of the initial discretization of the continuous system. These symbolic descriptions may not all be the absolute optimal description in terms of their

graph indeterminacy, nevertheless, if constraints are put on the 'grain-size' of discretization then distinct optimal graphs can be produced given particular constraints. The upshot of this is that many levels of non-equivalent symbolic descriptions of a single system are possible. This has particularly interesting implications for the relationship between symbolic models of cognition and the brain. It suggests that no deterministic symbolic model can capture neural computation adequately and that symbols in such models cannot be assumed to be atomic, but rather that they may always have content at some more fine-grained level of description (see also, [18]).

## 10. Conclusion

This paper has not attempted to define computation directly, but in the process of describing a method of characterizing computation in a diverse range of systems a definition has emerged. Computation is the behavior of a dynamical system. Unfortunately, computation is only useful if we know how that system's dynamics can be harnessed to automate the solution of a problem and acquiring this knowledge may be beyond us in most cases. The intrinsic computation of a system in terms of a particular measurement of its dynamics can, however, in principle, be characterized in terms of its grammatical class and effective algorithmic complexity regardless of its usefulness. When the grammatical characteristics of the task a system is assumed to be computing are known, then some measures of the system may be ruled out as the basis of that computation if the grammatical class of the dynamics of those measurements falls below that of the task in the Chomsky hierarchy. Although it is quite impractical to apply this technique to the brain we can still infer that some aspect of the brain's dynamics which is maintained near the transition to chaos underlies cognition. Concise symbolic descriptions of this computation must be stochastic, moreover, it may be profitable to treat symbols in such descriptions as non-atomic – potentially having content in a different symbolic characterization of the same underlying computation.

## References

1. D.H. Ackley, G.E. Hinton, and T.J. Sejnowski (1985), 'A Learning Algorithm for Boltzmann Machines', *Cognitive Science* **9**, 147–169.
2. P. Bak, C. Tang, and K. Wiesenfeld (1987) 'Self-organized Criticality: An Explanation of 1/f Noise', *Physical Review Letters* **59**, 381–384.
3. G. Chaitin (1975), 'Randomness and Mathematical Proof', *Scientific American* **233**, 47–52.
4. N. Chomsky (1963), 'Formal Properties of Grammars', in R.D. Luce, R.B. Bush, and E. Galanter, eds., *Handbook of Mathematical Psychology*, Volume 2. Wiley.
5. J.P. Crutchfield and K. Young (1989), Inferring Statistical Complexity. *Physical Review Letters* **63**: 105–108.
6. J.P. Crutchfield and K. Young (1990), Computation at the Onset of Chaos. In W.H. Zurek, (ed.)

- Complexity, Entropy and the Physics of Information.* (Santa Fe Institute Proceedings Volume VIII). Addison-Wesley.
7. W.J. Freeman and B.W. van Dijk (1987), 'Spatial Patterns of Visual Cortical Fast Eeg During Conditioned Reflex in a Rhesus Monkey.' *Brain Research* **422**, 267–276.
  8. F. Grúncis, M. Nakao, Y. Mizutani, M. Yamamoto, M. Meesmann, and T. Musha (1993), Further Study on 1/f Fluctuations Observed in Central Single Neurons During Rem Sleep. *Biological Cybernetics* **68**, 193–198.
  9. W. Heisenberg (1962), *Physics and Philosophy*. Harper and Row, 1962.
  10. J. Hintikka (1977), 'Quantifiers in Natural Languages: Some Logical Problems ii.' *Linguistics and Philosophy* **1**, 153–172.
  11. C. Isenberg (1978), *The Science of Soap Films and Soap Bubbles*. Tieto.
  12. S.A. Kauffmann (1989), 'Adaptation on Rugged Fitness Landscapes' In D.L. Stein, editor, *Lectures in the Sciences of Complexity*, volume 1 of Santa Fe Institute Studies in the Science of Complexity: Lectures Series. Addison-Wesley.
  13. S.A. Kauffmann (1989), 'Principles of Adaptation in Complex Systems', in D.L. Stein, ed., *Lectures in the Sciences of Complexity*, volume 1 of Santa Fe Institute Studies in the Science of Complexity: Lectures Series. Addison-Wesley.
  14. S.A. Kauffmann (1991) 'Antichaos and Adaptation', *Scientific American* **265**, 64–70.
  15. J.A.S. Kelso, S.L. Bressler, S. Buchanan, G.C. DeGuzman, M. Ding, A. Fuchs, and T. Holroyd (1992), 'A Phase Transition in Human Brain and Behavior' *Physics Letters A* **169**, 134–144.
  16. R.W. Kentridge (1993), 'Dissipative-structures and Self-organizing Criticality in Neural Networks With Spatially Localised Connections' In F. Eeckmann and J.M. Bower, (eds.) *Computation and Neural Systems*, 531–535. Kluwer Academic Publishers.
  17. R.W. Kentridge (1994), 'Critical Dynamics of Neural Networks With Spatially Localised Connections' In M. Oaksford and G. Brown, (eds.) *Neurodynamics and Psychology*. Academic Press.
  18. R.W. Kentridge (in press) *Cognition, Chaos and Non-deterministic Symbolic Computation: The Chinese Room Problem Solved? Think*.
  19. S. Kirkpatrick, C.D. Jr. Gelatt, and M.P. Vecchi (1983), 'Optimization Using Simulated Annealing' *Science* **220**, 671–680.
  20. W. Li, N.H. Packard, and C.G. Langton (1990), 'Transition Phenomena in Cellular Automata Rule Space' *Physica D* **45**, 77–94.
  21. A. Manaster-Ramer and Q. Yu (1993), 'Linguistic Mechanism, Physical Mechanism and the Secondary Non-r.e.ness of the Physical World' In *Workshop on Physics and Computation PhysComp '92*. Proceedings of the Workshop on Physics and Computation, October 2–4, 1992, Dallas, Texas., 24–33. IEEE Computer Society Press.
  22. M. Mitchell, P.T. Hraber, and J.P. Crutchfield (1993), 'Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations' *Complex systems*, submitted.
  23. N.H. Packard (1988), 'Adaptation at the Edge of Chaos' In J.A.S. Kelso, A.J. Mandell, and M.F. Schlesinger, (eds.) *Dynamic Patterns in Complex Systems*. World Scientific.
  24. T. Winograd (1983), *Language as a Cognitive Process*. Volume 1: Syntax, Addison Wesley.
  25. M.P. Young, K. Tanaka, and S. Yamane (1992), 'On Oscillating Neuronal Responses in the Visual Cortex of the Monkey', *Journal of Neurophysiology* **67**, 1464–1474.