The External Tape Hypothesis:
A Turing machine based approach to cognitive computation.

Andrew Joseph Wells

Thesis submitted for the degree of Ph.D.

The London School of Economics and Political Science.

Abstract.

The symbol processing or "classical cognitivist" approach to mental computation suggests that the cognitive architecture operates rather like a digital computer. The components of the architecture are input, output and central systems. The input and output systems communicate with both the internal and external environments of the cognizer and transmit codes to and from the rule governed, central processing system which operates on structured representational expressions in the internal environment. The connectionist approach, by contrast, suggests that the cognitive architecture should be thought of as a network of interconnected neuron-like processing elements (nodes) which operates rather like a brain. Connectionism distinguishes input, output and central or "hidden" layers of nodes. Connectionists claim that internal processing consists not of the rule governed manipulation of structured symbolic expressions, but of the excitation and inhibition of activity and the alteration of connection strengths via message passing within and between layers of nodes in the network. A central claim of the thesis is that neither symbol processing nor connectionism provides an adequate characterization of the role of the external environment in cognitive computation. An alternative approach, called the External Tape Hypothesis (ETH), is developed which claims, on the basis of Turing's analysis of routine computation, that the Turing machine model can be used as the basis for a theory which includes the environment as an essential part of the cognitive architecture. The environment is thought of as the tape, and the brain as the control of a Turing machine. Finite state automata, Turing machines, and universal Turing machines are described, including details of Turing's original universal machine construction. A short account of relevant aspects of the history of digital computation is followed by a critique of the symbol processing approach as it is construed by influential proponents such as Allen Newell and Zenon Pylyshyn among others. The External Tape Hypothesis is then developed as an alternative theoretical basis. In the final chapter, the ETH is combined with the notion of a self-describing Turing machine to provide the basis for an account of thinking and the development of internal representations.

Table of Contents.

List of Figures.

Acknowledgments.

I would like to thank a number of people for their help. My parents Joan and Arthur Wells both provided invaluable support when I began to study psychology and continue to do so. John McShane initially agreed to supervise the thesis and Ric Seaborne valiantly took on the burden when John left the department for a research post in industry. John's untimely death in a boating accident was a profound shock to all his friends and colleagues. He is very much missed. Julie Dockrell, who has since left for fresh pastures, wore the mantle of departmental admissions tutor for longer than might otherwise have been the case. Laura Markowe was a companion on many a Saturday morning. Bradley Franks read draft chapters and pressed the case for diagrams. As the only other cognitive scientist in the department he has been an important source of moral and intellectual support throughout. Liz and John Valentine provided staunch support and friendship at a critical time. Mike Oakesford also gave generous and valuable assistance. Simon Roberts has twice provided timely, insightful advice of the greatest clarity with humour and sympathy. Many other friends and colleagues have helped me directly and indirectly; in particular, Ian Dickson, Farquharson Cousins, Nick Braisby, Richard Cooper and Steven Haynes. Most importantly, I want to thank my wife Mia for everything. Without her the thesis might never have been completed.

Chapter 1. Introduction.

Cognitive science has recently enjoyed a resurgence of main stream interest in a style of theorizing and computational modelling known variously as "connectionism", "neural networks" and "parallel distributed processing", which has existed in one form or another since the pioneering work of McCulloch and Pitts in the 1940's. Within the connectionist[1] framework, there are numerous, sometimes substantial, differences of approach, but there is a unifying, shared belief that the rule based, symbol processing approach to cognitive computation, which has been dominant since the early days of computer modelling of mental structures and processes, has fundamental limitations. Proponents of the symbol processing approach have responded to the challenge of connectionism and a substantial debate about the foundations of cognitive science has been engendered. Since connectionism and symbol processing are both *computational* approaches to the study of cognition, one way to characterize the major differences between them is in terms of their relationship to Turing's machine model of computation which constitutes one member of the class of formally equivalent, basic models of routine computation. Roughly speaking, proponents of the symbol processing approach argue that the brain implements a Turing machine and that the mind can be described as a system in which symbolic expressions are processed according to rules. By contrast, connectionists argue that Turing's machine model is, by and large, irrelevant to the study of cognitive computation and urge the adoption of what is sometimes called "brain style" modelling, in which closer attention is paid to the neural basis of cognitive computation. At least one connectionist theorist, Smolensky (1988), has argued that connectionism may challenge the view that the class of well defined computations is co-extensive with the class of Turing machine computations. The position defended in this thesis claims that neither of these views is entirely correct and that a third approach based closely on Turing's original

---

[1]Throughout this thesis I have used the term "connectionism" as a generic label for the whole framework.

analysis can illuminate both the others.

1.1 Giant electronic brains.

The possible links between computers and brains have been evident from the start, and in the early days of their development, digital computers were sometimes called "electronic brains" (cf. Bowden, 1953). Not many people knew much about them, but they were said to be capable of prodigious feats of calculation far outstripping the capacities of the apparently rather feeble human intellect.

> "These machines are similar to what a brain would be if it were made of hardware and wire instead of flesh and nerves. It is therefore natural to call these machines *mechanical brains*. Also, since their powers are like those of a giant, we may call them *giant brains*."
>
> Berkeley (1949, p.1)

The early optimism, although accompanied by some limited successes in the production of apparently "intelligent" behaviour, quickly gave way to the realization that simulating the full scope of human cognition was a task of daunting scale and complexity. Curiously, perhaps, the capacity for more "intellectual" tasks such as chess and theorem proving turned out to be much easier to capture than the mundane business of "peripheral" tasks such as perception. The soubriquet "electronic brain" fell into disuse and unflattering comparisons were made, and continue to be made, between the computer metaphor and other, earlier technological metaphors. It is a commonplace that successive generations try to explain the mind in terms of their own technological innovations, and perhaps the sceptics are right to insist that the "computational metaphor" is just that; a metaphor which will, in due course, be seen to be no more substantial than its predecessors. The greater the effort made, the harder it seems to be to get a computer to produce anything like the flexibility of behaviour which is a characteristic product of the brain's activity.

1.2 Alan Turing and the universal computing machine.

Despite the practical failure to realize early hopes, many cognitive scientists are convinced that computational ideas form a much more principled basis for the study of cognition than the term "metaphor" implies. The basis of this conviction lies in the work of Alan Turing and other mathematical logicians of the 1930's who developed formal models of the processes involved in routine computation. In his seminal paper "On Computable Numbers, With An Application To The Entscheidungsproblem", Turing showed that machines were, in principle, behaviourally much more flexible than had previously been suspected. He did this by providing a detailed logical blueprint for the construction of a machine which he called the universal computing machine. In the context of his paper, the machine was applied to the computation of functions of real numbers, but its true significance lay not in its initial realm of application but in its modus operandi. The machine merited the adjective "universal" because it was a general purpose instruction obeying mechanism which could carry out any task for which a complete and unambiguous set of instructions could be written. Turing showed that the specification of a set of task instructions was formally equivalent to the specification of a machine to carry out that task, and hence that a general purpose instruction follower was ipso facto a general purpose machine because it could do, by simulation, any task which could be done by a machine built specially for the purpose.

1.3 Logical and Physical Descriptions.

One of Turing's central ideas was "that mental processes are correctly described in the logical model independently of the particular physical embodiment, and so can be embodied in a physical form other than the brain." Hodges (1988,p.9). It may be useful to think of this idea as proposing a two tiered explanatory hierarchy and hence as an intellectual forebear of Marr's (1982) influential three level model. Turing's logical model corresponds to Marr's algorithmic level, i.e. the formal specification of what needs to be done to complete the task in hand, and Turing's

physical embodiment description corresponds closely to Marr's implementation level[2].

A fundamental implication of Turing's work for cognitive science is that the operations of the mind, at least in so far as it engages in thinking and reasoning, can be described functionally in terms of sets of instructions or programs, independently of their medium of realization. This leads, at a minimum, to the possibility of studying mental processes with computer simulations, because a computer program could mimic the "relation structure" (Craik 1943) obtaining among the elements of a mental process. More importantly, if mental processes are correctly characterized functionally, a powerful argument can be developed for the independence of psychology from neuroscience.

1.4 Artificial Intelligence and Physical Symbol Systems.

A further radical consequence of Turing's work which obtains if the criterial features of mentation are purely functional, is that an appropriately programmed computer would be as capable of cognitive processing as a human cognitive agent. This possibility provides a theoretical basis for the discipline of Artificial Intelligence and for the Physical Symbol Systems hypothesis (PSSH), first articulated by Newell and Simon (1976). According to this hypothesis, computers can quite properly be thought of as "electronic brains", because brains and computers have common organizational principles. This claim leads quite naturally to a symmetrical claim about brains, viz. that they are "biological digital computers". On this approach, human behavioural flexibility depends primarily on the fact that the brain is computationally universal, rather than, for example, on its being made of a biologically responsive stuff.

The extent to which mental processes can be studied independently of their medium

---

[2]This invites a question about what, if anything, in Turing's work corresponds to Marr's computational level. For Marr, (1982,p.24), this level is "an abstract computational theory", which specifies the goal of a computation. The goal of Turing's 1936-37 paper was to provide a means "to somehow survey the class of all possible algorithms" Davis (1988a,p.154) so as to tackle "the Hilbertian Entscheidungsproblem", Turing(1936-37,p117). Turing's algorithmic specification of a universal machine showed, among other things, that the problem had no solution.

of implementation is an interesting and important question. Turing's abstraction of the logical steps of a computation from the details of its implementation certainly allowed him to describe the universal machine purely functionally in terms of finite sets of discrete states and discrete symbols. However, his analysis did not show, nor did he claim, that there were no physical constraints on the kinds of substrate which could be used to generate mind like behaviour using computational structures.

Consider, as an uncontroversial example, the can opener, whose function is to enable its operator to gain access to the contents of a sealed metallic container. The fact that there is more than one way of implementing the can opening function suggests, by analogy with Turing's treatment of mental processes, that "*can opening processes are correctly described in the logical model independently of the particular physical embodiment, and so can be embodied in a physical form other than the usual metallic one.*" While this is clearly true, and allows for diamond can openers, laser light can openers among others, it does not follow that can openers can be made from anything at all. The design of a can opener is constrained by facts such as the relative hardness of different materials which are relevant to the performance of the task. It would, for example, be difficult to make a can opener from rice pudding or from string and brown paper simply because these materials are too soft. Constraints on the suitability of various media for can opening arise from a clear understanding of what the functional description of can opening implies for its practical implementation, given the nature of the physical world as we currently understand it to be. Conversely, a given medium will be capable of supporting some functions but not others and an understanding of the nature and limitations of a medium will illuminate the study of the functionality it can support. A six storey building can be built of brick, but a sixty storey building cannot, because the structure will collapse under its own weight.

Given that a function as simple as can opening has constraints limiting the media in which it can be realized, it is inevitable that there will also be constraints on the media in which the complex functionality of minds can be realized. It is also highly

likely that the functions of the mind will be illuminated by a study of its medium of implementation, and it is possible that various hypotheses about the nature of mind, including the hypothesis that the mind is a computer, will be ruled out when the functionality of its medium of implementation is understood. Unfortunately, there is a regrettable tendency on the part of some cognitive theorists to move prematurely from the conditional "If the mind is a computer then mental processes can be studied independently of their medium of realization" to the bald assertion that "because the mind is a computer psychology is independent of neuroscience". One of the possibilities which needs to be considered is that neural tissue is not the sort of stuff in which binary switches can be implemented and hence that the mind is not a computer as they are commonly understood.

1.5 Turing machine architecture and the generic computer theory of mind.

Despite the cautionary tale about functionalism above, a computational account of mental phenomena is widely expected to succeed, and various theories of cognitive architecture have been developed on the basis of the idea that the brain realizes a Turing machine or digital computer of some kind. Turing machines and their computations are defined formally and discussed in Chapter 2. Informally, a Turing machine is a system consisting of two parts, a finite control machine and a potentially infinite, one dimensional tape. The tape is divided into squares, each of which can be blank or can contain a token of one of a finite alphabet of symbols. The control machine has access to the contents of just one square of the tape at any moment and can move along the tape in either direction. The control is equipped with a scanner with which it recognizes symbol tokens, and a printer with which it writes them. The single tape square to which the control has access is called the "scanned square" and the symbol on it is called the "scanned symbol". The control machine has a finite number of internal states and its operations are usually described in a machine table. Because the machine works with a finite symbol alphabet and has a finite number of internal states, its operations can be exhaustively described by specifying how it behaves for each combination of internal state and scanned symbol. The essential characteristics of a Turing machine are the distinction between the tape and the control, the potential infinitude of the tape and

the fixed finite nature of the control. The tape serves as a symbolic memory. Digital computers are essentially practical versions of Turing machines, i.e. they exhibit the separation of control from memory but their memories are more tractable than uni-dimensional tapes. They are also equipped with transducers which translate external inputs into symbolic codings and vice versa. The keyboard and VDU of a computer are transducers. In normal operation, information about the outside world consists exclusively of symbolic representations of stimulus energies written into memory by the transducers.

The first digital electronic computer, the ENIAC, spent much of its time idle because it suffered from a severe information input bottleneck as a result of its tiny internal memory. With the expansion of high speed memory in later machines, and with the development of the stored program concept it became possible to incorporate more and more of what was needed to complete a computation in high speed memory. The whole thrust of computer design thinking from von Neumann onwards was to cut down on the need for frequent communication with the low speed outside world. The goal was the communication of a completely specified task, i.e. both program and data, in a single transaction from the outside world, such that the machine could then complete its computation without the need for further external intervention. A properly functioning computer is a system in which everything of computational interest happens inside the machine. The environment serves as a source of data and as a repository for results, but computationally, the machine is autonomous.

Proponents of the physical symbol systems hypothesis and related theories are committed to the view that the brain implements a practical version of the Turing machine architecture just as digital computers do. Theoretical differences of course abound, but the commitment to a fundamental architectural isomorphism between minds, digital computers and Turing machines is a hall mark of what may usefully be called the "generic computer theory of mind", or "generic theory". According to the generic theory the senses provide input to, or function as, transducers which produce representations in the symbolic code with which the brain-ware Turing

machine does its computations. The cognitive computer does not have direct access to its environment but only to symbolic representations of it. Putnam (1975) provides a very clear example;

> "The Turing Machines I want to consider will differ from the abstract Turing Machines considered in logical theory in that we will consider them to be equipped with sense organs by means of which they can scan their environment, and with suitable motor organs which they are capable of controlling. We may think of the sense organs as causing certain 'reports' to be printed on the tape of the machine at certain times, and we may think of the machine as being constructed so that when certain 'operant' symnbols are printed by the machine on its tape, its motor organs execute appropriate actions. This is the natural generalization of a Turing Machine to allow for interaction with an environment."
>
> Putnam (1975, p.409)

Putnam was one of the first theorists to develop the comparison between Turing machines and minds and his work has been very influential. One of the major claims of this thesis is that Putnam's and other similar models are not in fact the most "natural generalization" of a Turing machine for purposes of cognitive modelling.

Jerry Fodor's (1980) arguments for methodological solipsism as a research strategy provide another example of the way in which ideas rather similar to Putnam's can inform the development of a cognitive theory. Fodor argues that computational processing depends purely on the syntactic relations among the elements of a computational system. This is very clear in the case of a Turing machine. Fodor also assumes that the mental Turing machine communicates with its environment solely by having access to "oracles"[3] which enter symbolic representations of ambient environmental energies on to parts of the tape from time to time. Following Putnam, Fodor describes this model as "a natural extension of the computational picture of the mind", from which it follows that if computational

---

[3]The term "oracle" was first used, in roughly Fodor's sense, by Turing in his 1939 paper on ordinal logics.

processes are purely formal then "the bearing of environmental information on such processes is exhausted by the formal character of whatever the oracles write on the tape", and the proper study for psychology is just the character of the internal symbolic representations. The argument is similar to one which he deploys for slightly different purposes in "The Modularity of Mind", as follows;

> "...if, as many of us now suppose, minds are essentially symbol-manipulating devices, it ought to be useful to think of minds on the Turing-machine model ...However ...Turing machines are *closed* computational systems...the rest of the world being quite irrelevant to the character of their performance...If, therefore, we are to start with anything like Turing machines as models in cognitive psychology, we must think of them as embedded in a matrix of subsidiary systems which affect their computations in ways that are responsive to the flow of environmental events."
>
> Fodor (1983,pp.38-39).

Newell (1980) provides a further example of the architectural assumptions of the generic theory. A physical symbol system, according to Newell, communicates with its environment by means of two operators "input" and "behave". Newell acknowledges that "without some reliable transduction from external structure to symbols, the symbol system will not be able to produce reliable functional dependence on the external environment", Newell (1980,p.167), but he does not find this at all problematic. He assumes that reliable transduction will be achieved and that "From a formal viewpoint, the operation of these two operators [input & behave] can just be taken as given, providing in effect a *boundary* condition for the internal behavior of the system." Newell (1980,p.147).

In general, accounts of cognitive architecture belonging to the generic theory tradition propose two sets of scanners. The first set consists of sensory transducers which convert ambient stimulus energy into tokens of symbols in the internal cognitive code, and the second set consists of mechanisms which read, write and manipulate expressions in the cognitive code produced by the sensory transducers. Cognitive computation is thus isolated from direct contact with sensory processes.

1.6 Connectionism and cognitive architecture.

Connectionists are unconvinced by a number of characteristics of the generic account of the basic structuring of human cognitive architecture. Some theorists (e.g. Kohonen, 1988), explicitly question the suggestion that the brain has mechanisms reliable enough to support the complex articulated data structures, random access location addressing and extended chaining of logical steps which are characteristic of digital computation. It is of interest to note that von Neumann expressed similar reservations as early as 1948 when he argued that the "logical depth" of digital computation was of a different order from that of human computation and that a new system of logic would be needed to understand complex automata of both natural and artificial origins. It is not widely recognized by psychologists[4] how farsighted von Neumann was in this respect. Those who believe that the Boltzmann machine (Hinton & Sejnowski, 1986) represents an entirely new departure will find food for thought in the following;

> "...there are numerous indications to make us believe that this new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is thermodynamics, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information."
>
> von Neumann (1987,p.407).

Apart from questions about the nature of the basic mechanisms, most connectionists take issue with the notion that cognitive computation consists of the manipulation of explicit symbolic representations, and numerous connectionist counter proposals to the generic view are now being developed. However, much connectionist work consists of the development of models of specific cognitive capacities using feedforward networks and the backpropagation learning algorithm and it is not clear exactly how to relate such models to each other or to models derived from generic theory. There is, as yet, no fundamental theory which unites connectionist research in the way in which Turing machine theory unites proponents of the generic theory,

---

[4]For an exception, see Boden (1988, p.2).

although useful theoretical work is being done, cf. Smolensky (1988), Churchland (1989), Bechtel & Abrahamson (1991), Clark (1993). One of the claims of the thesis is that a proper understanding of the nature of Turing's analysis of computation provides an account of the conceptual linkage between connectionism and the generic theory.

A fuller discussion of connectionism and its relations to the External Tape Hypothesis is given in chapter 7. As an introduction to the approach taken here, it is useful to think of connectionist models as having two distinct phases, which might be called the training phase and the operational phase. One common way in which a connectionist project is conducted is to identify a cognitive capacity, such as the capacity to form the past tenses of verbs or to exhibit the functional asymmetry of the Stroop colour word naming phenomenon, as a target. A network with a given basic architecture is then trained using a suitable corpus of exemplars until a predetermined criterion of performance is reached with respect to that target capacity. During the training phase, the strengths of the connections between nodes in the network are modifiable. When criterial performance has been achieved, connection strength modification is discontinued and the structure can be considered fixed. During the operational phase, the network's capacity to produce the desired associations is assessed. In some cases, such as the modelling of past-tense acquisition, the training phase is of primary psychological interest as a model of part of the process of language acquisition. In such a case the performance of the network in the operational phase is significant only in so far as it provides a measure of the extent to which performance criteria have been met. In other cases, for example models of concept acquisition, the operational phase is of more psychological interest and the training phase is used solely to induce the connectivity needed to establish the set of input-output paired associations which constitute the basic performance of the system. What is generally then of most interest is the study of the extent to which performance generalizes to unfamiliar inputs and the exploration of the internal structures of connectivity which support such performance. These have been investigated using techniques such as cluster analysis and principal components analysis and are explored for the insights they might

provide into the way conceptual information is stored in the human cognitive system. Another technique which has been used is to alter the connectivity so as to simulate the effect of a brain lesion, for example, and to note whether observed changes in performance bear some measurable similarity to the pathology of actual cases of brain damage.

The question which is raised briefly here, and discussed in more detail later, is how a network might best be characterized when it is considered as a fixed, entity, i.e. after its training has been completed, or, when a snapshot is taken of its structures at some moment during the training regime. The proposal made is that a fixed connectionist network, i.e. one which is not learning, is accurately construed as a finite state machine. The argument for this view is based on considering the set of inputs and outputs which such a network can handle, and the character of its internal activation states when it is in operation. In the light of the characterization of fixed nets as finite automata, the learning process can be thought of as transforming an initial finite automaton which may not compute a recognizable function, but none the less computes some function, into one which computes, to a more or less accurate degree, the associative function specified by the set of input-output pairs which form the corpus of training examples. The corpus of training examples, as well as providing data for the machine, also provide the means of specifying the error feedback which the machine receives at each training step.

If this view is correct it suggests that in so far as connectionist models are intended to have realist interpretations they can be construed broadly as proposing that the brain is a finite state automaton. The importance of this point is that it agrees very closely with the view of Turing's analysis of computation which the thesis argues is the proper view to take. The basic operation of a finite automaton is the state transition; this is a rule following operation, but it need not be explicitly rule governed. It is relatively easy to think of neurally plausible implementation schemes for state transitions. It seems quite reasonable, for example, to think of a state transition as a function from one stable pattern of neural firing to another, or as a network energy function of a Boltzmann machine. If we do think in this way, and

identify patterns of activation as the internal states of a connectionist network, then a system with n processing units or nodes each capable of k activation states, may be thought of as a finite state automaton with up to k^n internal states. These states are determined by the activation function thresholds and connection strengths of the nodes in the network. A learning rule applied to a network alters the set of activation patterns which the network produces, primarily by modifying the weighted connections between pairs of nodes. Thus a learning rule changes the set of internal states of an automaton. Since an automaton is identified by its internal states and its state transition table, changing the internal states amounts to constructing a new automaton. Both supervised and unsupervised connectionist learning techniques can be seen in this light as methods for constructing finite automata.

1.7 Implications of the finite state account of connectionist models.

Connectionists have not yet gone far enough in following through the implications of the above account because they have not considered sufficiently carefully how radically the functional relationship between the machine in the head and the external stimulus environment changes when the internal machine is hypothesized to be a finite state automaton rather than a Turing machine. In the latter case, the external environment is of limited direct interest because the computational environment of a Turing machine is its tape and the tape of the cognitive Turing machine is hypothesized to be in the head. If the internal machine is considered to be a finite state automaton the situation is very different. The fundamental separation of memory from control seen in a Turing machine is absent and an independent internal memory capable of storing symbol tokens representing states of affairs in the external environment is not essential. The only mandatory internal machinery is that required to manage state transitions. Finite automata use the same set of internal states for control and for memory and state transitions are directly driven by the input. The neural networks of McCulloch and Pitts (1943) provide an example. Such networks, which do not learn, are equivalent to finite state automata; a finite state automaton can be constructed to compute the same function as any neural network and vice versa (Minsky, 1967). This point may be taken to

support the claim made above about the characterization of connectionist networks when their connections are considered fixed. Neural networks are not equivalent to Turing machines although the claim that they are has sometimes been made. McCulloch and Pitts were clear about this and their original paper bears careful reading. What they actually say is this;

> "every net, if furnished with a tape, scanners connected to afferents, and suitable efferents to perform the necessary motor-operations, can compute only such numbers as can a Turing machine".
>
> McCulloch & Pitts (1943, p.37)

Thus, a neural net **connected to an appropriate environment** computes like a Turing machine. This suggests that if the brain is a neural network and hence also a finite state automaton, then, to argue that the cognitive system has the computational power of a Turing machine, should require an argument to the effect that aspects of the external environment constitute a tape, and that the sensory and motor pathways of the peripheral nervous system provide the scanners and transducers. To a first approximation, the argument should be that human cognitive architecture is a synergetic system of organism and environment realizing a Turing machine. That seems to be the direction in which connectionist philosophy naturally leads.

It is not clear that connectionists do, in fact, give the real environment such a prominent place in their models. Consider the following passage from Rumelhart, Hinton & McClelland (1986). Having argued in line with the suggestion above that a clear model of the environment is crucial and that both the history of inputs to the system and its responses may be relevant to that model, they then say;

> "In practice, most PDP models involve a much simpler characterization of the environment. Typically, the environment is characterized by a stable probability distribution over the set of possible input patterns independent of past inputs and past responses of the system. In this case, we can imagine listing the set of possible inputs to the system and numbering them from 1 to M. The environment is then characterized by a set of probabilities, p_i for i =

1,...,M."

It is clear from this quotation that the primary focus of interest in PDP models is not the environment, but the processing characteristics of internal configurations of nodes and the learning rules which modify the connections among them. This emphasis is in one way entirely natural, since it is the presence of hidden units which enables connectionist systems to transcend the limitations on the powers of single layer perceptrons so notably identified by Minsky and Papert (1969). It is also possible that the capacity to deal with real world complexity will emerge from these initial restrictions to impoverished environments. Nevertheless, the methodology is curious given that one of the stated goals of connectionism is to overcome the domain specificity and "brittleness" of symbol processing systems.

The input and output node layers of connectionist networks show a strong and perhaps more than coincidental resemblance to Newell's "input" and "behave" operators. In fact, in some respects they are weaker. Hanson & Burr (1990) remarked that the environment for a connectionist system will typically consist of "a set of stimuli defined in an arbitrary feature space." It may be, of course, that impoverished environments are chosen for sound methodological reasons. But it is dangerous to suppose that a model which works well in a restricted environment will also work under more realistic and complex stimulus conditions.

1.8 Arguments for a new model.

The moral to be drawn from both generic and connectionist architectures is that generic theory and connectionism, while differing (sometimes fundamentally) with respect to the details of structures and processing, nevertheless share the view that the important *computational* parts of the cognitive system are located entirely inside the skin of the organism which maintains contact with the external world by processing representations of it. Such a view is encouraged by the digital computer systems on which models are usually built and tested. Connectionism focuses on developing internal representational structure via associative learning, whereas generic theory focuses on exploiting existing internal structure by developing search

and proof techniques for managing symbolic knowledge bases. Neither position represents an entirely adequate approach to the problems of embodied cognition and a new approach is needed. The generic theory has produced models which are markedly brittle, despite the maximal behavioural flexibility which they enjoy, in principle, by virtue of their computational universality. Connectionist models by contrast, while less brittle and more able to handle degraded input, are known to have difficulty with tasks such as drawing inferences which require temporally extended sequences of processing. Even their celebrated capacity for generalization from training inputs to novel stimuli has been called into question (Norris, 1991).

A particular point of focus for the thesis is that neither the generic approach nor connectionism offers a satisfactory treatment of the environment. The central point is that both approaches suppose that what the cognitive system has to do is to build or manipulate internal structure which reflects, represents, or imposes structure on the external world. The question is why this should be thought to be the best way to proceed. Given that there is elaborate and relatively permanent structure in the external world, why not use it directly? Brooks (1990, 1991) has reported work, admittedly with rather simple and strictly bounded systems, in which one of the guiding principles is that the world can, and must, serve as its own model in this way. Brooks's work is discussed in Chapter 6.

It may be objected that such a suggestion represents a regressive move towards a behaviourist analysis in which the causes of action are sought in the organism's environment. The objection loses what force it might otherwise have when it is understood that the behaviour of a Turing machine is a function of two parameters, its current internal state and its current input. The choice of which of these to focus on is very much dependent on the nature of the activity the machine is engaged in. Likewise with human activity; sometimes we need to concentrate on what is happening internally, sometimes on what is going on in the outside world. Take a simple example. Suppose someone asks for directions and is told "*Go straight along the road until you come to the first set of traffic lights and turn left.*" How should we think about their subsequent behaviour? While they are walking along

the road, it might be appropriate to think of control as being exercised by the state of mind "*looking for a set of traffic lights.*" Behaviour will continue to be responsive to aspects of the environment, such as side turnings, roundabouts, etc. but is not controlled by them, except perhaps negatively, in the sense that these other things are recognized as "*non traffic lights.*" The point is that they do not lead to changes of internal state. When the traffic lights are reached, however, it might be appropriate to think of them as exercising control, because they, or at least the perceptions they generate, do lead to a change of internal state. Thus, the question of where control is said to be exercised is a matter of emphasis. Strictly speaking it is actually a matter for internal states and external structure equally. So it is with a Turing machine. A rather natural analogue of the pedestrian looking for traffic lights and then changing direction, is the Turing machine searching for a particular symbol on its tape and changing direction when it finds it. This sort of activity is fundamental to Turing's original analysis. While the machine is moving in a particular direction, and before it has found the required symbol, it is appropriate to think of its behaviour as being controlled by its internal state, and when the symbol is found as being determined by that symbol, which triggers a change of direction, but in fact a Turing machine's action is always locally caused by the pair (current state, symbol scanned). Further analysis of the relations among internal states, inputs and outputs is given in Chapter 2.6 and Chapter 3.8.

1.9 The External Tape Hypothesis (ETH).

The central claim of the thesis is that a computational account of cognition must be based on a proper understanding of the interactions between the human organism and the external environment. It is argued that these interactions are best modelled by regarding the embodied central nervous system as the finite state control of a Turing machine and selected aspects of the external environment as its tape. This approach is called the external tape hypothesis (ETH) and is derived directly from Turing's seminal analysis of routine computation. The ETH implies that the cognitive architecture is a synergetic system of organism and environment which realizes a Turing machine and the claim is that this is much the most natural generalization of Turing's original analysis. The argument is developed in detail in

Chapter 3. The term "synergetic" is not intended to convey the idea that the whole is greater than the sum of its parts, but to denote precisely, the notion of combined or co-ordinated action. Thus, the implication of the ETH is that cognitive activity is based on the combined and co-ordinated activities of an organism and its environment, interacting as the control and tape of a Turing machine. The approach is unbiased by the engineering considerations inherent in the design of digital computers and offers a new perspective on a variety of interesting problems in psychology. Cognition, according to the ETH, involves external objects directly. A crucial point is that cognitive processing need not involve internal, symbolic representations of the kind fundamental to the generic theory. There is a need for stable internal states, but these need not be representational in the sense of "designating" something other than themselves. The issues here are related to a fundamental distinction between tape memory and control memory which is introduced in Chapter 2 and subsequently explored and argued for extensively. It is a central claim of the ETH that this type of memory has been ignored and seriously underplayed by cognitive scientists, primarily because it does not figure prominently in the operation of digital computers owing to their design aims and construction methods. The claim is that internal states need not be semantically interpretable in the way argued by Pylyshyn (1984) and Newell (1980) for example. It is an empirical rather than a conceptual matter to determine whether there are representational states of the traditional kind. The ETH suggests that the primary relation between external stimuli and internal mechanisms is one of registration rather than representation. The view is consonant with, but not committed to, a connectionist account of internal states. The ETH goes beyond the predominant connectionist concern with the way in which stable *spatial* structures (often thought of in terms of sets of micro-features) form and interact with internal states, to the ways in which stable *temporal* structures of events and objects might also form and interact with internal states. This is an area with which connectionists are beginning to become concerned as difficulties become more manifest (Elman, 1990; Chater, 1991). Recent important work in this area has been reported by Cleeremans (1993). The ETH differs from the generic theory in a variety of ways, perhaps most importantly in allowing that the nature of the computational substrate may be of

fundamental importance to the development of a satisfactory account of human cognition. Evidence from theoretical computer science and from the emerging technology of analogue VLSI, which models neural circuits in silicon, is presented to support this suggestion.

1.9.1 Linear Tapes and Human motion.

The linear nature of a Turing machine's tape imposes on the control the need for repeated traverses to retrieve information. This constraint, which has been cited as a major barrier to the practical utility of the Turing machine, becomes a source of interesting hypotheses when Turing's machine is adopted as a serious model of the human cognitive architecture considered as a system in which the organism interacts with the external environment, because it highlights human mobility as a factor in cognitive processing. It may be a simple minded observation, but it is striking that, like Turing machines, humans do not have random access to different parts of the environment. Moving from A to B involves traversing all intervening points, and moving back to A from B involves traversing them again. It may be computationally inefficient, but it has to be done. Thinking of the organism, walking around, sensing its environment, as akin to the Turing machine control shuttling up and down its tape, suggests that human mobility may be connected to our memory capabilities in a fundamental way. In Chapter 6, neurobiological evidence is cited which supports this view.

Treating the mobile organism as a component of the cognitive system also aligns cognitive psychology more closely with robotics than has previously been conventional, and treating the brain and other parts of the CNS as a finite state automaton sharpens the links with neuroscience. The latter point is particularly important. If, as the physical symbol systems hypothesis suggests, the brain is organized like a digital computer, then arguments that neural structures are not, in and of themselves, representationally significant have considerable force. If, however, the brain is a finite state automaton, then structures realizing states of the automaton will have representational significance, because the states of a finite automaton both represent and control, and understanding the implementation of

cognitive states in the brain will need to be treated as a central part of the cognitive science enterprise. Churchland (1986) and Pellionisz (1988) offer arguments to this effect.

1.9.2 Parallel and Serial Operation.

The ETH assumes that the cognitive system is both parallel and serial in operation, thus suggesting that an either/or debate about the issue is sterile. Viewed from outside, i.e. from the perspective of an observer, the system can be seen to be producing behaviour in a serial fashion. This is literally so; it is not just as Smolensky (1989) puts it "an approximate description of the global behavior of a lot of parallel computation." The operations of the control, i.e. the brain, however, are obviously parallel. Curiously enough, although the early computer designers opted for serial operation for very good reasons, from the beginning parallel control was seen to be simpler in some respects. At a conference held in 1951 to inaugurate the Manchester University computer, Maurice Wilkes, read a paper in which he introduced the important concept of microprogramming. In the course of this paper he argued that control in a parallel machine was simpler because the electronic waveforms needed were easier to produce and of less critical shape.

> "In the case of a serial synchronous machine the waveform must rise at some critical moment relative to the clock and must fall at another critical moment, and its edges must be sharp. In a parallel asynchronous machine all that is needed is a single pulse whose time of occurrence, length, and shape are all non-critical."
>
> Wilkes (1951, p.182)

There is an important general point implicit in these remarks. Computers are now so reliable that it is easy to forget the precision of the engineering required to make them work as they do, and all too easy to attribute to neural tissue, the same functional qualities (cf. Dennett 1984, p.149 note 21; 1987, pp.231-2 on *wonder tissue*.). Johnson-Laird does something of the sort at the start of "Mental Models". Having argued that mind can be studied independently of brain because programs can be studied independently of hardware, an argument which is reviewed in

Chapter 5, he goes on to suggest that

> "The neurophysiological substrate must provide a physical basis for the processes of the mind, but granted that the substrate offers the computational power of recursive functions, its physical nature places no constraint on the patterns of thought."
>
> Johnson-Laird (1983, p.9).

This argument only goes through if it has already been shown that the brain is actually organized like a Turing machine and that mental processes stand in the same relation to brain processes as the software of a computer stands to its hardware. Johnson-Laird offers no argument to show that this is so. Without such an argument the observation above is of limited value.

1.9.3 Representational Structures and Thinking.

The ETH reduces the representational burden on internal structures and paves the way for a naturalistic approach to the semantics of mental states. If the primary relation between the brain and the external world is one of registration rather than representation, then there is no obligation to postulate internal structures which are articulated in ways which reflect the full combinatorial variety of object-event relations. It is argued that this approach not only reduces the representational burden on internal states, but also provides a way of considering how thinking and perception might be linked. It is clear that the cognitive system of the neonate is far from being a tabula rasa and that the perceptual system is innately structured to respond (i.e. to make state transitions) in highly specific ways to given forms of external input. Evidence that this is so comes from sources such as Spelke (1985, 1990) It is hypothesized in Chapter 7 that thinking is enabled by the development of internal state transition trajectories, i.e. virtual machine processes, produced by a Turing computable process of self-description originally described by Lee (1963). The process redescribes internally, sequences of states and the external events and objects which cause transitions between them, in a form which is decoupled from the specific triggering objects/events, and provides free floating schemata which are proposed as the "vehicles" of thought. The innate structures with which the neonate is endowed get the process off the ground. A case may be made for thinking that

such a process may underlie the process of representational redescription which Karmiloff-Smith (1992) argues is an important part of the process of normal intellectual development.

1.9.4 Program and Observer Perspectives.

A further interesting consequence of the ETH is that it brings into play more social considerations than have customarily been favoured in cognitive theories. One way to see what is involved here is to consider two possible perspectives on a computation, that of the machine and that of an observer. Newell (1980, p.166), describing the behaviour of a universal machine program, summarized the perspectives succinctly as follows;

> "From the perspective of the program there is no choice and no decision; it simply puts one foot in front of the other so to speak. From the perspective of the outside observer a choice is being made dependent on the data."

This is a very interesting observation. It is suggested that there may be grounds for identifying our own perspective on our actions with something like the program perspective and another's perspective on our actions with the observer's point of view. Thus it would appear that the observer is better placed to understand some aspects of our actions than we are ourselves. This suggests that a complete understanding of cognitive behaviour may require a social context in which feedback is available. G.H. Mead made just such a proposal in "Mind, Self, and Society" (1934). Mead's work, although rarely quoted by cognitive scientists[5], is remarkably contemporary in its preoccupations, which is particularly striking since it pre-dated not only the digital computer, but also the computer's prefiguration in the theoretical work of Turing and Church. von Neumann also, was concerned with the distinction between an observed system and its observer as early as 1932 (Schnelle 1988, p.543). Schnelle suggests that von Neumann's concerns then were important precursors of the ideas set out in his General and Logical Theory of

---

[5]The only reference I have encountered is again in the work of Allen Newell (1982,p.109 footnote 10).

Automata. The distinction between program and observer perspectives comes into play strongly in the context of the external tape hypothesis, by virtue of the fact that more of the cognitive machine is assumed to be on public view than is the case in most cognitive theories. Observers can see the objects which provide the computational symbols driving the cognitive Turing machine, and can see the responses of the machine. The only parts of the machine which are hidden are the internal mechanisms which constitute the control.

The ETH also makes contact with the "intentional stance" of Dennett (1978,1987). Dennett sees himself as balanced "firmly on the knife-edge between the intolerable extremes of simple realism and simple relativism" (1987,p.37). He wants to argue that the observer's ascription of intentional states to a system, resulting from the perception of data dependent choices, is objective because observable patterns of action do exist, and at the same time observer dependent and potentially indeterminate.

> "it is always possible in principle for rival intentional stance interpretations of those patterns to tie for first place, so that no further fact could settle what the intentional system in question *really* believed."
>
> Dennett (1987, p.40)

Treating the environmentally situated organism as a Turing machine seems to provide some support for Dennett's position. Some aspects of cognitive computation are visible since an observer can see what a person is doing. Other aspects, the internal states and transitions among them, are not visible. Hence ascriptions of intentional states, although based on objective patterns of behaviour, are defeasible because they are based on partial information. Further, even if state transitions were available for inspection, it would still be the case that a unique ascription of intentional content might be unavailable because multiple consistent interpretations of a Turing machine computation are always possible in principle.

1.10 Objections to the External Tape Hypothesis.

Numerous objections to the ETH have been raised, both by its opponents and by those who are sympathetic towards the point of view. One important problem is precisely that which from another angle makes it so attractive, namely the immediacy of its interaction with the environment. What models based on the generic theory seem to provide is an autonomous internal computational environment in which such paradigmatically cognitive activities as planning and anticipation can be carried out (cf. Dennett 1978, ch.5; Newell 1990). A Turing machine control, by contrast, stands in a completely deterministic relation to its input, and appears to offer no scope for judgment, planning and deliberation. There seem to be two main ways of approaching this problem and it is not clear which is the better.

The first approach accepts that although the relation is deterministic it allows more flexible responding than appears possible initially, particularly when the number of states of the system in question is large. This is obviously true of the brain. Even if it is supposed that all the information processing work of the brain is done by neurons, which is a conservative supposition, and that the operation of neurons is essentially digital, then the brain which has something like 10^{10} neurons would have approximately 2^{100} or 1.27×10^{30} total states. Even allowing for the fact that many of these neural states might be computationally identical, such a huge number means that behaviour can be dependent on input in highly complex ways and over extended time spans. This approach is discussed in more detail in the context of Turing's analysis of computation in Chapter 3.

The second approach argues for a distinction between primary and secondary computational relations which is associated with the way in which thinking might be derived from experience via self-description as mentioned in Section 1.9.3. This is treated at greater length in Chapter 7 in the context of representational redescription. The essence of this approach perhaps represents a weakening of the most radical version of the ETH in that it allows a limited place for explicit, semantically transparent symbolic representations.

Another problem area concerns the representational inadequacy of the state and

symbol form in which descriptions of Turing machines are normally couched. For the ETH to have any practical value, it will be necessary to have much richer descriptive forms for internal states and symbols.

> "In order that we can apply any insights which we may have about mechanisms we want this description to reflect the actual, concrete, structure of the device in a given state. On the other hand, we want the form of description to be sufficiently abstract to apply uniformly to mechanical, electrical or merely notional devices."
>
> Gandy (1980, p.127).

In the paper from which the quotation was taken, Gandy used hereditarily finite sets to extend the scope of Turing's original analysis. One of the attractions of Gandy's approach is that it shows how to incorporate parallelism into the control of a machine whose global behaviour is serial. Gandy's work is more fully discussed in Chapter 6. Alternative forms of description which might be used are those of Hoare (1985) and Milner (1989).

Another area of problems clusters around the character of a Turing machine tape and the symbols which are printed upon it. There are four principal characteristics as follows;

1) that the number of symbols be finite,
2) that the portion of the tape which can be observed by the control at any one time is finitely bounded.
3) that the tape itself is, in principle, of unbounded length.
4) that symbols remain unchanged once written, unless or until the control encounters them again.

With respect to 4), the ETH is not in greater difficulty than the generic theory even though objects in the world do not behave like Turing machine symbols. Both accounts have difficulties. Another aspect of the same general problem focuses on the kinds of things that could count as computational symbols. The physical symbol systems hypothesis, for example, suggests that the right kinds of things are neural circuits (Newell 1990, p.132). The ETH suggests that they are primarily objects and events in the external environment. However, the kinds of results reported by Hochberg (1968) show how difficult it will be to give a satisfactory account of the

physical characteristics of objects which give rise to form perception independently of a perceiver.

With respect to 3) the ETH faces no greater difficulty than any other computational account which requires computational universality and hence unbounded memory. In fact the proposal is more principled since large chunks of the external environment are potentially available as memory and the idealization is thus less severe.

With respect to 2), it is clear that the limited acuity of our sensory systems, attention span and short term memory mean that there are indeed bounds on how much of the environment can be scanned at any one time. The ETH implies that cognitive theories of attention should constitute a central part of cognitive science. Attention is discussed in greater detail in the context of Turing's analysis of routine computation in Chapter 3.

The problem with the finite bound on the number of symbols in 1) is more serious, in that there appears to be an unbounded number of environmental contingencies which could be sensed by an organism and which might therefore count as computational symbols for that organism. At least two separate sorts of problem can be discerned.

Kirk (1986, p.443), discussing what he takes to be a fundamental disanalogy between a machine tape and the world, poses one sort of problem. He says "there is no definite limit to the number of kinds of human performance that could produce a token of, say '3', nor any definite limit to the number of things that would be acceptable as a token of '3'." The point is that something can count as a token of '3' if it is intended by its producer to be such a token, and, if it can be perceived as such. Thus, as Kirk points out, we can perceive a pattern of stones in the desert or a vapour trail in the sky as a token of '3'. I think a possible line of defence to this objection is to distinguish the primitive symbols forming the "alphabet" of a system from the interpretation placed on them. A universal Turing machine may

have a much smaller repertoire of symbols than a machine whose operations it simulates and yet be capable of simulation by encoding the target machine's symbols as sequences of the smaller set which it can handle. Thus the vapour trail and the pattern of stones need not, perhaps, be considered as separate symbols but may be thought of as composites.

A second sort of problem with respect to environmental symbols is that many environmental quantities are continuous whereas Turing machine symbols are discrete. Headway may be made with this problem by considering the phenomena of categorical perception (cf. Harnad 1987), in which physically continuous stimuli are perceived to belong to discrete categories. Three more specific problems with the ETH are as follows[6].

5) The world, unlike a tape, cannot be run backwards.
6) The world, unlike a tape, cannot be blank.
7) Since there are identical symbols on the tape (world) and TM control (brain) the ETH trivializes the whole problem of perception (and perhaps learning).

Problem 7 is the most important and is addressed in Chapter 2.6. In outline, the argument presented is that the control and tape need have identical symbols on them only if the control is implemented literally as a machine table. However, since the machine table is a form of description of an abstract machine rather than a blueprint for the implementation of a real machine, there is no need for explicit symbols as part of an implemented control although they are so convenient as to be almost inescapable as a means of describing an abstract control.

Problem 6, that the world, unlike a tape, cannot be blank, is perhaps less difficult to resolve than it might initially seem. First, although a Turing machine's tape can be blank, it is not part of its definition that it ever must be and there are Turing machines, among them all universal machines, which must have non-blank tapes in order to function correctly. What is mandatory for a Turing machine is that no more than a finite number of squares can be non-blank at the start of a computation.

---

[6]I am grateful to Professor Yorick Wilks for raising these problems.

This appears to amount to the stipulation that the world can be only finitely complex. Whether this is a serious problem is hard to determine. Second, and perhaps more convincingly, if the objection is well founded, it ought also to apply to the generic theory. The generic theory proposes that human memory is modelled as the tape of a Turing machine. But memory, unlike a tape, also cannot be blank. The various forms of amnesia which might seem to refute this suggestion are best thought of not as blanking the memory but as affecting retrieval, or, in extreme cases, as destroying the tape. Thus problem 6 appears to apply to many computational theories of cognitive architecture if it applies at all and is not therefore, specifically a problem for the ETH.

Problem 5 can be resolved by considering the differences between the tape of a Turing machine and a reel of cinema film. These may be thought to be similar media in the sense that the one is divided into squares and the other into frames. If a cinema film is run backwards, effects appear to precede their causes which is, of course, physically impossible in the real world. This idea, applied to a Turing machine tape, appears to capture the intended force of the objection. The tape of a Turing machine, it is claimed, has the capacity of running backwards in time which is not characteristic of the real world in which time is uni-directional. However, for this objection to hold, it would have to be the case that the time of inscription of a symbol is uniquely associated with its position in the sequence of squares on the tape. It is an association of this kind which confers on a reel of film its own internal "film time" and hence the possibility of running it backwards. Consider a sequence of frames, $f_1 \ldots f_m$, of a cinema film. Each of the f_i has a unique time at which it was shot, $t(f_i)$, such that $t(f_i) < t(f_j) \Leftrightarrow i < j$. It is this biconditional association of position in the sequence of frames with time of shooting which gives sense to the notions of running a film "forwards" and "backwards". These notions, of course, apply only to "film time" and not to real time, because running a film backwards in "film time" requires forward movement in real time. There is no analogous internal notion of "tape time" in the case of a Turing machine, because there is no relationship of entailment between position on a Turing machine tape and the time of inscription of a symbol. In consequence, there is no

sense in which the tape can be said to run backwards. The proper terms for describing movement relative to a Turing machine tape are "left" and "right" not "forwards" and "backwards". It is, of course, quite possible for a sequence of state transitions and tape movements to occur which exactly reverses a prior sequence of actions, leaving the tape as it was before the pattern of symbols resulting from the original sequence was inscribed. However, suppose the original sequence to have been started at some arbitrary time t_1 and to have been completed k-1 time steps later at time t_k. To reverse the sequence requires a further k time steps t_{k+1} to t_{2k}. Thus time continues to move forward as a sequence is reversed. The sense in which reversing a sequence would have undesirable theoretical consequences would be that in which reversing the sequence also reversed the flow of time such that the time at the end of the reversal of the sequence was the original t_1 rather than t_{2k}. But this is not the case with a Turing machine. It appears therefore, that the only satisfactory formulation of the notion of running a tape backwards, i.e. reversal of a sequence of transitions, is one which does not damage the ETH.

1.11. Outline of the thesis.

In Chapter 2 formal definitions of finite automata and Turing machines are given and the properties of such machines are discussed. The application of the Turing machine concept to the study of cognitive architecture is expanded beyond the introduction given in chapter 1. In chapter 3, Turing's analysis of computation is discussed in detail and its divergence from the assumptions of the generic theory is documented. In Chapter 4, a short account of some of the principal features of digital computers and the history of their development is given with a view to reinforcing the notion that the digital computer is unsuitable as a literal model of human cognitive architecture. In Chapter 5. some prominent examples of the generic computer theory of mind are discussed in the light of Turing's analysis. In Chapter 6 the External Tape Hypothesis is developed in detail. Chapter 7 gives an account of the development of thought in terms of the ETH and self-reproducing machines. Chapter 8 summarizes the claims made, outlines areas of importance for further research and relates the ETH to other theoretical enterprises in cognitive science.

Chapter 2. Turing Machines and Finite Automata

In this chapter the formal foundations on which the ETH is based are described. A Turing machine is essentially a mathematical abstraction which describes a potentially infinite output such as the decimal expansion of π, in terms of a finite set of rules for its production, but Turing developed his ideas in the context of machines which might realize such sets of rules. Thus, in addition to the formal definition of a Turing machine, it is important to understand what kind of machine Turing had in mind and what one might be like if it were actually constructed. This is particularly important for the study of cognition where the concern is with the embodiment of intelligence rather than with the purely formal or functional means of its specification. Turing's genius lay in showing how the formal and the physical might be combined. What Turing aimed to produce was a mechanical model of the system consisting of a human "computer" working out a routine calculation with paper and pencil. The details of Turing's machine model and his analysis of computation are the basis for the ETH and are discussed in detail in Chapter 3. Turing's biographer suggests that he may have used the typewriter as his basic model for a machine, Hodges (1983, pp.96-98), whereas Hendriks-Jansen (1994) has claimed, much less plausibly, that Turing had in mind a manufacturing assembly-line as his basic model. Whatever the case, Turing was definitely a man with a practical bent, although apparently somewhat maladroit in his dealings with machinery, and it is clear that he had in mind the idea of a real machine, albeit one that was as simple as possible. In order for such a machine to do useful work it would need to be equipped with the means for reading symbols and for writing and erasing them. It would also need an indefinitely large supply of paper on which to record the potentially infinite results of its computations.

2.1. The description and definition of Turing machines.

Informally, a Turing machine consists of two parts. The first is a control machine which embodies or contains the rules defining the procedure it is to execute. The second part, which replaces the paper on which a human computer writes is a one-dimensional tape, divided into squares, each of which can contain one of the finite

number of symbols which the machine can recognize. Its format is like a single line taken from a page of squared arithmetic paper. The tape is assumed to be indefinitely extendable so as to be able to contain an indefinitely long symbolic expression. The control machine is equipped with a scanner which enables it to read the symbol on a square, and with a printer which enables it to write a symbol on a square or erase the symbol currently there. The control machine is also able to move to the left or the right relative to the tape, so as to scan a different square. The way in which the control machine embodies or contains the rules defining the procedure which it executes is a matter which requires substantial discussion. Turing's analysis, arising as it did from the consideration of a human computer, treated the rules as a set of "states of mind" which were defined functionally in terms of their relations to each other and to the inputs and outputs of the machine.

The Turing machine concept can be defined formally in a number of ways. These ways differ in detail, but turn out ultimately to be equivalent in the sense of encompassing the same set of functions. The following, slightly adapted, definition is taken from Lewis & Papadimitriou (1981,p.170);

Definition 2.1

A **Turing machine** is a quadruple (K,Σ,δ,s), where
K is a finite set of **states**, *not* containing the halt state h;
Σ is a finite alphabet of symbols, containing the blank symbol #, but not containing the symbols L and R;
$s \in K$ is the **initial state**;
δ is a function from $K \times \Sigma$ to $(K \cup \{h\}) \times (\Sigma \cup \{L,R\})$.

If $q \in K$, $a \in \Sigma$, and $\delta(q,a) = (p,b)$, then the machine M, when in state q and scanning symbol a, will enter state p, and (1) if b is a symbol in Σ, rewrite the a in the currently scanned tape square as b, or (2) if b is L or R, move its head in direction b relative to the currently scanned square. Since δ is a function, the operation of the machine M is deterministic and it will stop only when $\delta(q,a) = (h,b)$, or because the machine reaches a configuration for which the transition function δ is undefined. In the latter case the machine is said to "hang".

Definition 2.1 differs from Turing's original specification in a number of details. The most noticeable of these is that at each invocation of the transition function δ the machines of Def. 2.1 either move or print a symbol but do not do both, whereas Turing's original machine design included both printing and moving as part of each instruction. However, definition 2.1 coincides with Turing's definition in the fundamental matter which is the nature of the transition function. This is a function of two arguments, q and a, the first of which is the current internal state of the machine, and the second of which is the currently scanned symbol. The internal state and scanned symbol together determine the behaviour of the machine. Turing called the pair (q,a) a "configuration", and distinguished this from the internal state alone, which he called an "m-configuration", and from the complete description of the current state of a machine, including the sequence of symbols on its tape which he called a "complete configuration". These are important distinctions and the terms are used throughout the thesis in Turing's sense. Because both the set K of internal states and the set Σ of symbols are finite by definition, the number of possible configurations is also finite. In principle, the domain of the transition function δ can be the whole of K x Σ but need not be. It is not the case, for example, in the example machine M described below.

As an example of a rather simple Turing machine, consider the following machine $M = (K,\Sigma,\delta,s)$ where $K = \{q_0,q_1,q_2\}$, $\Sigma = \{\#,(,),X,Y,N\}$ and $s = q_0$. This machine is a slightly modified version of a machine described by Minsky (1967). The transition function δ can be described in terms of a machine table as follows;

q	σ	$\delta(q,\sigma)$	q	σ	$\delta(q,\sigma)$	q	σ	$\delta(q,\sigma)$
q_0	#	(q_2,L)	q_1	#	(h,N)	q_2	#	(h,Y)
q_0	X	(q_0,R)	q_1	X	(q_1,L)	q_2	X	(q_2,L)
q_0	(	(q_0,R)	q_1	(	(q_0,X)	q_2	(	(h,N)
q_0	)	(q_1,X)						

M is a machine which takes strings of left and right parentheses as input, and, when started in state q_0 on the leftmost element of such a string, eventually halts and prints 'Y' if the string is a member of a set P, 'N' if it is not. The tape is assumed to be blank throughout except for the expression which M is to evaluate. P is

defined recursively as follows as the set of strings of parentheses which are 'grammatical';

1. () is in P.
2. If E is in P, (E) is also in P.
3. If E and F are in P, EF (the concatenation of E and F) is also in P.
4. Nothing else is in P.

The machine table is a common way of describing the logical structure of a Turing machine control which dates back to Turing's original analysis. An alternative means of description which has been developed since his pioneering work is the state transition diagram, usually abbreviated to "state diagram". This is a very useful notation, which is often both more compact and easier to follow than a machine table. A state diagram for the machine M is shown in Figure 2.1.

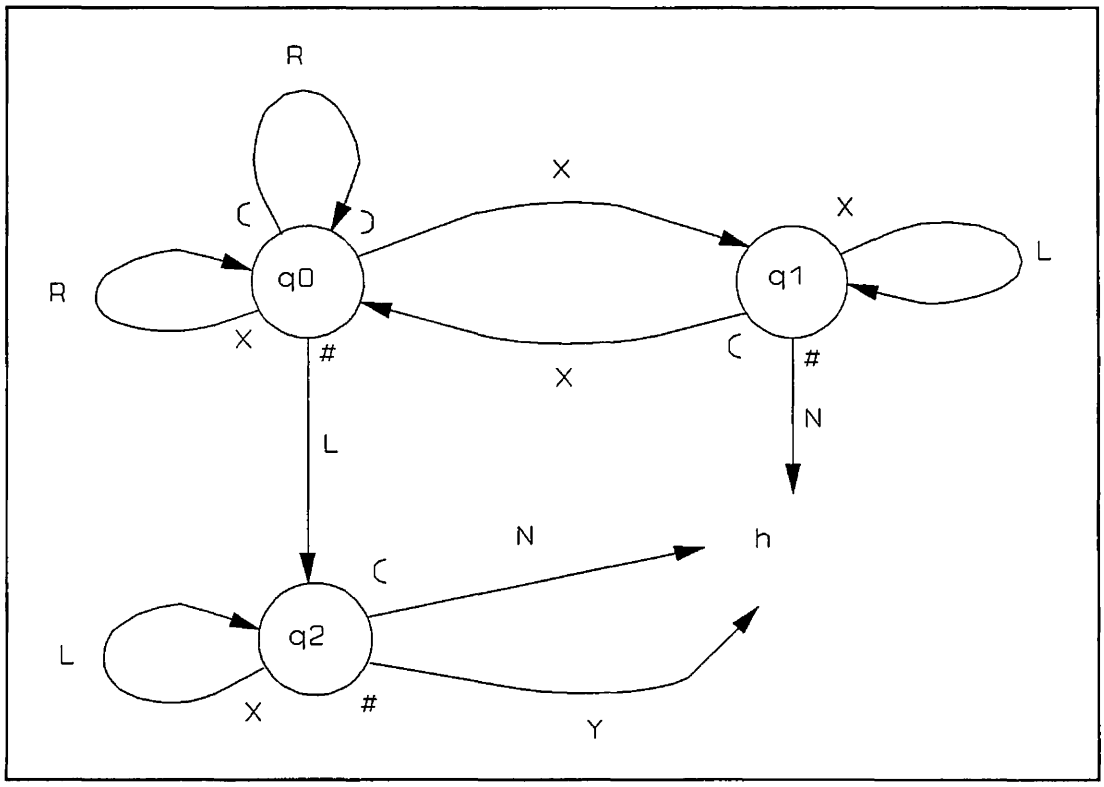

Figure 2.1 The State Diagram for machine M.

States are shown as circles labelled with their identifiers. State transitions are shown as labelled arrows. It is clear that a transition can be made from a state back to itself. The symbol near the tail of an arrow is the symbol on the currently scanned square of the tape, and the symbol near the middle of an arrow indicates

the action taken which is either a movement or a printing action. Since the sets of print symbols and movement symbols are disjoint there is no confusion. Comparison of the machine table and state diagram for machine M shows that they contain exactly the same information.

The operational characteristics of machine M and the strings it can evaluate are worth describing in greater detail, because they serve to introduce a number of important general features of the Turing machine. One distinction which is worth making at the start is between configurations with which the machine can deal and those with which it cannot. This is the distinction between recognizable and unrecognizable configurations. It is not the same as the distinction between well-formed and ill-formed strings as defined by membership of the set P. The set of recognizable configurations contains all those for which M can reach the correct decision, i.e. it contains all those configurations which consist of an unbroken sequence of parentheses contained between blanks, such that M is started in state q0 scanning the leftmost parenthesis of the expression. The point of making this distinction is to emphasize that a Turing machine computation requires not just that the string of input symbols meets some pre-defined criteria, but also that the relationship between the control and the input string is similarly constrained. This observation shows how closely intermeshed external structures on the tape and the internal organization of states of the control need to be in order for a computation to be executed correctly. The ETH hypothesizes that this intermeshing can be used to model the interlocking of organisms and environment which constitutes part of the ecology of a species.

An example of a recognizable, well-formed string which the machine M might be required to evaluate is '#()(())#'. Given this string M works as follows; from the starting configuration in which the machine is in state q0 scanning the leftmost parenthesis of the expression, it moves right still in state q0, until it encounters a right parenthesis, which it erases, writing an 'X' in its place. If the string is well formed, there will be a matching left parenthesis somewhere to the left of the current position. To determine whether or not this is so, the machine goes into state

q1 and searches to the left. If it encounters a '#' while searching, there is no matching left parenthesis, the string is not well formed and the machine halts and reports the result. If a left partner for the 'X-ed' right parenthesis is found, this is also erased and replaced with an 'X'. The machine then re-starts its rightward search in q0 for another right parenthesis. When a '#' is encountered in this state, the right end of the string has been reached which means that all right parentheses have been accounted for. The machine then goes into state q2 and moves left. In this state if it encounters a left parenthesis the string is ill formed but if it encounters the '#' at the left end of the string, the string is well formed. In either case the machine halts and reports accordingly. For the string #()(()# above, successive operations will produce the sequence of expressions #(X(()#, #XX(()#, #XX((X)#, #XX(XX)#, #XX(XXX#, #XXXXXXX#, and the machine will report the string to be well formed. What happens, in general, is that a target string is treated as a concatenation of substrings, which are dealt with one at a time moving from left to right. Within each substring pairs are matched working outwards from the deepest to the shallowest nestings. The technique can be applied recursively to strings of arbitrary length and embeddedness. Broadly speaking, the individual states of the machine can be described as follows; state q0 is a state characterized by left to right movement with respect to the expression on the tape. Its sole purpose is to find an occurrence of a right parenthesis. State q1 is a state characterized by right to left movement. Its basic purpose is to find a matching left parenthesis for the right parenthesis located by state q0. States q0 and q1 act as a pair, locating and 'X-ing' pairs of matching left and right parentheses. State q1 has an additional function, which is to report the target expression ill-formed if a left parenthesis is not found. State q2 is a right to left moving state. Its task is to determine whether there are any unmatched left parentheses in the expression. If there are it reports the expression to be ill-formed, if not, it reports the expression to be well-formed.

2.2. The memories of Turing machines.

A point of fundamental importance for the thesis is the fact that machine M can be said to have two sorts of memory. The first is the obvious symbolic memory

provided by the tape. The tape is the repository of the target symbolic expression and of all changes made to it while the machine is in operation. This memory is essentially static but fundamental. Without a tape, the class of computations which can be carried out is reduced. The second kind of memory is a dynamic memory, which might be thought of as a kind of working memory.

Before describing this memory, it is important to distinguish the claim being made from another which, emphatically, is not being made. It is not being claimed that the machine has any memory, either implicit or explicit, of the sequence of states it has passed through or of the number of moments of time during which it has occupied the current state. The machine M does not know that it is "in" a particular state at a particular time. Thus when it is in q0, for example, the machine does not know whether it has just been started, whether it has always been in q0 or whether it was in q1 at the previous time step and has just arrived back in state q0 as the result of a state transition caused by a '(' when the machine was in q1. Similarly, when it is in q1, even though at some prior moment of time the machine must have been in state q0, it has no knowledge or memory of this or of any other fact. The machine's horizons at a given moment, so to speak, are entirely local. Perhaps, more picturesquely, it might be described as living solely in the present or as being profoundly amnesic. Although, to an outside observer, a Turing machine like M has a history, the machine itself is entirely unconscious and knows nothing of its own history. To say this is not to claim that no Turing machine could be conscious. It might be that consciousness is a property, just like computational universality, of certain Turing machines which are organized in a particular way. Like universality, however, consciousness is not a defining feature of a Turing machine and machine M is certainly not conscious. Further, the control has no explicit knowledge of what is on the tape, even at the moment at which it scans a symbol. It does not know which tape square it is scanning because its horizons are limited to just the currently scanned square which is indistinguishable from any other square on the tape. Indeed, a simple Turing machine could not even be said to know that it had a tape.

The positive sense in which the machine can be said to have a memory other than the static memory of the tape, is that it can have an implicit memory for a symbol scanned at some indeterminate time in the past. Note that it is an implicit memory for a symbol not for a state. In the case of machine M, state q1 constitutes an implicit memory for the occurrence of a right parenthesis. Despite the very limited capacities which a machine such as M enjoys, this way of speaking is justified by the following considerations. First, the machine never scans a right parenthesis when in state q1, i.e. in a manner of speaking it never has any direct experience of right parentheses in q1. Thus any effect of a right parenthesis on the machine in q1 must be indirect or implicit. Second, however, a transition to q1 from q0 is made if and only if a right parenthesis is encountered when the machine is in q0. In the case of a null expression or an expression consisting entirely of left parentheses the machine never enters state q1. Thus it is apparent that state q1 is fundamentally connected to the occurrence of right parentheses in some way. The nature of this connection becomes apparent from considering the behaviour of the machine if it scans a blank i.e. '#' in q1. In such a circumstance the machine halts and reports that the target expression was ill-formed. What this means, looking at the behaviour of M from the outside, is that M has failed to find a left parenthesis to match a *previously scanned* right parenthesis. It is for this reason that it is appropriate to talk of the machine in q1 as having an implicit memory for the previous occurrence of a right parenthesis even though it has no awareness, of any kind, of this fact. Finally, since the implicit memory lasts precisely for as long as the machine remains in state q1 it seems appropriate to think of it as a kind of "short term" or "dynamic" or "working" memory. This terminology is perhaps not entirely felicitous because it is not a short term or working memory in the sense in which these terms are understood in cognitive psychology, cf. Baddeley (1986) for example, if only because there is no set limit on how long the implicit memory can last. Nevertheless some such usage captures the required distinction between the static memory of the tape and the more active memory of the control. The state diagram for M shows that the machine remains in q1 for as long as it takes to traverse the current sequence of 'X'es on the tape. Since expressions in parentheses may be arbitrarily long, there may be also be arbitrarily many 'X's on the tape and

hence there can be no upper bound on the time that the machine spends in state q1 implicitly remembering that a right parenthesis had been encountered. To avoid confusion with the normal psychological terminology, the implicit memory which machines may have for symbols as a result of the organization of their internal states is referred to from here onwards as "control" memory to distinguish it from tape memory. An additional point which may be worth emphasizing is that although the memory for a previously scanned symbol is implicit, the behavioural consequences of that implicit memory are, of course, quite explicit.

The fundamental place of control memory in the approach to cognitive architecture which follows from the ETH cannot be overstressed and the issues are discussed further in Chapters 3 & 6. The basic point is that if the brain is not a complete Turing machine but is a finite automaton like the control of a Turing machine as the ETH suggests, then control memory must form an important part of human memory resources. Another point which emerges from the above discussion of states is that they may have multiple, overlapping functions. State q1 implements the search for a left parenthesis at the same time as serving as an implicit memory for a right parenthesis. In more complex machines, which may have multiply embedded sequences of states, the complete functional description of a state may need to advert to descriptions at numerous levels of functioning. More is said about this in Chapter 3.

Another point which should also be mentioned here briefly is the caveat that internal states or *m*-configurations as they appear in machine tables and state diagrams are logical not physical entities. Part of Turing's achievement was to show clearly that the logical specification of a mental activity such as calculation could be specified independently of questions of its physical implementation. This does not mean, as the discussion in Chapter 1 shows, that there are no constraints on media for implementation. In practice, as the history of the development of digital computers shows, satisfactory implementations of logical states are hard to come by and the implementation of the set of instructions constituting a single logical state will typically be highly complex.

Furthermore, the internal states of a Turing machine are logically global, i.e. they are states of the whole machine. Thus the temptation to see a state diagram as a diagram of the parts of a physical machine must be treated carefully. There will always be numerous ways in which the logic of a system of states may be realized in practice. The interesting and psychologically important question of the relationship between logical and physical descriptions is discussed further below and in Chapter 3 in the context of Turing's universal machine description.

2.3. The description and definition of finite automata.

Two other classes of machine need to be discussed in this chapter. These are finite automata and universal Turing machines. The essential difference between finite automata and Turing machines is that the former cannot use a tape as memory in the way that Turing machines can. A deterministic finite automaton (DFA) consists of a finite state control machine and a tape. In these respects such a machine resembles a Turing machine. However, the control of a DFA can only read symbols from its tape, it cannot write on the tape, and it can move only to the right, assuming it to be started scanning the leftmost element of an input string. The tape is, therefore, nothing more than an input device which presents successive elements of a string of symbols to the machine. A subset of the finite set of internal states of the automaton is designated as the set of "final" or "accepting" states, and if the automaton ends up in one of these states at the end of a particular input string, that string is said to be "accepted" by the automaton, otherwise it is "rejected". The *language* accepted by an automaton is the set of strings it accepts. Formally, again borrowing from Lewis & Papadimitriou (1981, p.51), a deterministic finite automaton can be defined as follows;

Definition 2.2.

A **deterministic finite automaton** is a quintuple $D = (K,\Sigma,\delta,s,F)$ where
K is a finite set of **states**,
Σ is a finite alphabet of **symbols**,
$s \in K$ is the **initial state**,
$F \subseteq K$ is the set of **final states**,
and δ, the **transition function**, is a function from $K \times \Sigma$ to K.

The rules by which the next state is selected are built into the transition function, as is the case with Turing machines. Thus if D is in state $q \in K$ and the symbol read from the input tape is $\sigma \in \Sigma$, then $\delta(q,\sigma) \in K$ is the uniquely determined state to which D passes.

There are two principal differences between definition 2.2 and definition 2.1. First, definition 2.2 specifies a set of "final" states rather than the halt state h, and second the transition function of definition 2.2 does not specify an output symbol or a movement relative to the tape. Both of these points have been raised informally in the introduction to finite automata above. Apart from these differences it is readily apparent that the definition of a deterministic finite automaton is very much like that of a Turing machine. Hence it is appropriate to think of the control of a Turing machine as a deterministic finite automaton with added output and movement capacities. Some authors, e.g. Lipschutz (1976) use the term "finite state machine" to describe a finite automaton which is capable of output.

There is one further class of finite machines, the non-deterministic finite automata, which need to be mentioned here. Such machines have a transition relation rather than a transition function describing their actions. A non-deterministic finite automaton is one in which a number of alternative actions may be available for a given configuration. Although non-determinism might seem to be a powerful property of a machine, the classes of deterministic and non-deterministic finite automata are provably equivalent because a non-deterministic machine can always be converted into an equivalent deterministic one. It has been argued, for example by Nelson (1989), that the class of non-deterministic finite automata is the appropriate class of machine models for modelling cognitive behaviour. For the purposes of the thesis however, the distinction between deterministic and non-deterministic automata is not of primary importance and all further references to finite automata are to deterministic machines unless otherwise specified. The primary distinction which is explored in the thesis is that between Turing machines and finite automata.

2.4. The computational capacities of finite automata.

Because they cannot write on their tapes and because they cannot change direction and revisit previously scanned squares, finite automata are more restricted in the computations they can perform than are Turing machines. The limits of their computations are well understood. They are restricted to recognizing members of the class of regular languages which are those languages which can be defined by regular expressions. Informally, the regular languages are those in which the amount of memory needed to determine whether or not a string is a member of the language can be fixed in advance and is dependent solely on the structure of the language and not on the length of the input string. This relates very naturally to the idea of a finite automaton as a machine which cannot use a tape as an auxiliary memory. The memories of finite automata are limited to what can be built into their internal state structures. One of the interesting features of the machine M, is that the language which it accepts, i.e. the set of strings which are elements of the set P defined above, is not regular. Thus, although the machine has a simple structure and the function it computes is also simple, it is a function which cannot be computed in its full generality by a finite automaton. The reason is easily understood. Consider expressions of the form $(^k)^k$ for some natural number k, which consist of k instances of '(' followed by k instances of ')'. Thus $(^3)^3 = ((()))$. All such expressions are elements of P. Machine M evaluates such an expression by traversing the k left parentheses and then ticking off successive matching pairs from the inside out, i.e. ((XX)), (XXXX), XXXXXX. Suppose, however, that the expression was to be evaluated by a finite automaton. Because elements of the string cannot be revisited the machine would have to be structured so as to increment a count of the number of left parentheses as each was presented and to decrement the count as matching right parentheses were presented. A machine such as F in Figure 2.2 would suffice for the task with $k = 3$.

State q0 is the starting state for the machine and also its only final or accepting state. It accepts all and only those strings for which it is in q0 when the end of the string is reached. Apart from accepting strings such as ((())), it also accepts (()()) and ((())(())) and any other of the countably infinite set of strings of parentheses

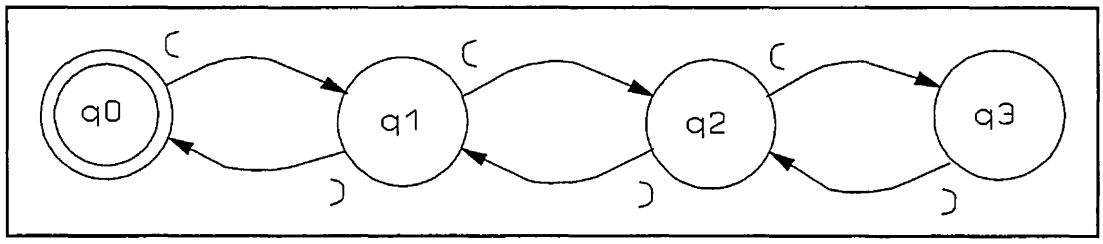

Figure 2.2. F: A finite automaton for parenthesis string recognition.

which do not contain any nesting deeper than 3. However given any expression containing a substring of $(^k)^k$, for $k > 3$, the machine will hang in q3, because its input will be a left parenthesis and the configuration "(",q3 is one for which no transition is defined. Machine F is, therefore, limited to accepting only a subset of the elements of P, and it is clear that for any fixed machine with n states $q_0,\ldots q_{n-1}$, which has to count its inputs in this way, it will always be possible to specify an element of P which contains a substring of the form $(^n)^n$ which the machine cannot accept. Thus no finite automaton can be structured which can accept all and only the strings in P. This is, of course, a limitation but it is worth flagging some considerations which are mathematically perhaps of no interest but which might be of considerable interest in the context of real creatures solving real problems in real time. Suppose a limit on the depth of nesting had been established such that it was known that the machine would not need to evaluate strings with nesting deeper than some arbitrary value n. In such a case an adequate automaton could be constructed for the task. Might there be any reason to prefer a machine of type F rather than type M? At first sight it seems improbable. Suppose, for example, the depth of nesting to be limited to 50. An automaton following the design of machine F would have to have 51 states in order to deal with inputs of maximum depth of nesting, whereas machine M with its three states would be adequate. However, if efficiency, defined as the number of state transitions needed to evaluate a string, is taken as the criterion of assessment the picture looks rather different. Considering strings of the form $(^k)^k$, a finite automaton designed like F has to make just 2k state transitions in order to complete such tasks, whereas machine M has to make $2(k+1)^2+2k$ transitions. This means that machine M becomes relatively less and less efficient as the size of the input increases. For a depth of 50, $2(k+1)^2+2k = 2701$. Clearly the example is artificial and its connection with psychological subject matter is

remote, but it may be that the obvious limitations of finite automata have shifted attention unduly away from their possible advantages. By way of conclusion it is worth noting that automaton F provides another example of implicit memory. State q1 implicitly remembers that one more left than right parentheses have thus far appeared in the input, q2 that two more left than right have appeared and so on.

2.5. Universal Turing machines.

Whereas a finite automaton is essentially a Turing machine without a tape, a universal machine, architecturally, is just like any other Turing machine, i.e. it consists of a finite state control and an indefinitely extendable tape. The differences come in the organization of its internal states and the nature of its data. The first machine table for a universal machine was constructed by Turing in his seminal paper Turing (1936-7). A universal machine is a *single* Turing machine, which can carry out *any* of the countably infinite number of different tasks or computable functions for which a special purpose Turing machine can be specified. At first sight the task of constructing a universal machine seems impossible. The difficulty lies in the finitude of K and Σ, the sets of states and symbols respectively, which define a Turing machine. Since the universal machine, *U*, is a Turing machine it has, by definition, a fixed, finite set of states K and a fixed, finite alphabet of symbols Σ. It would seem, therefore, that for any given specification for *U*, we could always think of a task which required a machine with a set K' of states where $|K'| > |K|$ and/or an alphabet Σ' where $|\Sigma'| > |\Sigma|$. Such a task would be beyond the capacity of *U* which would therefore not be universal. The problem is similar to the difficulties faced in trying to design a finite automaton to tackle arbitrary depths of nesting in the parenthesis checking task.

Turing's solution, which has not been improved upon in the essentials, required two techniques, the symbolic encoding of machine tables and the interpretive execution of symbolic encodings. First he showed that a code could be defined using a fixed, finite alphabet Σ, to represent any Turing machine. Turing called the encoding of a machine a "standard description", and it is analogous to a computer program. Second he showed that a machine table could be specified which would take the

"standard description" of a Turing machine as input, and produce the same output, for given input data, as the encoded machine would have produced. So, the universal machine consisted of a finite state control which could interpret and execute the standard description of any other Turing machine, plus a tape on which a standard description could be written as well as any other input. It is important to be quite clear that a universal Turing machine requires *both* a "program" or "standard description" *and* data on its tape, whereas a special purpose Turing machine requires only data. Turing's machine table for *U* was of the same kind as that for M above, in that it specified exactly how *U* was to behave while leaving open the question of its possible physical realizations.

In one sense the universal machine was special purpose because it was "hard-wired" to carry out a single function, the execution of a standard description. This can be called the interpretive function. In another sense, however, it was multi-functional because its behaviour depended on the encoding of the particular Turing machine written on its tape. The machine's behavior was changed by substituting one encoding on its tape for another, which changed the function computed while leaving the "hardware" interpreter unchanged. Thus the essential difference between a special purpose and a universal Turing machine is that the latter is "programmed" to compute a function, whereas the former is "hard-wired". Minsky (1967) gives an interesting and straightforward construction for a universal machine which has just over twenty internal states and uses a binary encoding for its target machines.

Two points about universal machines which are relevant to their use as models of cognitive architecture are the following. First, because universal machines are Turing machines they conform to definition 2.1 and have a fixed alphabet of symbols and a fixed set of internal states. Their characteristic flexibility is derived entirely from their being structured so as to execute encodings of the machine tables of other Turing machines. This means that any cognitive theory which hypothesizes that the mind is, or contains, a universal Turing machine has to be committed to the notion that the mind deals with inputs which are rendered into a standard internal code. If this is so, it ought to be possible to obtain some evidence for the nature

of the internal code, for the basic symbolic resources from which it is structured and for the mechanisms which implement such encodings. If such evidence is not forthcoming after extensive empirical enquiry, the probability of the hypothesis being correct must be reduced. Fodor (1975) was one of the first theorists to take this challenge seriously. The second point is a point about efficiency. Universal machines are necessarily less efficient than the machines they simulate because for each configuration of the target machine, the universal machine must execute a set of transitions of its own to determine what the appropriate instruction for the target configuration is and to carry it out. Using a programmed version of Minsky's (1967) universal machine and an appropriate encoding of machine M, the universal machine required more than 80000 transitions to evaluate the binary equivalent of the input string '#()#' for which M requires 10 transitions. Part of this prodigality derives from the linearity of the tape, but a modified version of the machine using two tapes still required more than 5000 transitions to carry out the same task. In modern digital computers which are essentially practical versions of universal machines, these costs are minimized or masked in two ways. First computers use addressing schemes which provide much more efficient memory access than the linear tapes of universal Turing machines and second, computers achieve state transitions at speeds of the order of micro-seconds or better, whereas neurons operate in the milli-second speed range. Despite these savings, however, the inefficiency of the universal machine method of function execution must lead to questions about its suitability as the architectural basis for the human cognitive system in which temporal efficiency is a matter of obvious importance.

2.6. Logical and physical machine descriptions.

The issue of the relationship between logical and physical machine descriptions which was raised briefly earlier in the chapter needs some further discussion, particularly in the context of one of the objections to the ETH raised in Chapter 1 which can now be considered in the light of the formal definitions of Turing machines and finite automata discussed above. Problem 7 objects that since there are identical symbols on the tape and control of a Turing machine, the ETH trivializes the whole problem of perception and perhaps also the problem of

learning. It will be useful to have a name for the problem and a suitable one appears to be "the identical symbol problem".

The image of an implemented Turing machine which supports the thinking leading to the identical symbol problem must suppose the control to be an explicit implementation of the machine table plus a reader. The reader observes the tape to pick up the input symbol, checks the internal state record to see what the current state is and uses this pair, i.e. the configuration, to determine which entry in the machine table is to executed. Having found the appropriate entry, the output symbol is inscribed on the tape, the state record is updated, and the next instruction can be executed. A Turing machine implementation of this kind is described by Wilks (1975). The fundamental question is whether a Turing machine control implementation must always have an explicit machine table and a reader. If the identical symbol problem is to be a problem for all implementations of Turing machines the answer must be yes, but it is far from clear that this is so, even though our experience with computers pushes us in this direction because computers are explicitly syntactic machines.

Consider the parenthesis checking machine M and how it might be implemented. First M needs perceptual capacities to read symbols from its tape. These can be minimal. What is required is not recognition but discrimination. M does not need anything like the capacity to say "Now I am perceiving a '#'". Because M's symbol alphabet has just four symbols, its perceptual system needs only to be able to pass one of four distinct messages to the control. These messages might be four different voltage levels, four different positions of a lever, four different neural firing rates over a given interval or whatever else might reliably signal four different inputs. Similarly at the output side, we might imagine M to be equipped with something like a daisywheel, or more flexibly, the print head of an ink-jet printer, which responds differentially to appropriate output signal differences. As a slight complication, output capacity must include movement left or right but it is not hard to see how this might be done.

In addition to input and output capacities, an implementation of the transition function is also required. The way in which this is to be done lies at the heart of the identical symbol problem. Multiple internal states provide the capacity for a machine be able to make more than one response to a given input. If a given input is always paired with the same output, then no more than one internal state need be postulated. Indeed, in such a case the notion of internal state is rendered superfluous because the state notion explains nothing which is not captured by the simple and more parsimonious specification of the associative pair. It was just this superfluity which early behaviourists originally hoped to demonstrate with respect to the idea of mental states. The demonstration that a one state universal Turing machine is not possible, ought then, in theory, to be taken to constitute a refutation of radical behaviourism, because it shows that there are mechanical processes which cannot be reduced to a set of I-O pairs. If a behavioural capacity is to be given a deterministic form, i.e. specified by an input output function, and yet to encompass the many cases in which a single input can be paired with more than one output, a second argument is needed to cope with the additional variability because a function can be a one-to-one, or a many-to-one but not a one-to-many relation. The role of the "state" parameter in the specification of a deterministic machine is to provide that second argument. As is apparent from the machine table method of specifying a transition function, the state need be nothing more than a column index. However, there is nothing in the functional specification of the state notion which precludes its being realized in other, less syntactic forms.

Consider machine M in receipt of input 'X'. One of three different responses is required, (q0,R), (q1,L) or (q2,L), with the choice depending on the current state of the machine. On the assumption that the input can be organized in terms of, say, a set of distinct voltage levels, it is perfectly plausible to think of the internal states of a real M being implemented as, say, a system of servomechanisms and a switching network. Such a system would be an embodiment of the transition function of M, but need not contain anywhere, an explicit rendering of the machine table. It might assist clarity to call such a system a non-tabular implementation of a Turing machine control. Given the notion of a non-tabular implementation the

identical symbol problem can be restated as the claim that all Turing machine control implementations must be tabular, and the ETH can be stated as including the claim that non-tabular control implementations are perfectly plausible.

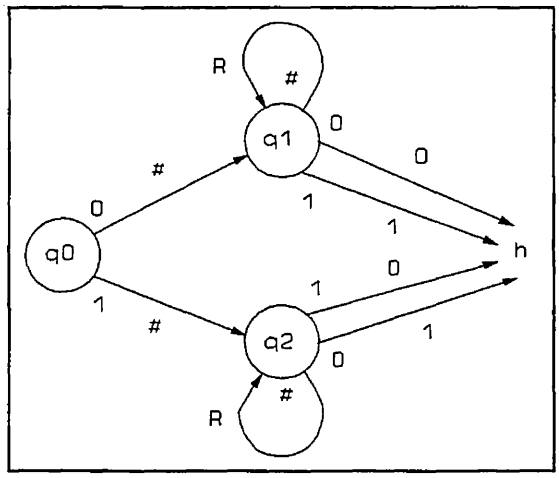

Figure 2.3. The state diagram for XOR1

For a familiar example of a function which can be given both tabular and non-tabular implementations consider the logical function XOR of two binary inputs. This is commonly specified as a truth table. A Turing machine to implement this function can easily be specified. XOR1 with K = {q0,q1,q2}, Σ = {#,0,1} and s = q0 is such a machine. The state diagram for XOR1 is shown in Figure 2.3. XOR1 reads its first input in q0, replaces it with a blank and goes to state q1 if the input was a "0" and to q2 if it was a "1". In either of these states the machine then moves right to scan the second input, which is replaced by the value of the function, following which the machine halts. XOR1 is a simple machine, but if the pairs 00,01,10,11 are defined as single symbols an even simpler machine XOR2 can be specified with K = {q0}, Σ = {00,01,10,11,0,1,#} and s = q0. The state diagram for XOR2 is shown in Figure 2.4. XOR2 is particularly simple because it embodies the function, as it were, directly. It is of interest in the current context because it has a non-tabular implementation which is shown in Figure 2.5.

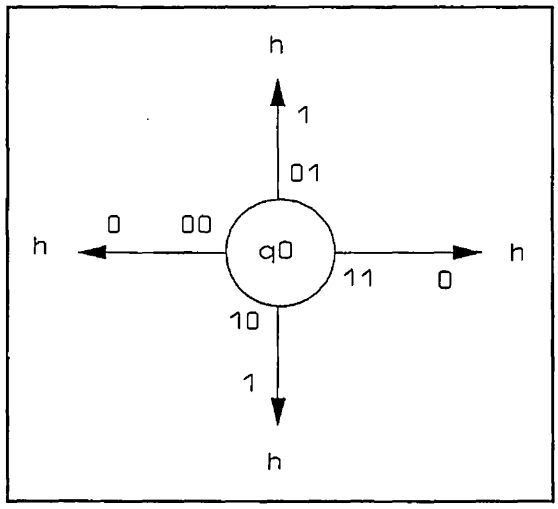

Figure 2.4. The state diagram for XOR2.

Figure 2.5 shows a network of threshold elements. Two of these take input, one produces output and three, using connectionist terminology, are hidden units. If the

two inputs together are specified to constitute a complex symbol of the kind defined in the alphabet of XOR2, and if the output is printed on a square of the tape, then it is apparent that the network constitutes a realization of the control automaton of a Turing machine. One might imagine the network to be contained within a black box which is schematized by the dotted line in Figure 2.5. Clearly the network is functionally equivalent to XOR2, it does not contain any explicit symbol tokens, it is non-tabular and is therefore not a system to which the identical symbol problem applies. These points alone are insufficient to establish that the network is uniquely an implementation of XOR2 because it is also functionally equivalent, in the input-output sense, to XOR1. However, the point at issue is not whether a given implementation can be identified uniquely as the implementation of a particular Turing machine but whether a non-tabular implementation of a Turing machine control can be specified. The fixed XOR network of Figure 2.5 shows that such an implementation can be specified. This is of course to be expected from the work of McCulloch & Pitts (1943) who demonstrated the formal equivalence of finite automata and neural networks composed of fixed threshold elements.

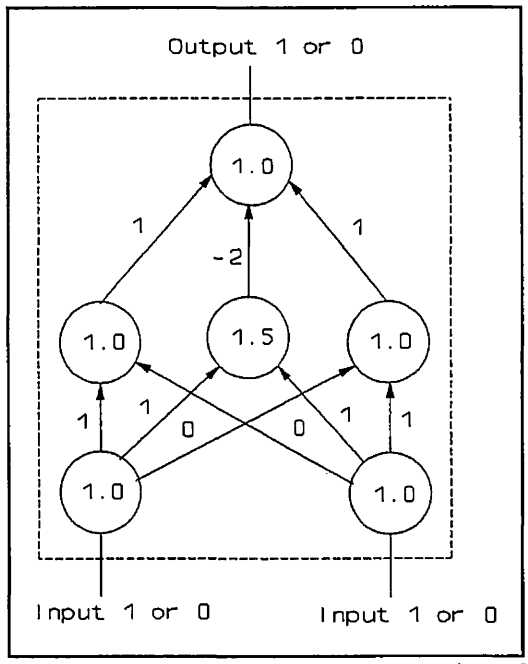

Figure 2.5. A non-tabular implementation of XOR2.

It is sometimes suggested, as discussed in Chapter 1, that McCulloch & Pitts demonstrated the logical equivalence between neural networks and Turing machines rather than finite automata. The basis for the suggestion may be a remark made by McCulloch in the discussion following von Neumann's talk at the Hixon symposium in 1948. McCulloch is reported as saying of his work with Pitts, "What we thought we were doing (and I think we succeeded fairly well) was treating the brain as a Turing machine". Aspray & Burks (1987,p.422)

Examination of the 1943 paper shows that McCulloch's later remark was perhaps not recorded entirely precisely. What the 1943 paper proves is that a McCulloch-Pitts net can be used to model the finite automaton which constitutes the control of a Turing machine. In order for a net to compute exactly what is computed by a Turing machine it has to be supplemented by perceptual organs and a tape. "It is easily shown: first, that every net, if furnished with a tape, scanners connected to afferents, and suitable efferents to perform the necessary motor-operations, can compute only such numbers as can a Turing machine; second, that each of the latter numbers can be computed by such a net; and that nets with circles can be computed by such a net; and that nets with circles can compute, without scanners and a tape, some of the numbers the machine can, but no others, and not all of them." McCulloch & Pitts (1943, p.37).

Another important point is that although the network connection strengths and thresholds are fixed in the network of Figure 2.5 it is clear that a network of this type could be trained to achieve the functionality displayed using error backpropagation. Notice, in this instance, that the tape plays no essential role as an external memory, and that the system can therefore, properly be thought of as a finite state machine with output. It is this sort of example which, in part, justifies the claim made in Chapter 1 that a connectionist network, whose connection strengths have been fixed after training, constitutes a finite automaton.

To summarize, two main classes of computing machines have been defined and discussed. These are the classes of Turing machines and finite automata. The class of Turing machines includes both mono-functional and universal machines. The latter are those which are organized so as to treat part of their input as the specification of another Turing machine whose coded machine table they can decode and simulate. The class of finite automata can be divided into deterministic and non-deterministic automata, but the two classes are equivalent in computational power.

In addition to these basic definitions, a number of points which are of importance

to the main argument of the thesis have been introduced. The first of these is that Turing machines have two sorts of memory, tape memory and control memory. Control memory is an implicit memory for one or more specific symbols which have appeared on a Turing machine's tape at some point in the past. The second point which relates to the issue of implementation is the demonstration that the "identical symbol problem" need not affect all Turing machine implementations. Non-tabular machine implementations are not open to this objection.

The third point is the claim that a fixed neural network has the power of a finite automaton. This is a point which was first proved by McCulloch & Pitts (1943) with respect to networks constructed from fixed threshold elements. The novelty, if any, of the present approach is to claim that this result extends to networks whose architecture is such as to make them amenable to training of the kind studied by contemporary connectionism. It has been suggested that the capacity to learn falsifies the claim that a connectionist network has the power of a finite automaton because a network with hidden layers and the capacity to adjust its connection strengths can compute any arbitrary function and is thus a Turing machine[7].

It is hard to see how this can be the case for connectionist networks as they are currently studied. The critical point is that with regard to its input, output and potential for using auxiliary memory, a connectionist network is like a finite automaton. The sequence of inputs is not subsequently available to the network on a tape and the outputs of the network do not modify its inputs. Thus, like a finite automaton, a connectionist network has no auxiliary memory available to it and is therefore restricted, in its memory capacity, to whatever internal capacity it has in the fixed system of nodes and connections among them which constitutes its architecture. The implication of this fact is that unless such a network has infinite internal capacity, there will be functions, of which the parenthesis checking function of machine M is one, which can be computed by a Turing machine but not by the network, which cannot therefore be a Turing machine. The question then is whether

---

[7]I am grateful to Professors Wilks and Marslen-Wilson for raising this objection.

a network can have infinite internal capacity, and the answer seems to be no. Infinite storage capacity would have to be bought by making connection strengths infinitely variable or by making thresholds infinitely sensitive, and neither of these options is plausible for realistic modelling and certainly not for implementation.

The claim that connectionist networks are Turing machines by virtue of their capacity to be trained to implement arbitrary functions appears to take the flexibility of universal Turing machines to be a defining characteristic of Turing machines generally. That is not the case. Machine M is an example. It is a Turing machine, but it computes only a single function. If it computed a different function it would be a different Turing machine. What distinguishes a Turing machine from a finite automaton and also from a connectionist network is the capacity to use an indefinitely expandable auxiliary memory. A connectionist network which was capable of multiple internal states and which was able to take input from a tape and to write output on that tape is a different matter. A system of this kind would have the requisite capacity. But such a system would be more than just a network, and connectionists, with some exceptions, typically do not think of their nets as embedded in a larger system in this way. They are, of course, right to concentrate on the formidable learning capacities of networks with hidden unit layers, but they need also to think about how these nets should be brought into co-ooperative interaction with structured environments.

In summary, Chapter 2 has prepared the ground for the detailed presentation of Turing's analysis of computation in Chapter 3 and for the subsequent elaboration of the External Tape Hypothesis. This chapter has defined and discussed Turing machines and finite automata. The major issue to which the thesis is devoted is which of these classes of machine is appropriate for modelling the human mind. The generic theory, introduced in Chapter 1 argues that the mind supervenes on the brain and that the brain should be considered as implementing a Turing machine. The ETH argues, on the basis of Turing's analysis of computation, that it is incorrect to think of the brain as a Turing machine. The brain should be construed as a finite automaton. This suggests, since minds obviously have at least the power

of Turing machines, that mind does not supervene on brain, but extends to those portions of the external environment with which the finite control system of the brain interacts. Thus mind is a property of the situated organism and the environment serves as an auxiliary memory for the human just as the tape serves as an auxiliary memory for a Turing machine.

Chapter 3. Turing's Analysis of Computation.

The analysis of computation in terms of abstract machines which established Alan Turing as one of the most important mathematical logicians of the twentieth century is contained in his paper Turing (1936-7). The significance of this paper for the External Tape Hypothesis is twofold. First, it provides a machine model, of greater complexity than the Turing machines commonly discussed in the literature, which makes an excellent vehicle for a discussion of the nature of control memory. The concept of control memory was introduced in Chapter 2. It is the memory which a Turing machine's states provide and is distinct from its tape memory. Normally such memory is not thought of as particularly important because it is fixed and is, as it were, swamped by the unbounded memory provided by the tape. However, if the hypothesis that the brain realizes the finite state control automaton of a Turing machine rather than a whole Turing machine including a tape is correct, then control memory must be of primary importance for modelling human cognition. This represents a major departure from the tradition of modelling in cognitive science which has been referred to as the generic theory in Chapter 1.

The second, more fundamental, contribution of Turing's analysis to the ETH is that it provides evidence for the hypothesis that the brain should be modelled as the finite state control of a Turing machine, rather than as a whole Turing machine including a tape. The evidence consists of Turing's own arguments to the effect that the brain is a finite automaton and that the tape is an external, auxiliary memory. The idea that the brain might be modelled as a whole Turing machine, and, in particular, the idea that the tape of a Turing machine might provide a model for human memory are not ideas which are found in Turing's paper. They are the ideas of a later generation of theorists whose primary inspiration was the digital computer rather than the abstract model on which the computer was based. The idea that human memory can be modelled as the tape of an internal Turing machine in fact contradicts Turing's arguments for the separation of the tape which is an external, unbounded memory resource from the mind of the human computer which is a finite, bounded resource.

3.1. Turing's introduction to computing machines.

In his famous paper on computable numbers, Turing (1936-7) introduced the notion of a computing machine by considering what a person did in the process of computing a number. In 1936 when the work was written, a "computer" was a person who calculated rather than a machine.

> 'We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions $q_1, q_2, \ldots, q_R$ which will be called "*m*-configurations". The machine is supplied with a "tape" (the analogue of paper) running through it, and divided into sections (called "squares") each capable of bearing a "symbol". At any moment there is just one square, say the *r*-th, bearing the symbol S(r) which is "in the machine". We may call this square the "scanned square". The symbol on the scanned square may be called the "scanned symbol". The "scanned symbol" is the only one of which the machine is, so to speak, "directly aware". However, by altering its *m*-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously. The possible behaviour of the machine at any moment is determined by the *m*-configuration q_n and the scanned symbol S(r). This pair q_n, S(r) will be called the "configuration": thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (*i.e.* bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or left. In addition to any of these operations the *m*-configuration may be changed. Some of the symbols written down will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure. It is my contention that these operations include all those which are used in the computation of a number.'
>
> Turing (1936-7, pp.117-118)

The first point to notice about this definition is that Turing derived the idea for a computing machine from considering a human "computer" in the process of calculating a real number using paper and pencil. Thus, although the Turing machine is abstract, it is an abstraction from a very familiar situation and the distinction between computing paper and human memory is neither arcane nor abstruse.

A second important point is the distinction Turing makes between an "*m*-

configuration" which is an internal state of the finite control part of the system, and a "configuration" which is a pair consisting of an *m*-configuration and an external symbol. Configurations rather than either *m*-configurations or symbols alone control the behaviour of the machine from moment to moment. The implication is that although behaviour is directly responsive to external input it is not uniquely controlled by that input; behaviour is a function of two parameters, the input and the current internal state. This point was discussed in Chapter 2.

The final, and most important, point for present concerns is that the person doing the computing is compared with "a machine which is only capable of a finite number of conditions." Turing's claim is that the appropriate model for the mind of the person is a finite automaton. The automaton is supplied with a tape which is the analogue of the paper on which a human computer calculates. Thus, the original Turing machine was a model of a system consisting of a person working with pencil and paper, and at the outset Turing established a clear distinction between the automaton with a finite number of states which models the human mind and the tape which models the paper on which the human computes.

This is important because there is already a prima facie incompatibility between Turing's picture and those painted by various proponents of the generic theory. Pylyshyn (1984, p.69), for example, maintains that "a computer (or a brain, for that matter) is more appropriately described as a Turing machine than as a finite-state automaton, though clearly it is finite in its resources". Pylyshyn's argument for this claim is slightly different from the language of thought and symbol systems arguments described briefly in Chapter 1. Pylyshyn's argument is based on the notion that a sequence of state transitions can count as a computation only if the constituent states are capable of semantic interpretation, and his claim is that this is true of the states of Turing machines but not of finite automata. Hence if the mind is to meet the dual requirements that its states be computable and semantically interpretable, it must be thought of as a Turing machine. Pylyshyn's approach has been criticized by Nelson (1987). If the approach described in this thesis is correct, Pylyshyn is right about computers but wrong about brains.

Briefly, the issues can be resolved by considering different ways of thinking about what constitutes a computational state. Turing distinguished three possible construals of the notion of "state", to each of which he gave a different name. The first is the "state of mind"; this is what Turing refers to as an *m*-configuration. It is an element of the set $\{q_1,...q_n\}$. The second is an ordered pair (q_i,S_j), where S_j is an element of the alphabet of the machine. Turing called such a pair a "configuration" and it is configurations which determine the moment by moment behaviour of a machine. In terms of the discussion of Chapter 2, the elements of configurations constitute the arguments to the transition function $\delta(q,a)$. The third construal of the "state" notion is the global picture of the machine at a given time. It includes the current configuration plus the entire sequence of symbols on the tape of the machine and the position of the reading head with respect to that sequence. Turing called this the "complete configuration". It seems reasonable to require that successive complete configurations must be semantically interpretable for a computation to be of any value, but there can be no requirement that either configurations or *m*-configurations must be semantically interpretable independently of the context in which they occur. By insisting that states of mind must be semantically interpretable Pylyshyn is, in effect, insisting that they must be construed as complete configurations, and various passages in Pylyshyn (1984) support this reading. However, it is clear that Turing did not intend the tape of the machine to be construed as part of the "state of mind", and hence that Pylyshyn's proposal is in conflict with Turing's analysis.

Another example of the conflict between Turing and the generic theory is to be found in the influential work of Newell and Simon (1976) who describe the mind as a physical symbol system, which is a universal machine by definition, and thus clearly not a finite automaton. Both Pylyshyn and Newell & Simon, identify the mind with a whole Turing machine, whereas Turing clearly identifies the mind with just the finite state control of a Turing machine, and not with the tape as well.

Newell & Simon make a point about the study of Turing machines which bears on the discussion of control memory. "The finite state control system was always

viewed as a small controller, and logical games were played to see how small a state system could be used without destroying the universality of the machine. No games, as far as we can tell, were ever played to add new states dynamically to the finite control -- to think of the control memory as holding the bulk of the system's knowledge." Newell & Simon (1976, p.44) From a technical point of view there would, of course, be interest only in the attempt to reduce the number of states needed for a universal controller, cf. Minsky (1967, Ch.14) but it by no means follows that this is the right tactic for psychological research. Indeed, given that Turing had good reasons for thinking of the human mind as a finite control system, it seems clear that the predominant focus of interest on symbolic structures on the tape is simply misleading from the point of view of psychological research into memory. If Turing's model is taken seriously as a model of human cognitive architecture, then knowledge is indeed encoded in the states of a finite automaton and so is memory. Turing is explicit about this point; '...by altering its *m*-configuration the machine can effectively remember some of the symbols which it has "seen" (scanned) previously.' Turing (1936-7, p.117). This suggests the implicit storage of symbols in states of the control and opens up the whole question of how knowledge is represented. The possibility of implicit representation makes plausible a very different picture of cognitive architecture from the traditional computer based model.

3.2. Turing's definition of a computable number and his characterization of mind.

Further evidence for the identification of the human mind with a finite automaton comes from Turing's definition of a "computable number". Computable numbers are the raison d'être of his paper and Turing defined them in the opening sentence as "the real numbers whose expressions as a decimal are calculable by finite means." The point of particular interest in the present context is the reason Turing gave for the restriction to finite means, namely "the fact that the human memory is necessarily limited." Turing (1936-7, p.117). This observation lies at the heart of Turing's analysis.

The fact that a number is computable by finite means does not imply that the number must have a finite decimal expression. π, for example, is a computable number. Similarly, the fact that a computable number may have a non-terminating decimal expansion is not a reason for claiming that it cannot, in principle, be computed by a finite agent. What is required is that the finite agent be capable of cyclic processing and have access to unbounded resources of time and space. The essence of the Turing machine is thus the specification of a set of interactions between a finite agent and a potentially unbounded reservoir of external symbolic resources. Such a specification constitutes the production process for a particular computable number. Seen in this light, the identification of the (bounded) human memory with the (unbounded) tape of a Turing machine which is characteristic of the generic theory quite clearly runs directly counter to Turing's original analysis.

Turing's argument for the assertion that the mind is "necessarily limited" is brief but clear. It is similar to the argument he advanced to show that the number of symbols allowed for computation must also be finite. Both arguments are presented in §9 of his paper. The argument for a finite number of symbols rests on the claim that 'If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent.' Turing (1936-7, p.135). This argument has a physical basis which is characteristic of Turing's style and which distinguishes his approach from purely abstract characterizations of computability. The emphasis on physical constraints on computability is, of course, entirely appropriate when considering questions about the nature of human cognitive architecture. Turing is making two particular points with his argument. The first is that for mechanical computation we must be able to regard symbols as "literally printed" Turing (1936-7, p.135, footnote †) on a unit square and unless we are prepared to think of the square as divisible into an indefinitely large number of pixel like regions, we have to acknowledge a limit on the number of distinct points onto which symbols can be mapped. The second point arises from the fact that symbols must also be recognized. Members of an infinite set of symbols constrained to be printed on a square of a certain fixed size would not always be pairwise discriminable owing to

limitations on the resolving power of a finite recognition system. Thus the argument for a finite alphabet of symbols is an argument about perceivability. If the difference between a pair of symbols were made arbitrarily small they would become indistinguishable and deterministic computation would be impossible. But, as Turing pointed out, the effect of accepting such physical limitations on symbols is not serious, because compound symbols, consisting of sequences of tokens from the fixed alphabet, can always be used instead.

The related argument for a finite number of states of mind is equally briefly stated but also perfectly clear. 'We will also suppose that the number of states of mind which need be taken into account is finite. The reasons for this are of the same character as those which restrict the number of symbols. If we admitted an infinity of states of mind, some of them will be "arbitrarily close" and will be confused.' (Turing, op.cit. p.136). This is an assertion of the supervenience of states of mind on the brain. In his biography of Turing, Hodges (1983, p.108) suggests that by the time the work on computable numbers was written, Turing was becoming "a forceful exponent of the materialist view". A finite bound on the number of states of mind follows from the finitude of the brain and the supervenience of states of mind on the brain. A significant point about this argument is Turing's further observation that "the restriction is not one which seriously affects computation, since the use of more complicated states of mind can be avoided by writing more symbols on the tape." Turing (1936-7, p.136). Clearly this observation has weight only if "states of mind" and "symbols on the tape" are distinct parts of the system.

In summary, Turing's paper explores the notion of a computable number, which is a number whose expression as a decimal is calculable by finite means, i.e. by an algorithm. The algorithm will, in many cases, require unbounded resources of time and space. There is, therefore, a fundamental distinction to be made between the fixed structure realizing the algorithm and the resources needed for temporary storage and for recording the output of the computation. From consideration of the example of a human engaged in a routine calculation, Turing developed a machine model to capture the essential processes involved; the model consisted of a finite

control automaton which modelled the algorithm embodied in the mind of the human, and an indefinitely expandable tape which modelled the paper on which the calculation was worked. The identification of the human mind with the finite control system was supported by an argument for the claim that the human brain was capable of being in only finitely many different states.

3.3. Turing's later work on computers.

Turing's later work on computers is consistent with the ideas of his 1936 paper. In 1946 he produced a report for the National Physical Laboratory, in which he provided a detailed design for a stored program computer to be called the ACE. The report had elements in common with von Neumann's influential draft design document for the EDVAC computer which Turing recommended should be read in conjunction with his report. This is discussed in more detail in Chapter 4. Although von Neumann suggested in passing that there was a correspondence between computer memory and the nervous system, there is no suggestion of such a parallel in Turing's ACE document. With respect to the storage requirements, Turing's outline is very much of a piece with the thinking of his 1936 paper. "It is evident that if the machine is to do all that is done by the normal human operator it must be provided with the analogues of three things, viz. firstly, the computing paper on which the computer writes down his results and his rough workings; secondly the instructions as to what processes are to be applied; these the computer will normally carry in his head; thirdly, the function tables used by the computer must be available in appropriate form to the machine. These requirements all involve *storage of information* or *mechanical memory*." Turing (1946, pp.20-21). Notice the clear distinction between the computing paper on which results are written and the instructions which are in the head of the computer.

In 1947 Turing gave a lecture about the ACE project to the London Mathematical Society, in which he was quite explicit about the relationship between universal Turing machines and digital computers and also about the nature of the memory of digital machines. Digital computers were, he said, "practical versions of the universal machine." Turing (1947, p.112), and their memories were analogous to

the infinite tape of the universal machine. The major problem was that access to a one dimensional tape memory was unsatisfactory for a practical machine because of the time spent shuttling up and down looking for a particular piece of information. "This difficulty", he remarked, "presumably used to worry the Egyptians when their books were written on papyrus scrolls. It must have been slow work looking up references in them, and the present arrangement of written matter in books which can be opened at any point is greatly to be preferred." Turing (1947, p.107) The organization and size of the memory of the proposed ACE computer was clearly of paramount importance and Turing used a good deal of the lecture to talk about it, "because I believe that the provision of proper storage is the key to the problem of the digital computer, and certainly if they are to be persuaded to show any sort of genuine intelligence much larger capacities than are yet available must be provided." Turing (1947, p.112). This is an interesting observation because it shows that Turing was thinking even at that early stage about the possibility of digital computers displaying intelligence and had identified a large memory as a crucial requirement. However, the fact that a computer would require a large symbolic memory with tape like characteristics in order to display intelligence is not an argument for supposing that human memory is of the same form, and Turing did not advance such an argument despite his interest in machine intelligence and his likely sensitivity to useful parallels between minds and computers. The reason why a computer requires a large internal symbolic memory is precisely because it lacks the complex perceptual apparatus which enables humans to engage in direct transactions with the external environment as required. By contrast, a computer which lacks such communicative ability must be provided in advance with all the information which will be relevant to the performance of whatever task it is engaged on plus storage for intermediate results and so forth. Hence the requirement for a large symbolic memory. It is clear that pre-storage of all the relevant information needed by a mobile machine interacting with a real environment poses formidable problems of both technical and conceptual origin. Technical problems include the development of representational and search methods which can cope with the ever present threat of combinatorial explosion, and conceptual problems include the difficulty of determining which aspects of the environment are relevant to a

machine.

Finally, there are references in the very well known paper Turing (1950) which point to the identification of symbolic computer memory with the paper on which a human computer calculates and not with human memory. In the 1950 work Turing described the digital computer as a machine intended to carry out any operations which could be done by a human computer. The human computer "has an unlimited supply of paper on which he does his calculations" Turing (1950, p.43) and the digital computer has a store which "is a source of information, and corresponds to the human computer's paper" Turing (1950, p.44). Slightly further on, this correspondence is re-iterated. "The computer includes a store corresponding to the paper used by a human computer. It must be possible to write into the store any one of the combinations of symbols which might have been written on the paper." Turing (1950, p.47). These remarks are firm indicators of a continuing link in Turing's thinking to the idea of the universal machine and to the distinction made there between the finite state control which modelled human memory and the tape which modelled the paper on which the human computer wrote.

Turing also explicitly recognized a distinction, at least at one level, between the nervous system and computers. "The nervous system is certainly not a discrete-state machine. A small error in the information about the size of a nervous impulse impinging on a neurone, may make a large difference to the size of the outgoing impulse." Turing (1950, p.57). This is a somewhat puzzling observation because it appears to be in conflict with the notion of a "state of mind" which is a fundamental component of the Turing machine concept and of Turing's analysis of computation. The "state of mind" in the 1936 paper was certainly intended to be a discrete state in the sense in which Turing used the term in the 1950 paper. In an article written to mark the fiftieth anniversary of the Turing machine concept, Turing's biographer, Andrew Hodges, suggested that Turing's thinking about this issue was incomplete. "...we cannot feel that Turing had arrived at a complete theory of what he meant by modeling the mental functions of the brain by a logical machine structure." Hodges (1988, p.11). One possible answer is that Turing was

feeling his way towards an analysis of the issues in terms of a hierarchy of levels of explanation such as that proposed by Marr (1982). In Marr's terms the "state of mind" as Turing understood it is a construct at the algorithmic level whereas Turing's discussion of nervous activity is pitched at the implementation level. This, of course, is not an explanation of how discrete algorithmic states can be obtained from continuous nervous activity, but it replaces the apparent conflict with a question about how it might be possible to construct reliable discrete states from assemblies of real neural elements. von Neumann also gave some consideration to these issues at about the same time and wrote a notable early paper on the topic, which is reprinted in Aspray & Burks (1987, ch.13). von Neumann proposed a multiplexing scheme to control the global behaviour of circuits made from unreliable basic components and showed how the probability of a signalling error could be made arbitrarily small. From a different perspective, the phenomena of categorical perception, cf. Harnad (1987), whereby continuously varying physical dimensions are perceived as having discrete characteristics, suggest that Turing's analysis of mental functioning in terms of discrete states of mind enjoys considerable empirical support at an appropriate level of analysis. Notwithstanding the problems in understanding how to reconcile Turing's apparently divergent remarks, they do, in any case, provide further evidence for the suggestion that he was not trying to explain the workings of the nervous system in terms of digital computation.

To summarize, the evidence of Turing's published work over more than a decade clearly supports the claim that to treat the tape of a Turing machine or the symbolic memory of a computer as a model of human memory is to make a major departure from the Turing machine concept as its originator understood it. The generic computer theory of mind makes just such a departure and cannot therefore be considered to enjoy the support of Turing's theoretical analysis. The ETH, by contrast, by treating the brain as a finite automaton and modelling memory as control memory, argues for precisely the distinction that Turing made.

3.4. Turing's universal machine design.

Part of the difficulty with the concept of control memory is that for the sorts of

reasons given by Newell & Simon (1976) which were briefly discussed earlier it has been a neglected topic. In a modern computer, control memory is nothing more than the contents of the registers of the CPU plus the set of basic operations which constitute the behavioural repertoire of the processor. The control of Turing's universal machine is a large, complex automaton which provides numerous interesting examples of the nature and uses of the control memory concept. For this reason it is worth examining in greater detail. From the point of view of the ETH it is the size and complexity of the automaton rather than the fact that it functions as a universal interpreter which is of primary interest. To understand the structures of the control it is necessary to introduce Turing's encoding scheme for the target machines which the universal interpreter was to simulate and to say something about Turing's notation and construction methodology. These are particularly interesting in the context of implementation issues.

The universal machine operates by interpreting and acting upon the description of another machine which is known as the "target machine". This description, which Turing called the "standard description", has to conform to certain syntactic conventions. Each instruction has five components; if one thinks of the instruction as a whole as a condition-action rule the first two components constitute the left hand side or condition, the remaining three the right hand side or action. The five components, in the order in which they appear in an instruction, are the current state (q_i, $1 \leq i \leq m$), the scanned symbol (S_j, $0 \leq j \leq n$), the output symbol (S_{ij}), the tape movement indicator (M_{ij}) and the next state (q_{ij}). This formulation differs slightly from the definition of a Turing machine given in Chapter 2. In that definition an instruction had four terms only, because a machine could either print a symbol or move, but not do both as part of the same instruction. The point is a purely technical detail. Turing's syntax specified that states were to be encoded by "D" followed by i occurrences of "A", symbols by "D" followed by j occurrences of "C", and tape movements by "L", "R" or "N" for left, right or no movement respectively. Thus, for example, the encoding of the instruction q_1,S_0,S_3,R,q_4, was the string "DADDCCCRDAAAA". Instructions were separated from each other by semi-colons. Turing also specified that target machines had to be started in state q_1

= "DA" scanning a blank tape, and that the encodings for "blank", "0" and "1" had to be S_0 = "D", S_1 = "DC" and S_2 = "DCC" respectively. These specifications can be understood in the light of Turing's primary concern with computable numbers. His formalism specified machines which printed out the binary encoding of a real number. Turing called this encoding the "sequence" computed by the machine. When he came to consider the universal machine, one of the requirements was that it too should print the sequence computed by the target machine. This would only be possible if the encoding used for the digits "0" and "1" was known in advance and was the same for all targets. The encodings for "0" and "1" are the only two which the universal machine "interprets" in the sense of substituting what they stand for. The specific encoding for "blank" was needed in order to make the first configuration of every target machine the same and hence to provide a starting point for the universal machine's interpretive operations.

As a familiar example, consider machine M of Chapter 2. The definition of M, and its machine table are reproduced below. M = (K,Σ,δ,s) where K = {q0,q1,q2}, Σ ={#,(,),X,Y,N} and s = q0. The machine table is;

q	σ	δ(q,σ)	q	σ	δ(q,σ)	q	σ	δ(q,σ)
q0	#	(q2,L)	q1	#	(h,N)	q2	#	(h,Y)
q0	X	(q0,R)	q1	X	(q1,L)	q2	X	(q2,L)
q0	(	(q0,R)	q1	(	(q0,X)	q2	(	(h,N)
q0	)	(q1,X)						

As it stands, M is not suited to simulation by Turing's machine, partly because its starting state is not q1, partly because its starting symbol is not a blank, and partly because its instructions are specified as quadruples rather than as quintuples. Further, Turing did not specify a distinguished halt state. His machines would halt if they reached a configuration for which no action was defined. However, with minor modifications, a standard description for M can be produced. The first requirement is to change the state identifiers so that the starting state is q1. The halt state will be q4. Since no instructions are specified for q4, a transition which specifies q4 as the next state will cause the machine to hang which is the desired

result. The second requirement is to change the quadruples to quintuples. This is easily done as follows; first, whenever M prints an output symbol it does not move. So the movement symbol 'N' can be used in these cases. Second, whenever M does move, it does not print a symbol. The same effect can be had by specifying that the output symbol be the same as the input symbol. The use of 'N' as a movement symbol introduces a conflict with its use in the alphabet of M as an indicator of the status of a string at the end of a computation. This can be got round by replacing 'Y' and 'N' for 'Yes' and 'No' with '1' and '0'. It is also evident that the condition that the machine be started in state q1 scanning a blank could easily be implemented at the cost of an additional state. However, this is omitted here as the exercise is intended simply as an example. Putting the changes in place gives a revised machine table for a machine M' = (K,Σ,δ,s) where K = {q1,q2,q3}, Σ = {#,(,),X,Y,0,1} and s = q1. M' is functionally equivalent to M. The machine table for M' is;

q	σ	δ(q,σ)	**q**	σ	δ(q,σ)	**q**	σ	δ(q,σ)
q1	#	(#,L,q3)	q2	#	(0,N,q4)	q3	#	(1,N,q4)
q1	X	(X,R,q1)	q2	X	(X,L,q2)	q3	X	(X,L,q3)
q1	(	((,R,q1)	q2	(	(X,N,q1)	q3	(	(0,N,q4)
q1	)	(X,N,q2)						

All that is then required is to specify standard encodings for the symbols '(',')', and 'X' and to make the appropriate substitutions. Setting '(' = DCCC, ')' = DCCCC and 'X' = DCCCCC, the machine table for M' in standard description format appears as follows;

DADDLDAAA;
DADCCCCCDCCCCCRDA;
DADCCCDCCCRDA;
DADCCCCDCCCCCNDAA;

DAADDCNDAAAA;
DAADCCCCCDCCCCCLDAA;
DAADCCCDCCCCCNDA;

DAAADDCCNDAAAA;
DAAADCCCCCDCCCCCLDAAA;
DAAADCCCDCNDAAAA

Two points are worth making. First it is clear that the encoding scheme could, in principle, be applied to transform any machine description into a standard description. Perhaps more important for present concerns is the obvious price paid for the behavioural flexibility of the universal machine. That price is complete inflexibility with respect to the encoding conventions. What this implies is that if, as the generic theory suggests, thinking is essentially the activity of an internal Turing machine, a process of encoding external stimulus energies into expressions in an internal alphabet which corresponds, however weakly to the process exemplified above, must be hypothesized to exist.

In addition to the alphabet used to encode standard descriptions, Turing's universal machine required additional symbols to act as markers. One set of temporary markers was specified; Turing used the letters "u", "v", "w", "x", "y" and "z" for this purpose. He also used the colon, a double colon serving as a single symbol, and an inverted and reversed lower case "e", as permanent markers. For convenience here, the double colon is replaced by the asterisk "*" and the inverted "e" is replaced by "e". Turing called the digits "0" and "1" "figures" or "symbols of the first kind"; all other symbols were "symbols of the second kind".

The tape of the universal machine was one-way infinite to the right of a fixed starting point which consisted of the symbol "e" printed on two consecutive squares. Turing considered the tape to consist of two alternating sequences of squares. Counting squares from the left, he called the odd numbered sequence "F-squares" and the even numbered sequence "E-squares". E-squares were used exclusively for markers which could be erased, whereas the symbols on F-squares formed an unbroken sequence. Once a symbol was printed on an F-square it was never altered. Thus the end of the sequence printed on F-squares could be identified by finding two consecutive blank squares and the start by finding the marker string "ee". A symbol 'β' on an F-square, and also the F-square itself were said to be marked by a symbol 'α', if 'α' appeared on the E-square immediately to the right of the F-square containing 'β'. Marking is a highly significant operation in the context of the universal machine. For example, it is the method by which the

current configuration is compared with the configuration of an instruction in the standard description of the machine being simulated. The universal machine was started on a tape which was blank except for the standard description of a target machine printed immediately to the right of the marker string "ee" and terminated with the symbol "*".

Given the large symbol alphabet and the task of producing a universal interpreter which Turing was tackling de novo, it is unsurprising that the finite state control of the first universal Turing machine is a complex structure of several hundred states. To handle the complexity Turing developed an economical notation for the description of commonly used processes which includes ancestors of the concepts of a subroutine and of a formal parameter. It is not, however, a programming language in the modern sense, but a way of abbreviating the machine table description of a structural unit. The notation can be seen to point in two rather different ways when one is thinking of implementation. These different implementation possibilities are profoundly interesting in the context of the study of real cognitive systems because they relate to the nature of control memory. They can be explored by studying Turing's notation with the help of state diagrams.

The essential elements of the notation were what Turing called "skeleton tables" or "*m*-functions". An important example is shown in Table 1.

m-config.	*Symbol*	*Behaviour*	*Final m-config.*
	e	L	f1(C,B,α)
f(C,B,α)	not e	L	f(C,B,α)
	None	L	f(C,B,α)
	α		C
f1(C,B,α)	not α	R	f1(C,B,α)
	None	R	f2(C,B,α)
	α		C
f2(C,B,α)	not α	R	f1(C,B,α)
	None	R	B

Table I. The skeleton table for f(C,B,α).

The process described in Table I finds the leftmost occurrence on the tape of a symbol "α" if there is one. The scanning head is moved to the left until the end of tape marker "e" is encountered. Then it is moved to the right looking for the target symbol but also checking for the occurrence of two consecutive blanks indicating the other end of the tape and showing that the target symbol does not occur. Whereas an ordinary machine table would have named configurations in the first column, Table I has the parameterized expressions $f(C,B,\alpha)$, $f_1(C,B,\alpha)$ and $f_2(C,B,\alpha)$. These are *m*-configurations and the table as a whole is a skeleton table. Parameters C and B are also *m*-configurations and α is a symbol. A final transition is made to C if α is found and to B otherwise.

Turing said that "skeleton tables are to be regarded as nothing but abbreviations: they are not essential." While this is true in a sense, it displays Turing's characteristic modesty and underestimates the value of the technique. Two points are particularly worth making. First, the *m*-function notation is very economical although it can be difficult to comprehend. Second, the technique invites the specification of complex structures in terms of simple parts. Turing used this technique wherever possible in the specification of the control of the universal machine even when simple, much more efficient alternative state structures could have been devised. In fact, all but 14 of the 420 states in the universal machine control were specified in terms of *m*-functions. To understand the technique both Turing's notation and state diagrams are useful. In Figure 3.1 the state diagram of f(C,B,a) and its relation to the machine table are shown.

The states f, f1 and f2 of Figure 3.1 are the realizations of $f(C,B,\alpha)$, $f1(C,B,\alpha)$ and $f2(C,B,\alpha)$ respectively and the target symbol α is 'a'. The three internal states of the structure and the relations among them are what would be common to any instance of the *m*-function. The transition arcs to states C and B are shown rather than the final states themselves. Although the "f-unit" as it can usefully be called, is an abstract logical structure, there is no reason why it should not be thought of as a structural unit akin to a logic chip. This kinship becomes even more apparent

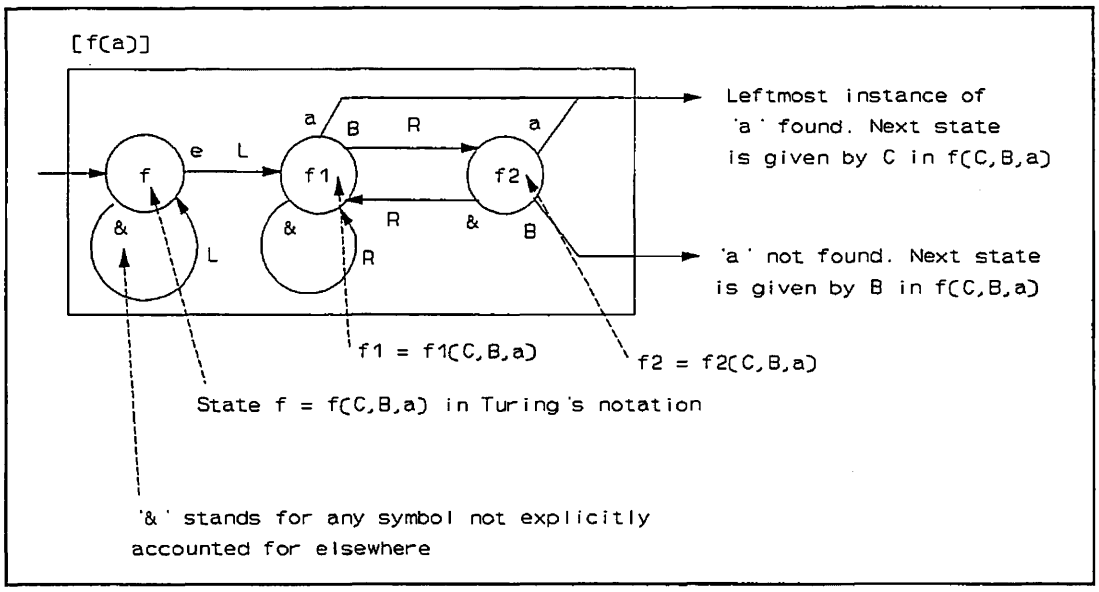

Figure 3.1. The f-unit, the basic building block of Turing's machine.

when its use in the specification of a more complex structure is examined. Suppose the task was to search for the first instance of symbol 'x' followed by the first instance of symbol 'y' if an 'x' was found, with a final transition to state C given both 'x' and 'y' found, and a final transition to state 'B' if either was not found. In Turing's notation this would be expressed as f(f(C,B,y),B,x), i.e. a nesting of two f-units. f(C,B,y) is specified as the state to which a transition is made if 'x' is found[8]. By contrast, in f(C,f(C,B,y),x), the search for 'y' is carried out only if 'x' is not found, because f(C,B,y) is parameter B in f(C,B,x). State diagrams for both of these nested structures are shown in Figure 3.2.

Turing used a variety of other structures in addition to the f-unit as foundations for the universal machine construction. Like the f-unit these structures reflect the organization of the tape.

Figure 3.3 shows the structures f'(α), q(α) and con(α) as well as the f-unit f(α). f'(α) is an f-unit augmented by a single state which positions the control one square

[8]The order of evaluation may initially be confusing for programmers. The outermost function is carried out first, followed by whatever is conditionally specified to follow. For example, in f(f(C,B,y),f(C,B,z),x) a search for 'x' is made first followed by a search for 'y' if 'x' is found or for 'z' if 'x' is not found.

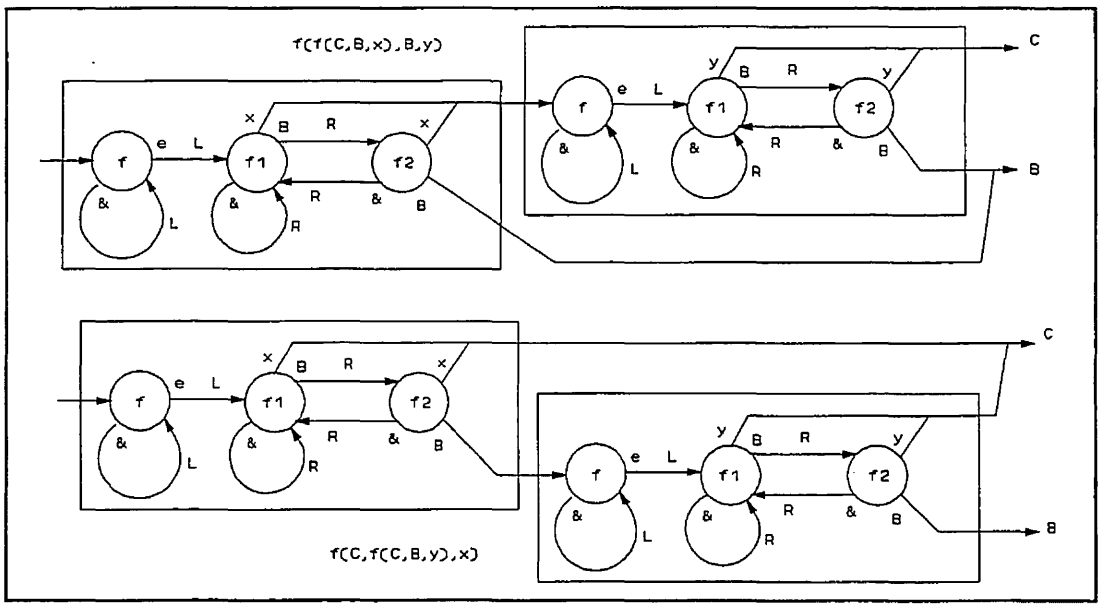

Figure 3.2. Examples of function specification using nested f-units.

to the left of the symbol found. In such instances the symbol found is almost invariably a marker. q(α) finds the rightmost occurrence of the symbol α. It is assumed that there is at least one instance of α on the tape. Con(α) is used to mark a configuration with the symbol α. A configuration is a sequence of symbols on F-squares of the form DA(A*)D(C*), where (A*) and (C*) stand for zero or more occurrences of A and C respectively. Configurations determine the operations of Turing machines, hence their identification is a matter of fundamental importance. Since a configuration is a sequence on F-squares, the E-squares between them can be used for markers. DxAxDxCxCx is how the configuration DADCC marked with 'x' would appear on the tape of the universal machine.

In addition to structures for searching and marking, structures for erasing markers are also needed. Figure 3.4 shows the e(α), eall(α) and er(M) structures. e(α) is an f-unit to which a single state has been added which erases the symbol found. eall(α) has the same state structure as e(α) but the transition after an instance of the target symbol has been erased, reruns the whole process. er(M) is a simple structure which erases any markers found on the tape.

The final *m*-functions which constitute components of the universal machine are those which write output on the tape. They represent a second order of complexity

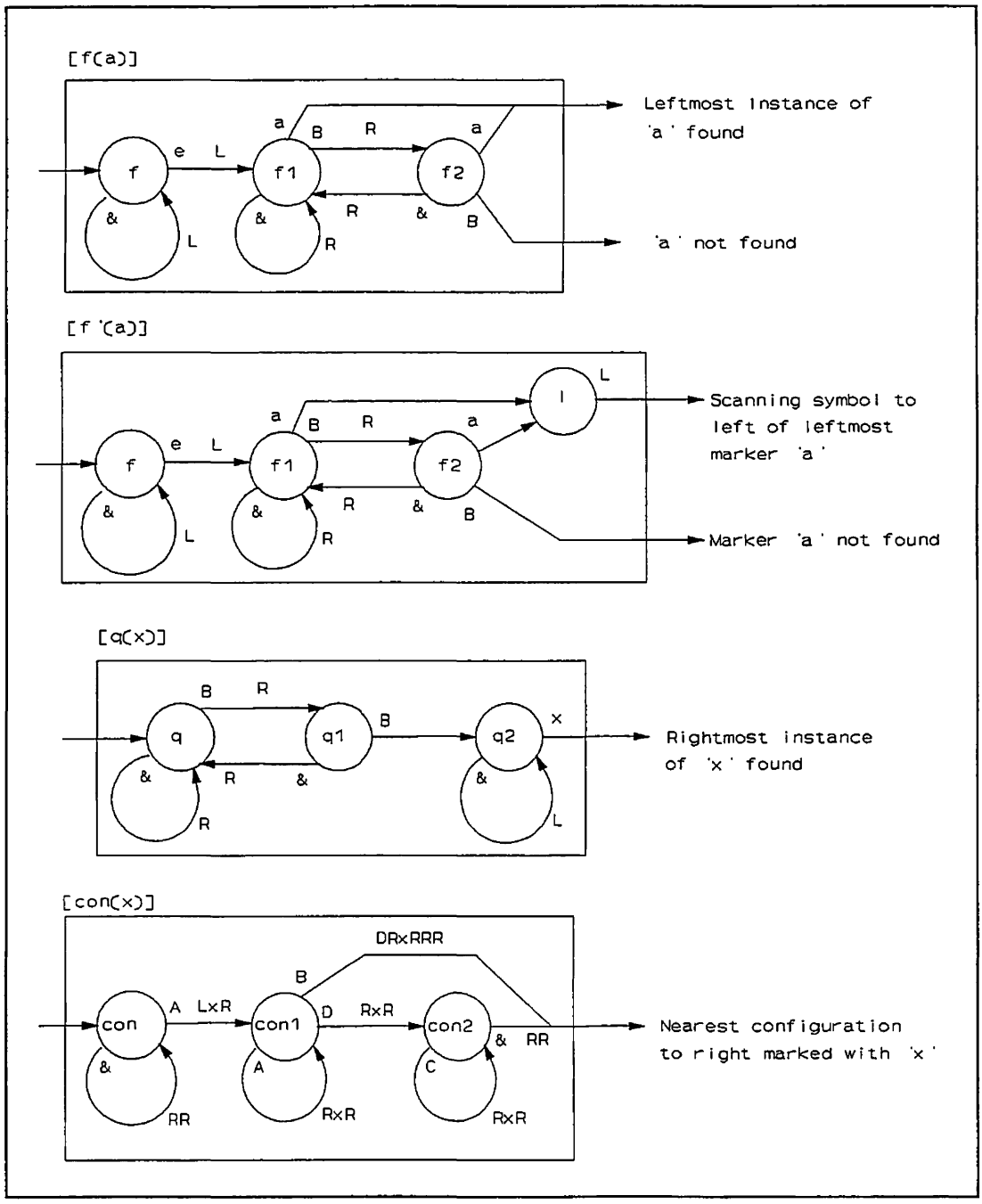

Figure 3.3. *m*-functions for searching and marking.

as they include other *m*-functions as parts. Ce(v), pe(D) and pe2(0,:) are shown in Figure 3.5. ce(v) copies each symbol marked by a 'v' to the end of the tape and erases the markers. pe(D) is a component of ce(v). It uses an f-unit to position the scanning head over the first F-square and then scans successive F-squares until a blank one is found. Given Turing's conventions, the first blank F-square will be at the end of the used portion of tape. pe2(0,:) illustrates very clearly the inefficiency to which simple concatenation of structures can give rise because it

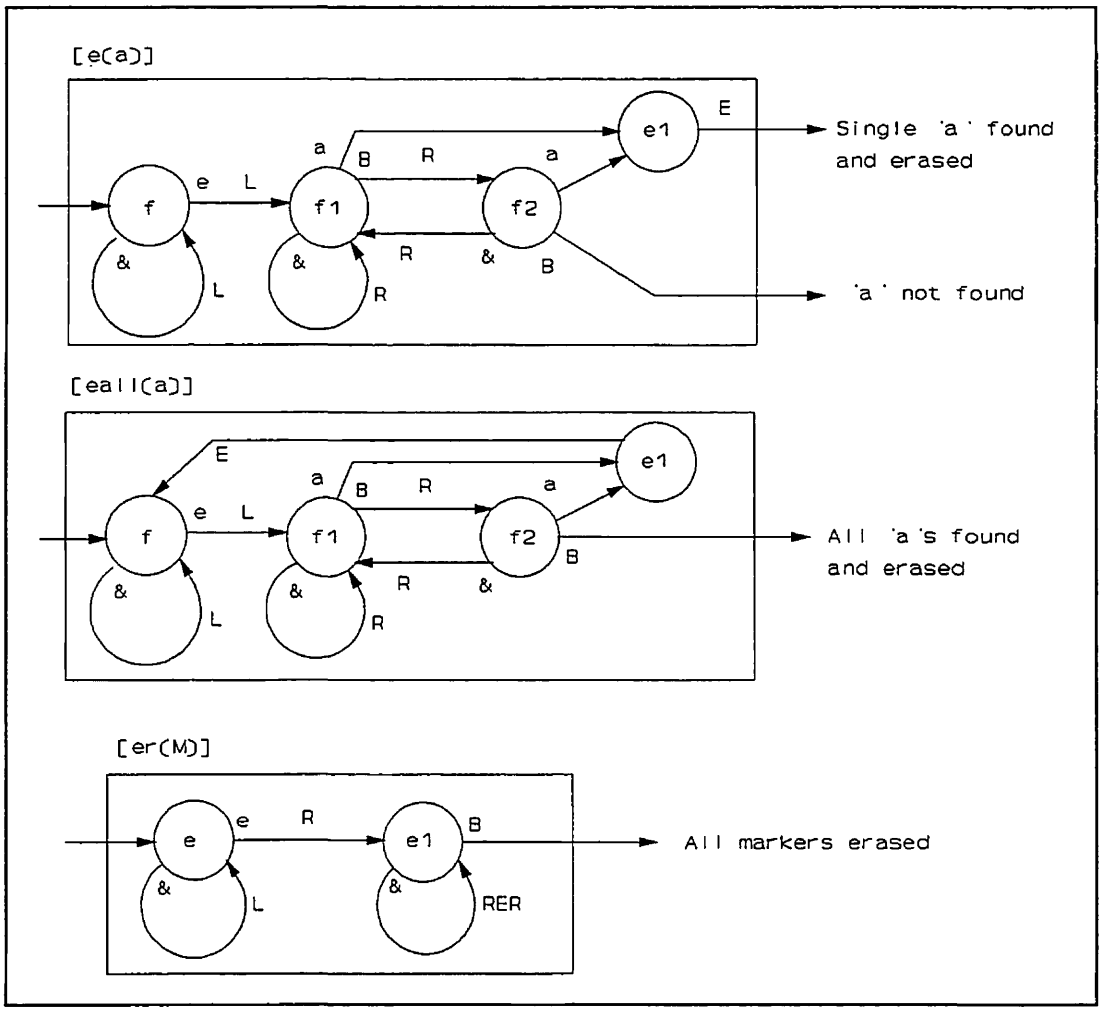

Figure 3.4. *m*-functions for erasing markers.

involves an entirely unnecessary traverse of the used portion of the tape in both directions.

With these preliminaries in place, Figure 3.6. shows the complete set of structures comprising the control of Turing's universal machine and Table II shows the machine table as Turing presented it. Figure 3.6 provides the basis for a discussion of control memory and implementation options. The circles indicate single states, the solid rectangles *m*-functions, and the dashed rectangles indicate regions of the structure which have a common purpose and are defined as multi-levelled *m*-functions. The previous figures provide all the components needed to complete the full state diagram. A brief description of the processing cycle will motivate the discussion. The machine works by writing out successive complete configurations of the target machine, interspersed with the sequence of '0's and '1's which

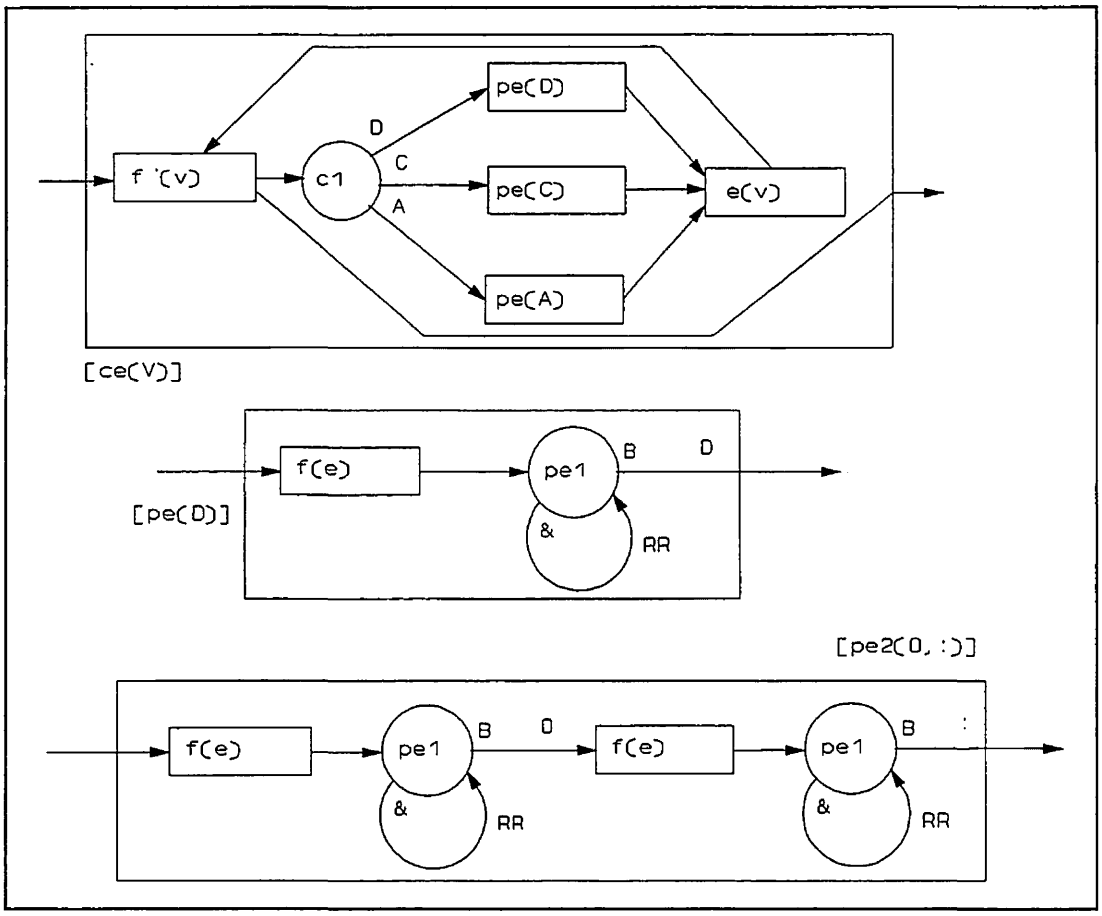

Figure 3.5. *m*-functions for copying and printing

constituted the number computed by the target. The machine is started on a tape which is blank apart from the standard description of the target machine which is written on F-squares at the left end of the tape and terminated with the symbol '*'. Processing is initialized by having the machine write the first configuration, marked with 'y' at the end of the tape. This will always be DyAyDy indicating that the target machine was started in state q1 scanning a blank. The matching configuration in the standard description of the target is found by a pattern matching search using the 'kom' and 'kmp' structures. The 'kmp' structure is particularly interesting both for the brevity of its definition and for the complex task which is executed by a system composed of nothing other than f-units and erasers. It is an outstanding example of the concatenation of simple structures to produce a complex outcome and is discussed further below. Once the appropriate instruction in the standard description of the target has been identified, the 'sim' structures mark this instruction in preparation for the next stage of processing. The configuration, which

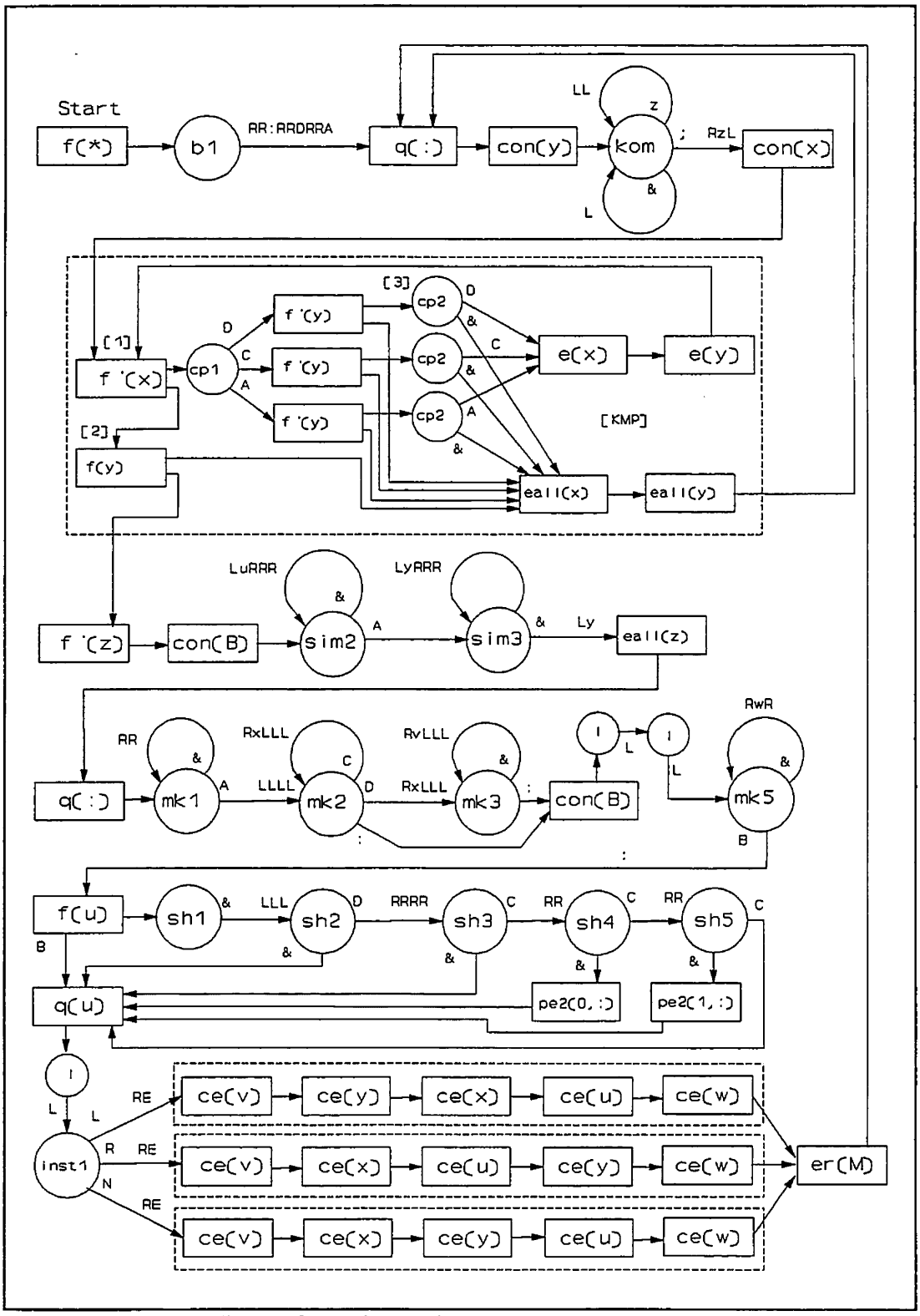

Figure 3.6. The state diagram for Turing's universal machine.

was used to identify the instruction, is now redundant and is marked with blanks. The operations, i.e. the symbol to write and the movement indicator are marked with 'u', and the next state is marked with 'y'. Any remaining 'z' markers from

the pattern matching process are erased. Having marked the instruction, the machine then marks out the most recent complete configuration on the tape. The components of this, along with the appropriate parts of the marked instruction in the standard description make up the next complete configuration. To simulate the movement

m-config	symbol	operations	final *m*-config
b			f(b1,b1,*)
b1		RR:RRDRRA	anf
anf			q(anf1,:)
anf1			con(kom,y)
kom	;	RzL	con(kmp,x)
kom	z	LL	kom
kom	not z or ;	L	kom
kmp			cpe(e(e(anf,y),sim,x,y)
sim			f'(sim1,sim1,z)
sim2	A		sim3
sim2	not A	LuRRR	sim2
sim3	not A	Ly	e(mk,z)
sim3	A	LyRRR	sim3
mk			q(mk,:)
mk1	not A	RR	mk1
mk1	A	LLLL	mk2
mk2	C	RxLLL	mk2
mk2	:		mk4
mk2	D	RxLLL	mk3
mk3	not :	RvLLL	mk3
mk3	:		mk4
mk4			con(l(l(mk5)),B)
mk5	Any	RwR	mk5

mk5	None	:	sh
sh			f(sh1,inst,u)
sh1		LLL	sh2
sh2	D	RRRR	sh3
sh2	not D		inst
sh3	C	RR	sh4
sh3	not C		inst
sh4	C	RR	sh5
sh4	not C		pe2(inst,0,:)
sh5	C		inst
sh5	not C		pe2(inst,1,:)
inst			q(l(inst1),u)
inst1	α	RE	inst1(α)
inst1(L)			ce5(ov,v,y,x,u,w)
inst1(R)			ce5(ov,v,x,u,y,w)
inst1(N)			ce5(ov,v,xy,u,w)
ov			er(anf)

Table II. The machine table for Turing's universal machine.

of the target machine relative to its tape, Turing used a convention in which the *m*-configuration was written immediately to the left of the scanned symbol in the complete configuration. Thus the string D C C D C C D A D C D C C D C C represents a machine in state q1 scanning the '0' of the string '11011'. Suppose the instruction for configuration D A D C specified a move left and a transition to state q2, the complete configuration D C C D C C D A D C D C C D C C would need to be replaced by D C C D A A D C C D C D C C D C C. Turing managed this with the following marking convention. The original complete configuration is marked by the 'mk' structures as DvCvCvDxCxCxD_A_D_C_DwCwCwDwCwCw, where the underscore '_' indicates marking with a blank. The part of the instruction

in the standard description indicating the output symbol and a transition to state q2 would have been previously marked as DuCuDyAyAy by the 'sim' structures. The simulation of movement is achieved by writing out the marked parts in the appropriate order. Turing did this with the set of ce(α) structures which is discussed below. For a move left the order is 'v','y','x','u','w' which, for the example given, yields D C C followed by D A A followed by D C C followed by D C followed by D C C D C C. Having written out the new complete configuration the machine makes a transition to state 'anf' to begin the next instruction matching cycle.

3.5. Control memory in Turing's universal machine.

The most significant portions of the universal machine control for demonstrating the control memory concept are the 'kmp' set of structures and the 'ce(α)' structures. The 'kmp' structures carry out a symbol by symbol configuration matching process given a pair of configurations, somewhere on the tape, one of which has been marked with 'x's and one with 'y's. Turing defined 'kmp' as a complex *m*-function with a single entry point, and two exit points. The two exit points implement a branching operation which is conditional on the success of the configuration matching process. Thus the function of 'kmp' is to compute the conditional "*If the marked configurations are the same go to 'sim', else go to 'anf'*".

The first important point to note is how '*same*' is implicitly defined in terms of the symbol structures which can appear on the machine's tape and the possible routes through the 'kmp' function. Two configurations are the same if and only if the machine fails to find a 'y' marker while executing f(y), structure number [2] in the state diagram for 'kmp' (Figure 3.6). Consider a hypothetical tape expression DxAxDx#####DyAyDy, where ##### stands for an arbitrary number of intervening squares with no 'x' or 'y' markers on them. 'kmp' works through this expression by finding the first 'x' marker and noting the symbol it marks. Then it finds the first 'y' marker and notes that the symbol it marks matches the symbol marked by the 'x' marker. The first 'x' marker and the first 'y' marker are deleted and the process is repeated on the expression D_AxDx#####D_AyDy. This cycle is

repeated twice more successfully but on the fourth invocation the machine finds neither an 'x' nor a 'y' marker. This must signal that the configurations matched. Consider the alternatives. First, the expressions marked with 'x' and 'y' might have been of different lengths. In such a case the instances of either the 'x' or the 'y' marker would be exhausted first leaving at least one instance of the other marker still on the tape. If the 'x' markers had been exhausted there would still be at least one 'y' marker on the tape which would be found by the machine executing structure [2]. If the 'y' markers had been exhausted the machine would not have entered structure [2] because an 'x' marker would have been found by f'(x), and the lack of a 'y' marker would then have been detected by the appropriate f'(y) unit. Alternatively a pair of symbols might have failed to match at some stage in the processing. In such cases also, the machine would not have entered structure number [2] and the mismatch would have been detected by one of the 'cp2' states. Thus the only case in which the machine can be executing structure [2] and can fail to find a 'y' marker is when there were equal numbers of 'x' and 'y' markers to begin with and each pair marked a token of the same symbol. Bearing in mind that the 'kmp' structure is 'internal' to the machine because it is part of the state structure of the control automaton, and that the marked expressions are 'external' because they are on the tape and the tape, by definition, is an external, auxiliary memory resource, the notion of 'same' is derived from the interaction of internal and external structures and is implicit in the actions of the control. Given the standpoint of the ETH which equates the brain with the control of a Turing machine, it follows that cognitive operations resulting in intentional notions like 'sameness' may also result from, or be modelled as, the interactions of internal and external structure.

A second important point about the 'kmp' *m*-function is that it provides a very clear example of a second form of implicit control memory for symbols. The first form, discussed in Chapter 2, was a functional type of implicit memory. It was appropriate to identify state q1 of machine M as maintaining an implicit memory for a right parenthesis because of its subsequent behaviour. The form discussed here is a positional type of implicit memory. The relevant elements of the structure are

the 'cp1' state, the three f'(y) *m*-functions to which it leads and the three 'cp2' states to which they lead. The example is particularly clear because it does not matter how symbol recognition is carried out by the 'cp1' and 'cp2' states. The first aspect of the structure to note is that the three f'(y) structures are functionally identical in terms of their internal operations. This amounts to the claim that if they were real objects which could be plugged into three positions in a printed circuit board they could be swapped around without affecting the functionality of the system. Despite their internal, functional identity their functional roles in the wider system of the 'kmp' *m*-function are different. One functions as an implicit memory for a 'D', one for a 'C' and one for an 'A'. Two points justify this claim. The first is that exactly three f'(y) structures are needed because the machine has to deal with three different symbols at this stage of processing. If only two f'(y) units were available the machine could not make all the necessary discriminations and would not be able to carry out the configuration matching process. The second point is to note what would happen if, for example, the 'cp2' state, numbered [3] were to make a transition to e(x) if the symbol marked by 'y' were a 'C' rather than a 'D'. This would constitute an error and would lead to failure of the configuration matching process. The reason is that the path to the 'cp2' state numbered [3] is that used when the symbol scanned by the 'cp1' state is a 'D'. Thus the f'(y) unit that lies on the path between the 'cp1' state and the 'cp2' state numbered [3] functions as an implicit memory for a 'D' by virtue of its position in the larger 'kmp' structure.

Positional implicit memory is a significant feature of the control memory of Turing machines in the context of the ETH and there is good reason to believe that position is a significant organizational principle in the nervous system. Kuffler, Nicholls & Martin (1984, p.6), for example, describe a fundmental principle of organization in the brain in the following terms; "The quality or meaning of a signal depends on the origins and destinations of the nerve fibers, that is, on their connections." Thus it would seem that the nervous system makes substantial use of a form of memory which is characteristic of finite automata. If the brain is better thought of as a finite automaton rather than as a whole Turing machine as is claimed by the ETH, and if, in consequence, human memory should be thought of as modelled by control

memory rather than tape memory, then it is of course important that the principles of operation of control memory should be plausible for the brain.

The idea of positional implicit memory is interestingly linked to an apparently rather plausible hypothesis about the relationship between memory and modular structure. Control memory appears to imply modular structure in a way that symbolic, tape memory does not. 'kmp' again provides an interesting example. The point to notice is that although the behaviour of the machine in the states represented by each of the three f'(y) units is essentially identical, there must be no cross-talk between them because they are subserving different memory functions. The logical state structure thus implies an implementation constraint given the assumption that memory is non-symbolic. This leads to an interesting secondary way of interpreting state diagrams. Complex state diagrams look very much like printed circuit board diagrams and although this can be very misleading the resemblance is not entirely coincidental. Although a node in a state diagram is properly thought of as identifying a global logical state of a machine, it can also be thought of as a marker for a sub-assembly of parts which might be used to construct a machine with the set of logical properties exhibited in the state diagram. The possibility of using a simple network to implement the XOR function discussed in Chapter 2 suggested as much, as does Turing's method of constructing complex *m*-functions from multiple copies of simple units. Nodes linked by arrows imply sub-assemblies linked causally, and nodes unlinked by arrows imply autonomous sub-assemblies. The linkage between position and content seen in control memory is not a principle which is characteristic of the symbolic memories of computers or the tapes of Turing machines. The fundamental point about these memories, although they are modular in the sense of being divided into separate locations, is precisely that they are undedicated in terms of content. It seems clear, therefore, that to distinguish, as Turing did, between external symbolic memory, and internal, brain like, control memory, is to make a distinction which accords to a first approximation with what is known about the brain.

The other part of the universal machine construction which is of interest, both from

the point of view of the organization of control memory and as an example of Turing's ingenuity in using existing definitions, is the set of ce(α) structures which implement the copying of marked regions of tape in order to construct successive complete configurations. As is apparent from Figure 3.6 these structures are divided into three sets of five units, the choice among the three sets being determined by the movement symbol in the active instruction of the target machine. Although the set of ce(α) structures accounts for 315 of the 420 states of the machine, the organization is entirely regular and straightforward. The major structuring principle is the division into three separate paths corresponding, as already observed, to the three possible movement symbols. It is appropriate to think of each path as constituting an implicit memory for its originating movement symbol, if only because it is the identity of this symbol which determines the order in which the different marked regions are dealt with within the path. Each ce(α) unit consists of an f'(α) unit which finds the first occurrence of a symbol marked with α, a 'c1' state which determines what action to take, three pe(β) units which print the appropriate symbol 'D','C', or 'A' at the end of the tape, and an e(α) unit which erases the marker. This unit makes a transition back to the f'(α) unit to repeat the process which is terminated when no further instances of the marker are found. Control is then passed to the next ce(γ) unit. The large scale replication of functional units, which is the most striking aspect of the ce(α) set of structures, simplifies what would otherwise be a complex control problem at the cost of a considerable increase in the size of the machine.

3.6. Styles of implementation.

This observation leads naturally to a question about implementation style. It is important to remember that although it is tempting to treat diagrams like Figure 3.6 as a sketch for a printed circuit board, it is in fact a picture of logical structure which is compatible with a variety of realizations. Two broad strategies, at opposite ends of the spectrum are of particular interest. There is the strategy of using multiple replications of functional units with minimal control circuitry and content specified by position which is clearly seen in Figure 3.6 or there is the strategy of using fewer functional units with parameter passing and more complex control

circuitry. These might reasonably be called the MAXH-MINC (maximum hardware-minimum control) and MINH-MAXC (minimum hardware-maximum control) strategies. Consider the ce(v) unit shown in Figure 3.5. Most of its structure is identical to that of all the other ce(α) units. The only variation is in the identity of the marker which is relevant at the finding and erasing stages. So the question very naturally arises, when considering implementation, whether it might be possible to find a way of passing this variable information to a real structure at execution time. If this could be done, the physical replication of units could be avoided in favour of a single unit plus an appropriate stream of instructions. The seeds of the idea are to be found in Turing's notion of a skeleton table and it is this which makes his own assessment of skeleton tables as mere abbreviations less than generous. One obvious way of passing variable information is in the form of explicit symbols. Figure 3.7 shows a schematic view of the sort of mechanism which might be constructed pursuing the MINH-MAXC strategy. It is purely notional and is intended solely as an illustration.

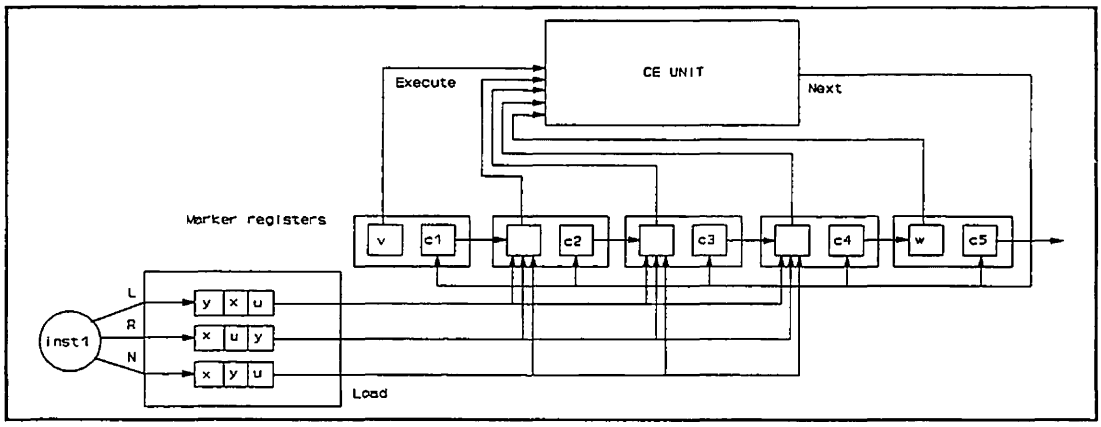

Figure 3.7. A hypothetical implementation scheme using only one ce(α) unit.

From whatever structure implements 'inst1', control is passed to a 'Load' unit which contains symbolic information about the marker parameters for the second, third and fourth cycles through the CE unit. The first and fifth are fixed as 'v' and 'w' respectively which saves a certain amount of control wiring. The load unit installs the appropriate values in the variable marker registers and initiates the execution cycle which starts with the first marker register containing 'v'. This is passed to the CE unit, whose internal operation would be as shown

in Figure 3.5. After all the 'v' markers are exhausted a signal is sent on the 'next' line. Precise details are unimportant but the idea is that on the first pass the 'next' signal would activate 'c1' in the 'v' marker register which would pass control to the second marker register. This would send its marker to the CE unit and set 'c2' ready to receive its own 'next' signal. And so on.

If the ETH is correct, discussion of implementation options is particularly important and the issues which are considered here in the context of Turing's universal machine are issues which would arise with respect to the implementation of any complex machine. The ETH proposes a MAXH-MINC implementation, but if the cognitive system is an implementation of a Turing machine, roughly of the same type as digital computers as the generic theory suggests, then there is a strong presumption in favour of the minimal replication, symbol based, complex control, MINH-MAXC strategy. Some of the reasons for this are considered in Chapter 4. What this strategy does, in effect, is to exchange hardware for software. One aspect of the trade-off is ease of modifiability, and hence greater flexibility of function at the cost of increased control complexity, and probably at the cost of slower execution given the need to initialize components like marker registers.

One profound difference between the two strategies is the need, in the MINH-MAXC case for the passing of symbolic parameters, whereas in the MAXH-MINC strategy, which would be exemplified by a direct implementation of Turing's design, what is needed is an activation signal rather than a symbolic parameter. In the former, but not the latter case, explicit symbolic information is passed and thus it is appropriate to think of the system as a symbolic information processor. When the brain is considered, it seems almost certain that signals rather than explicit information are what is passed at the level of neurons and this tends to suggest, along with the evidence for the importance of position as a principle of brain organization, that the brain implements a MAXH-MINC strategy and should perhaps be thought of primarily as a signal processor rather than a symbolic information processor. It is important to be clear just what this claim amounts to and it is

easiest to understand first in the context of the Turing machine. That the Turing machine as a whole is an information processor is not in question. The information which is processed is found on the machine's tape and the transformation of input into output which constitutes the function computed is a paradigmatic information process. The question at issue is whether the finite state control of a Turing machine is also an information processor and the answer to this question appears to depend on the implementation strategy adopted. It seems perfectly possible for a system as a whole to be an information processor while its separate parts are not. Broadly speaking, the MAXH-MINC strategy implies a view of automata which takes them to be signal processors, and the MINH-MAXC strategy implies a view which takes them to be information processors. Thus the suggestion that the brain, construed as a finite control automaton is not an information processor does not entail or imply the false claim that the cognitive system as a whole is not an information processor. What is at stake is the description of the internal system of components and their arrangement which implements the control system for the information processes which the system as a whole executes.

To summarize the arguments of Chapter 3 thus far: from consideration of the example of a human engaged in a routine calculation, Turing developed a machine model to capture the essential processes involved; the model consisted of a finite control automaton which modelled the algorithm embodied in the mind of the human, and an indefinitely expandable tape which modelled the paper on which the calculation was worked. The identification of the human mind with the finite control system, and not with the tape, was supported by an argument for the claim that the human brain was capable of being in only finitely many different states. Turing's further published work over a period of more than a decade supports this understanding of his analysis of computation. To treat the brain as a whole Turing machine and human memory as modelled by symbolic tape memory is thus a misuse of the model.

It has further been argued on the basis of an analysis of Turing's universal machine construction that a positional form of control memory, in addition to the functional

form of control memory discussed in Chapter 2, can be identified and that a distinction can be made between implementations of finite control automata which take them to be information processors and those which take them to be signal processors. Turing's machine model is compatible with both styles of implementation. On the basis of the above, the idea that the brain is a finite, signal processing automaton is perfectly compatible with the view that it is internally non-symbolic but that it is a component of an information processing cognitive system which makes use of explicit symbols externally. The ETH thus appears to be well supported by Turing's analysis of computation.

3.7. Serial and parallel architectures.

Two further points are worthy of brief consideration here. First is the issue of serial versus parallel architectures and processing. It has sometimes been suggested that the brain cannot be a Turing machine because it is a parallel system whereas a Turing machine is a serial system. This criticism might be thought to apply to the ETH as well, even though it denies that the brain is a Turing machine, because a finite automaton is also a serial processing system in that it passes from one global state to the next. The first point to make is to distinguish the issues of architectures and processing. A serial architecture is essentially one built according to the MINH-MAXC strategy. In such an architecture simultaneous processes are impossible if each process requires the same operation and only one hardware component implementing the operation is available. This is the case with most commercial computers which have a single CPU. A parallel architecture is one in which multiple processors are available for a given operation thus allowing process parallelism. It is apparent, however, that although a serial architecture rules out parallel processing, a parallel architecture does not rule out sequential processing. The architecture of the ce(α) units, for example, would be parallel if a MAXH-MINC implementation were built, but processing would still be sequential. Parallel architectures are discussed in greater detail in Chapter 4. For the present however, the fact that the brain has a parallel architecture does not rule out its functioning as a sequential system at the level of global state transitions.

3.8. Configurations, thought and behaviour.

The second point to make concerns the nature of the relationship between the control of a Turing machine and its inputs. One of the most valuable features of the generic computer theory of mind is that it seems to give a good account of how it might be possible for the cognitive system to operate so as to make thought more or less independent of the external stimulus environment. The picture of the mind as an internal Turing machine communicating with the outside world via sensory and motor transducers appears to give just the right sort of account, because internal computational processing is independent of the world and yet connected to it. At first sight it might appear that this semi-autonomy is sacrificed by the ETH, because the ETH takes the concept of a configuration, i.e. the combination of current internal state and current symbol scanned which determines the next step of a Turing machine, to be a relation between the organism and the world whereas for the generic theory it is an internal relation between different parts of the neural Turing machine. Thus it might appear that the possibility of autonomous internal processing is ruled out by the ETH.

In one sense this is true and part of the purpose of the ETH is to encourage a view of the organism as more closely tied to the external environment than the generic theory suggests. However, the grip of the environment as a component of computational configurations is much less unyielding than might at first appear. As before, it is necessary to examine Turing machines of considerable internal complexity to get a true feel for what the relationship between symbol and state amounts to and what is ruled out by supposing configurations to involve the external world.

It was observed in Chapter 2 that the point of having a system with more than one internal state was to enable more than one response to a given stimulus to be possible while still retaining a deterministic approach to processing. What this suggests is that it is not independence from external input which promotes autonomy but an appropriately extensive set of internal states. The fact is that human organisms are not insensate, unsituated entities in the way that computers are. In

ordinary circumstances the awake human cannot choose not to see or hear or feel. But this does not diminish the human capacity for voluntary action. Thus the immediacy of interaction with the environment implied by the ETH is not a recipe for ironclad determinism. More significantly, in the Orwellian world of the Turing machine, some symbols are more equal than others. Consider the f-unit, the basic building block of Turing's machine, whose state diagram is shown in Figure 3.3. For the f-unit in state 'f', its inputs are divided into two classes, instances of 'e' and everything else, the point, of course, being that the occurrence of an 'e' causes a state transition to state 'f1' while any other input leaves the machine in state 'f'. Once in state 'f1', 'e' loses its more equal status and becomes just another animal in the farmyard while the target symbol and the blank become the 'privileged' members of the alphabet. One might even want to assign the target symbol higher status than the blank because it represents the acme of the f-unit's processing. Another way of expressing the same point might be to say that different states are sensitive to different symbols. The familiar example of driving a car while pre-occupied and arriving at a destination with a very uncertain grip on how the journey was achieved shows that this kind of differential sensitivity is characteristic of humans as well. It is not that the cues to which one responds automatically when driving are unimportant but that they are means to an end rather than ends in themselves. Similarly, one might plausibly regard the sequence of symbols which the f-unit has to traverse in pursuit of its target symbol as a means to an end. This is not to deny that the behaviour of the Turing machine is determined by its configurations. What it does imply is that the internal state provides the element of control. Perhaps it would not be too confusing to suggest that behaviour is determined by configurations and controlled by internal states. This idea can be interpreted in a way which has an air of plausibility to it when thinking of the sources of ordinary behaviour. If I were walking in the mountains and were overtaken by an unexpected thunderstorm, I would be exercising a remarkably intransigent notion of freedom if I continued to walk without putting on my waterproofs and took no heed of the possibility of a lightning strike. That I would, in such circumstances take the appropriate steps to weatherproof myself and get off an exposed summit is to allow that my behaviour is determined by the weather. But

it is not a form of determination which robs me of freedom to act since my behaviour in acting thus is controlled by my internal state. Furthermore, the internal state which recommends the donning of waterproofs is subordinate to the overarching state or set of states which constitute my wanting to continue my walk without being soaked or blasted by a thunderbolt. It is very interesting to note that the nesting of states within structures in complex machines echoes this dependence of human behaviour on hierarchical mental state structures. The processing of an f-unit in Turing's machine always constitutes a stage in a higher level process and the processes which have f-units as their constituents may themselves be components of still higher level processes. The erasers in the 'kmp' *m*-function are a good example. These structures have f-units as components and are themselves components of the pattern matching process which identifies the appropriate configuration in the target machine. Thus, although it is clear that the moment by moment behaviour of the machine is determined by its configuration, it is less easy to say exactly which process is controlling behaviour. Is it the f-unit, the eraser or the pattern matching process carried out by the 'kmp' *m*-function, or is it a combination of all of these or even of some still more remote constellation of state structures. Quite clearly then, the control of behaviour by internal states, as distinct from the determination of immediate behaviour by the current configuration, can extend over lengthy time sequences and embrace multifarious inputs. This is a view which seems at least as plausible as the idea that behaviour is controlled by a semi-autonomous internal computer which is isolated from direct contact with the world in which the behaviour it controls has to happen.

In conclusion, both Turing's analysis of computation and the design methods he used to develop the universal machine appear to support the claims of the ETH that the brain should be considered as a finite control automaton rather than a complete Turing machine. Control memory turns out to be a complex, multifaceted concept which is at least as plausible a model of human memory as the more familiar model based on symbolic, tape style memory. In addition the range of interactions between internal states and external symbols appears to offer a plausible basis for an account of the relationship between human cognizers and their environments.

Clearly such an account needs elaboration. What the present chapter has shown is that it is not a task which is doomed from the start.

Chapter 4. Digital Computer Models.

In Chapter 3, Turing's analysis of computation was used to support the basic claim of the ETH that the brain should not be considered to implement a whole Turing machine, but the finite state control of a Turing machine. This suggestion runs counter to a wide variety of well established and respected theories which have been grouped together under the general label of the generic theory. It is characteristic of generic theorists to argue that the brain does implement a Turing machine of some kind and that the mind is essentially the program or set of programs which are executed by this machine.

Clearly it would be absurd to suggest that generic theorists are committed to the idea that the brain contains an infinite, linear tape which is traversed by a finite state control, and it is relevant to ask what the generic theory proposes in the absence of such an arrangement. Broadly speaking, digital computers have provided the model for the architecture which the brain is hypothesized to implement and the purpose of this chapter is to examine briefly what the commitments of such a model are.

4.1. The architectural commitments of the generic theory.

Clearly, the major commitments are to an internal separation of memory from control and to explicit symbolic expressions. Although the generic theory is not committed to the notion of a linear tape, it is committed to the notion of an internal memory system which is symbolic, and which may provide both data and programs for the executive system of the cognitive computer. The memory system is separate, at least functionally, from the control system which manipulates the symbolic expressions contained in the memory to produce such characteristic phenomena of cognition as reasoning and decision making.

4.1.1. Location addressing.

One of the reasons why the generic theory is not committed to the idea of a linear tape and a mobile control is the difficulty of access which such a medium imposes. It is clear, therefore, that the generic theory must be committed to some form of

addressing scheme other than traversing a linear tape. The most common form of addressing scheme in digital computers is location addressing. One way to think about such a scheme which retains a contact with the material of Chapter 3, is in terms of a Turing machine with a finite tape, each of whose squares has a hard-wired connection to the control. The connection may be thought of as enabling the control to read and write from the square as though it were scanning it. On the assumption that the difference in distance between the nearest and most remote squares contributes little if anything to the access time, such hard wired connections provide constant time access to each element of the memory. The central processors of digital computers enjoy connectivity to their memories of essentially this kind even though the memory is normally two-dimensional. It is reasonable to consider the byte as the unit of computer memory analogous to the square of a Turing machine tape. Human memory may also be essentially two-dimensional if the cerebral cortex is its seat.

The advantages of a memory whose locations are systematically addressable extend beyond the simple fact of constant time access, important though this is, to the organization of both programs and data. One of the pronounced inefficiencies of Turing's universal machine is that it has to hunt through successive instructions of the standard description of its target machine in order to find the one to execute. By contrast it is standard practice in computers to arrange the instructions in sequential locations so that the next to be executed is normally to be found in the location following the location of the current instruction. Only when the current instruction is a conditional branching instruction will this not normally be so since the alternatives cannot both be next in line. The process whereby the central processor gains its next instruction, known as the instruction fetch, is therefore both fast and simple by comparison with the contortions of matching and marking which Turing's universal machine has to carry out to achieve the same effect.

4.1.2. Virtual architecture.

The advantages of addressable memory locations are perhaps even more noticeable when it comes to organizing data items. Although the basic unit is the byte or tape

square, which can hold a single symbol, larger aggregates are normally required for computing. Given a system of addressable locations, larger scale structures can easily be defined. This is done in terms of address arithmetic given a base address which serves as the entry point to the structure. Suppose, for example, that a two dimensional array of n x n squares or bytes were required given a linear tape as the medium. From a base address k at which the first element of the first row, element (1,1) was to be stored, the second element of that row would be at address k+1, the third at k+2 and so on up to the n'th and final element of the first row (1,n) at address k+n-1. The second row with element (2,1) would start at address k+n, and its final element would be at k+2n-1. The final row would start at address $k+n^2-n$ and the final element of the whole array (n,n) would be at address $k+n^2-1$. To access a given element at location (r,c) a simple calculation of k+n(r-1)+c-1 gives the offset from the base address. This is an example of a virtual structure which uses address arithmetic. There are few limits, other than space available and the imagination, on the types of virtual structure which can be specified in this way. The utility of such a structure, from the point of view of the programmer, depends of course on the programmer's being able to think of the structure in terms of its virtual characteristics rather than having to make the requisite address calculations, and much of the effort in developing programming languages has had the provision of virtual structure as a goal. When, as is commonly the case, the programmer has no access to the real machine structures but only to virtual structures presented through the constructs of a programming language, the language is taken to define a virtual machine or virtual architecture. Many cognitive scientists believe that the notion of virtual architecture is one of the keys to understanding the relationship between mind and brain, and claim that the mind is a virtual architecture. It is broadly for this reason that theorists of such persuasion tend also to argue that the study of mind can be carried out independently of the study of brain because what is of psychological interest is the virtual architecture whereas what is of neuroscientific interest is the physical architecture which provides the medium in which the virtual architecture is implemented. Given that thoughts appear to obey principles of rationality, for example, whereas neurons obey principles of biochemistry, the attraction of such a position is considerable.

4.1.3. Virtual architecture and control circuitry.

Although the attractions of thinking of mind as a virtual architecture are considerable the view has costs as well as benefits. The costs are best appreciated by thinking of the control circuitry needed for the real architecture which supports the virtual structures. The need for address arithmetic and addressable locations have already been introduced, and the latter have been seen to entail a finite store with fixed data paths between control and memory locations. This implies a great deal of wiring and switching capacity, in addition to the capacity for transmitting symbols rather than just signals. The distinction between symbols and signals was introduced in Chapter 3. Taking a crude example, if we imagine a line of people, each standing within touching distance of the next, a signal can be passed down the line by each poking the next in the ribs, whereas a symbol requires the passing of a piece of paper with the symbol written on it as well as the poke in the ribs to draw attention.

The extent of the constraints on implementation arising from the need for control circuitry to manage information access and transfer can be understood from brief consideration of the way they are managed in digital computers. Digital computers operate almost exclusively with two state basic memory units known as bistables or flip-flops. These units have been chosen for logical simplicity and because they offer the easiest means of achieving the required reliability.

> "Although other numbers of states are possible, and ternary (three-state) machines have been proposed occasionally, digital technology has developed exclusively to handle binary information. There are several reasons for this. The first is the requirement for high reliability... The second reason is the simplicity of the logic design for binary representations... A final reason... is that no one has ever found striking advantages from the resulting processing structure in having more than two states."
>
> Siewiorek, Bell & Newell (1982, pp.66-67)

Two states units can store only one binary digit or bit and have to be aggregated in order to represent a wider variety of states or entities. Typically, the eight bit byte

is the basic unit of aggregation and can represent $2^8 = 256$ different states. Encoding schemes such as ASCII assign symbolic values to various of the 256 states. The fact that symbolic representations are encodings over aggregates of binary digits means that the hard wired data paths from memory to the control have to have at least as many bits as there are bits in the representation. Contrasting this with the case of a signal for which a single wire suffices, the amount of wire needed by symbolic encoding can be seen to be considerable.

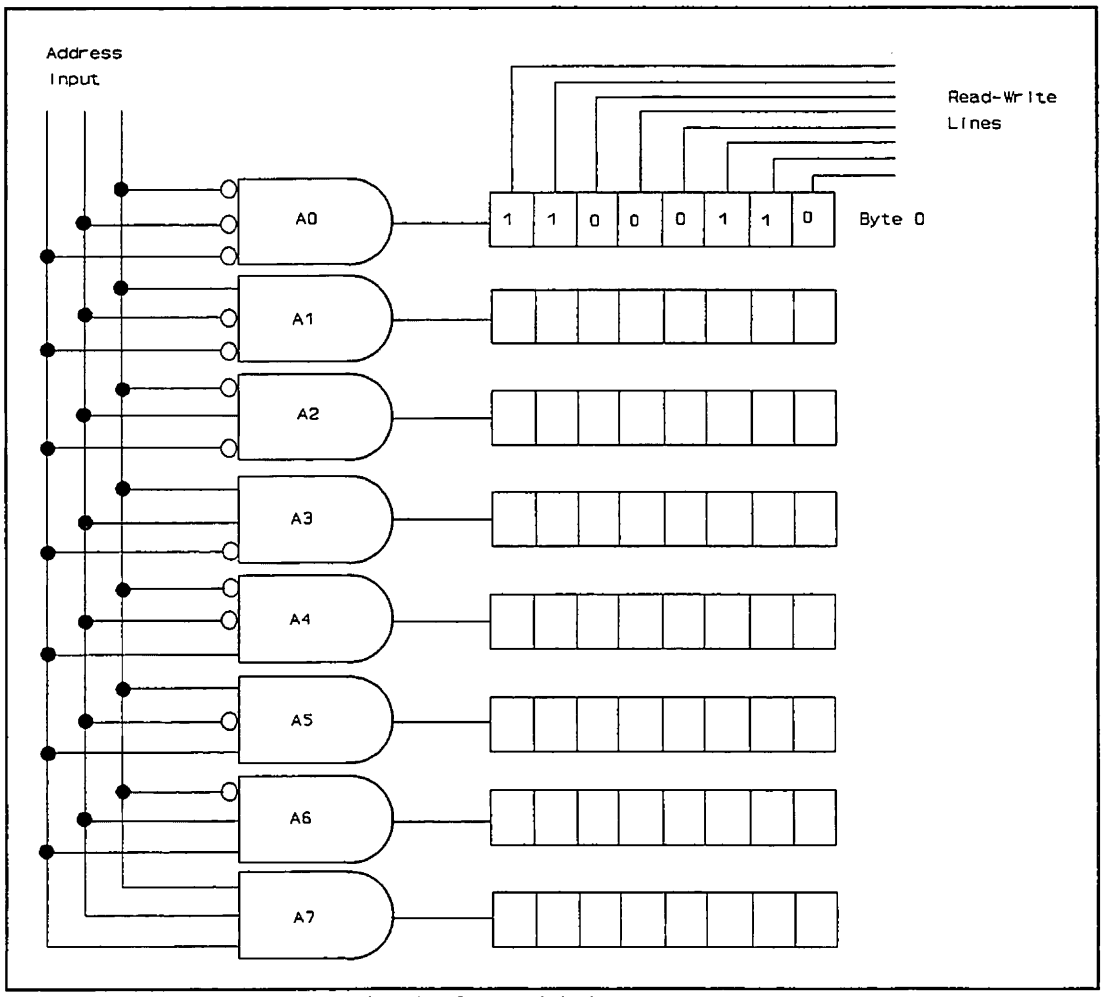

Figure 4.1. A three bit address decoder for an eight byte memory.

The load imposed by a location addressing scheme is even more substantial. Consider the simple address decoder for an eight byte memory shown in Figure 4.1. The addresses are three bit numbers and are presented on the address input lines shown at the left of the figure. The address lines are connected to eight AND gates labelled A0 to A7. The gates produce an output when all three of their inputs are

positive. The connections from the address lines to the gate inputs are shown as filled circles. The unfilled circles represent inverters, which turn an input of '0' from the address line into an input of '1' at the gate and vice versa. Inspection of the pattern of inverters shows that each of the eight possible addresses produces an output from just one of the AND gates. The address pattern '000' produces an output from gate A0, pattern '001' from gate A1 and so on. This output is the address selection mechanism which, in conjunction with the appropriate control signals on the read-write lines, will either send the pattern stored in the byte to the control or write into the byte the pattern supplied by the control. For simplicity, the read-write lines are shown connected only to Byte 0. Clearly the number of locations which can be addressed by k address lines is 2^k. Thus for a memory of 2^{20} bytes, i.e. 1 Mb. organized using an addressing scheme of the kind shown, twenty address lines would be needed plus 2^{20} AND gates each with 20 inputs. In practice, decoding strategies which are more economical in terms of the number of gates needed are generally used, but they tend either to be slower than the type shown or to require more complex circuitry at memory locations. A further important point about all such systems is the need for almost flawless reliability in the base components. If the address decoder is to work correctly, addresses must appear correctly on the input lines, and the multi-input AND gates must also work precisely so as to select the item required. Similarly, the read-write lines must operate correctly. All of these requirements impose wiring and reliability constraints of a kind which it may not be plausible to suppose that neurons support.

It may be objected that content addressing rather than location addressing is clearly the method of choice for the neural system, particularly since human memory appears to work in such a way. It is far from clear, however, that content addressing eases the burden of control circuitry required. On the assumption that the memory consists of a series of independently modifiable locations, which seems to be required for a Turing machine, the control problem is in some ways worse for content addressing schemes than for location addressing and may explain why the great majority of computer architectures use location addressing. A variety of content addressing schemes have been proposed but they all have in common the

specification of a tag or key which may be part of the data item required or may constitute a separate but connected field. The requisite item is found by comparing the relevant parts of the contents of each location with the key or tag to find a match. Thus it may be necessary, in principle, to compare the contents of every location in the memory with the search key in order to find the appropriate data item. The control circuitry needed to manage the search process is at least as complex as that required for an address decoder, and read-write circuitry is needed just as in the location addressing scheme. Furthermore, content addressable memories tend to be slow because of the search process, may yield multiple matches to the search key which need to be dealt with and also pose problems when data have to be written to the store. While on the subject of content addressable memories, it is appropriate to note that Turing's universal machine uses content addressing to locate the appropriate instruction in the standard description of its target machine, using the configuration from the most recent complete configuration as the key.

It seems most likely that when people think about content addressing and its advantages they have in mind the sort of associative memory which is characteristic of connectionist networks in which the input constitutes the search key, the set of connection strengths of the network constitutes the stored data and the output constitutes the recalled item. However, this style of memory is not available to theorists who claim that the brain implements a Turing machine, first because the control is not separated from the memory and second because the memory does not consist of a set of independently modifiable locations whose contents are symbolic.

It seems clear, therefore, that the claim that the brain implements a whole Turing machine carries substantial implications for the amount of control circuitry needed and imposes rigorous requirements on the reliability of the basic components. It has been suggested that these requirements make the claim implausible.

> "The principles of computer memories can hardly be realized in biological organisms for the following reasons: i) All signals in computers are binary

> whereas the nerual signals are usually trains of pulses with variable frequency. ii) Ideal bistable circuits which could act as reliable binary memory elements have not been found in the nervous systems."
> Kohonen (1988,p.12)

One further point which is of some interest is the implication that if the brain is a Turing machine rather than a finite automaton its memory will be both less efficient and of lower capacity. The reason is that control and memory states are not distinguished in a finite automaton and thus the physical substrate can serve both purposes simultaneously as discussed in Chapters 2 and 3. Assuming the brain to implement the control circuitry for accessing a tape as well as whatever is required to implement locations, it is clear that a large amount of neural capacity will simply be unavailable for memory purposes in a way that is not the case for a finite automaton implementation.

A further point which is of great significance is that in addition to the burden of control circuitry which the internal Turing machine model entails, the generic theory has also to specify how external stimulus energy is translated into symbolic input which the internal computer can use and how the outputs of that computer can be translated into behaviour. The characteristics of these mechanisms and the conceptual problems to which they lead are explored further in the discussion in Chapter 5. For the present, it is clear that they also represent an additional call on neural resources which may be substantial.

In summary, in the light of Turing's analysis which specifically separated the finite state control unit modelling the brain, from the tape modelling the paper on which a human computer might write, and given the costs in terms of control circuitry which would be needed if the brain were indeed structured like a digital computer of some kind, the thesis that the brain does implement a Turing machine of a kind must be questionable. In the remainder of this chapter, a short review of some of the history of the development of computers and the means of using them is presented. It is apparent that many of the decisions were dictated by engineering considerations which while essential for the development of efficient machines

contribute to the implausibility of the view that the mind resembles or is architecturally similar to a digital computer. Computer design is essentially concerned with harnessing the power of the universal machine concept and, at least until recently, has been directed towards the MINH-MAXC end of the implementation dimension for a variety of sound engineering reasons. Both of these choices focus attention on the exploitation of a large auxiliary memory. However, in the absence of evidence either that the brain implements a universal machine interpreter or that evolution has adopted the MINH-MAXC strategy, and given the principled distinction which Turing made between human memory and auxiliary memory, there is no good reason to suppose that the computer provides a satisfactory model of the architecture of the mind. Indeed, the ETH suggests that the strategic choices made in the development of computer architecture are precisely the wrong ones from the point of view of developing a model of human cognitive architecture. This is not, of course, to claim that the architecture of the mind could not be simulated on a computer nor that the mind is not a computational system. If the mind is a mechanical system then the Church-Turing thesis strongly suggests the possibility of simulating its operations in the form of a program. But this is not at all the same as claiming that the architecture of the mind is itself organized in the way a digital computer is. Not all computational machines are universal. The existence of an infinity of Turing machines which compute but are not universal demonstrates this point.

4.2 The First Electronic Computers.

Although Turing's work was in print in 1937 and the first electronic computer was not commissioned until 1946, the theory of computable numbers appears not to have had a direct impact on the development of the very earliest machines which were constructed in the U.S.A. Throughout the 1930's the most sophisticated calculating machines available were analogue differential analyzers, most notably those built by Vannevar Bush and his colleagues at MIT, which approximated solutions to differential equations, cf. Goldstine (1972, Chapter 10) and electro-mechanical digital calculators based on Herman Hollerith's tabulating machines which were used for the construction of astronomical tables among other things.

4.2.1 The ENIAC[9].

In 1943 funds were made available to the Moore School of Electrical Engineering at the University of Pennsylvania to construct an electronic computer for the Ballistic Research Laboratory of the Ordnance Department of the United States Army. Herman Goldstine, who was the Ballistic Research Laboratory's representative at the Moore School during this period said

> "...one of the main functions of the Ballistic Research Laboratory was the production of firing and bombing tables and related gun control data. ...The automation of this process was to be the *raison d'être* for the first electronic digital computer."
>
> Goldstine (1972,p.135)

The machine was formally accepted by the U.S. Government in 1946 and operated successfully until it was retired to the Smithsonian in 1955. The ENIAC was, by modern standards, a physical giant but a computational midget. It was 100 feet long, 10 feet high, 3 feet deep and weighed 30 tons. In operation it consumed 140 kilowatts of power. The clock had a period of 10^{-5} seconds, and the machine performed some 330 multiplications per second. For all its impressive size, it had storage for only 20 ten digit decimal numbers. Nonetheless, it represented a huge step forward both in the sophistication and reliability of its engineering and in its speed of operation which was some 500 times faster than its closest electro-mechanical rival, the IBM Automatic Sequence Controlled Calculator (Goldstine 1972,p.117). The ENIAC owed its speed to the use of electronic rather than electro-mechanical components. Although this is now standard practice it was both controversial and risky at the time. Some engineers claimed that the necessary reliability could not be achieved and that the machine would never operate successfully. By proving the doubters wrong, the engineering team led by J.P. Eckert and J.W. Mauchly ushered in the modern era of computing. Interestingly, in view of future developments, the ENIAC was a parallel machine in which many operations proceeded simultaneously. The machine had thirty basic units. Twenty

---

[9]Electronic Numerical Integrator and Computer

accumulators for the twenty numbers mentioned above, one multiplier, one combined divider and square-rooter, three function table units which provided additional storage for fixed tabular data, an input unit and an output unit, two control units and a master programmer which provided overall direction for the parallel operations of the machine.

The major problem with the ENIAC was programming. "This was a highly complex undertaking and was one of the reasons why it was to be a unique machine. This aspect of the machine was unsatisfactory, as the evolutionary process was to reveal." Goldstine(1972,p.160). The difficulty lay in the fact that the ENIAC had to be reconfigured physically, by means of numerous switches, every time a new calculation was to be done. Not only was the programming procedure liable to error; it was also very time consuming and meant that the machine was at a standstill for much of the time. While the strategy of programming by reconfiguration now seems hopelessly inefficient, Hockney and Jesshope (1988, pp.9-10) make the following observations;

> "...the architecture of the ENIAC was rearranged for each problem, by using the plugboard to rewire the connections between the units. One could say that the algorithm was literally wired into the computer. It is interesting that such ideas are beginning to sound very 'modern' again in the 1980's in the context of MIMD[10] computing, reconfigurable VLSI[11] arrays, and special-purpose computers executing very rapidly a limited set of built-in algorithms. However, the time was not ripe for this type of parallel architecture in the 1940's ... The difficulty of programming parallel computers is a recurring theme that is still with us today."

Some of the reasons for the difficulty of programming parallel computers are discussed further below. From the point of view of the ETH, what is particularly interesting is the fact that parallel architectures, by their nature, appear to be best suited to special purpose tasks. Since the brain is a parallel machine, this provides

---

[10]Multiple Instruction Stream/Multiple Data Stream.

[11]Very Large Scale Integration.

a further argument that it is likely to have evolved to compute special purpose tasks rather than as a general purpose universal processor. The claim that the brain does not include an auxiliary, tape-like memory is consistent with this suggestion.

4.2.2 John von Neumann and the Stored Program Concept.

Architecturally, the ENIAC was originally conceived as an electronic version of a differential analyzer (Hockney & Jesshope 1988, p.9), and, as mentioned above, it had to be re-arranged for each problem. In 1947, von Neumann, who had learned about the ENIAC after a chance meeting with Goldstine, showed "how to convert it into a centrally programmed computer in which all the programming could be done by setting switches on the function tables" Burks (1966, pp.7-8). This meant that the machine was, in effect, turned into a primitive stored program computer. Another way of conceptualizing what was done is to think of the ENIAC in its original form as a protean mechanism which could be configured as an arbitrary Turing machine to compute a specific function. von Neumann showed how to configure the ENIAC as a universal Turing machine. It was a significant development. As Goldstine (1972, p.233) says "Although it slowed down the machine's operation, it speeded up the programmer's task enormously. Indeed, the change was so profound that the old method was never used again." Two points should be marked. First the fact that configuring the ENIAC as a universal machine slowed down its operation. This would be expected given the costs of interpretive execution. Second, the significant point is that the loss sustained in this way was more than offset by the gains in efficiency with regard to the task of programming, because the basic speed of the machine was so high. Because of this very high operating speed, the scope of the ENIAC was potentially much greater than its designers had originally supposed, but this speed exposed two major flaws. The programming problem has already been noted. The other flaw was its very small storage capacity, which made it unsuitable for problems which generated large amounts of intermediate data.

von Neumann, as a consultant to the EDVAC[12] project, and members of the ENIAC, EDVAC and IAS[13] computer project teams, produced various reports from 1945 onwards, dealing with the logical design of high speed computers, in which proposals were put forward to overcome the difficulties experienced with the ENIAC. The ideas reported in these papers "were widely circulated and served many people as textbooks on logical design and programming" Randell (1973, p.352). The first, and perhaps the most influential of them was the "First Draft of a Report on the EDVAC" von Neumann (1945), which outlined what has since become known as the "von Neumann architecture."[14]

4.2.3 The Design of the EDVAC.

The importance of the early documents for present purposes is the light they shed on why modern computers, which continue to share many fundamental features with their ancestors, are built as they are. When analogies are made between the operations of the computer and those of the mind, it would be unfortunate if technological fixes, no matter how ingenious, were mistaken for theoretical necessities. The EDVAC, for example, was designed to use stored programs not out of logical necessity, but for efficiency reasons.

> '...it would make little sense for a computer to produce the results of a calculation rapidly, only to wait idly for the next instruction. The solution was to store the instructions internally with the data: what has come to be called the "stored program concept".'
>
> Davis (1988a,p.166).

The draft report on the EDVAC argued for a number of features in a high speed computer. First, because the machine would be used primarily for calculations it

---

[12]Electronic Discrete Variable Computer.

[13]Institute for Advanced Study.

[14]The "First Draft" engendered a controversy about who was responsible for the stored program concept. von Neumann's name was the only one on the report which was prepared by him, and was not initially intended for circulation beyond the group whose ideas it collated. Other members of the group, in particular Eckert and Mauchly, felt that their contributions were not recognized when the report was made public.

should have specialized arithmetic organs. Precisely which operations should be built in was, and still is, a matter for discussion[15]. The arithmetic processors or central arithmetic unit (CA) would constitute one major part of the computer. The second part would be a device which exercised central control (CC) over the sequencing of operations. The point to notice here is the radical change from the design of the ENIAC. In the new design, it was anticipated that the instructions to the machine would be stored in the same way as its data, i.e. as numbers in the memory, rather than being wired in as they were in the ENIAC. Thus the EDVAC and its successors were designed to function as universal Turing machines. It was anticipated that this would lead to much faster programming. The third requirement was for a large memory (M). This was needed to enable the machine to hold the program instructions and to deal with complex numerical problems (various types of partial differential equations being specifically mentioned) which might need large amounts of storage for intermediate results.

In addition to C (CA & CC) and M, a computer would need an input device I and an output device O which would manage the transfer of information between C and M and an external recording medium R, which would be punched cards, magnetic tape or wire, or some other suitable technology. It was envisaged that communications would, as a rule, be between R and M and not between R and C. Further discussion of the input and output devices can be found in Burks, Goldstine & von Neumann (1947) where it is clear that the major preoccupation was to find a suitable way of using the computer to control its own external storage so as to minimize human intervention. As Rosen (1969,p.14) says "The early scientific computers were designed in accordance with a philosophy that assumed that scientific computing was characterized by little or no input or output." von Neumann argued that once a machine had been given the instructions for a particular task, "it must be able to carry them out completely and without any need for further intelligent human intervention." While this is exactly the sort of arrangement which

---

[15]Current arguments about two strategies are continuing the debate started by von Neumann. Proponents of RISC (reduced instruction set) computers favour unsophisticated but extremely fast primitive instructions from which more complex instructions are composed, while proponents of CISC (complex instruction set) computers argue for more sophisticated primitive instructions which are slower than the RISC primitives but provide complex capabilities in fewer steps.

is suitable for high speed calculations, it is much less satisfactory as the basis for a model of the cognitive system which is characterized by the constancy of its interactions with the external environment and by the sensitivity and range of its input and output systems.

4.2.4. Computer design and ideas about the nervous system.

The idea that computers were built specifically to facilitate calculation and that their design was shaped largely by engineering considerations is in conflict with a line of thought which claims that the first computers were explicitly modelled on what was known about the nervous system at the time and hence provide a basis for models of the cognitive system. Norman (1986, p.534) for example suggests that "the architecture of the modern digital computer - the so-called Von Neumann architecture - was heavily influenced by people's (naive) view of how the mind operated." A similar point is made by Boden (1988, p.2); "In designing the digital computer, von Neumann was influenced not only by Turing's earlier work on the theory of computation, but also by some novel ideas about the logical functions of the brain", which, she goes on to say, were due to the work of McCulloch and Pitts. It is easy to overplay the extent to which these were significant influences. Norman cites Wiener (1961) and the series of conferences on cybernetics sponsored by the Macy Foundation, e.g. von Foerster (1952), as evidence for his suggestion. What these documents really show is not so much an influence of brain function on computer design, but a belief that there are interesting and important parallels between computers and brains. This belief is grounded in the fundamental assumption that the primary mode of operation of neurons is digital. Wiener, for example, argued that the computer, "must represent almost an ideal model of the problems arising in the nervous system", because the "all-or-none character of the discharge of the neurons is precisely analogous to the single choice made in determining a digit on the binary scale". Hence, he claimed that "The synapse is nothing but a mechanism for determining whether a certain combination of outputs from other selected elements will or will not act as an adequate stimulus for the discharge of the next element, and must have its precise analogue in the computing machine." Wiener (1961, p.14). This is a claim which is not as well supported by

current evidence as it would have appeared to be in the 1940's. Current evidence suggests a much more complex picture in which the effects of particular synapses depend on their type, their position on the target neuron and the states of the numerous ionic currents which modulate the electrical activity of the neuron, cf. McCormick (1990).

The early documents discussed above which deal with practical computer design issues show only a peripheral link with the then current understanding of brain function. The paper by Burks, Goldstine & von Neumann (1947) which deals with the design of the computer built at the Institute for Advanced Study in Princeton, makes no mention at all of any parallels with the nervous system, nor of any influence of concepts from neuroscience. It is almost exclusively concerned with detailed discussion of the technical problems associated with the arithmetic and logical control organs of the machine. In the earlier EDVAC discussion document there are two types of suggestion of linkage between computer and nervous system concepts. First there is the suggestion of a correspondence of parts C and M of the computer with associative neurons in the brain and second there is the fact that the construction was described using models of switching elements based on the idealized neurons of McCulloch & Pitts (1943).

With regard to the first suggestion, two points should be made. First, it is an isolated claim which is not otherwise referred to and which has no impact on the subsequent discussion. Second, it relies on a distinction between sensory and motor areas of cortex on the one hand, and associative areas on the other in which the sensory and motor areas are assumed to be dedicated to particular functions, while the associative areas are assumed to be undedicated and the hall mark of higher mental capacities. Nadel, Willner, & Kurz (1986,p.223) call this view into question; "the implicit brain theory of the 1940's and 1950's which legitimated the abiological stance of cognitive science appears to be wrong in all important respects. The brain is not wired up in a way which could support the general-purpose functions of classical Associationism." Zeki (1993, p.4) has suggested, even more trenchantly, that the doctrine of associationism "retarded our present notion of the

organization of the visual cortex and of brain function by well over a century." Perhaps it is just as well that the influence of brain theory on computer design was not too extensive.

The second apparent type of linkage between computer concepts and the nervous system by way of McCulloch-Pitts neurons, in von Neumann's hands at least, was primarily a methodological device aimed at clarifying the relationships between the elementary units out of which a computer might be built and the arithmetic and logical functions it was required to execute. von Neumann suggested that "The ideal procedure would be to treat the elements as what they are intended to be: as vacuum tubes. However, this would necessitate a detailed analysis of specific radio engineering questions at this early stage of the discussion, when too many alternatives are still open...All this would produce an involved and opaque situation...In order to avoid this we will base our considerations on a hypothetical element...This simplification is only temporary, only a transient standpoint, to make the present preliminary discussion possible." von Neumann (1945, pp.29-30). The use of a hypothetical element allowed von Neumann to discuss circuits, for example for particular arithmetical operations, independently of the details of their implementation. It appears to have been this, rather than any deep notion of correspondence between brains and computers, which was responsible for the adoption of the McCulloch-Pitts formalism. Some may find a pleasing irony in the fact that a neural model was used to distance the logic of a design from its implementation. As mentioned above, by 1947 when some of the general principles were much clearer, analogies with the nervous system had entirely disappeared from the preliminary design document for the IAS computer.

A further point is that von Neumann was also very well aware of the important disanalogies between the functioning of the nervous system and the functioning of computers. The EDVAC document, for example, contains as many examples of disanalogies between brains and digital computers as it does analogies. What the foregoing discussion shows, therefore, is that although von Neumann, Wiener, and other computer pioneers were undoubtedly interested in parallels between the human

nervous system and computer circuits, the suggestion that computer design was heavily influenced by nervous system ideas overstates the case.

4.2.5. The case for serial processing.

The five parts of a computer described in the EDVAC draft design document, CA, CC, M, I and O, plus R which is properly considered as an extension of M, constitute the structural heart of the von Neumann architecture. The most important remaining feature in the "First Draft" is the argument for serial as opposed to parallel processing. As has already been noted, the ENIAC was designed as a parallel machine, but was eventually configured to operate as a serial machine in order to ease the complex task of programming it, a task which as Hockney and Jesshope pointed out remains difficult in parallel machines today. The point about parallelism is that it provides an excellent way of increasing the throughput of a machine with relatively slow components. What it means, of course, is that the machine requires more of those components.

> "This way of gaining time by increasing equipment is fully justified in non vacuum tube element devices, where gaining time is of the essence, and extensive engineering experience is available regarding the handling of involved devices containing many elements...For a vacuum tube element device on the other hand, it would seem that the opposite procedure holds more promise."
>
> von Neumann (1945,p.363).

The argument is founded on the state of the technology and engineering experience in 1945. The vacuum tube was a bulky, expensive and error-prone piece of equipment. Thus it was wise to use as few as possible. On the other hand it was extremely fast by the standards of the day. von Neumann reported that the fastest relays available had reaction times of 5 milliseconds at best, and more commonly 10 milliseconds or more, whereas the reaction times of vacuum tubes could feasibly be made as short as 1 microsecond. Thus a single vacuum tube device operating serially might in principle give about the same performance as some 5000 relay devices operating in parallel. These figures led to the eminently sensible conclusion,

> "The device should be as simple as possible, that is, contain as few elements as possible. This can be achieved by never performing two operations simultaneously, if this would cause a significant increase in the number of elements required."
>
> von Neumann (1945,p.364).

4.3. Examples of von Neumann architectures.

In the relatively few pages of the "First Draft", von Neumann, with his incisive insights and customary clarity of expression, set down principles which heavily influenced the logical design of computers throughout the 1950's and 1960's. The thesis that substantial performance gains could be had by improving the speed of components rather than multiplying their numbers went against the conventional wisdom of the time but has been fully justified. One of the most telling indications of the continuing success of the von Neumann architecture is the striking similarities which are found when ancient and modern instruction sets are compared. Three examples are given and the differences can be seen to be quantitative rather than qualitative.

4.3.1. The EDSAC order code.

One of the first practical stored program machines was the EDSAC (Electronic Delay Storage Automatic Calculator) which was built at Cambridge and became operational in 1949 (cf. Wilkes & Renwick 1950; Renwick 1950). The EDSAC had storage for 1024 binary numbers of 17 digits each and could also handle two adjacent storage locations so as to yield numbers of 35 digits. Thus its data types were long and short numbers. The machine operated at a frequency of 0.5 megahertz. The instruction set or order code, as it was known, consisted of eighteen instructions. Eight of the instructions were for arithmetic operations, one carried out logical AND, three were for data transfers, two were conditional branching instructions, two were concerned with input and output, and the other two were for checking and halting the machine.

4.3.2. The 6502 instruction set.

The 6502 microprocessor (cf Leventhal 1979) is representative of the generation of

eight bit microprocessors which launched the current computer revolution and took computers into homes, offices and schools in the late 1970's and early 1980's. The 6502 has fifty six instructions and can address a memory of 2^{16} or 64k bytes. Direct comparisons with the EDSAC order code are somewhat artificial because the 6502 would always be used in conjunction with other devices which would manage input and output for example, whereas the EDSAC order code was complete. Nevertheless, the similarities are considerable. The 6502 runs at up to 2 megahertz. Its basic data types are eight and sixteen bit unsigned numbers. There are nineteen arithmetic instructions, five logic instructions, sixteen data transfer instructions and fifteen conditional branch and jump instructions. The remaining instruction is a null operation which has a variety of uses for timing, debugging and so forth. The most notable differences between the two instruction sets are that the 6502 has no multiplication instructions, that it has stack and interrupt servicing instructions, and that it has more sophisticated memory addressing modes. Use of a stack facilitates subroutine management and recursive computation, interrupt servicing instructions enable peripheral equipment to be managed in a way that had not been invented when the EDSAC was built, and complex memory addressing modes facilitate the management of programmed data structures such as arrays.

4.3.3. The 80386 instruction set.

The Intel 80386 microprocessor (cf. Crawford & Gelsinger 1987) is a modern 32 bit processor which represents the state of the art in microprocessor design in the mid 1980's. It can access up to four gigabytes, i.e. 2^{32} bytes, of physical memory, and up to 64 terabytes , i.e. 2^{46} bytes, of virtual memory. It is capable of running at clock speeds of 16-20 megahertz which makes it as much as forty times faster than the EDSAC. When considered in conjunction with the 80387 floating point processor with which it was designed to operate, the processor supports unsigned, two's complement, binary coded decimal and floating point numbers of up to 80 bits, plus character and bit strings. There are 226 instructions in the instruction set, supporting a range of activities which go well beyond the capabilities of both the 6502 and the EDSAC to include capabilities such as multi-tasking and virtual memory access. There are twenty integer arithmetic instructions, six bit

manipulating instructions, sixteen conditional byte setting instructions, twenty three conditional branching and jump instructions, eleven data transfer instructions, and over seventy floating point instructions. Among this prodigal array of capabilities, one particularly striking feature of the 80386 is that only two instructions are concerned with input and output other than to and from the main memory. Two of the EDSAC's eighteen instructions were also dedicated to these functions. The situation is not quite as solipsistic as this bald statement might imply, since the 80386 has a separate memory space of 64 kilobytes devoted to the control of peripheral devices. However, it is interesting to note the views of the chip's designers on the question of input and output to external devices.

> "...the addressing and protection of I/O devices is quite different from the addressing and protection of program code and data. Typically, an I/O device has only a few control ports, requiring only a small number of bytes of addressable storage, and there are only a small number of devices in the system... On the other hand, program code and data require many thousands, even millions of bytes of addressable storage, and need a different protection mechanism."
>
> Crawford & Gelsinger (1987, p.76)

4.3.4. Summary.

The view expressed by Crawford and Gelsinger is of a piece with the whole design philosophy of serial machines from von Neumann to the present day. The emphasis is on what happens internally, not on what is going on outside. While it is clear that great progress has been made in the design and manufacture of electronic devices for computing, even advanced processors like the 80386 are still recognisably related to their ancestors like the EDSAC. This is to be expected. Advances in computing technology have been cumulative and extrapolations to future developments are frequently attempted (cf. Siewiorek, Bell & Newell, 1982, p.7; Hockney & Jesshope, 1988, p.3; Hack, 1989, p.262). From the point of view of understanding cognitive architecture, however, there is little evidence that the considerations which shaped the design of the von Neumann architecture are relevant to determining the nature of human cognitive architecture.

4.4. The return to parallel architectures.

In the 1980's as hardware costs dropped, multiprocessor parallel systems became economically feasible. Even so, much hardware development effort was still devoted to increasing clock speeds, decreasing logic gate delays and otherwise improving on the details of serial computation, rather than moving towards multiprocessor systems. This process of successive refinements of serial technology has some clear physical limits.

4.4.1. The limits of serial technology

The CRAY-2, a pipelined vector computer, which, although parallel in many aspects is recognizably a successor of the EDVAC and EDSAC machines, demonstrates the way in which the limits of performance improvement are being reached. Perhaps the most striking observation about the CRAY-2, is that its performance depends on keeping the wires connecting its circuit boards short (Hockney and Jesshope 1988, p.121-122). The CRAY-2 has a maximum wire length of 16 inches, whereas the CRAY-1 had a maximum wire length of 48 inches. The difference in signal propagation time between wires of these two lengths is approximately 3 nano-seconds, and the components of the CRAY-2 are so fast that savings of nano-seconds are significant. Signal propagation delays impose a lower bound on improvements in the clock rate for a computer and thus negate the benefits of faster logic gate switching times. The solution to this problem is to pack the components on a circuit board closer together so as to reduce wire length, but this means that heat dissipation then becomes a significant problem. As a colourful illustration of the problems, Hockney and Jesshope (1988, p.564) compare the task of cooling the CRAY-1, which has lower chip densities than the CRAY-2, with "putting a 1 kW electric element into a biscuit tin and trying to keep it just above room temperature". It is apparent from considerations of this kind that physical limits are now being reached which will prevent further cost effective refinements of the serial model (Hack 1989), and that the development of large scale multi-processing will be essential for continued increases in computer speed. Experimentation with parallel systems has been ongoing since the very early days, and, as remarked earlier, the ENIAC which proved the viability of the whole electronic computing

enterprise was originally operated as a parallel machine. Until comparatively recently, however, serious interest in multi-processor computers has been the province of research groups. This is partly a result of the very high costs involved prior to the development of VLSI technology, but is also a result of the unwillingness both of suppliers and users to retool and learn new programming methods. Despite such entrenched attitudes much has been learned about parallel architectures which is of relevance to cognitive science.

4.4.2. Parallel tasks and parallel architectures.

One important general finding has been that once the von Neumann model is set aside, questions about architectures become intimately linked to questions about tasks. The nature of a problem determines the kind of architecture needed for the most efficient solution, and, conversely, a particular type of architecture will be better suited to certain kinds of problems than others. This is not a question of absolute capability, but of efficiency, an issue which has been discussed by cognitive scientists (cf. Thagard 1986; Krellenstein 1987; Ramsey 1989). Hockney & Jesshope (1988, p.432) point out that to obtain optimum performance from any computer it is necessary to tailor programs to the architecture of the machine. The difference between parallel and serial computers lies in the ratio between the performance of good and bad programs, which rises from a factor of two or three with serial machines to ten or more with parallel machines. In an analysis of the prospects for general purpose parallel computing, Hack (1989) concurred with this view and went on to suggest that

> "...the efficient utilization of a parallel architecture requires that the user becomes directly involved in the detailed control flow of their application program... The most serious challenge to the user is that the most appropriate options are not always obvious, and are very much a function of the specific hardware configuration."
>
> Hack (1989, p.273)

Hack also argued that experimentation with different implementation alternatives of an algorithm for a particular parallel hardware configuration was unavoidable if an efficient match between the requirements of the algorithm and the capabilities of the

hardware were to be achieved. The fundamental reason for the diversity of possibilities is the range of structural options open to the designer of a parallel system. While the von Neumann design was the architecture of choice, the serial processing mode meant that the relation between the central processor and memory was not fundamentally problematic. Since there was only one processor, a high speed bus carrying register sized chunks of information to and from a unitary memory store as fast as the processor could deal with them was the obvious way to do things. With multiple processors, the issue of structural organization becomes much more intricate. How many processors should there be? Should each processor have its own memory and communicate by passing messages, or should processors share a large memory and communicate by having data in common? Is there a best design for general purpose parallel computers? This is a much more difficult question than might at first appear. The difficulty lies in the fact that different problems have quite different orders of parallel decomposability. Suppose a task has been identified as requiring twenty primitive instruction executions for its solution. If each instruction were independent of all the others then a machine with twenty processors could be utilized to complete the task in a single twenty fold parallel step. At the other extreme, if a chain of dependencies existed such that the twentieth instruction depended on the result of the nineteenth, which itself depended on the result of the eighteenth, and so forth, down to the second which depended on the result of the first, then no parallel machine could compute the solution faster than a serial machine. Thus the design of a general purpose parallel machine is a difficult problem. One answer is to build machines with configurable processor arrays which are dynamically allocated to the elements of a task at run time. One such machine is the Connection Machine (Hillis, 1985). The problem with machines of this kind is that raw performance has to be sacrificed in order to manage the synchronization of and communications among the various processors, thus shifting the problem to another design dimension (Greenbaum 1989). The evidence seems to be that really high performance from a parallel system is only obtained when the machine is built to match, or happens to match, a particular task.

4.4.3. Control regimes in parallel systems

Apart from questions of structure and the relation of processors to memory, the issue of control organization is one which comes to the fore in parallel systems as a result of removing the constraints of the von Neumann architecture. This is to be expected in the light of the distinction between the MAXH-MINC and MINH-MAXC strategies introduced in Chapter 3. As with questions of structure, it is the seriality of the von Neumann design which provides the primary constraint. If access to the memory is made for just one item at a time, then it is almost inevitable that control should consist of taking an instruction followed by its operands. It is for this reason that the von Neumann model is sometimes also known as the control flow model (Sharp, 1985, p.18). When instructions, data and multiple processors are all available simultaneously, however, other control regimes become possible. Perhaps the most interesting of these from the psychological perspective is the data flow model (cf. Hockney & Jesshope, 1988, 248-250; Fox & Messina 1987; Gelernter 1987; Sharp, 1985). In a dataflow computation, instructions do not execute in a predetermined sequence under the control of a program counter. Instead, an instruction waits until it has received all its operands, then executes its operation and passes the results to those instructions which are waiting for them as operands. A program for a dataflow computation is represented as a directed graph. The nodes of the graph represent mathematical or logical operations, and the edges represent the flow of data from one operation to another. Sequencing arises from the decomposition of a problem into digraph form rather than from an algorithm which explicitly defines the execution order of parts of the problem statement. Evaluation of a data flow computation may be either demand driven or data driven. In the demand driven mode, execution is driven by requests for data from operations which need operands. In the data driven mode execution is driven by the availability of inputs. The demand driven mode can be thought of as pulling data through the system towards the output, whereas the data driven mode can be thought of as pushing data through the system from the input. Sharp (1985, p.126) suggested that a data flow machine might be built by implementing data flow graphs directly in hardware.

Chapter 5. Computers, Models and Cognitive Theories.

The material of Chapters 3 and 4 demonstrates that the generic theory is only one way in which computational ideas can be deployed in support of a theory of cognitive architecture, and gives rise to the claim that it is not the best way. Although it was the dominant theory until the revival of interest in connectionism, the analysis of Chapter 3 shows that the generic theory does not follow from Turing's analysis and, in fact, contradicts Turing's arguments which claim that the brain should be considered as a finite state machine. Furthermore, Chapter 4 shows that the requirements which the generic theory imposes on the medium of implementation cast doubt on the plausibility of the approach when the brain is the target medium. It might therefore be expected that accounts of the foundations of the generic theory would provide justification for its basic tenets. In fact however, as this chapter demonstrates, this is not the way in which such accounts proceed. Generally, the claim that the brain is organized as a Turing machine is taken as a basic assumption, and the consequences of that assumption are then spelt out. Much of the literature is highly developed and the points made are skilfully argued, but, from the point of view of the ETH much of the work is ultimately wasted effort because the basic assumption is never argued for.

The reasons for this state of affairs are obviously complex, but in view of the importance of the claim that the basic assumption of the generic theory is unwarranted, it seems appropriate to trace some of the threads. First, it appears that many theorists have drawn their inspiration primarily from digital computers and only secondarily from Turing machines. This is not surprising, in view of the treatment of Turing machines in influential early texts. Miller, Galanter, and Pribram for example, in their very influential book "Plans and the Structure of Behavior" made reference to Turing's work, but not directly to the paper on computable numbers, and suggested that "One consequence of taking Turing's theorem seriously is that it directs attention toward the electronic computer as the right kind of machine to simulate human behavior." Miller, Galanter & Pribram (1960, pp.46-47). Undoubtedly the computer is the right kind of machine to

simulate behaviour, but it is all to easy to move from that view to the very different view that computer architecture provides the right kind of model for the architecture of the mind. Second, since it is clear that digital computers are, as Turing himself argued, essentially practical versions of universal Turing machines, it must seem, if one is unfamiliar with the detail of Turing's analysis as distinct from the formal definitions of the Turing machine developed in the subsequent literature, that the Turing machine concept and the digital computer concept are essentially equivalent in the motivation they provide for a theory of cognitive architecture. Since the memory of a digital computer appears to provide a natural model for human memory, and since the distinction between hardware and software seems to map so naturally on to the distinction between mind and brain, it is understandably easy to slide from a computer based theory of cognitive architecture to the assumption that such a theory enjoys the theoretical support of Turing's analysis of computation. Since, further, it is understood that Turing's model is in a sense as general a model as it is possible to obtain, it would appear that no independent argument for the basic assumption that the brain implements a Turing machine, is needed.

The conclusion to be drawn from the above is that study of Turing's original paper is needed in order to appreciate the nature of his argument. However, the paper is not widely available. Apart from the journal in which it was originally published, which appears not to have a very wide circulation, the paper has been reprinted only once in the collection edited by Davis (1965). Part of the reason for this may lie in the involved construction which Turing used which contains a number of technical slips. Because the result is of such importance, the Turing machine concept has been presented in numerous other, formally equivalent but technically much more accessible, forms thus rendering the original form of presentation obsolete. [Wang 1957 perhaps] From the point of view of mathematicians and computer scientists whose main interest is in developing and extending the theory of which Turing's paper is one of the progenitors the result rather than the arguments for its presentation is primary, and if such centrally involved researchers take this position there is little reason why psychologists should be inspired to read the paper.

The problem which this situation leaves for the proponent of an alternative account like the ETH is one of coming to grips with the foundational literature, since what has to be questioned is not specific arguments but the whole structure of thinking on which the generic theory is based. The purpose of this chapter, therefore, is to look at influential discussions of some of the foundational issues with a view to clarifying certain areas of debate.

5.1. Putnam on the Mind-Body Problem.

It is appropriate to begin with Putnam's (1960) treatment of the mind-body problem in terms of Turing machines which both Wilks (1975) and Jackendoff (1987) acknowledge as probably the earliest reference to make the link between Turing machines and psychological theory explicit. The first point about this paper which is important for present purposes is Putnam's claim that "Any Turing machine is completely described by a *machine table*,..." Putnam (1960, p.365). What Putnam should have said, strictly speaking, is that any Turing machine *control* is completely described by its machine table. That the description of a complete Turing machine requires reference to the sequence of symbols on the tape as well as to the structures of the control is very clear from Turing's analysis. In section 9 part I of his paper, Turing clearly included the tape as an essential part of the description of a machine. "We know the state of the system if we know the sequence of symbols on the tape, which of these are observed by the computer (possibly with a special order) , and the state of mind of the computer." Turing (1936-7, p.136). In section 9, part III of his paper, he also discussed a model of computation in which the notion of a state of mind was replaced by a note of instructions which enabled the computer to carry out one step of the computation and write the next note. Of such a model he said that "... the state of progress of the computation at any stage is completely determined by the note of instructions and the symbols on the tape." Turing (1936-7, pp.139-140). The sequence of symbols on the tape formed part of what Turing called the "state formula" which constituted the complete description of a machine at a given time. Thus it is misleading to assert, as Putnam did, that the machine table alone provides a complete description of a Turing machine. It is misleading because it diverts attention from the question of the relation between tape and

control and how this should be understood.

In a later paper, Putnam (1967), Putnam discussed what he called the "natural generalization" of a Turing machine to allow for interaction with an environment. This generalization, quoted in detail in Chapter 1, suggests that we should think of the brain as implementing a whole Turing machine, equipped with sensors to monitor the environment and to write appropriate "reports" on the tape from time to time. Quite apart from Putnam's own influence, this view has been propounded in the influential work of Jerry Fodor, who was one of Putnam's students. Neither Putnam nor Fodor argues for this generalization.

It is interesting to note that Putnam explicitly makes the point that the appropriate model for a real computing agent is the finite automaton. "...I am going to consider a hypothetical 'community' made up of 'agents', each of whom is in fact a Turing Machine, or, more precisely, a finite automaton." However, he immediately goes on to say, "(Of the many useful equivalent definitions of 'finite automaton', the most useful for present purposes is the one that results if the definition of a Turing Machine is modified by specifying that the tape should be *finite*.)" Putnam (1967, p.409). Again, he offers no argument for the claim that the definition of a finite automaton as a Turing machine with a finite tape is the most useful one for the discussion of theoretical issues in psychology.

A clue to the thinking underlying Putnam's choice of model comes somewhat later in the paper. He says "Usually we think of a Turing Machine as having a memory in the form of a paper tape upon which it prints symbols; however, this can be regarded as mere metaphor. Instead, in the case of a finite automaton, i.e. a Turing Machine whose tape is finite instead of potentially infinite, the tape may be thought of as physically realized in the form of any finite system of memory storage. What we mean by a 'symbol' is simply any sort of *trace* which can be placed in this memory storage and later 'scanned' by some mechanism or other." Putnam (1967, p.412).

It is a sad irony that Putnam prefaced the comments above with the suggestion that they "may perhaps prevent some misunderstandings". Instead, they are potential sources of confusion. First, in Turing's analysis, the paper tape form of memory is "no mere metaphor". Paper is very much part of the ordinary process of computation as Turing visualized it. He argued that the two-dimensionality of a sheet of paper was inessential but never suggested that the paper could be dispensed with altogether. "In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, *i.e.* on a tape divided into squares." Turing (1936-7, p.135)

To argue as Putnam does encourages four types of confusion. The first is to make it easy to confuse tape memory with control memory, the second is to ignore the distinction between finite means of production and potentially infinite product which is part of the essence of the Turing machine concept, the third is to blur the distinction between the "inner" finite part of the machine, and the "outer" storage medium, and the fourth is to blur the distinction between a symbol and an internal state of the system. Turing did not regard a 'symbol' as "any sort of *trace*". In a footnote he went to some lengths to define a symbol as a set of points of a unit square occupied by printers ink.

Putnam's position is understandable when one is thinking of the Turing machine as an abstract model of a digital computer, but to argue as he does is to ignore the motivation for Turing's construction, and, in particular, the distinction between the internal, finite set of "states of mind" which constitute the control of the Turing machine, and the external, potentially infinite tape on which the results of the calculation are written. It is worth recalling Turing's point that the definition of computable numbers as those which are calculable by finite means rests on "the fact that the human memory is necessarily limited" while the numbers include many whose expressions are unbounded. "I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the

real parts of the zeros of the Bessel functions, the numbers π, e, etc." Turing (1936-7, p.116).

5.2. Functionalism.

Apart from his discussions of the nature of the Turing machine concept, the early Putnam is notable for his functionalist stance. "The functional organization (problem solving, thinking) of the human being or machine can be described in terms of the sequences of mental or logical states respectively (and the accompanying verbalizations), without reference to the nature of the 'physical realization' of these states." Putnam (1960, p.373). Putnam initially defended this functionalist thesis with an argument to the effect that human mental states were Turing machines states. In Putnam (1973) he rejected Turing machine functionalism specifically and in Putnam (1988) he argued that computational models of the mind/brain were in general insufficient for cognitive psychology, because "We cannot individuate concepts and beliefs without reference to the *environment*." Putnam (1988, p.73). This is a view with which the proponent of the ETH can agree while maintaining that the Turing machine is an appropriate model for cognitive architecture. The point is developed in Chapter 6. For the present discussion, what is most relevant is the way in which the central distinction between "mental" or "logical" states on the one hand and their "physical realization" on the other has been deployed. There is a noticeable tendency in the literature to run together arguments that should be kept separate.

Johnson-Laird (1983) provides an example. He starts his discussion of functionalism by making reference to Craik's notion of a "relation-structure". Craik developed this idea in the context of working out what it was about a model which illuminated its explananda. His answer was that a model had a similar relation-structure to that which it explained, meaning by this that it worked in the same way as the process it paralleled but was in some useful way simpler, quicker or more convenient. Craik proposed the hypothesis that thought models reality symbolically, i.e. he suggested that we think by constructing internal symbolic models which have a similar relation-structure to that part of external reality which they model. One

way, as pointed out by numerous theorists, in which such a capacity might be advantageous is in allowing its possessor to predict what might happen by thinking through a model rather than working blindly by trial and error. Craik's notion of similarity of relation-structure appears to have much in common with Putnam's notion of functional isomorphism. For Putnam, two systems are functionally isomorphic if "*there is a correspondence between the states of one and the states of the other that preserves functional relations.*" Putnam (1973, p.291).

Johnson-Laird says of the relation-structure notion that "the importance of this idea has become clear since the development of programmable computers. Once you know the way in which a computer program works, your understanding of it is in no way improved by learning about the particular machine on which it runs on this occasion or that. The same program may be translated into completely different codes for controlling different makes of computer that operate in different ways, and yet it is the same program that computes the same function however it is physically realized -- whether the machine uses cogs, hydraulic valves, vacuum tubes, or silicon chips." Johnson-Laird (1983, p.9).

While one might take issue with some of the details of the argument, in the main it seems unexceptionable. However, Johnson-Laird goes on to claim "There is a major lesson for cognitive science here...the mind can be studied independently from the brain. Psychology (the study of the programs) can be pursued independently from neurophysiology (the study of the machine and the machine code). The neurophysiological substrate must provide a physical basis for the processes of the mind, but granted that the substrate offers the computational power of recursive functions, its physical nature places no constraints on the patterns of thought. This doctrine of *functionalism*...has become commonplace in cognitive science." Johnson-Laird (1983, p.9).

There are two principal flaws in Johnson-Laird's argument. First, the fact that a computer program stands in a certain relation to the hardware on which it is executed contains a lesson for cognitive science only if it is known that the relation

between minds and brains is of the same sort. But this is not known, it is hypothesized. At best therefore, Johnson-Laird is entitled to the claim that IF the mind stands in the same relation to the brain as a computer program stands to its underlying hardware then the mind can be studied independently of the brain. The "IF" is of primary importance and Johnson-Laird's failure to acknowledge the need for it is a clear example of the tendency, mentioned at the start of the chapter, for theorists to assume what has to be demonstrated. In addition to this, Johnson-Laird also appears to be suggesting that functionalism, as a theory of the relation between mind and brain, implies that the brain is structured as a universal interpreter. This seems to be the intended force of the claim that the physical nature of the substrate in which mind is realized places no constraints on the patterns of thought provided that it "offers the computational power of recursive functions".

Philosophically, functionalism is a thesis about the individuation conditions on mental states which claims that what individuates a mental state is not the substrate in which it is physically realized but the functional role it plays in a sequence of mental activity. Thus a state is individuated by considering its connections to other states and to its inputs and outputs. Functionalism, if true, is likely to apply to all sorts of systems other than universal machines and certainly does not imply that any system which can be described functionally is a universal machine. Any Turing machine can be described functionally, but not all Turing machines are computationally universal. Undoubtedly, a program for a computer is an example of a functional characterization, par excellence, but this does not support the conclusion that all functionally characterizable systems are programmable computers which appears to be the conclusion Johnson-Laird is trying to draw. The effect of his argument is to run together the idea of functionalism with the idea that the mind is the program of the brain.

A rather similar, but perhaps less contentious, presentation of the foundations of the generic theory has been given by Jackendoff (1987), who states the case as follows; "...the information content of data and programs...can be stated independently of physical instantiation in any particular computer...Thus there is a sense in which,

like the mind, the information in the computer is autonomous -- inhabits a separate domain -- from the (mere) hardware that supports computation...the computer analogy suggests the following hypothesis: just as we need not deal with the actual wiring of the computer when writing our programs, so we can investigate the information processed by the brain...independent of questions of neurological implementation. This approach is often called *functionalism*; the idea behind this term is that the function rather than the physical substance of the brain is significant in studying the mind...It is now routine to speak of the information in the brain as *mental representations* and of the processes operating on such representations as *mental processes*. In short, the mind is taken to stand to the brain as the software and data of the computer stand to the hardware." Jackendoff (1987, pp.15-16). Jackendoff explicitly makes the point that the computer analogy suggests a hypothesis about the relation between mind and brain rather than demonstrating the independence of the one from the other, but the autonomy of mind is taken as a given and the final sentence somewhat contradicts the idea that the computer analogy is a hypothesis and not demonstrable fact.

What functionalism does highlight as an issue for careful consideration is the nature of the relationship between functional and physical descriptions. In Chapter 1.4 the issue was briefly discussed. What seems clear is that there is a range of degrees of constraint which merit discussion. Consider the functional description "An instrument for eating". This is a description which enjoys freedom from constraint in at least two dimensions. First it applies to a range of utensils which meet the description in different ways such as cups, spoons, knives and forks, and second it applies to utensils which are of the same functional type, e.g. spoon, but are made of different materials such as metal, wood or plastic. By making the functional description more specific, e.g. "An instrument for eating liquid foods" the constraints on its realization can be increased, and it seems possible to tighten the constraints to an arbitrary degree. Compare "A vehicle for travelling from London to Edinburgh" with "A vehicle for travelling from London to Edinburgh in less than one hour." Despite the possibility of tightening constraints arbitrarily, however, there will always be a gap, at least in principle, between functional and physical

descriptions. There are at least two reasons why this is so. First, functional and physical descriptions serve different purposes and use different vocabularies, and second, there seems to be no clear limit to the power of the imagination to conjure up different ways in which specific functions can be implemented. It seems unlikely then that a simple theory delineating the entire range of relationships between functional and physical descriptions can be constructed.

As far as the relation between mind and brain is concerned, if it turns out that the brain does implement a general purpose computer or universal Turing machine then the links between function and substance will be as loose as it is possible for them to be, but it is worth emphasizing again that this is a hypothesis not a demonstrable truth and it may be that function and substance are more closely linked in the case of mind and brain than the generic theory suggests. von Neumann, for example, in his address to the Hixon symposium in 1948 suggested that the only practical way of describing what constitutes a complex function like the capacity to make visual analogies might be to give a description of the connectivity of the visual part of the brain. Far from its being "mere hardware", it may be that the study of the brain is the best way to understand its functions.

5.3. Functional and physical description of Turing machine control states.

For present concerns, although it is a major contention of the thesis that the brain is not organized as a general purpose computer, it is argued that the brain is part of a computational system which, together with aspects of the environment realizes a Turing machine, and hence it is claimed that the functional states of the brain may be thought of as analogous to the states of the control of a Turing machine. Thus, rather than trying to encompass the full range of connection between functional and physical descriptions, it is appropriate to consider the relationship between the functional and physical descriptions of Turing machine control states.

A helpful concept in this discussion is Pylyshyn's concept of an "instantiation function". Pylyshyn deploys the concept to explain the relationship between the physical and computational (i.e. functional) states of a computer. The point that he

wants to make is that we need a way of picking out certain physical properties of a computer if we want to describe its computational states, because most of the indefinitely large number of physically discriminable properties such as its colour and its mass which might be mentioned in a physical description are irrelevant to its computational description. The properties which are relevant are those properties of components which are meant to react in particular specified ways to changes in their input conditions. The instantiation function is, therefore, a prescriptive specification of two sets; first, the set of properties of the machine which are relevant to the specification of its computational states and second the set of values of those properties which define a computational state or states. A given computational state is thus "an equivalence class of physical states indistinguishable from the point of view of their function in the machine's abstract computational description." Pylyshyn (1984, p.56). As a pertinent example consider the binary states which consititute the primitive computational states of a digital computer system. These are nominally known as the '0' state and the '1' state. In digital computer systems, various different schemes have been used for assigning voltages to these states. Generally logical states are defined in terms of bands of voltage sometimes with a forbidden region between them. Thus the '0' state might be defined as the band between zero and one volt, and the '1' state as the band between two and five volts, Cripps (1977, p.3). In this case, the instantiation function represents a simple mapping of voltages onto computational states. A simple instantiation function of this kind might then form part of a more complex instantiation function to map the relevant physical states of the transistors constituting an AND gate, say, onto the requisite computational states of logical AND. An example of a physical property of such a transistor device which is irrelevant to the specification of its computational properties, although essential for its proper functioning, is the state of its power supply.

It is clear that the specification of the instantiation function specifying the computational states of a whole computer would be a massively complex task which nobody would attempt because there would be no point to it. Because computers are structured hierarchically, appropriately chosen instantiation functions at the

lowest logic levels coupled with rigorous construction methods ensure that aggregations of physical units will correctly map onto the compositions of logical functions which they represent. Whether such a logically perspicuous, hierarchical type of instantiation function can be described for the relation between the physical and computational states of the brain is a matter for empirical investigation.

The essential difference between physical properties and computational states as far as computers are concerned is that their physical properties are intrinsic qualities of the media out of which they are constructed whereas their computational states are prescribed via an instantiation function in terms of equivalence classes of relevant physical properties and values of those properties. It is the prescriptive nature of computational states which makes it possible for them to be implemented in multiple media, given that a medium has physical properties with the requisite characteristics.

It is clear from the example of the AND gate, that the instantiation functions of digital computers are such that there will be some physical properties, such as colour, which are strictly irrelevant to their computational states, and others, like power supplies to logic switches which are essential to the maintenance of correct logical functioning, but do not contribute directly to that functioning themselves. The latter properties are part of what Pylyshyn (1984) calls the functional architecture. The fact that many essential properties of a machine may not contribute directly to its computational states raises the question of the proportion of the physical mass of a machine which does contribute to its computational states. It is easy to imagine how it might be possible to develop some kind of index of efficiency based on something like the ratio of the mass of the machine and its power consumption to the number of logical decisions made, for example. Clearly the early machines like ENIAC were very inefficient while the brain is extremely efficient.

It is important to note that considerations related to efficiency impose a variety of physical limits in the real world. A flying bird with the mass of an elephant, for example, is impossible because muscle is not a sufficiently effective power source

to raise such a mass off the ground. It is therefore pertinent to ask whether there might not also be physical limits related to efficiency constraining the media suitable for implementing cognitive processes. One way to envisage how this might happen is to imagine an instantiation function which makes such efficient use of the physical properties of its medium that redundancy is minimal. The mapping from physical to computational states is such that almost every physical property has a computational role. It would still be the case that one could give separate logical and physical descriptions of the functioning of such a creature, but it might be the case that the constraints were such as to ensure that the implementation medium was uniquely suited to its tasks. Might it not be the case that such minimal redundancy turned out to be an essential feature of the engineering of an intelligent creature, given the other physical demands such as mobility which the organism would need to support? Perhaps the brain is a medium of this sort. Against this of course would have to be set the fact that the brain is capable of effective operation under a variety of conditions of loss and damage and exhibits a form of redundancy via replication or multiplexing. However, a creature whose computer made inefficient utilization of its medium might turn out to be so heavy that its muscle power was insufficient to enable it to escape from a potential predator which its intelligence had enabled it to identify. The idea that intelligence is based on a computational architecture whose medium is not an essential part of its specification is so much an article of faith that it is important to try to envisage situations in which this might not be the case. It may not be biological chauvinism as Block (1980) argues to maintain that intelligence is exclusively a property of brains.

The idea that the efficiency with which physical resources are utilized might be a useful parameter to consider when trying to understand the nature of cognitive computation suggests that it is appropriate to try to understand the human instantiation function or functions rather than seeking to isolate the study of cognitive processes from the study of brain processes. If, as is likely to be the case, human computational states make much more efficient use of the brain than our computational artefacts do of their media, it may turn out that detailed understanding of neural circuits is essential to a detailed understanding of cognitive processing.

von Neumann's suggestion about visual analogy being best understood in terms of the organization of the visual areas of the brain may turn out to be yet another of his prescient insights.

The functioning of transistors in computer circuits provides a final example related to instantiation functions and efficiency. The junction transistors commonly used in these circuits can be operated in three modes known as active region, saturated and cutoff, Bartee (1981, p.186). For digital switching purposes transistors tend to be operated at or near the extremes of their range, that is in either saturated or cutoff mode. The active region mode of operation is not used for switching purposes and hence would not be relevant to an instantiation function which used the saturated and cutoff modes to provide instantiations for logic '1' and '0'. However, Mead (1989) has shown that using the active region of transistors to implement a variety of non-linear, analogue computational primitives makes much more efficient use of their intrinsic properties, and he and his colleagues have begun to explore the simulation of human neural circuits using the intrinsic analogue properties of silicon circuits as computational primitives rather than driving them to the extremes of their operational ranges to implement binary switches. Some of the work of this group is described further in Chapter 6. The direction in which it clearly points is that the intrinsic nature of the computational medium may be much more important for understanding natural computation than digital models allow.

It seems quite clear, therefore, that the relationship between the logical or functional states of Turing machines and physical realizations of them is a complex issue which cannot be settled by fiat. While instantiation functions are prescriptive in the case of computers, the human instantiation function, if such can be found, is a result of evolution and needs to be discovered not stipulated. It does not seem to be the case as Johnson-Laird suggests, that functionalism implies that the mind must be organized as a general purpose computer.

5.4 Functionalism and Multiple Instantiation.

The relationship between functionalism and the argument from multiple instantiation, i.e. the claim that the medium of implementation is irrelevant to the functional specification of a system because the same functional system can be implemented in different media, has been the subject of a relatively recent exchange in the journal "Cognitive Science" (cf Thagard 1986, 1987; Krellenstein 1987; Ramsey 1989).

Thagard (1986) claims that the argument from multiple instantiation constitutes the core of the functionalist position and should, in principle, rule out parallel hardware as irrelevant. This, he says, makes functionalism "computationally naive", because parallelism allows "qualitatively different kinds of algorithms for intelligent operations." Thagard further claims that the Turing machine model, while mathematically satisfactory, is "seriously defective" as a model for understanding intelligence because it ignores the constraint on human intelligences that they must operate in real time. Intelligence, says Thagard, "should be viewed as relative to the environment in which behavior must occur." Clearly, Thagard has in mind here the Turing machine model as deployed by generic theorists.

Both Krellenstein (1987) and Ramsey (1989) take issue with Thagard's construal of functionalism. Krellenstein argues that the primary concern of functionalism is not with different means of physical instantiation as Thagard suggests, but with the distinction between a (virtual) machine and a program. Krellenstein also argues that

> "A program running on a parallel machine that produced some sort of intelligence will also run on a serial machine, and this is enough to show the hardware irrelevant for explaining the nature, if not the evolution, of that particular intelligence."
>
> Krellenstein (1987, p.155).

Thagard (1987) accepts the "in principle" argument, but suggests that the practical aspects of the speed up provided by parallel hardware are more significant than Krellenstein allows. A similar position has been taken towards Krellenstein's argument by Clark (1989, pp.121-122).

Ramsey's (1989) critique of Thagard's position, asserts that multiple instantiability is not the basis of functionalism[16], and also that functionalism does not license the neglect of neuroscience. Ramsey argues that Thagard's attack depends on two claims about functionalism T1[17] and T2, and two claims about parallel systems T1' and T2'.

T1 Functionalism depends on an explicit distinction between hardware and software.

T1' Parallel systems blur the distinction between hardware and software.

T2 Functionalism asserts that considerations of physical implementation are largely irrelevant to the characterisation of psychological states.

T2' Parallel systems support qualitatively different kinds of algorithms from those available on von Neumann style stored program serial machines.

Thagard, says Ramsey, uses T1' to argue that the distinction made in T1 is not well founded and T2' to argue that T2 is false. Hence functionalism is false and so is the argument from multiple instantiability. Ramsey mounts a defence which maintains that T1 is not a thesis of functionalism and that T2 properly understood is not undermined by T2'. Ramsey makes the case against T1 as follows. First he distinguishes two possible construals of functionalism, F1 and F2.

F1 Psychological states are determined by their functional or causal roles and not by first order physical properties.

F2 Psychological states are determined by their roles in programs and not by the way those programs are realised in hardware.

F1, says Ramsey, is a proper statement of functionalism in psychology. F2, which has a superficial similarity to F1, makes it appear as though functionalism requires

---

[16]Multiple *instantiability* in principle, is a theoretical consequence of Ramsey's construal of functionalism. Multiple *instantiation* in practice, would be contingent on the existence of suitable physical media.

[17]These labels are not taken from Ramsey's original paper. I have used them in order to try to clarify the complex points he makes.

a hardware - software distinction, but Ramsey points out that there are two ways of understanding the term "program". It may be used to refer to a causally inert description such as a flowchart which simply describes the workings of an algorithm, or it may be used to refer to a set of commands in a programming language which causes a machine to behave in such and such a way. The two uses of the term "program", which may be called the "descriptive" and "operational" uses are easily confused because a clearly written program, intended to be operational, may also serve as a description of the algorithm which it implements[18]. However, functionalism requires only the descriptive use of the term "program" which is independent of the existence of hardware. Thus, even though F2 may be true when properly construed, it does not license T1, hence T1', which may also be true, does not show functionalism to be false.

Ramsey goes on to suggest that an equivocation with respect to the term "hardware" undermines Thagard's argument from T2' to the falsity of T2 and thus to the downfall of functionalism. Ramsey claims that "hardware" may be used to refer both to the "stuff" from which a system is made and to the "architecture" of that system. Thagard's concern in T2' is architectural. Since he argues from T2' to the falsity of T2, he must be claiming that "considerations of physical implementation" in T2 is also a reference to architecture. Ramsey denies that this is the correct reading of the functionalist claim in T2. Functionalism, says Ramsey, claims that the nature of the "stuff" out of which a system is constructed is irrelevant, i.e. you could realise functionally equivalent systems in brain or in silicon. However, functionalism does not claim that architectural considerations are irrelevant. Thus Thagard's argument from T2' to the falsity of functionalism also fails.

The clarification of the issues provided by Ramsey's analysis suggests that Thagard's arguments against functionalism are not just untenable but also

---

[18]While this is particularly true of logic programming languages like Prolog, c.f Clocksin and Mellish (1984, p.vii, & p253), it is noteworthy that much of the effort which has gone into developing other new computer languages has also been intended to enable programmers to produce operational specifications of algorithms descriptively.

unnecessary, because his main point is compatible with functionalism. The argument from multiple instantiation is an argument about stuff rather than architecture, says Ramsey, and thus quite compatible with the notion that considerations of parallelism may be relevant to psychology.

5.5. A theoretical case for parallelism.

The attack on Thagard by Krellenstein, asserting the reducibility, in principle, of any parallel machine to a serial one is the kind of argument frequently advanced by those who wish to claim the theoretical high ground, and such arguments leave a residual feeling of unease even among the most hardened of those to whom practical considerations such as real time responding are of fundamental importance. Wells (1993) discusses recent work in mathematical logic which provides an argument for the importance of parallelism in principle. The case is based on work by Shepherdson (1975, 1988) and by Gandy (1980) which develops a more general understanding of the nature of computation than is provided by Turing's original analysis. This work is motivated by a desire to provide firm foundations for the claim that anything which can be computed by a machine is calculable. The point at issue is succinctly put by Gandy. He distinguishes two claims which he calls Theorem T and Thesis M.

> **Theorem T.** What can be calculated by an abstract human being working in a routine way is computable.
>
> **Thesis M.** What can be calculated by a machine is computable.

Gandy claims that Turing (1936-7) provides a proof of Theorem T but not of Thesis M. One of the reasons why Thesis M. is not proven by Turing's original work, is that it is possible for a machine to print, or otherwise operate on, an arbitrary number of symbols simultaneously whereas Turing's analysis, modelled on the way in which a human computes with paper and pencil, is restricted to serial operations on just one symbol at a time. The justification of Thesis M. must, therefore, take parallel working into account. One way of doing this is to define aspects of parallelism in terms of extensions to the standard single tape, single read-write head

Turing machine, and to show that such extensions do not add to the set of computable functions. It can be shown, for example, that parallelism in the form of multiple tapes and/or multiple read-write heads does not increase computational power. This is accomplished by showing that the squares of multiple tapes can be mapped systematically onto the squares of a single tape, and that the operations of multiple scanning heads can be tracked and performed by a single head. Lewis and Papdimitriou (1981, pps.198 ff) provide a proof. Shepherdson discusses cases for which the reductive tactics above do not succeed;

> "In the case usually considered, of computation over total structures, i.e., ones whose functions and relations are defined for all arguments, parallel procedures are no more powerful than serial ones, for one can obviously serialize a parallel procedure by subdividing the time scale. This is no longer true if there are partial functions, e.g. the function f defined by
>
> $f(x) = x$ if $f_1(x)$ is defined or $f_2(x)$ is defined
> $=$ undefined otherwise,
>
> obviously cannot be computed by any serial procedure because it might choose the wrong one of f_1, f_2 to evaluate first."
>
> Shepherdson (1988,pp.584-585).

The point is that if a function is undefined for a given argument it will not return a value in a finite amount of time but will behave like a computer program in an infinite loop. Hence, with a serial procedure in a case, say, where $f_2(x)$ is defined but $f_1(x)$ is not, if $f_1(x)$ is tried first the computation will never terminate. With two processors, however, the computations of $f_1(x)$ and $f_2(x)$ can be started simultaneously and independently of each other; $f_2(x)$ will, in due course, yield a value to feed to f(x) and the attempted non-terminating computation of $f_1(x)$ can be discarded as irrelevant. Hence a parallel system is more powerful than a serial one for this type of function. The argument depends, of course, on the assumptions made about the capabilities of the processing units which evaluate the functions $f_1(x)$ and $f_2(x)$. In particular, if it is assumed that the units proceed in a series of steps and can be stopped and restarted where they left off, then the capabilities of a pair of processing units operating in parallel can be simulated by a serial procedure

which alternately computes single steps of $f_1(x)$ and $f_2(x)$. However, as Shepherdson says, "even if this is possible it seems to be an unnatural and inefficient trick for serialising what is essentially a parallel procedure" (Shepherdson 1975, p.469).

It would appear, therefore, that theoretical justification for a parallel system rests, in part, on the nature of the domain over which the system computes. If the domain is such as to include some arguments for which the functions computed by the system may be undefined, then a parallel system is, given appropriate assumptions, more powerful than a serial one. It seems perfectly possible, indeed highly likely, that the domains over which psychological computations are defined will include partial functions. Gandy's work is considered further in the development of the ETH in Chapter 6.

5.6. The Physical Symbol Systems Hypothesis.

An influential line of thought which has been systematically developed for some twenty years by Allen Newell, often in conjunction with Herbert Simon (Newell & Simon, 1972, 1976; Newell 1980, 1982, 1990) is the Physical Symbol Systems Hypothesis (PSSH). The symbol systems hypothesis is one of the major contributors to the standpoint described in the thesis as the generic theory. Newell is a determined advocate of a view which claims that the brain realizes a symbol system which is a universal machine by definition, and that the characteristic flexibility of intelligence is a consequence of the computational universality of the brain. In some circles the PSSH has acquired the status of a "central dogma" (Pinker & Mehler, 1988). Pinker (1994, p.78) suggests that the symbol systems hypothesis is "as fundamental to cognitive science as the cell doctrine is to biology and plate tectonics is to geology."

5.6.1. Definition of a Physical Symbol System.

In a systematic statement of the symbol systems hypothesis in their 1975 Turing Award Lecture, Newell and Simon defined a physical symbol system as follows;

> "A physical symbol system consists of a set of entities, called symbols,which are physical patterns that can occur as components of

> another type of entity called an expression (or symbol structure)... A physical symbol system is a machine that produces through time an evolving collection of symbol structures. Such a system exists in a world of objects wider than just these symbolic expressions themselves."
>
> Newell & Simon (1976, p.40)

Two other central ideas were also defined. These are the ideas of **designation** and **interpretation**. Designation is the means by which a symbol or expression is related to an object, and interpretation is the process by means of which designating structures of symbolic expressions are evaluated. The thing to have in mind when trying to visualize a physical symbol system, is a general purpose computer running LISP.

> "The type of system we have just defined is not unfamiliar to computer scientists. It bears a strong family resemblance to all general purpose computers. If a symbol-manipulation language, such as LISP, is taken as defining a machine, then the kinship becomes truly brotherly."
>
> Newell & Simon (1976, p.41).

The physical symbol systems hypothesis, based on the notion of a system defined above is simple but of very broad scope.

> "A physical symbol system has the necessary and sufficient means for general intelligent action."
>
> Newell & Simon (1976, p.41).

Newell and Simon maintained that the PSSH was nothing new in principle, but simply a systematization of a collection of more or less inchoate ideas which had been arrived at by a variety of people as a result of a number of influences. The influences identified were the development of formal logic, Turing's work on the theory of computation, the stored program concept and the idea of list processing (Newell & Simon 1976, 42-46). The PSSH was intended to make a quite specific architectural assertion about the nature of intelligent systems, based on a proprietary

notion of a symbol, which went beyond the theoretically motivated roots of the discipline of computation provided by Turing. Newell and Simon also explicitly acknowledged the influence of the evolution of digital computers on the form of the PSSH, thus making clear its empirical nature. The evidence adduced for the hypothesis in the 1976 paper was scanty. Newell and Simon in fact suggested that the primary evidence for the PSSH was negative, i.e. the absence of competing hypotheses as to how intelligent activity might be achieved.

5.6.2. The Nature of Symbols.

In his 1980 paper, Newell provided a more detailed account of the PSSH in which he argued for the necessity and sufficiency of symbol systems as the basis for intelligence rather than simply making the assertion as he and Simon had done in the earlier paper. From the point of view of the ETH and the analysis of Chapter 4 it is striking that Newell saw no need to argue for the assumption that the brain contains a symbol system. Given that a symbol system is a universal machine by definition, Newell's hypothesis is equivalent to the claim that the brain contains or implements a universal Turing machine. Indeed, part of his argument includes showing that his schematic symbol system is universal by showing how to implement a Turing machine in the formalism. Thus the symbol systems movement is a clear example of a program which assumes part of what it needs to prove.

Quite apart from this fundamental problem there are problems of detail with Newell's analysis. The symbol concept is pivotal to Newell's argument which hinges on the contention that the properties of symbols and symbol processes in physical symbol systems as Newell describes them, are essential properties of every universal machine. It is argued here that Newell's arguments do not establish his case.

It is important from the outset to distinguish various uses of the root term "symbol" and its cognate forms and this is done using a subscript notation. In particular there is a distinction to be made between "symbol_N" and "symbol_T". Symbol_N is a notion which is internal to the concept of a physical symbol system. A symbol_N

implements the capacity of "designation" introduced in the 1976 paper by providing "distal access" to an object, process or symbol other than itself. "Designation" amounts in effect to a scheme for binding symbolic identifiers to processes. $Symbol_T$ is a term which denotes a member of the alphabet of simple symbols which are used by a Turing machine when reading from or writing to its tape. The essential property of these symbols is the nominal property of type identity. A token of a given symbol must be indistinguishable from any other token of the same symbol and distinct from every token of any other symbol. The simple unadorned term "symbol" is used in the discussion when neither of the specific terms applies and in direct quotation.

5.6.3. An outline of Newell's argument.

In outline, Newell's argument runs as follows. $Symbols_T$ are sufficient for the specification of machines which compute single fixed functions. These are machines like ordinary Turing machines or finite automata. In order to construct a universal machine, however, decomposition of the input to the machine into two parts must be introduced, such that one part of its input is the program of the target machine and the other part is the data which the target machine would receive as input. This decomposition may be done directly with $symbols_N$ or by using strings of $symbols_T$ and an addressing scheme on the unbounded tape of a Turing machine which together amount to a $symbol_N$ system. $Symbolisation_N$ is thus a necessary condition for a universal machine. Finally, the physical symbol system hypothesis asserts that $symbolisation_N$ is a concept of sufficient generality to encompass all other instances of symbolization in cultural symbols and symbolic artefacts generally. $Symbolization_N$ is thus the crucial concept which unites the primitive mechanisms at the lowest level of a system with the mental entities at the highest level and with entities in the external environment within which a system exists.

5.6.4. The argument in detail.

The detailed explanation of symbol systems starts with a specific example which Newell describes in detail, and which turns out to look very much like a slightly abstract version of a Lisp interpreter. The resemblance is intended, and Newell

reiterates the view of the 1976 paper that this is a good thing. The description of the example system is followed by the assertion that symbol systems constitute a class of universal machines.

> "Symbol systems form a class -- it is a class that is characterized by the property of universality."
>
> Newell (1980, p.147)

This claim is followed by a discussion of universality and an introduction to various formulations of the class of effective procedures and the difficulties which attend the construction of universal machines. One of these difficulties concerns the fixity of machines and leads to the requirement for the partitioning of the input into a description of the machine to be simulated and a description of the contents of the target machine's tape. Newell asserts, with respect to the input that

> "the basic decomposition into two parts has far-reaching consequences -- it guarantees the existence of symbols."
>
> Newell (1980, p.149)

Newell clearly means $symbols_N$ here, in accordance with the outline argument above. By this stage of the argument then, Newell has described an example symbol system, discussed the nature of universality and made two important claims as follows;

C1: Symbol systems are universal machines.
C2: Computational universality guarantees the existence of $symbols_N$.

Newell demonstrates the truth of C1 for the example system by showing that it can be used to simulate a universal Turing machine, but has, at this point, no general characterisation of symbol systems to justify the claim for the class as a whole. One might therefore expect the development of the argument to continue with a characterization of the class and a proof that C1 is indeed true for the whole class. Instead Newell takes what he admits to be the curious step of making C1 true by

definition. "Symbol systems are the same as universal machines." (p. 154). Newell takes this step because he recognises

> "that we do not have an independent notion of a symbol system that is precise enough to counterpoise to a universal machine, and thus subsequently to prove their equivalence."
>
> Newell (1980, p.155)

Newell argues, however, that the apparently arbitrary equation of symbol systems and universal machines is legitimate because

> "we have *discovered* that universal machines always contain within them a particular notion of symbol and symbolic behavior, and that this notion provides us for the first time with an adequate abstract characterization of what a symbol system should be. Thus, tautologically, this notion of symbol system, which we have here called physical symbol system, is universal."
>
> Newell (1980, p.155)

This claim suggests that on examination a universal machine is always found to use a particular notion of a symbol ($symbol_N$) and a notion of symbol processing, and it is these notions which define symbol systems and all other universal machines. Thus the argument turns out to be a more complex account of the simple assertion of the 1976 paper. But it is certainly far from obvious that every universal machine contains the concept of $symbol_N$. A striking point about universality is that very little machinery is needed to achieve it.

Newell tries to buttress his claim by asserting that every universal machine exhibits in some form all the essential properties of any other universal machine. This may be so, but unless it is clear what those essential properties are, the observation does not advance the argument. Newell makes the suggestion that although there are differences among universal machines, which may in some instances be critical, "these differences are not critical for the nature of symbols." (Newell, 1980, p.155)

It is worth expending considerable effort to be absolutely clear about just what Newell's claim entails, and whether there is a case for it, because the notion of symbol involved, $symbol_N$, is pivotal not just for Newell's account in the 1980 paper, but for much of his work over the decade to 1990 on the SOAR architecture. At the heart of the notion is the concept of designation, for which Newell provides the following definition.

> "*Designation.* An entity X designates an entity Y relative to a process P, if, when P takes X as input, its behavior depends on Y."
>
> Newell (1980, p.156)

Designation, according to Newell, has two primary features. It grounds the notion of symbolization in the behaviour of a process and it provides, as Newell puts it, "action at a distance" (1980, p.156). The means for providing this distal access are to be found in the early list processing languages identified as an influence on the development of the symbol systems hypothesis in the 1976 paper. Newell was part of the development team of the IPL (Information Processing Language) languages IPL-1 to IPL-5. List processing has been described as "One of the most significant events that has ever occurred in programming" (Sammet, 1969,p.388) and IPL-5 was widely implemented. Distal access was achieved by having addresses as the elements of lists. These addresses provided access to other lists. Thus the concept of designation is essentially the concept of addressing.

An example of designation is the use of symbolic identifiers in LISP to denote functions. This can be a highly complex business, (cf. Steele 1984; Wilensky 1986). A simple introduction is provided by Tatar (1987, chapter 3). When a user defines a function in LISP using DEFUN, which is a function definition function, a binding is set up between the identifier which names the function and the code which constitutes it. Thus, the line

```
(defun square (x) (* x x))
```

defines a function named "square" which takes a single argument "x" and multiplies it by itself. Once the definition above has been evaluated, calls to the function will

yield the appropriate numerical results.

```
(square 3)
9
(square (square 3))
81
```

In terms of Newell's definition, the identifier "square" designates the function which calculates the product of a number with itself, relative to the process of evaluation carried out by the LISP interpreter, because, when the interpreter takes the identifier as input, its behaviour depends on the function code. The identifier is bound to the function code by storing the address of the code as part of the information associated with the identifier. Thus designation is a name for the method of location addressing which is characteristic of digital computers.

Once the concept of designation and its roots in the construction of list processing language entities is understood, the operation of interpretation is easy to put into perspective. Newell defines interpretation as follows;

> "*Interpretation.* The act of accepting as input an expression that designates a process and then performing that process."
>
> Newell (1980, p.158)

What should be clear from the above examples is that the notions of designation and interpretation are technical notions describing the internal processing of a computational system, and are particularly related to how a digital computer obtains the operands for its instruction processing. This is not just a feature of Newell's presentation in 1980. It remains central in his "Unified Theories of Cognition" (1990), where he refers explicitly to the role of $symbols_N$ as providers of distal access to non-local structure. The argument there is simple. It is a natural law, says Newell, that processing in the physical world is always local, i.e. takes place in a limited spatial region. If this were not so, there could be action at infinite distances in contravention of physical law. Localization implies that as tasks grow in size, there will come a time at which additional structure from outside the local

region will have to be accessed in order to complete the task. $Symbols_N$ which designate non-local structure are the means by which such distal access is managed in computational systems.

> "The symbol token is the device in the medium that determines where to go outside the local region to obtain more structure."
>
> (Newell, 1990, p.74)

5.6.5. Designation and Representation.

The argument that $symbols_N$ are necessary for computational universality includes an important equivocation on the term "designation". In some instances Newell uses it to talk about the representational aspects of symbolization, in others about the specifics of distal access mechanisms. The equivocation is disguised to some extent by the fact that Newell takes a very broad view of the concepts which the term encompasses. Immediately prior to the definition of designation given above, Newell says,

> "We call this concept *designation*, though we might have used any of several other terms, e.g., *reference, denotation, naming, standing for, aboutness,* or even *symbolization* or *meaning*. The variations in these terms, in either their common or philosophic usage, is not critical for us. Our concept is wholly defined within the structure of a symbol system."
>
> Newell (1980, p.156)

The terminological looseness of the above allows Newell to make plausible sounding claims like the following;

> "Designation is at the heart of universality. For one machine to behave as an arbitrary other machine, it must have symbols that designate that other."
>
> Newell (1980, p.157)

The flaw in the argument is a failure to distinguish the designatory capacity of the symbol tokens which provide distal access in a given system, from the symbolic

expressions which provide the descriptive reference to a target machine. The need for a distinction can be seen immediately by noting that distal access to unbounded regions of structure is required, not just by universal machines, but by any machine, e.g. a parenthesis checker, whose input may be of unbounded size. Hence the capacity for distal access and universality must be distinct. The issue can be clarified by distinguishing direct from virtual input. A direct input is defined as one which is causally related to the hardwired behaviour of a machine. Thus, to a first approximation, the symbol alphabet of a Turing machine provides its direct inputs, and a binary machine instruction is a direct input to a digital computer. A direct input may consist of more than one element i.e. it may consist of a string of Turing machine symbols, but the length of the string must be fixed. A virtual input, by contrast, consists of expressions constructed from elements of direct input which may be taken, by an external observer, to refer to or to represent entities external to the system. Virtual inputs are things such as strings of Turing machine symbols of arbitrary length or elements of high level languages. Virtual inputs express logical rather than causal relations, and the trick with universal machines is to arrange their direct inputs in such a way as to produce the same output as a machine whose direct inputs had the causal structure mirrored by the logic of the virtual inputs to the universal machine.

One point of the definition of designation as Newell gives it is to tie the concept of designation to the concrete behaviour of a process. Designation is thus a local capability, properly ascribed to a specific process with particular input and output capabilities. The essence of this capacity is its finitude not its universality. Designation is a property of direct inputs. Representation, by contrast, is a function of virtual inputs, and it is representation, not designation, which is at the heart of universality.

Turing's universal machine provides a clear example of the distinction between direct and virtual inputs (cf. Chapter 3.4). Direct inputs produce the sequence of figures which would be computed by the target machine, whereas virtual inputs are indirectly responsible, via matching and copying, for the sequence of complete

configurations which describes the target machine's computations. The first point to notice is that the "figures" produced by a machine in Turing's original scheme, i.e. the elements of "the sequence computed by the machine" are just the binary digits 0 and 1. These digits are mapped via the standard description scheme to the expressions "DC" and "DCC" respectively. Similarly, the starting state of every machine is mapped to the expression "DA" and the direction of travel operations, left, right, and no movement are mapped to the primitive symbols "L", "R" and "N". The importance of this scheme is that it provides all the designatory capacity needed for the universal machine to interpret the description of the target machine. All that is required of the other symbols used by the target machine, is that they be coded consistently throughout, and in such a way as to reflect systematically and accurately the instructions defining the target machine. They do not have, indeed cannot have, a fixed designatory mapping to the primitive symbols of the universal machine. Consider, for example, the standard description of machine M in Chapter 3.4. The symbol "X" of M is mapped to the expression "DCCCCC" and the symbol "(" is mapped to "DCCC". This is not a designatory mapping, because it makes no direct contact with the inner mechanics of the universal machine, which would still work properly, provided all the necessary changes were made to the standard description of M if the mapping were the other way round. By contrast, if the mappings for 0 = "DC" and 1 = "DCC" were changed, the internal structure of the universal machine would have to be reprogrammed for it to produce the correct output of those target machines which print '0's and '1's. Likewise with the mappings for the start state and the direction of movement indicators. The simulation mechanism is sensitive to these mappings. Thus the direct inputs to Turing's universal machine are the elements of its symbol alphabet, plus the expressions, "DA", "DC", and "DCC". The direct involvement of the expression "DA" can be seen in the universal machine state "b1" and the involvement of expressions "DC" and "DCC", in states "sh2", "sh3", "sh4" and "sh5". Unlimited representational capacity via virtual inputs is achieved with the finite designatory capacity of the direct inputs. To argue as Newell does, that designatory capacity is fundamental is correct in one sense, but misses the crucial relation between designation and representation.

5.6.6. External Reference.

The technical flaw described above is symptomatic of a related but much deeper problem with Newell's hypothesis which is a direct consequence of the assumption that an internal Turing machine model is the correct basis on which to understand cognitive architecture. Given such a model, external inputs are virtual inputs to the cognitive machine and must be translated into direct inputs by transducers. The solipsistic bias inherent in such a position, which is characteristic of all versions of the generic theory, is particularly evident in Newell's recent work, and is manifest in his treatment of the issue of the relation between symbols in a symbol system and the external world. The point to notice in the 1980 treatment which foreshadows the later remarks is that although designation is a relation which is ultimately intended to cover all instances of reference both internal and external, it is defined as a concept which is wholly internal to the structure of a symbol system.

> "The prototype symbolic relation is that of access from a symbol to an expression, not that of naming an external object. Thus, it is an implication of the formulation, not part of its definition, that the appropriate designatory relations can be obtained to external objects (via chains of designation)."
>
> Newell (1980, p.169)

It is significant that Newell takes it to be an "implication" rather than a hypothesis, that the appropriate relations can be found. This follows from the assumption that the cognitive architecture is a Turing machine implemented in the brain. Given that the mind does make adequate contact with the external world, then, because it is a Turing machine, it must be the case that there are appropriately structured transducers etc. which manage the transfer of information from outside to inside and vice versa. Hence there can be no difficulty in principle in the production of adequate direct input. This seems to be the reason for Newell's very sketchy treatment of the problems of reference and intentionality which concern virtual input. He raises the question,

> "How does it ever happen that a symbol actually *refers to* or *denotes* or *designates* or *stands for* or is *about* something outside the system

itself?"

Newell (1990, p.78)

The answer is apparently rather simple assuming there to be knowledge systems, (Newell 1982) which are symbol systems with a well defined level above the symbol level in which knowledge is the medium and rational principles govern behaviour. Knowledge, says Newell, is about its domain which is external to the system. It is a way of describing a system from an observer's perspective (Newell, 1990, p. 78). Symbol systems realize knowledge systems, hence they are about the same external things as the knowledge systems they realize. The realization is achieved by "implementing representation laws so that the symbol structures encode the knowledge about the external world" (Newell, 1990, p.79). And that, apparently, is that.

> "That's all there is to be said about intentionality, except for the details of how to build symbol systems that approximate knowledge systems closely enough."
>
> Newell (1990, p.79)

The easy optimism is a consequence of the fact that Newell is concerned with direct rather than virtual input, with processing rather than with representation. He says, for example

> "...that complex systems require symbol structures, in the sense of embedded symbol tokens that provide distal access, is separate from the notion of representation. It is a requirement that arises from the constraints of processing, not from the contraints of establishing relations to the external world."
>
> Newell (1990, p.75)

This approach is of a piece with the view, consistent throughout his work since the early papers on the Logic Theorist in the 1950's (cf. Newell, Shaw, & Simon 1958), that it is purely a tactical question to concentrate on internal symbolic processing and not to deal with perceptual and motor systems.

> "The difficulty is that the total system is too complex to handle all at once, and the sorts of considerations that go into perception and motor action seem too disparate to integrate. So the strategy is divide and conquer."
>
> Newell (1990, p.160)

Divide and conquer is perfectly respectable as a scientific strategy, of course, but it is worrying that the strategy should so neatly consign to the sidelines precisely those aspects of the total system which were peripheral to the von Neumann model of computation, when it seems entirely obvious that people and computers are at opposite ends of the spectrum of environmental connectedness. It also demonstrates again, if further demonstration be needed, how deeply ingrained is the assumption that the brain is a whole Turing machine, served by perception and motor processes rather than a finite state machine connected to its memory by perception and motor processes such that the whole system implements a Turing machine.

5.6.7. System levels and the brain.

A discussion of one further aspect of Newell's approach is relevant to the aims of this chapter. This is the question of how the levels of computational systems as he describes them are hypothesized to map onto the brain in the human cognitive system. This is not an issue on which much time is spent in the 1980 paper, but in "Unified Theories of Cognition", a substantial effort is spent in Chapter 3 on describing how neural analogues of the various system levels described in earlier chapters might be identified.

Newell identifies, four different phenomenal worlds, which are characterized by the time scale on which actions in those worlds occur. Each world comprises a number of levels which are grouped into bands. Thus the worlds are known as the biological band, the cognitive band, the rational band, and the social band (Newell, 1990, pps. 121-122). The area of particular relevance for present concerns is the set of four levels comprising the cognitive band. Newell suggests that these occupy time scales from approximately the 10 millisecond level at which neural circuits (rather than individual neurons) operate, to the 10 second level at which the units

of cognitive tasks are manifest (Newell, 1990, p.140). He argues, persuasively, that the time available for cognitive operations is bounded from above by experimental psychological data, and from below by the operational speed of neurons. Hence, the operations predicted by the symbol systems hypothesis are constrained to occur within the time band identified. The methodology is admirable but the conclusions are open to serious question. The methodology locates the basic operation of distal access at the level of neural circuits which operate at approximately the 10 millisecond time scale. The process of distal access involves locating the appropriate distal structure, (which would be done via its address in a digital computer implementation) and retrieving the information stored in the structure at the given address. According to Newell, this process of access and retrieval is carried out in the brain by neural circuits. Addressing of the kind required, as discussed in Chapter 4, is achieved in computers by constructing the memory and logic devices from highly reliable bistable devices and doubts about the plausibility of such schemes as the basis for the implementation of neural computation have been raised.

There is a fundamental question about the suitability of neural tissue as a substrate for the process described. It is also not clear that information retrieval is the kind of thing that the brain does at the neural circuit level. The point about distal access is to bring remotely stored information back to the local site. While neurophysiological evidence is consistent with the idea that mental representations can be defined as states of activity of brain cells (Changeux and Dehaene, 1989), it is not clear that information retrieval and the integration of multiple sources is carried out by successive distal accesses in the way proposed by the PSSH. It seems more likely that simultaneous activity at multiple sites is integrated by global temporal synchronization (Damasio 1989).

Even if it were plausible to explain single acts of distal access in the way proposed by the PSSH, there is the equally serious difficulty posed by the need to combine these basic acts into sequences comprising higher level actions. The problem exists because the PSSH methodology assumes that the way to get higher level functionality is by chaining sequences of lower level acts. The top level of activity

proposed in the cognitive band in Newell's account occurs at a time scale which is three orders of magnitude greater than the time scale for elementary acts of distal access. The implication of this difference is that the composed operations at the top cognitive level will involve possibly thousands of elementary acts at the lowest cognitive level. Thus the structure is logically deep. High level programming languages, which also rely on chaining extensive sequences of low level operations, have substantial hidden control structures, often in the form of system stacks to manage the associated sequencing and access problems. Newell's candidate general architecture SOAR, also makes essential use of a structure called the "context stack" to manage its decision cycles (Newell 1990, pps.171 ff.). However, nowhere in the part of his book devoted to the mapping from the PSSH to neural structures is there any mention of a stack. The basic implausibility of proposing logically deep styles of storage management as characteristic of the brain has been known, and written about since the early days of computers.

> "...whatever language the central nervous system is using, it is characterized by less logical and arithmetical depth than what we are normally used to."
>
> von Neumann (1958, p.81)

But if a stack is not postulated, the operational composition of elementary acts is compromised. In the absence of any convincing alternative hypothesis about the management of such logical depth, it is likely that there is something wrong with the idea that the structures which support rational, intelligent action in the brain are like those which support processing in a digital computer.

5.7. Pylyshyn's analysis of cognitive computation.

Many of the difficulties attendant on Newell's position arise from the basic, unsupported assertion that the brain implements a universal Turing machine. Similar problems arise in the work of another influential theorist, Zenon Pylyshyn, who has written extensively about the computational foundations of cognitive science. Pylyshyn's primary concern, in his book "Computation and Cognition" (1984), is to establish a science of cognition on firm foundations. Pylyshyn comes

to grips with the philosophical issues involved in the study of cognition in a way which Newell seems to believe is unnecessary. Pylyshyn offers arguments which purport to show that if the idea of cognition as computation is taken as a literal empirical hypothesis rather than just as a metaphor, then it can be demonstrated that the cognitive system must be structured as a physical symbol system. In particular, Pylyshyn claims to show that cognitive computation must involve the transformation of structured symbolic expressions rather the state transitions of a finite automaton. Thus, he claims, that the brain must be considered to instantiate a Turing machine. Pylyshyn's arguments are important for the ETH because, if they are correct, they show that the ETH must be false given its commitment to the view that the brain should be thought of as a finite automaton.

5.7.1. Psychological Explanation.

Pylyshyn's approach to cognitive computation is rooted in his approach to psychological explanation. When we wish to explain, rather than simply to describe, behaviour, he says, we have to appeal to the beliefs, knowledge and goals, i.e. to the mental representations, of those individuals whose behaviour is under investigation. Informally, a paradigmatic explanation sentence is of the form "X did Y because he/she believed (hoped, wished, feared, expected, etc.) that P. In such a sentence, X is an actor, Y a behaviour, and P a description of an intentional content. Thus "Mrs. Jones ran for the bus because she believed that there wasn't another one for several hours" explains the running behaviour of Mrs Jones by appealing, inter alia, to the content of one of her current beliefs. The issue which Pylyshyn identifies as a fundamental problem is how to link the intentional contents of thoughts to actions: the problem with claiming that intentional contents really are the causes of behaviour is that it is hard to see how such contents could form the basis of a respectable science. They seem to be the wrong sorts of entities to be involved in causal transactions.

> "How is it possible for properties of the world to detemine behavior when the properties are not causally related in the required sense to the functional states of the system, which is what we seem to be claiming when we say, for example, that what determines the

> person's behavior in rushing to the phone is something like his anticipation of, or desire to obtain help?"
>
> Pylyshyn (1984. p.26)

One manifestation of the difficulties is that intentional state types appear to cross classify the natural kind categories which might provide the necessary causal agency (Fodor, 1975). There is no simple mapping from, say, the characterization of a situation as an emergency, to a type of physical event which explains how an emergency causes the behaviours associated with responding to it, e.g. the behaviour of summoning assistance. This is because there are no principled constraints on the physical ways in which an emergency can occur and be responded to.

5.7.2. Semantics, Symbols and Implementation.

Pylyshyn's solution is that desires to obtain help, and other intentional phenomena, are not literally causes of behaviour. The causes of behaviour are physically instantiated internal representations of the phenomena which they represent, consisting of symbolic expressions in an internal code which is realized in some physical substrate. Thus, on Pylyshyn's account, scientifically respectable psychological explanation requires a tri-level explanatory hierarchy. There is the semantic characterization of an event, there is the symbolic representation of that event which mirrors all the relevant aspects of the semantic characterization, and there is the realization of the symbolic representation in a particular physical system which is causally efficacious.

Pylyshyn claims that "the classical view" of computing assumes that computer systems also have three distinct levels of organization (Pylyshyn 1989, p.57). These are the semantic level, the symbol level and the physical (or biological) level. The semantic level explains why appropriately programmed computers do what they do, by appealing to the contents of their data structures, the symbol level describes the structures in which the system's knowledge is encoded, and the physical level explains the system as a physical rather than a functional object. Pylyshyn explicitly

associates the semantic level with the knowledge level of Newell (1982)[19]. The levels of computer systems thus described can be seen to map neatly onto the explanatory requirements of materialist theories of cognition, and Pylyshyn argues that cognitive science should be based on the strong hypothesis that minds literally are computational systems of the kind outlined above. Clark (1989) has described such systems, in which there is a close mapping between the entities found at the top level of analysis and their syntactic computational analogues, as "semantically transparent".

Given such a tri-level system, it is appropriate to ask how the symbol level is organized because this level is underdetermined from the point of view of cognitive theory. The semantic level characterizes behaviour in terms of reasons, goals etc., and the physical, or implementation, level, characterizes behaviour in terms of neural impulses, and so forth, which will be mapped onto computational states at the symbol level via an instantiation function. In one sense then both of these system levels are satisfactorily constrained, at least in principle, the semantic level via our everyday mentalistic talk, the implementation level by what is known about the physical medium of the brain. Pylyshyn claims that the symbol level must be organized as a Turing machine if it is to provide the appropriate linkage between the semantic and implementation levels.

---

[19]Foster (1990) has argued that this is confused because there is a substantial difference between the knowledge level and the semantic level.

> "Whereas Newell is very careful to separate the knowledge level view of a system in terms of beliefs, goals and the principle of rationality, all implemented somehow (but we do not want to say how), Pylyshyn's view relies crucially on those distinctions being reflected at the syntactic level."
>
> Foster (1990, p.28)

I think Foster is wrong to distinguish the semantic and knowledge levels in this way for two reasons; first, although Pylyshyn certainly does claim that semantic distinctions are mirrored at the finest grain in the symbolic structures encoding them at the syntactic level, he is also quite clear that the semantic level is an autonomous system level, constrained by a principle of rationality (Pylyshyn 1984, p.38). Second, although Newell certainly characterizes the knowledge level as a distinct system level, he is quite clear that a body of knowledge constituting the knowledge level of a system is realized by representations existing at the symbol level (Newell 1982, p.100; 1990, pps 78-80).

Like Newell, Pylyshyn takes the behavioural plasticity of the universal Turing machine to be one of the important indicators that cognitive processing may be a form of computation. "This extreme plasticity in behavior is one reason why computers have been viewed all along as artifacts possibly capable of exhibiting intelligence." Pylyshyn (1984, p.53). For this reason he is committed to the view that the cognitive computer is computationally universal and hence must be a Turing machine.

Pylyshyn stresses the importance being able to attach semantic interpretations to symbols. "This quality of symbols and of computational states, whereby they can consistently be given a semantic interpretation, is not the only thing that makes useful computation possible; but it is one of the most important characteristics shared by computation and cognition." Pylyshyn (1984, p.63). The point that he wants to make is that we can see from the example of computers how the semantic characteristics of a program can be translated into effective symbolic processes which, as it were, shadow the semantic characteristics so as to produce transformations of their inputs which produce interpretable outputs respecting the semantics of the function which the symbolic processes execute. The reason why this is important is that "Although computation may not be the only way to realize a semantically described process, it is the only one we know how to achieve in a physical system. Put another way, nobody has any idea how it might be possible, even in principle, to build a system whose behavior is characterized in terms of semantic rules without first describing the system in terms of operations on symbol structures." Pylyshyn (1984, p.63) This is a further reason for thinking of the mental computer as a Turing machine.

Pylyshyn develops two more detailed arguments about the need to treat cognitive states as symbolic expressions on the tape of the brain's Turing machine. The first is intended to establish that cognitive states must be implemented as symbolic expressions rather than as sequences of state transitions. The argument is based on an analysis of the following putative cognitive rules;

> R1: If you *perceive* an event *E* as an instance of the category "emergency" (call this cognitive state S_1), then create the goal to get help (call this S_2), and
>
> R2: If you have the *goal* of getting help, and you *know* that help can be obtained by telephone (call this combined state S_4), then create the goal of locating a telephone (resulting in state S_5).
>
> Pylyshyn (1984, p.63)

The question which arises is how to represent the cognitive states S_1, S_2, S_4, S_5. Pylyshyn makes two points about the rules connecting these cognitive states which bear on the question; first he notes that we describe R1 as holding between S_1 and S_2 by referring to what these two states represent. Second he notes that single cognitive states can have complex contents in the sense of having distinct components. S_4, for example, is a combined goal and knowledge state. Pylyshyn uses these two points to argue that we cannot represent the two rules computationally in terms of transitions between two internal states as follows;

R1: S_1 --> S_2
R2: S_4 --> S_5

The reason why state transitions are unsatisfactory according to Pylyshyn is that internal states are atomic entities which cannot have constituent components. Thus they are unable to reflect the fact that the rules depend on the constituent components of the cognitive states.

> "For instance, R2 must refer to the S_2 component of the S_4 state in a way that identifies it as the same representational content that appears in R1. Thus, R2 cannot be expressed simply as a rule between S_4 and S_5; the inclusion of two distinct parts of S_4 must be explicitly represented in R2, and one part must be identified as being identical to the consequent state of rule R1."
>
> Pylyshyn (1984, p.64)

All these distinctions and sub-distinctions as well as others must be represented, says Pylyshyn "by some functional feature of the state." and this, he claims,

demonstrates the need for symbolic expressions. One shortcoming of this argument is that it rests on the assumption that a given cognitive state, say S_1, has to be represented by a single state of an automaton. This is an assumption for which there does not seem to be any justification. The particular difficulty in Pylyshyn's example is that the antecedent of R2 is a conjunction of terms one of which relates to the consequent of R1. This means that R2 cannot be specified in terms of a single state transition but it does not preclude a more complex state transition treatment. What is needed is something like the following;

> R2: If you have the goal of getting help (call this S_2), then create the goal of searching your knowledge base for a means of obtaining help (call this S_3), and if you are searching in S_3 and you find "telephone call" as a means of obtaining help then terminate the search (call this S_4), and create the goal of locating a telephone (resulting in state S_5).

This elaboration of R2 yields the following set of state transitions expressing rules R1 and R2.

> R1: S_1 --> S_2
> R2: S_2 --> S_3; S_3 --> S_4; S_4 --> S_5

In effect, R2 is rather like one of Turing's *m*-functions. One might, for example, think of the f-unit of Chapter 3 as implementing the cognitive state "Find the first instance of α" where "α" is the symbolic parameter to the *m*-function. A cognitive state might be taken to consist of a related set of internal states of this kind. There is a starting state and one or more final states connected by a chain of other states of arbitrary length and complexity. In a similar way, it would be possible to add functionality to maintain the other internal distinctions which Pylyshyn says are important, such as the fact that S_1 is a state resulting from perception rather than from inference. It seems therefore that even if Pylyshyn is right to claim that a single internal state cannot adequately represent a cognitive state with complex content, he is not justified in concluding that structured sets of internal states will not suffice.

Pylyshyn's second argument for the need for symbolic expressions is based on the potentially unbounded generativity of certain cognitive capacities, language being an obvious example, and thinking also, if one accepts something like Fodor's (1975) view of the relation between thinking and language. Pylyshyn argues that if a finite characterization of the processing underlying such capacities is to be achieved then they must be described in terms of operations on formal expressions rather than as sequences of state transitions. This is a much sounder argument. As the analysis of Chapter 2 showed, the processing of strings of symbols generated by recursive, specification, such as the parenthesis strings processed by the machine M, can only be achieved in general by a Turing machine.

However, it is not clear that Pylyshyn has established his claim that the *brain* must be organized as a Turing machine. Like Newell, and like many other generic theorists, he simply assumes that the brain is the exclusive locus of cognitive computation and that universality is the basis of cognitive flexibility. Neither of these points is argued for. What Pylyshyn clearly recognizes as a problem, however, in a way that Newell seems not to have done, is the way in which his theory isolates the computational processing of the cognitive computer. The point is that because the cognitive computer is hypothesized to be part of the brain and a universal machine, there is a need for input systems to translate external stimuli into the neural symbolic codes used by the cognitive computer and a need for output systems to translate the codes output by the computer into suitable behaviour.

The task of characterizing input systems is demanding because, as Pylyshyn acknowledges, there are semantic constraints on the functioning of these non-cognitive mechanisms which he calls transducers. "Because the output is to serve as the basis for the only contact the cognitive system ever has with the environment, it should provide all (and only) cognitively effective information. It should not provide information that could never serve as the basis for a cognitive distinction...On the other hand, the output must provide the basis for all potential distinctions that could show up in cognitive phenomena...Consequently, the output of the set of transducers available to an organism *must preserve all distinctions*

present in the environmental stimulation that are also relevant to the explanation of some behavioral regularity." Pylyshyn (1984, p.158).

As a result of these constraints, Pylyshyn's version of the generic theory, and by implication all other versions as well, face what appears to be an unanswerable question about how the transducers which are deemed to be non-cognitive, are supposed to determine the cognitive relevance or otherwise of their input. The problem appears to be of the same order of difficulty as the frame problem in Artificial Intelligence and it is far from clear that there is a solution to the frame problem. On the assumption that a computational architecture for cognition must be structured as Pylyshyn suggests, however, there is little option other than to accept the challenge and try to live with it. Pylyshyn is disarmingly frank about the nature of the problem, and acknowledges a number of constraints arising from transducers being considered parts of the functional architecture. These include the constraints that the input to a transducer must be stated in the language of physics and that the output of a transducer must be an atomic symbol or at best an n-tuple of symbols.

It is evidence of the grip which the generic theory exerts that Pylyshyn does not consider his own arguments to constitute a reductio of his position, given that a clear way to avoid the requirement for transducers is to abandon the idea that cognitive representations are internal symbolic encodings like the symbolic expressions on the tape of a Turing machine or in the memory of a computer. There are at least two reasons for favouring this move. The first, as Chapter 3 showed, is that it is consonant with Turing's analysis of computation and the second is that transducers seem to have an impossible job to do. The major cost of making the move is that it is no longer possible to construe the relation between mind and brain as like that between the program of a computer and its hardware. In particular, it requires the abandonment of the idea that the brain instantiates a universal computer.

5.8. Combinatorial syntax and structure sensitive processes.

The final arguments examined in this chapter are some of those used by Fodor & Pylyshyn (1988). Although their paper was designed as a critique of connectionism, it is useful for the light it sheds on the commitments and assumptions of certain aspects of the generic theory which Fodor and Pylyshyn refer to as the "Classical" approach to cognitive computation. Classical models of the mind, say Fodor and Pylyshyn, are minimally committed to the structures of Turing machines and von Neumann style computers, in the sense that "the kind of computing that is relevant to understanding cognition involves operations on symbols" (Fodor & Pylyshyn 1988, p.4). Fodor & Pylyshyn claim that classical computational models are "Representationalist", because they propose that "there are states of the mind which function to encode states of the world" (Fodor & Pylyshyn 1988, p.7). To present matters in this way, is already to finesse at least one of the issues of substance, which is the question whether the mind is in the head or involves external objects in some relation other than encoding. There is characteristically no argument for this central point which is simply assumed. It is taken to be the hallmark of a cognitive analysis that it should appeal to (internal) representations. "...any level at which states of the system are taken to encode properties of the world counts as a *cognitive* level; and no other levels do." (Fodor & Pylyshyn 1988, p.9).

Classical computationalism is also fearlessly materialist. "...neurons implement *all* cognitive processes... by supporting the basic operations that are required for symbol-processing." (Fodor & Pylyshyn 1988, p.11). Classical theories are distinguished by their commitment to "complex" mental representations or "symbol structures" and to "structure sensitivity of processes". The point of the latter claim is to establish that classical mental operations apply to mental representations by virtue of their form. A paradigmatic mental process maps classes of representations to other classes, with classes being determined by their abstract structural descriptions. Hence an operation which infers a single term from its appearance as the antecedent of a conjunction could infer either or both of, P from P & Q and (A v B v C) from (A v B v C) & (D v E v F), by virtue of the structure sensitivity of the process. Fodor and Pylyshyn are at pains to point out the seriousness of their

commitment to the physical realization of these characteristics of the classical architecture;

> "the symbol structures in a Classical model are assumed to correspond to real physical structures in the brain and the *combinatorial structure* of a representation is supposed to have a counterpart in structural relations among physical properties of the brain... the Classical theory is committed not only to there being a system of physically instantiated symbols, but also to the claim that the physical properties onto which the structure of the symbols is mapped *are the very properties that cause the system to behave as it does*....a Classical model is very different from one in which behavior is caused by mechanisms, such as energy minimization, that are not reponsive to the physical encoding of the structure of representations."
>
> Fodor & Pylyshyn (1988, p.13-14)

In a long footnote (note 9, pp.13-14), Fodor and Pylyshyn examine the conditions which have to hold on a mapping from expressions to brain states, if the causal relations among the latter are to reflect the structural relations among the former. For the scheme to work properly, there have to be two components; "(a) the definition of a primitive mapping from atomic symbols to relatively elementary physical states", and "(b) a specification of how the structure of complex expressions maps onto the structure of relatively complex or composite physical states." The latter specification will most likely be formulated in terms of a recursive mapping, since arbitrarily complex expressions must be dealt with.

A puzzling feature of this analysis, and one which indicates a degree of carelessness in its formulation, is the claim that the mapping discussed is an "instantiation function" as defined by Pylyshyn (1984) and discussed above. This claim cannot be correct given the nature of the instantiation function in Pylyshyn (1984). The mapping discussed by Fodor and Pylyshyn (1988) is a mapping from symbolic expressions to physical states. This sort of mapping must, in general, be one to many (and thus, mathematically, not a function), if it is to do a philosophically respectable job, because there is, in principle, no limit to the kinds of physical stuff which might instantiate a symbolic expression (silicon, beer cans, regiments of

Chinese etc.), and also no requirement that different tokens of the same symbolic expression be realized in the same way on different occasions (cf. Fodor 1975). By contrast, the "instantiation function" of Pylyshyn (1984) is a mapping from "an equivalence class of physical states" to "a computational state of the machine" (Pylyshyn 1984, p.56). This is a many to one mapping and is properly described as a function.

The task Fodor and Pylyshyn set for themselves in imposing constraints on the physical medium with the mapping they discuss, is very different from the task of specifying computational states by aggregating physical states of a given medium via an instantiation function. In the latter case one can at least choose mappings which make the best possible use of natural groupings of physical states for example. Trying to specify a mapping from computational states to physical states is very much more difficult. Numerous awkward decisions have to be made. The most important of these is the specification of atomic symbols. Unless these are known, the mapping cannot be got off the ground. The atoms have to be identified in order to get a feel for the problem of developing a mapping which will preserve a sufficiently high signal to noise ratio to provide stable computational performance. But even if that were possible the whole project seems ill considered. Unless you know something about the physical properties of the proposed medium in advance, there is no way to tell whether or not it will support the desired atoms in the right sort of way, and if you do know something about the medium then you will start from there.

Fodor and Pylyshyn do not acknowledge just how strong are the constraints on the medium which their theoretical position imposes. They readily dismiss "the (absurd) hypothesis that cognitive architectures are implemented in the brain in the same way as they are implemented on electronic computers" (Fodor & Pylyshyn 1988, p.55), while apparently retaining the requirement for arbitrarily deep logical structure which drove the development of just those computers. There is a striking irony in their claim that "The physical requirements of a Classical symbol-processing system are easily misunderstood." (Fodor & Pylyshyn 1988, p.57).

Nevertheless, Fodor and Pylyshyn offer a variety of arguments for the classical model. The first of these is that thought is productive in the sense that the representational powers of the cognitive system are unbounded under appropriate idealization (Fodor & Pylyshyn 1988, pps.33-37). Since this unbounded expressive capability is produced by finite means, we must treat the system of representations as expressions belonging to a generated set. Hence they have combinatorial structure, and thus imply a symbol system. Fodor and Pylyshyn conclude from this that "the mind cannot be a PDP" (p.33), because "Connectionist architectures cannot, by their very nature, support an expandable memory, so they cannot support productive cognitive capacities." (p.35). This is very much like the second of Pylyshyn's arguments discussed above. If mind is known to be implemented solely in the brain, then Fodor and Pylyshyn have a case. But Fodor and Pylyshyn need, and do not provide, an argument to the effect that the brain is the sole locus of the mind. The productivity argument alone does not suffice.

Fodor and Pylyshyn's second argument is based on the notion that cognitive representation is systematic. The examples they use are linguistic, and the point that they want to make is that the ability to deal with some sentences is intrinsically connected with the ability to deal with other sentences, by virtue of structural commonalities in their form. Thus, they say, the ability to understand the sentence "John loves the girl" entails and is entailed by the ability to understand the sentence "The girl loves John". Systematicity is thus a property of mastery of the syntax of a language. The point of the argument is that

> "*thought is systematic too*, so there is a precisely parallel argument from the systematicity of thought to syntactic and semantic structure in mental representations."
>
> Fodor & Pylyshyn (1988, p.39)

Just as you don't, apparently, find people who can understand "John loves the girl" but not "The girl loves John", so too you don't, apparently, find people who can think the thought that "John loves the girl", but not the thought that "The girl loves John". "...systematicity arguments infer the internal structure of mental

representations from the patent fact that nobody has a *punctate* intellectual competence." (Fodor & Pylyshyn 1988, p.40).

This argument, and two further arguments have a similar form. Fodor and Pylyshyn claim that cognitive competences are all fundamentally systematic and that the only way to explain this systematicity is to appeal to processes which are structure sensitive.

> "Cognitive capacities come in structurally related clusters; their systematicity is pervasive. All the evidence suggests that *punctate minds can't happen*... The only mechanism that is known to be able to produce pervasive systematicity is Classical architecture."
>
> Fodor & Pylyshyn (1988, pps.49-50)

There are various reasons for caution about this conclusion. First, it is notoriously difficult to argue from task performance to a mechanism. Fodor has made the point himself; "there are, in general, lots of mechanisms that can perform a given task, so that inferences from a task to a mechanism are up to their ears in affirmation of the consequent" (Fodor 1990, p.207). Thus, even if Fodor and Pylyshyn are right with their systematicity claims, they may still be wrong in their mechanistic assertions. The neuropsychological literature is a ready source of findings which suggest that the systematicity of normally functioning cognitive capacities, such as it is, is the result of the simultaneous activation of mechanisms which have very little in common with the, admittedly very sparse, outline provided by Fodor and Pylyshyn (cf. Walsh 1978; Shallice 1988). In the face of an increasing amount of evidence, it is also surprising to find Fodor and Pylyshyn claiming that "our knowledge of how cognitive processes might be mapped onto brain tissue remains very nearly nonexistent" (Fodor & Pylyshyn 1988, p.57).

Like the arguments proposed by Newell and by Pylyshyn, those of Fodor and Pylyshyn take as their starting point the view that the brain is organized as a Turing machine. Many other examples of the same assumption can be found in the literature. Because the starting point is an assumption, it is not explicitly argued for

and is thus difficult to attack, other than indirectly by pointing to the problems it poses and to its departure from Turing's analysis. An alternative is to develop a different account. The foundations of such an account are presented in the next chapter.

Chapter 6. The External Tape Hypothesis.

The starting point for the External Tape Hypothesis is the fundamental claim that the Turing machine was not, and was not intended to be, simply a model of the inner workings of the mind. It was a model of a system of states of mind interacting with external symbols. In consequence, the Turing machine model does not imply, much less entail, that cognitive processes must involve manipulation of *internal* symbolic expressions. Turing gave no indication that internal states would or should be constituted in such a way as to include symbolic expressions as parts. The different functions of tape and control may require entirely different types of explanation and principles of organization. The definitions of both states of mind and symbols were pared down by Turing to the barest essentials but there is no doubt about their relative locations, the one within and the other without the organism. The Turing machine is thus a micro-world in which a minimal mind interacts with a minimal environment. What is profoundly interesting is that such a spartan system can nevertheless be configured as a universal process simulator. What is also clear is that such exiguous notions of state and symbol need to be fleshed out if they are to ground a theory of cognitive architecture which locates the mind in the brain and in the world. The preceding chapters of the thesis have argued for a number of points which demonstrate the plausibility of the ETH as an alternative model of cognitive computation. These points are summarized below;

1) Turing's analysis of computation, described in Chapter 3, is based on a principled distinction between the finite state control of a Turing machine which models the mind of a human computer, and the tape which models the paper on which the human writes.
2) The generic computer theory of mind takes the brain to implement a whole Turing machine rather than just the finite state control. In this respect it contradicts Turing's analysis.
3) The causal nexus of the Turing machine is the configuration, which is a combination of the current internal state and the currently scanned symbol. In terms of definition 2.1 of Chapter 2, the internal state and the symbol

scanned are the two parameters q, and a, of the transition function $\delta(q,a)$. These two parameters enable a deterministic system to be capable of more than one response to a single input. Multiple internal states enable multiple responses to single inputs. The ETH proposes a model of cognitive computation which is faithful to Turing's analysis and locates the internal state, parameter q, inside the organism and the symbol, parameter a, outside the organism. The ETH is thus committed to a view of cognitive states as involving aspects of the external environment because cognitive states are identified with the computational configurations of the system. The existence of multiple internal states rather than autonomous internal symbol structures is hypothesized to provide the behavioural flexibility of the system.

4) Identifying the brain with the finite state control of a Turing machine implies that human memory must be treated predominantly as control memory rather than as tape memory. Two forms of control memory, functional and positional were identified in chapters 2 and 3 and some indication of their potential was given.

5) If the brain is a finite state control machine, then the flexibility and responsiveness of the cognitive system does not result from its being structured as a universal machine. It is more likely that the cognitive system consists of a large variety of interacting, special purpose mechanisms.

6) Using Pylyshyn's notion of an "instantiation function" which is a mapping from the physical states of a system to its computational states, it is clear that instantiation functions might differ with respect to the extent that intrinsic properties of a medium function to implement computational properties. An extremely efficient implementation might use almost all the intrinsic properties of a medium. Knowledge of these properties would therefore be essential to a proper understanding of the computational properties of the system as a whole. It is quite possible that the computational properties of the brain are very closely related to its physical properties. There seem, at least, to be good reasons for doubting that the brain instantiates binary primitives in the way that computers do.

The central claim of the ETH is, in one sense, entirely traditional, but in another sense quite radical. It is traditional in that it claims like the generic theory that cognitive states should be understood as the configurations of Turing machines. It is radical because it denies that the brain alone implements the cognitive Turing machine. This gives the claim that cognitive states are configurations a completely different character. In particular, it means that aspects of the external environment enter into cognitive states.

To develop the ETH there is a need for theories of various kinds to put flesh on the formal bones of Turing's analysis. First, there is a need for a theory of internal states to understand how such states can be implemented and to explore the relations between structure and function in systems of states such as the control automata of Turing machines where there is no separation between structure and function of the kind found in universal machines and digital computers. Part of such a theory will be the development of accounts of the different types of control memory which were introduced in Chapters 2 and 3.

Second there is a need for a theory of symbols to determine what kinds of entities other than conventional alphanumeric tokens can serve as the symbolic components of causal configurations. This theory must include an account of locations and of the conditions of stability and permanence which characterize conventional computational symbols and will need to have counterparts in theories of external computational objects.

Third, there is a need for a theory of computational configurations to tackle the enormous complexity of the systems of interaction which exist between external entities and internal states and among sets of internal states. Approaches to automata theory like that of Rosenschein (1985), can contribute to this enterprise, and the ecological approach pioneered by Gibson (1966,1979) is also likely to provide a source of insights. Because the human organism is essentially mobile, another requirement is an understanding of the significance of movement.

6.1. Developing an account of internal states.

Turing distinguished what he called automatic machines, whose behaviour was completely determined, from those he called choice machines, whose next moves were only partly determined by their current configuration and would need to be determined by a choice made by an external operator. The machines he discussed in the 1936-7 paper were all automatic machines as are modern computers. Haugeland (1981) also describes computers as automatic formal systems, but uses the notion of automaticity somewhat differently from Turing. For Turing a machine was automatic if and only if it was completely deterministic. For Haugeland, a machine is automatic provided it can proceed without external intervention. A non-deterministic machine, plus a set of heuristics to do the choosing when necessary, is automatic in his terms.

For present purposes the crucial point is to see that a deterministic procedure need not be carried out automatically, using the latter term in Haugeland's sense. There is an important distinction to be drawn between determinism and automaticity. This distinction highlights a difference between Turing machines and digital computers on the one hand and people on the other which is entirely obvious but worth stating because it affects the conception of an internal state which is needed for the ETH. The difference is that the behaviour of computers and Turing machines is both deterministic and automatic whereas the behaviour of humans even when computing deterministic procedures is rarely, if ever, automatic, because it involves operations of the will. For instance, if I were to ask you to carry out, with paper and pencil, the multiplication of 284613 by 18737265, you would no doubt be able to do so using a deterministic algorithm which you learned as a child. But the deterministic nature of the algorithm would not make your behaviour automatic if only because you could decide to stop and leave the computation unfinished at any time. Further, the involvement of the will in human action is not dispensable from a psychological point of view. Cognitive and conative processes are intrinsically connected.

The issue of automaticity links with the question of whether the brain is a finite automaton or a Turing machine in an interesting and important way. If the brain

is in fact a whole Turing machine, then automaticity at the level of global state transitions is a desirable feature because the brain is an automatic system over whose operation we have relatively little conscious control, and global state transitions, on this model, are internal operations of the brain. Further, it is the automaticity of the brain viewed as a computer which protects the generic theory from the need for homunculi. If, however, the brain is not a whole Turing machine but the finite control system only, then automaticity at the level of global transitions is an embarrassment, for the sorts of simple reasons outlined in the previous paragraph, viz. that humans are able to stop and start deterministic computations at will. Volitional, motivational and affective phenomena must be incorporated in accounts of the aetiology of behaviour at such a global level of analysis, if the account is to be psychologically plausible.

What seems to follow is that when states of mind are thought of as global, as Turing intended, rather than as internal relations between control processes and symbol structures in the brain, a psychologically adequate characterization of state transitions will have to acknowledge a range of degrees of automaticity and will have to include accounts of attentional and motivational processes where automaticity is not assumed. These requirements, allied to the experientially obvious but theoretically obscure phenomena of consciousness suggest that a full account of the notion of internal state will need a great deal of development.

The ETH is therefore committed to an understanding of the notion of "internal state" which spans a wide range of descriptive phenomena at a variety of levels of abstraction. At the level of abstract analysis, the notion is Turing's notion of a "state of mind" which is global and unitary. At the level of everyday psychological description the "internal state" is a complex assemblage of, for example, attentional, motivational, affective and rational phenomena which are likely to depend on the interacting operations of multiple mechanisms. The question is whether these different levels of analysis can be brought together in a way which preserves Turing's state notion. What is required is an analysis which retains the abstract unity of the notion of internal state while allowing that the realized, physical state

may be complex. As a simple example of the required contrasts consider the following different ways of describing the states of a car. Abstractly we might describe the car as moving forward in first gear or second gear, as moving in reverse, as stationary etc. These global descriptions correspond to a description of a Turing machine as being in such and such a functional state. When we come to describe what implements the state of "moving forward in first gear", however, it is clear that a complex description is needed of the interactions of the various parts of the vehicle which contribute to this state. The states of the fuel system, of the electrical system, of the cylinders, of the gearbox, of the wheels, etc. all constitute elements of the global description.

6.1.1. Gandy's principles for mechanisms.

Typically, when thinking of Turing machines, one tends to think of their global states as being irreducibly unitary and thus as unsuitable vehicles for the structural complexities which are clearly constitutive of real cognitive systems. However, it was apparent from the discussion in Chapters 2 & 3, that it is possible to think of the state diagram of a Turing machine as something like a logical blueprint for a system of parts which could implement the set of transitions constituting the machine. Gandy (1980) has made an important theoretical advance by describing, in very general terms, the constraints which such complex structures composed of multiple, parallel subassemblies must meet if sequences of their states are to implement Turing computable functions.

Using the formalism of hereditarily finite sets, Barwise (1975), Gandy proposed four principles which he proved had to be satisfied by any discrete deterministic device if its successive states were to form a computable sequence. In order to describe a machine, one starts by assigning labels to represent the basic components and their states. "We suppose that labels are chosen for the various parts of the machine -- e.g., for the teeth of cog wheels, for a transistor and its electrodes, for the beads and wires of an abacus. Labels may also be used for positions in space (e.g., for squares of the tape of a Turing machine) and for physical attributes (e.g., the color of a bead, the state of a transistor, the symbol on a square)." Gandy (1980, p.127).

Functional aggregates of components, which constitute complex parts, are described by forming a hierarchy of sets of labels. The global state of a machine is constituted by the set of sets thus formed and state transitions are defined in terms of a "structural function" which describes "the transition between physical states with some persistent elements." Gandy (1980, p.129).

The four principles for mechanisms describe and constrain the hierarchical systems which can be constructed in this way. The first principle states that any machine can be described by giving a structural set $S_M \subseteq$ HF of state-descriptions, together with a structural transition function, F: $S_M \rightarrow S_M$, which describes the state transitions of the machine. The importance of the structural function is that it applies to potentially complex structures with parts, and allows the definition of transitions in which some parts persist, while others are changed. In this way the joint seriality and parallelism of a process can be described. The process is serial because the structural function maps the global states of a structure at successive moments, and it is parallel in that it describes potentially multiple simultaneous local changes in the structure which occur in the space of a single state transition.

The other principles place restrictions on the set of state-descriptions and on the transition function. The second principle, called "The Principle of Limitation of Hierarchy", requires that there be an upper bound on the set-theoretic rank of the state-descriptions, i.e. that the number of levels of descriptive structure must be finite. This means that there must be a finite limit to the nesting of parts within parts, therefore that the parts of a machine may not be infinitely complex.

The third principle called "The Principle of Unique Reassembly" states that any machine must be constructible from parts of bounded size which can be labelled so that there is a unique way of putting them together. This prevents the specification of an infinitely large machine in a way which is not forbidden by the second principle, i.e. by having infinitely large rather than infinitely complicated parts. Thus, for example, a gear wheel with an arbitrarily large number of teeth would be disallowed, as would a lever of unbounded length. The principle also enshrines the

physical fact that the indentity of an object is a function of the arrangement, rather than the individual identities, of its parts.

> "According to quantum mechanics... any two electrons must necessarily be completely identical, and the same holds... for any two particles whatever, of any one particular kind... What distinguishes the person... is the *pattern* of how his constituents are arranged, not the individuality of the constituents themselves."
>
> Penrose (1990, p.32)

The final principle, "The Principle of Local Causality" is the most important. In Turing's analysis of computation, the next state of a machine was determined solely by its current state, and the current scanned symbol. Turing justified this restriction by appealing to the finite perceptual capabilities of a human calculator. "We may suppose that there is a bound B to the number of symbols or squares which the computer can observe at one moment. If he wishes to observe more, he must use successive observations." (Turing 1936-7, p.136). This is an instance of the dependence on human capacities which Gandy's analysis seeks to replace. The justification for the principle of local causality which replaces the perceptual bound, "lies in the finite velocity of propagation of effects and signals: contemporary physics rejects the possibility of instantaneous action at a distance" Gandy 1980, p.135). Newell (1990) bases an argument for distal access on the same physical considerations. The principle of local causality requires that the next state Fx of a machine can be assembled from a set of overlapping regions, each of which is produced by a "causal neighbourhood" of x which is of bounded size. This amounts to the claim that the causal whole is the sum of its causal parts. A helpful example of the principle in action is the way in which the global states of a connectionist network are computed. Each node in a network, except in the case of a fully connected net, takes its inputs from one subset of the other nodes and delivers its output to another subset of those nodes. The activation level of a node, and the weights on its connections to its neighbours are all determined locally. Hence, the next state of any node in a network can be computed by considering a restricted "causal neighbourhood" of other nodes and connections, and the next state of the whole network can be arrived at by sequential consideration of the set of such causal

neighbourhoods. If this were not so, the simulation of connectionist networks on serial digital computers would not be possible. The principle of local causality could also be taken to imply the validity of orthodox scientific tactics which work towards an understanding of the whole of a complex structure or process via piecemeal study of its parts.

An important general point about the principles is the following; 'It is perhaps worth emphasizing how unrestrictive the principles are. Unlike most automata and algorithms which have been proposed, our treatment does not depend on singling out any set of "elementary" operations.' Gandy (1980,p.145).

The unrestrictiveness of Gandy's principles has a number of consequences for cognitive theorizing. First they show that it cannot be demonstrated that cognitive processing has to involve explicit, internal symbolic expressions in order to be "computational". A system is computational if and only if it meets the requirements of Gandy's four principles and these do not specify that explicit, internal symbolic expressions are required. Symbolic expressions will often be convenient, relatively easy to implement, etc. but they are not necessary. A connectionist network is an example of a type of computational system which meets the requirements without involving structured expressions. This calls into question certain arguments to the contrary. Pylyshyn (1984), for example, has argued that explicit symbols are constitutive of computation and hence must be involved in cognition if it is indeed a kind of computation. "...so long as we view cognition as computing in any sense, we must view it as *computing over symbols*. No connectionist device, however complex, will do, nor will any analog computer;" Pylyshyn (1984, p.74). Gandy's position may be thought somewhat more secure because he provides proofs for the four principles whereas Pylyshyn's argument about computation is informal and intermingled with ideas about the requirements for cognitive explanation.

Second the unrestrictiveness of the principles seems to show that connectionism will not rewrite the bounds of the computable as suggested, for example, by Smolensky (1988). Although Gandy explicitly intended the principles to cover discrete devices

only, his interpretation of what constitutes a discrete device is liberal. 'In principle the state of any concrete device can be adequately described by specifying, to a certain degree of approximation, the relevant physical parameters (chemical composition, pressure, current flow and so on) of sufficiently small regions of space. (By using the words "concrete", "mechanical" and the like, we intend that these regions will always be very much larger than the size of an atom).' Gandy (1980,p.130) The approximations to continuous systems provided by the computer realizations of connectionist nets seem to fall within the scope of Gandy's principles and thus not to transcend the boundaries of Turing computable functions.

The unrestrictiveness of the principles, taken in conjunction with the fact that the brain, in Turing's scheme realizes a finite state machine and not a Turing machine, suggests that the traditional abiological stance of cognitive science may need revision in order to encompass a satisfactory notion of internal state. There are two reasons for this. The first is that the principles suggest that there need not be a privileged symbol level at which distinctively "computational" operations take place. The second is that the identification of brain with a finite state machine suggests that there is no simple distinction to be made between program and hardware. Taken together, these two points suggest that a full understanding of cognitive processes will require consideration of a much wider range of levels and linkages between them than has traditionally been supposed. The principled distinctions which exist between the various levels of a computer system, for example, may simply not exist in the brain. It may be, for example, that a rather detailed account of the properties of neurons will be required to explain the computational capacities of the brain as a result of the efficiency of its instantiation function. To argue thus is not to claim that psychologists should stop doing psychology and start doing neuroscience. It is simply to claim that the principled separation of program from hardware which exists in computers and in universal Turing machines, and which ensures that, at a certain level of analysis, the program level is clearly separable from the hardware which executes it, need not be characteristic of the brain as a finite control machine.

6.1.2. Mead's analogue silicon modelling technique.

In chapter 5, brief mention was made of the work of Carver Mead, cf. Mead (1989) on the synthesis of analogue circuits in silicon. This work is relevant to developing an understanding of the way in which internal computational states might be realized in the brain. Consider first, by contrast with the generic theory, what kind of account is being sought. The discussion can usefully be based on Turing's distinction between states of mind, i.e. *m*-configurations, and states of the system, i.e. complete configurations. In the generic theory, the expressions "computational state" and "internal state" are generally both taken to refer to what Turing would have called complete configurations, i.e. combinations of the state of the internal Turing machine control and the state of the internal symbolic memory, and a change of state is generally thought of as involving a transformation of some symbolic entity in the latter part of the system. For the ETH, this is the wrong way to think about internal states, which should be thought of solely in terms of the *m*-configuration part which combines control and memory storage in the same set of states, as does a connectionist network. Thus, if one wants to understand the states of such a system it is not appropriate to ask what symbolic encodings it might contain. An investigation of the properties of its implementation medium is required, so as to see what sort of an instantiation function it might support and hence how it might support the complex physical interactions which constitute implementation of a set of logical states.

Mead's work can be seen as part of an attempt, based on physical principles, to understand the low level capacities of neural circuits which might be capable of supporting higher level, e.g. cognitive, phenomena. The basis of Mead's work is a conviction that a fundamental understanding of the performance of any efficient, complex system must be based on understanding how the intrinsic properties of the medium are exploited. One way to understand why this work is being done now and what its significance might be is to think in terms of research having come full circle since the beginning of the computer age with respect to the relation between mind and brain. The circle starts with Turing who took an important step towards separating the study of the logical functioning of the mind from the details of its

instantiation in the brain by abstracting the notion of a "state of mind" from the details of its implementation in the brain. Turing had almost nothing to say about the constitution of these states, apart from arguing that there could be only finitely many of them and that they could be organized systematically in the service of routine computation. The separation of function from substance allowed for the development of computers which performed logical, apparently mind-like tasks on the basis of clear organizational principles. These principles were synthesized and understood at a time when the kind of detailed knowledge of cortical micro-circuitry which is now current was still some way in the future. Much of the modern study of cortical circuitry depends on techniques developed since the late 1970's, Douglas & Martin (1990, p.433). The impressive speed and range of the new computers suggested that perhaps the mind was organized as a computer. Aspects of neural functioning, such as the action potential suggested a possible basis for binary coding in the brain which might function as the binary encoding at the basis of computer operation. The development of high level programming languages whose syntax more closely approximated that of natural languages than did the syntax of machine languages confirmed the possibility of a variety of levels of understanding of computer systems which were redolent with insights for those seeking to understand the relationship between brain and mind. The computer thus came to serve as a model of how the mind might be organized. Hence the generic theory. However, some forty years of experience has served to demonstrate that for most cognitive tasks, the computational power of the brain is orders of magnitude greater than that of the fastest supercomputers, even though the computational primitives of the latter are millions of times faster than neurons. Now the point has been reached where the brain is once again the focus of attention. If its computational processes can be understood the potential will be there for another explosion of growth in computer technology. As Mead demonstrates, the fabrication techniques for very large and wafer scale integration have reached a level of sophistication where it becomes possible to think of building systems which are neurally realistic with respect to both numbers of processing elements and their processing style.

From the point of view of the ETH, work of this kind must be of fundamental

importance in the long term for the development of cognitive theory. The brain, according to Turing's analysis, is a control automaton with an astronomically large but finite number of states. Most computer technology is not of assistance in understanding the structures and processing of such automata because, with rare exceptions, it has been dedicated to minimizing the control circuitry and concentrating on maximizing the internal memory space available. The point, of course, is that if the control is structured as a universal interpreter it is provably sufficient to compute any computable function and everything else can be done in software. The fundamental difference that thinking of the brain as a finite automaton makes is that structure and function are inseparable and cannot be separated in the way that software and hardware can be separated in a computer. Limited insights into the ways in which sets of states of an automaton might be aggregated to produce complex functionality can be obtained from the construction and study of large automata like Turing's original machine, but it is quite clear that study of the brain can be expected to yield increasingly important insights as both neuron staining techniques and techniques for live monitoring of brain function continue to improve. It is both sobering and exciting to realize just how limited have been the techniques with which current knowledge of the brain has been won and thus how much new progress can be expected with the techniques currently being developed. It is only since the 1980's, for example, with the advent of tracers like horseradish peroxidase that morphological and functional properties of single neurons in neocortex have been able to be identified simultaneously, Douglas & Martin (1990, p.391).

Mead's work is based on the following considerations. First that "neural systems evolved without the slightest notion of mathematics or engineering analysis...But evolution had access to a vast array of physical phenomena that implemented important functions. It is evident that the resulting computational metaphor has a range of capabilities that exceeds by many orders of magnitude the capabilities of the most powerful digital computers." Mead (1989, p.5). The second point is the maturity of transistor fabrication technology as mentioned above. Mead's proposal is to use this technology, evolved for digital computation, as a modelling substrate

in the service of the study of the brain. The method is to develop a silicon nervous system.

> "The constraints on our analog silicon systems are similar to those on neural systems: wire is limited, power is precious, robustness and reliability are essential...The effectiveness of our approach will be in direct proportion to the attention we pay to the guiding biological metaphor...First we will describe the relevant aspects of neural wetware at the level of abstraction where we will be working. We will then develop the operations that are natural to silicon, and examine how they can be used to implement certain known neural functions." Mead (1989, pp.7-8).

Early results from the research programme have been encouraging and suggest that it is indeed a valuable method. The silicon retina (Mahowald & Mead, 1991) is an interesting example. Its performance is intriguing, particularly the fact that it is subject to some of the same illusions as the human visual system. Implicit support for a proposal like the ETH is quite evident;

> "The interplay of context and adaptation is a fundamental principle of the neural paradigm. It also imposes some interesting constraints on neurally inspired circuits... change is a necessity for neural systems... This requirement for change firmly situates a neural circuit in the world that it observes, in contrast to digital circuits, whose design implicitly assumes separation between the system and the outside world."
>
> Mahowald & Mead (1991, p.44)

6.1.3. Conrad's trade-off principle.

An analysis of some slightly different issues which are relevant to questions about the nature of the states of automata can be found in the work of Michael Conrad (1974, 1985, 1988). Conrad argues that there is a trade-off principle relating a variety of fundamental factors in computational systems;

> "*A system cannot at the same time be effectively programmable, amenable to evolution by variation and selection, and computationally efficient.* The von Neumann computer opts for programmability. The trade-off theorem suggests that an alternative domain of computing is in principle possible, where programmability is exchanged for efficiency and adaptability. Biological systems, as the products of evolution, must operate in this alternative domain."

Conrad (1985, pps. 464-465)

To understand the arguments for the trade-off principle it is crucial to understand the physical basis of the analysis. Information processing systems are "systems which dissipate energy in certain interesting (or highly selective) ways." (Conrad 1974, p.83), and Conrad maintains that programmable systems are essentially much less efficient than non-programmable systems, because of the way they use their physical resources.

The basic idea is quite simple but depends on a potentially controversial notion about what sorts of processes can be called computational. The standard answer is that the definitive notion of a computable process is the Turing machine. Conrad argues that this view is unjustifiably restrictive.

> "Turing machines... are particular models of computation. They are particularly useful as reference points for evaluating the amount of computational work performed by arbitrary dynamical systems, not as delimiting the class of behaviors admitted to be forms of computing."

Conrad (1988, p.287)

Conrad argues, on the basis of a strong interpretation of the Church-Turing thesis, that any physical process can properly be treated as a computational process.

> "...all physically realizable dynamics *are equivalent to computation* in that they can be simulated by a von Neumann computer under the idealization that space and time bounds can be ignored."

Conrad (1985, p.468)

The difference between a Turing machine and an arbitrary physical system, says Conrad, is not that the one computes and the other doesn't but that the former has a potentially universal simulation capability, i.e. is programmable, whereas the latter will usually have a very restricted simulation capability, i.e. is not programmable.

Conrad developed what he calls the "trade-off principle" to explore the differences between programmable and non-programmable computational systems. The principle can be used to show that programmability is purchased at the cost of efficiency. The reason for this cost is that programmability implies controllability, and exercising control is an energy intensive business. This is a fundamental aspect of physical processes generally. The internal combustion engine in a car, for example, converts only a fraction of the energy it consumes into motion. This is due, in part, to the large amount of energy required to control the sequence of explosions which delivers power to the wheels. Computation is also a physical process in real, as opposed to abstract, machines and Conrad's argument shows that controlling computational processes necessarily decreases the energy available for task specific performance just as control of any other physical process decreases the available task energy. This is clearly true of digital computers, and is evident in the large amount of heat which has to be dissipated when they are run. This aspect of the trade-off principle also explains the very notable differences in efficiency between universal machines and the machines they simulate. The description of a target machine on a universal machine's tape is a control process which constrains the subsequent behaviour of the universal interpreter.

Perhaps more surprising is an aspect of the trade-off principle which shows that programmability and evolutionary adaptability are incompatible. One might be inclined, intuitively, to suppose that the general purpose capacity which is conferred by programmability would be consistent with adaptability. Newell (1990) for example, argues strongly for this point. Conrad develops an argument which shows this is an unreliable intuition. He starts by arguing that for a system to be evolvable it must be capable of "accepting" at least one structural change. "A system accepts a structural change if its performance improves or if it is capable of lasting long enough to accept another change, eventually leading to an improvement." (Conrad 1988, p.294). The argument for this threshold condition rests on probabilistic analyses relating the time scale of evolutionary processes to the likelihood of multiple, simultaneous, structural changes. Conrad then argues, using a proof similar to Turing's proof of the unsolvability of the halting problem, that the

problem of ascertaining whether a programmable system meets this condition, is, in general, unsolvable, even when the criterion for an "acceptable" structural change is very weak. By contrast, there are structural changes which non-programmable systems can be shown to "accept" in the required sense. Thus at a minimum, "programmable systems are not as effectively structured for evolution as nonprogrammable systems." (Conrad 1988, p.293).

Conrad's arguments lead towards a conclusion which supports the ETH. The brain is clearly an evolved system, and hence likely to be non-programmable, i.e. it is unlikely to be organized like a digital computer or a universal Turing machine. This supposition is reinforced by that aspect of the trade-off principle which shows that it will also be more efficient if it is non-programmable. There is no reason, however, why the brain should not have evolved so as to include among its huge number of internal states, a set which functions as the finite state control of a universal machine, whose programs are to be found in the environment. This yields a system which is both efficient and responsive to its environment. Conrad draws much the same conclusion;

> "...it would be possible to build a structurally nonprogrammable computer that is nevertheless effectively programmable at an interpretive level. There is a model for such a machine: People can read and follow rules despite the fact that the human brain, as a product of evolution, must be structurally nonprogrammable."
>
> Conrad (1985, p.475)

6.1.4. Summary.

To summarize the discussion of internal states viewed from the perspective of the ETH, the following points are particularly noteworthy. Turing's analysis leaves open the question of how such states are implemented. The discussion of chapters 2 and 3 suggested that memory was implicit and positional in the control automata of Turing machines, so the hypothesis of the ETH is that these forms of memory will also be characteristic of the brain and some evidence was cited in chapter 3 which suggests that this is indeed so. Sets of states computing deterministic functions need

not be executed automatically. This gives rise to the requirement for an analysis of attentional, affective and motivational mechanisms as proper parts of internal states. Gandy's principles for mechanisms show how complex, structured systems of parts can function as "states" of Turing machines. Mead's work with analogue silicon systems suggests that the intrinsic properties of the brain may be fundamental to its computational functioning, and Conrad's trade-off principle suggests further reason for doubting that the brain implements a programmable computer of the kind we are familiar with. The conclusion to be drawn from this discussion is that the ETH argues for the relevance of neurobiology and other relatively low-level investigative disciplines to the construction of a proper understanding of internal states. It is important, however, to be aware that cognitive states are not identified with internal states in the ETH. Cognitive states are identified with the configurations, (and possibly complete configurations) of the organism-environment Turing machine. Thus it is entirely appropriate to argue both for the indispensability of neurobiology to the understanding of cognitive processes and also for the irreducibility of cognitive states to neural states. Cognitive states involve external symbolic entities because they are Turing machine configurations not *m*-configurations.

6.2. Developing an account of external symbols.

For an abstract Turing machine, the set Σ of symbols which constitutes its basic inputs and outputs is arrived at by definition and is generally thought of as a set of alphanumeric symbols. The alphabet may be small since it is known that all computable functions can be defined over the alphabet $\Sigma = \{0,1\}$. Binary alphabets have proven most suitable for digital computers because binary number systems and binary logic are both easily amenable to implementation in digital electronics. The essential formal property of symbols for computation is type identity, such that each token of a given symbol is unmistakably identifiable as such and unmistakably distinguishable from tokens of any other symbol. Conventional alphanumeric symbols are particularly clear exemplars since they have few other intrinsic properties, but there seems to be no reason to suppose that entities with properties in addition to type identity might not serve as computational primitives. Because Turing was investigating the computation of real numbers expressed in binary

notation, he was concerned exclusively with numerical symbols, but in consequence of his work it is clear that computation is a very general type of process which might be carried out over basic elements of many kinds provided, at a minimum, that they meet the type identity condition. Appropriate entities are those which can be reliably identified and paired with internal states to act as configurations which determine behaviour.

Because the ETH locates the "tape" of the cognitive Turing machine in the environment, its machine table must be taken to describe a system of organism - world interactions, rather than a system of interactions between states of the brain which function as symbolic representations of external events and objects, and states of the brain which process those representations. This poses an immediate problem. Organisms interact with a wide variety of external inputs and not just with conventional alphanumeric symbols. Indeed, it is likely that there is an indefinitely large possible set of inputs which might impinge on the human organism. However, a machine table is, by definition, a finite structure. Thus it is clear that if a machine table is to make reference to external entities, either these must be restricted to conventional symbols, or a way must be found to characterize the notion of symbol in a way which allows for the richness of organism-environment interactions, but at the same time imposes sufficient constraints for finite realization. Given that there is no principled reason why conventional symbols should be privileged, the latter option is preferable. One promising way to proceed is to think in terms of the relationship between the primitive symbols of a universal Turing machine and the encodings of target machine symbols constructed out of them. This suggests that perhaps there may be symbolic primitives among the inputs which humans receive which serve as the basis for more complex, probably learned, symbols. A clear place to look for these would be in the behavioural repertoires of neonates and young infants. If one were to observe systematic differences of response to specific classes of stimuli, at ages sufficiently early to making learning an improbable explanation, there would be grounds for suggesting that these classes of stimuli functioned as basic symbols.

Recent research in child development strongly suggests that babies are equipped at, or very shortly after, birth with a range of reactions to specific classes of stimuli. These results are in marked contrast to the prior Piagetian orthodoxy which argued that children at birth were equipped only with very general sensory and discriminative capacities. Neonates are particularly responsive to auditory stimuli within the frequency range of the human voice, and they show special sensitivity to human speech sounds, Aslin, Pisoni & Jusczyk (1983); Eimas, Miller & Jusczyk (1987). Infants also appear to be specifically equipped to recognize human faces. By two to three months when infants are sensitive to pattern structure they prefer a schematic face to scrambled arrangements of features, Maurer (1985). More recently, Johnson & Morton (1991), Morton & Johnson (1991) have shown that neonates preferentially attend to stimuli with a face like arrangement of elements. Johnson & Morton hypothesize that two neural mechanisms underlie this performance. The first, which they call "CONSPEC" is a sub-cortical mechanism which is functional at birth. The second, "CONLERN" is a cortical mechanism which takes input from "CONSPEC" and controls face recognition from about the age of two months. Being able to distinguish faces from non-faces reliably is logically the same sort of process as being able to distinguish '0's from '1's reliably, and there seems no reason to deny that faces can act as computational primitives given that binary digits can. From the point of view of the ETH, the combinations (FACE,CONSPEC) and (FACE,CONLERN) can be thought of as configurations of the cognitive Turing machine.

Pre-linguistic infants also seem to be sensitive to a distinction between animate and inanimate objects based on the different ways in which they move, Mandler & Bauer (1988), and Premack (1991) has argued that movement is the basis for a built in capacity to distinguish intentionality from causality. He argues that infants are hard-wired so as to divide objects into two classes, those which are self-propelled and those which are not. He suggests that the induced movements of non-self propelled objects are the basis for perceptions of causality and that the free movements of self-propelled objects are the basis of perceptions of intentionality. More generally, the infant's perception of objects appears to make use of a variety of built in

mechanisms which instantiate four principles, boundedness, rigidity, cohesion and no action at a distance, Spelke (1990); Karmiloff-Smith (1992, Chapter 3).

The famous visual cliff apparatus developed by Gibson & Walk (1960) also suggests that infants are sensitive to marked environmental discontinuities. The visual cliff was used to show that by the time babies can crawl they are sensitive to potentially dangerous drops and behave accordingly, and other types of studies, for example by Arterberry & Yonas (1988) indicate that kinetic depth perception is present at an even earlier stage, by three or four months of age. The perception of dangerous versus safe in terms of potential drops appears to have the logical qualities required for a primitive symbol.

6.3. Developing an account of configurations.

The accounts of states and symbols described above lead very naturally to a consideration of the interactions between them. Once the machine table of a Turing machine is seen as describing a system of organism-environment interactions, a new perspective on cognitive computation becomes possible. There is a profoundly suggestive analogy to be drawn between the complex interlocking of internal control states and symbol structures on the tape of a Turing machine on the one hand and the interlocking of organisms and their environments on the other. This analogy suggests that Turing's analysis of computation is an ideal basis for a formal understanding of human ecology. Turing's machine model shows that to understand the cognitive system requires understanding not just the organization of the brain but also the organization of the environment and the system of relations which constitutes the basis of interaction between brain states and environmental entities. The crucial generalization, for psychological as distinct from logical approaches to computation, is to broaden the scope from the treatment of conventional symbolic entities such as digits, which constitute the symbolic resources with which Turing was exclusively concerned, to any sets of entities which, when paired with sets of brain states to implement configurations, constitute the definitions of effective procedures. The point to press as the focal issue for the development of psychological theory is not the central logical feature of universality which Turing

demonstrated such that any clearly describable entity or process can be represented in conventional symbolic terms and simulated by a single machine, but the central ecological feature of polymorphism such that any clearly describable systems of relations between sets of type identifiable entities and states of the brain which result in systematic action can be considered as computations.

6.3.1. Gibson's concept of affordance.

J.J. Gibson developed an ecological theory of perception which has marked commonalities with the ETH. Gibson makes the central point that

> "*animal* and *environment* make an inseparable pair. Each term implies the other. No animal could exist without an environment surrounding it. Equally, although not so obvious, an environment implies an animal (or at least an organism) to be surrounded."
>
> Gibson (1979/1986, p.8)

Gibson's concept of "affordance" implies an involvement between external objects and organisms which is somewhat similar to the relation between an external symbol and an internal state which, from the perspective of the ETH, constitutes a computational configuration. Gibson described affordances as follows; "The *affordances* of the environment are what it *offers* the animal, what it *provides* or *furnishes*, either for good or ill...[It is] something that refers to both the environment and the animal in a way that no existing term does. It implies the complementarity of the animal and the environment." Gibson (1979/1986, p.127). The nature of the complementarity was further described as follows: "An affordance cuts across the dichotomy of subjective-objective and helps us to understand its inadequacy. It is equally a fact of the environment and a fact of behavior. It is both physical and psychical, yet neither. An affordance points both ways, to the environment and to the observer." Gibson (1979/1986, p.129).

It is a common misrepresentation of Gibson's claim to suggest that an affordance represents only one pole of the environment-subject complementarity. Part of the difficulty stems from Gibson's claim that affordances are directly perceived. This

is often taken as the claim that the perception of affordances involves no internal processing at all, a claim which many critics have found incoherent. But Gibson does not claim that direct perception involves no internal processing. "The inputs of the receptors have to be processed, of course, because they in themselves do not specify anything more than the anatomical units that are triggered." Gibson (1979/1986, p.251). The theory of direct perception is not a denial of internal processing but part of an attempt to try to understand why percepts are meaningful, and what Gibson wants to deny is that external stimuli are given meaning as a result of *information processing*. His claim is that meaningfulness is a consequence of a prior, fundamental, organizational principle such as adaptation, which shapes cognitive structures so as to relate environment and behaviour.

The problem of meaning is a difficult one for all computational accounts of cognition. For the generic theory, a central part of this difficulty is to understand how it is that internal, computational states can stand in the relations to external objects which they must have in order to maintain the activity of the organism, given that they are also supposed to be semi-autonomous. This problem does not arise for the ETH which maintains that external objects contribute directly to the computational configurations of the organism. The criticism that such direct involvement of the environment renders the organism totally stimulus bound was discussed in Chapter 3. It appears that the richness of the organism's set of internal states refutes this criticism. Understanding the meaningfulness of inputs is not, of course, explained by the direct involvement of external objects in cognitive states, but suggests that the beginnings of an account may be found in the evolutionary history of the species, and a determination of those aspects of environments which would have been particularly salient for our ancestors. For the present it is worth re-iterating one of the central points of Turing machine theory which is that the set K of internal states and the alphabet Σ of elementary symbols have, in a sense, to be made for each other. States and symbols "mean" what they do by virtue of the behaviour which they cause. A configuration is an interlocking pair. This is true not just of mono-functional Turing machines but also of universal machines although in the latter case, a higher order meaning is also given through the encoding

convention for states and symbols of the target machine.

6.3.2. Rosenschein's situated automata approach.

Gibson's approach to the relation between organism and environment was largely informal but a theorist who has given a prominent place to the interaction between machine and environment in a more formal account is Rosenschein (1985). Rosenschein was concerned with the concept of knowledge and with understanding what is being claimed when a machine is said to know a proposition. He shows that it is possible to give a formal account of knowledge for a system which does not contain interpreted symbolic expressions, by deriving a notion of the information content of the system's internal states. Rosenschein describes two ways of understanding claims to knowledge as two different mappings between the real world and a formal model of "knowledge" as an operator in an epistemic logic based on the modal logic S5, Hughes & Cresswell (1968). In the formal model, the truth value of a knowledge claim is evaluated in terms of an "epistemic accessibility relation" over possible worlds, which is such that a claim to knowledge in world w, is supported if and only if the claim is supported in all possible worlds w', accessible from w. Rosenschein distinguishes two ways of understanding the epistemic accessibility relation in terms of computational systems.

According to the standard, interpreted symbol structures approach, the epistemic accessibility relation is understood in terms of a machine's knowledge base. A machine can be said to know a proposition either if that proposition is explicitly coded as an interpreted sentence in its knowledge base (the syntactic conception), or, if it can be derived from other explicitly coded sentences using an appropriate set of logical rules (the semantic conception). The different states of the knowledge base constitute representations of the possible worlds relevant to assessing a knowledge claim. Under the semantic conception of knowledge, a machine could be said to know the infinite corpus of consequences of its knowledge base. Newell (1982), for example, takes this view of the knowledge of a machine. Rosenschein suggested that a serious problem with either conception was that knowledge, thus understood, was not an objective property of the way a machine was embedded in

the world but depended on the interpretation given to whichever of its symbolic structures were taken to encode facts about the world. Under a different interpretation, the machine would possess different knowledge. This is clearly a problem which applies, not just to theories in A.I. which were Rosenschein's principal concern, but also to any version of the generic theory which takes knowledge to consist of interpreted symbolic expressions stored on the tape of the cognitive Turing machine. One answer to the problem of multiple interpretations, discussed by Pylyshyn, suggests that in practice the interpretation may be fixed. "If, however, we equip the programmed computer with transducers so it can interact freely with a natural environment...it is far from obvious what if any latitude the theorist...would still have in assigning a coherent interpretation to the functional states" Pylyshyn (1984, p.44). Pylyshyn recognized, however, that this somewhat ad hoc solution might still lead to debate over the propriety of ascribing a particular semantic content to states thus constrained.

Rosenschein's second approach avoids this problem and is also much more in keeping with the aims and motivation of the ETH. He proposed "to ground the notion of knowledge in objective correlations between machine states and world states." Rosenschein (1985, p.352). Since different world states constitute the possible worlds over which the epistemic accessibility relation is defined, the key to this alternative conception is to establish a link between machine states and states of the world. Rosenschein's model is based on a deterministic finite automaton with output, connected to an environment which can be in one of a (presumably very large) number of states. The states of the environment are defined as a lattice of instantaneous world conditions ordered in terms of their generality. If world conditions are taken to be sets of states of affairs, the ordering can be thought of set theoretically such that more general conditions contain more specific conditions as subsets. The environment generates the inputs for the DFA and responds to its outputs. A link between world states and inputs to the automaton is established in terms of the "*strongest postcondition*" function which is the most specific world condition, notated as ϕ/σ, that can be guaranteed to hold at time t', given that condition ϕ holds at time t and that the input to the automaton at t is σ. The

function is extended recursively to deal with sequences of inputs and in this way Rosenschein arrives at the notion of the most specific world condition which can be guaranteed to hold after the machine has experienced a given sequence of inputs. At this stage, a link has been established between world conditions and inputs to the automaton but not between world conditions and states of the automaton. Rosenschein observes that for each state s of an automaton it is possible to define an associated language L_s which is the set of input sequences which leave the machine in state s when it is started in its distinguished starting state s_0. Languages serve as the means for establishing a link between world conditions and states of the automaton. Specifically, Rosenschein defines $L(\phi)$, the language of condition ϕ, which is the set of input sequences picking out world condition ϕ, and $\phi(L)$, the condition of language L, which is the strongest, i.e. most specific, world condition guaranteed to hold after the occurrence of any sequence of L. Rosenschein uses $\phi(L)$ to establish the desired link between a machine state and a world condition by defining the *information content* of a state s as $info(s) = \phi(L_s)$. Finally he shows that the epistemic accessibility relation can be defined in terms of the information contents of states of the automaton, such that a state of the automaton has a given information content in world w, if and only if it has that content in all possible worlds w' accessible from w.

The latter approach to knowledge, which Rosenschein calls the "situated-automata" approach has one particular feature which deserves comment from the point of view of the ETH. As Rosenschein observes, the situated-automata approach "indicates how propositional content can be assigned systematically to arbitrary computational states that are not prestructured as interpretable linguistic entities, and thus it serves as at least prima facie evidence against the need for a language of thought in order to achieve full semanticity." Rosenschein (1985, p.356) Given that the ETH also proposes a model in which a deterministic finite automaton interacts with an external environment, Rosenschein's analysis suggests that a workable account of knowledge will be possible within the framework of the ETH.

6.3.3. The significance of movement.

Rosenschein's approach was explicitly intended to facilitate the design of mobile robots, and, more generally, "embedded agents", which are computer systems that sense and act upon their environments, Kaelbling & Rosenschein (1990). In Chapter 1 it was suggested that one of the features which makes the Turing machine attractive as a model of cognitive architecture is the fact that it includes motion relative to the tape as an essential aspect of its functioning. When the Turing machine is thought of as a model of an organism interacting with its environment, this feature suggests that human mobility is a fundamental characteristic of the cognitive system.

A digital computer, as discussed in Chapter 4, characteristically accesses data by using address arithmetic to select a hardwired path to the required location. A Turing machine traverses its tape and finds its data by pattern matching. This data access method is, of course, one of the principal reasons why the Turing machine is an unsatisfactory model for high speed, program driven computation, but it might be much more satisfactory as the basis for understanding the data processing of an organism, such as the human being, which is essentially mobile. The intuitive attraction of the idea is based on the relative permanence and immobility of the physical environment. Barring acts of war, earthquakes, volcanic eruptions and the activities of the construction industry, the physical environment can be counted upon to remain much the same over the sorts of time scales which people are sensitive to. This is true at a variety of spatial scales. Buildings remain where they are from day to day, and a note written on a sheet of paper and placed in a filing cabinet, can normally be retrieved at will.

Objects take up space, and moving around to gain access to them is an essential feature of everyday experience. The generic theory, of course, relies on the permanence of objects and the natural environment just as much as the ETH does. There is no point in having a memory system which consists of representations of an external environment unless the represented environment has at least some relatively stable features. But if environmental stability is granted, and if a way can

be found of relating directly to the environment without thereby being stimulus bound, then the utility of internal symbolic representations as means of freeing the organism from the deterministic tyranny of the local stimulus environment becomes questionable.

There is evidence to suggest that our basic sense of orientation to the physical environment is maintained by information derived from movement, rather than from the perception of gravitoinertial force as is widely believed (Stoffregen & Riccio, 1988). It is also clear that the neural system is configured to respond primarily to change, and change is most clearly produced by movement.

> "Most of the input to our sensory systems is actively generated by our body movements. In fact, it is fair to view all sensory information as being generated either by the movement of objects in the world around us, or by our movements relative to those objects."
>
> Mead (1989, p.127)

The generic theorist might respond by distinguishing sensory information from perceptual information and arguing that perceptual information results from an encoding process which transforms the information about change registered by the senses, into the kind of static, representational codes which constitute the elements of structured symbolic expressions. It may be impossible to prove that this approach is wrong, but there is evidence (Freyd 1983, 1987) for "dynamic representations" which appear to have a rather different character.

The ETH suggests that we must regard the cognitive system as a system which is necessarily associated with movement. The primary information processing capabilities of the organism are dedicated to extracting information from a dynamically changing sensory flux whose characteristics are, at least in the case of vision, derived to a considerable extent from the organism's own movements. A point that is easily missed if mobility is thought of simply as one among the many behaviours in the repertoire of an organism, is that mobility has to be associated with a variety of other capacities if it is to be useful. Generally, it helps to be able

to see if you can move, but seeing will only help you if you can process the information which vision provides at a rate which is appropriate to your speed of motion. It is, for example, of little use to be able to sense the chasm yawning at your feet, if the cognitive system is unable to process the information fast enough to trigger an appropriate behavioural response before you step over the edge.

> "...the mobile organism most likely to succeed is one that is willing and able to act quickly on messy and even inconsistent data, is able to perform sensory-processing tasks in real time, is preferably able to integrate the data received through various modalities and deploy it flexibly in new situations, and is generally an all-round biological achiever."
>
> Clark (1989, p.63)

Movement also appears to be involved in triggering processes that result in the formation of memory traces via the long term potentiation of synapses in the hippocampus (Cotman & Lynch 1989). An optimal pattern for trace induction corresponds to a naturally occurring "theta" rhythm which appears when an animal is exploring its environment (Larson, Wong, & Lynch 1986). These neurobiological considerations may provide some indication as to why the classic mnemonic technique known as the "method of loci" (Anderson 1990, pps. 200-202) promotes accurate recall. To use this method a person imagines a path through a familiar area. To remember a series of objects, the person "mentally walks" along the path, associating the objects with places encountered along the path. Recall is achieved by taking another mental walk down the path, retrieving the associated objects as successive "locations" are "reached".

If motion in the external world has something to do with memory, as the ETH suggests, then real world analogues for tape locations have to be found if the model is to be anything other than figurative. A reasonable natural analogue for a tape square is the idea of a place or location, somewhere like Trafalgar Square perhaps, or a particular lecture theatre, and there is a certain amount of psychological evidence which suggests that place can be associated with memory performance. Smith, Glenberg & Bjork (1978), for example, reported that memory for paired

associate lists of words was significantly better in the room in which they were learned than in another room.

There is also evidence from studies with rats, that specific neural ensembles in the hippocampus function as place recognizers (O'Keefe & Nadel, 1979; O'Keefe, 1989), and at least one connectionist model has been constructed (Zipser, 1986). Hippocampal place cells store information about the current environment and the animal's location within it. O'Keefe (1989, p.239) makes the interesting point that map construction in rats appears to be motivated by curiosity, rather than by a non-cognitive factor such as the reduction of hunger or thirst. O'Keefe and Nadel also intended their original theory to make contact with the neuropsychological evidence relating to amnesia, although this aspect of the work has been controversial (Horel 1979; Squire, 1979).

6.3.4. Brooks' studies of mobile robots.

One other approach to the study of intelligence which is particularly relevant to the ETH is the work of Brooks (1990,1991). Brooks claims that A.I. has foundered on the issue of representation, and that thinking about representation is the wrong way to approach the problems which have to be solved. He motivates his discussion with some reflections on the timecourse of biological evolution. On the basis of these reflections, he argues that the essential basis for intelligence is not the development of sophisticated symbolic representations but "the ability to move around in a dynamic environment, sensing the surroundings to a degree sufficient to achieve the necessary maintenance of life and reproduction." Brooks (1991, p.141). Brooks suggests that the traditional "decomposition by function" analysis of intelligent systems into peripheral perceptual and motor modules interacting with a central symbolic information processor, is not the best way to proceed. He argues for an alternative "decomposition by activity" approach. "An alternative decomposition makes no distinction between peripheral systems, such as vision, and central systems. Rather, the fundamental slicing up of an intelligent system is in the orthogonal direction dividing it into *activity* producing subsystems. Each activity, or behavior producing system individually connects sensing to action. We refer to

an activity producing system as a *layer*. An activity is a pattern of interactions with the world." Brooks (1991, p.146). The methodology inspired by this approach is to begin by building a very simple autonomous system and testing it in the real world. One example is a mobile robot which avoids hitting things. Once this is working properly, an incremental layer, operating in parallel with the first layer can be added and tested. The example which Brooks gives is the activity of trying to visit distant visible places. These two activities working together constitute a machine which tries to visit distant places but avoids hitting obstacles on the way. The control interactions between the layers have to be carefully engineered, but the advantage of the approach is that it gives an incremental path from very simple systems to complex systems. At each stage it is only necessary to build a small piece and to interface it to a complete working system. The methodology has been used to construct mobile robots which exist in a real office environment and behave in a variety of ways.

Brooks argues that with multiple layers each contributing sensing and acting, there is no central representation of the world in the system. The machine is simply a collection of competing behaviours. Brooks hypothesizes that "much of even human level activity is similarly a reflection of the world through very simple mechanisms without detailed representations." Brooks (1991, p.149).

From the point of view of the ETH, it is very interesting to note just how many parallels there are between Brooks's "decomposition by activity" approach to hardware design and Turing's methodology for constructing the control of the universal machine. Turing's hierarchical method of construction, based on multiple copies of simple *m*-functions such as the f-unit, hard-wired to produce particular types of behaviour, is strongly analogous to Brooks's use of layers of simple finite automata. Furthermore, Turing's *m*-functions individually connect sensing to action just like the layers of Brooks's machines. When one has in mind the model of the Turing machine as a complex system of states of mind interacting with an external environment the likeness between the two proposals is striking. Perhaps this should not be too much of a surprise. If the picture of Turing's analysis of computation

given in Chapter 3 is correct, it is appropriate to view the brain as a complex multi-functional finite automaton just as Brooks suggests. From an evolutionary point of view, it seems highly plausible also to suppose as Brooks does, that human cognitive capacity arose as a result of successive modifications to machines which were already functional.

6.3.5. Representations.

Finally, it is appropriate to discuss the nature of representations in the ETH. It is essential for this purpose, to bear in mind Turing's distinction between *m*-configurations which model the "states of mind" of the human computer, and configurations which consist of pairs (q,a) of *m*-configurations and symbols, where the *m*-configuration, q, is internal to the machine and the symbol, a, is external. The point of this distinction, as far as the ETH is concerned is that it supports a distinction which needs to be made between cognitive states which are world involving and mental states which are states of the brain. Whereas the generic theory argues that cognitive states are states of the brain and hence, by analogy with digital computers, must involve symbolic representations, the ETH claims that it is only mental states that are states of the brain and that these, therefore, need not involve explicit symbolic representations of the type found on the tapes of Turing machines and in the symbolic memories of computers. From the discussion of internal states in chapters 2 and 3, it is apparent that the ETH supports the idea that internal states, i.e. mental states or states of the brain are representational, but the primary mode of representation is by virtue of their implicit memorial capacities rather than because they are structured as semantically transparent, syntactic analogues of external world entities and events. Given that the functions and organization of the various parts of the brain are not, as yet, completely understood, it must be the case that the representational capacities of neural tissue and its organization are also not yet completely understood. It may be, as discussed in the next chapter, that some parts of the brain do operate in a more explicitly symbolic mode than the above analysis suggests. However, the ETH is strongly committed to the view that, until proved otherwise, the parsimonious hypothesis about mental states as opposed to cognitive states is to think of them as representing implicitly by

virtue of their various levels of functionality in the economy of the brain as a whole. By contrast, cognitive states consist of configurations of mental states and external symbolic objects and represent accordingly. The importance of the mental state component of cognitive states is that they provide the control of the system's operations. A distinction was made in Chapter 3 between the determination of behaviour and the control of behaviour. It was suggested there that external objects and events could usefully be thought of as contributing to the determination of behaviour while internal, mental states, controlled it. One point of the distinction was to describe how the nesting of *m*-functions, which appears to be characteristic of complex automata, can lead to complex behavioural outcomes which are best thought of in terms of multiple sources of control, even though, at any given moment, only a single configuration is operative. The representational content of cognitive states will depend at least in part on whether or not the external components of those states have consensually agreed referents. The discussion of representation is continued in the next chapter in the context of links between the ETH, connectionism, and the representational redescription hypothesis of Karmiloff-Smith (1992).

Chapter 7. The External Tape Hypothesis, Connectionism and Cognitive Development.

In the first part of this chapter the relationship between connectionist networks and the finite control automata of Turing machines is discussed. The discussion is initially limited to feedforward networks trained using the backpropagation algorithm because these are the networks which have figured most prominently in psychologically oriented applications. The purpose of the discussion is to establish a link between connectionist research and the ETH. There has been considerable debate over the theoretical status of connectionist models. Smolensky (1988), for example, argued that they may constitute a new class of computing machines, whereas Fodor & Pylyshyn (1988) argued that connectionism is best thought of as an implementation strategy for what they call "classical" architecture, i.e. the view of cognitive architecture as a Turing machine implemented in the brain. Much of the debate has been focussed on the claim that structured, symbolic representations are needed for an architecture to be capable of modelling the cognitive system. In a response to Fodor & Pylyshyn, Smolensky (1991) accepted that any satisfactory model of the cognitive system would need to satisfy two informal principles which he took to be fundamental to Fodor and Pylyshyn's position. These are the requirements a) that models of thought must have composite structure because thoughts have composite structure and b) that models of mental processes must be sensitive to this composite structure because mental processes are sensitive to the composite structure of thoughts. Smolensky suggested, however, that connectionist methods for implementing these requirements, such as tensor product variable binding, were sufficiently different in their commitments from traditional symbol processing models as to count as quite different theoretical approaches. Nevertheless, Smolensky accepted the long term need to show how composite structure and structure sensitive processing could be incorporated in connectionist models.

From the point of view of the ETH, of course, the case for structured, symbolic representations of the kind required by Fodor & Pylyshyn rests, at least in part, on

the disputed assumption that the brain implements a Turing machine. Thus it appears that Smolensky has tied himself more closely than need be to the ideas of the symbol processing school about what the fundamental requirements are. What the ETH has to offer the debate is the suggestion that a radical rethinking of the relation between symbolic structures and internal mechanisms is appropriate. The suggestion is not that significant structure is not needed, but that a fundamental distinction must be made between the kind of symbolic structures available external to the organism and the kind of internal control structures that are needed to process elements of the external structure.

7.1. The ETH and connectionism.

The ETH suggests, following the basic claim that the brain should be thought of as a finite automaton, that connectionist networks should also be thought of as finite state control automata. To make this suggestion is not to belittle the achievements of connectionism. The demonstration that arbitrary functions from input to output can be learned from training examples and error feedback is an obviously important contribution. The suggestion that connectionist networks should be thought of as finite automata opens up a discussion about the amount and kind of internal structuring which is needed in a network and suggests reasons why feedforward nets are insufficient for modelling cognitive phenomena and that more attention should be paid to relevant structure in the external environment. The external environment is an essential part of the computational capabilities of the cognitive system in the same way that the external tape is an essential part of the computational capabilities of the Turing machine.

For this argument to work it must be shown that connectionist networks are correctly construed as finite automata. It is clearly the case that fixed neural networks are correctly construed thus, cf. McCulloch & Pitts (1943); Arbib (1987, Chapter 2). The question then is whether the learning of which multi-layer connectionist networks are capable changes the characterization. In chapter 2 a relatively brief argument to the effect that it does not was advanced. In this section the capacities of multi-layer feedforward connectionist networks which learn by

backpropagation are examined in more detail, and the argument of chapter 2 is reinforced. The discussion is based on a generic example feedforward network and the equations determining its behaviour adapted from Simpson (1990). Some connectionists have stressed the "brain-like" properties of feedforward nets, but Smolensky (1986) considers them to constitute a distinct analytical level, intermediate between the neural and symbol levels, and Douglas & Martin (1990, pp.436-7) have drawn attention to a variety of important differences between feedforward nets and cortical circuits.

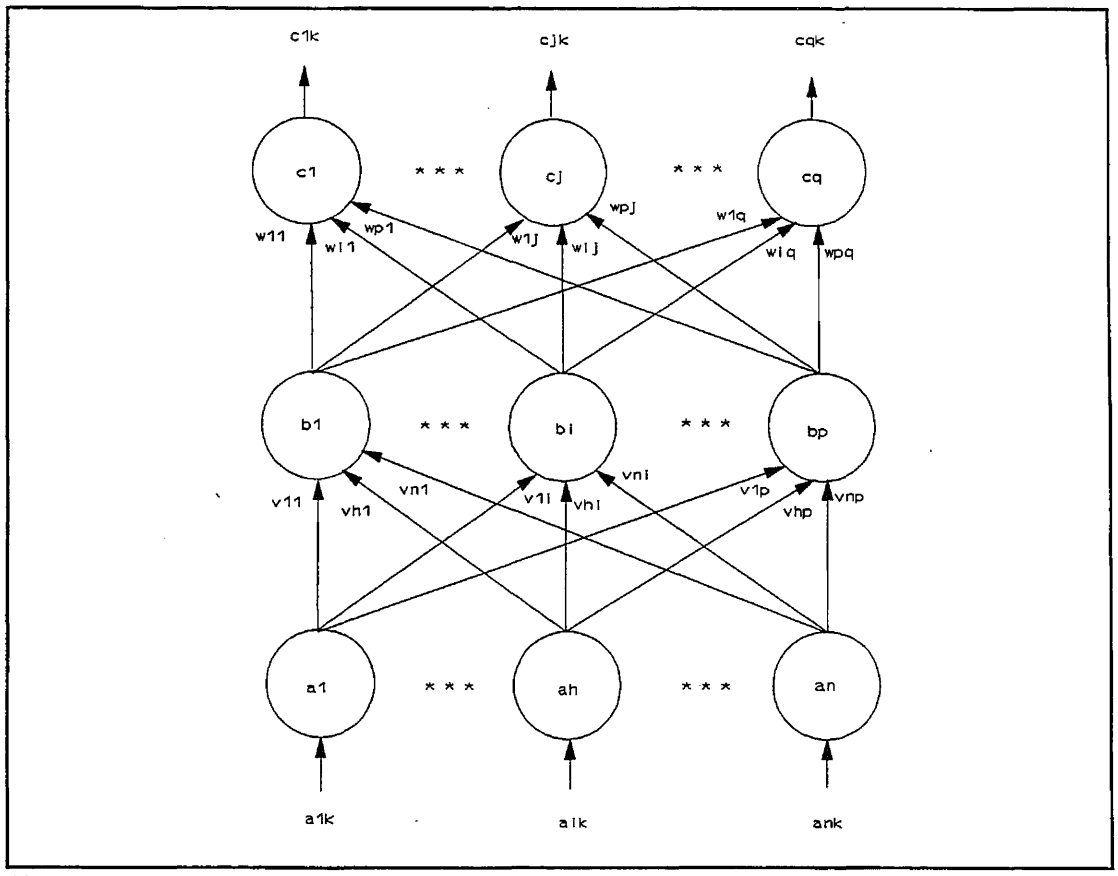

Figure 7.1. A generic feedforward network.

Figure 7.1 shows the topology of the basic feedforward network. There are three layers of nodes. $a_1 \ldots a_n$ is the input layer, $b_1 \ldots b_p$ is the "hidden" layer, and $c_1 \ldots c_q$ is the output layer. Subscripts *n,p,q* represent the fact that there may be different numbers of nodes in each layer. In many connectionist models there are fewer nodes in the hidden layer than in either the input or output layer, a property which encourages the formation of encodings which will generalize to novel inputs. The

connections between the input and hidden layers are labelled $v_{11}...v_{1n}$, and those between the hidden and output layers $w_{11}...w_{pq}$. The connectivity shown is full with each node at a given level having connections with every node at the next level. Other patterns of connectivity are possible, but full connectivity is the norm and other patterns do not alter the operation of the backpropagation algorithm. It is also possible to have multiple hidden layers of nodes, but again this does not alter the operation of the backpropagation algorithm. Pairs of input-output patterns (A_k,C_k) are defined with elements $a_{1k}...a_{nk}$ and $c_{1k}...c_{qk}$. The pattern elements may take arbitrary real values, but are most frequently taken to be binary elements representing the presence or absence of features in the domain over which the semantics of the associative pairs are defined. The input and output patterns are commonly interpreted as feature vectors.

Two types of tasks are commonly studied with feedforward nets. The first is the task of reproducing an arbitrary mapping from input to output. Typically the psychologically oriented researcher chooses a set of input output mappings which has an appropriate interpretation in a specific domain, for example the mapping between present and past tense forms of verbs. The task for the network is to learn the mapping such that when it is presented with an input vector A_k it produces as output the matching vector C_k. In some cases final success is the only issue of interest but in others such as the learning of past tense verb forms, where there are developmental phenomena such as over-regularization to account for, the learning trajectory as well as the final performance is of interest. The second type of task is classification or categorization, in which the input vectors A_k are treated as members of categories C_k. In a typical case a subset of the patterns constitutes the training set and the remainder the test set. The task of the network is to induce the "rules" for category membership from the training set alone so as to exhibit generalization by classifying the members of the test set correctly without explicit training on them. The generic method for training feedforward nets is as follows;

1) The network is initialized with a suitable set of starting values. Random values, usually sampled from the interval $[-1,+1]$, are assigned to each of the v_{hi} and the

w_{ij}. These are the starting values for the connection strengths between nodes in the different layers. Random values are also assigned to the activation thresholds of the nodes in the hidden and output layers. The set $\Theta = \{\theta_i \mid 1 \leq i \leq p\}$ is the set of activation thresholds for the hidden layer nodes and the set $\Gamma = \{\gamma_j \mid 1 \leq j \leq q\}$ is the set of activation thresholds for the output layer nodes. Thresholds are not needed for the input layer nodes because these pass the elements of the input pattern to the first layer of connections unchanged.

2) The training phase. In the training phase, the patterns in the training set are presented to the network one at a time. One presentation of the complete set of training patterns constitutes an *epoch*. A large number of training epochs may be needed for the network to reach criterion performance. Within an epoch the presentation order of training patterns would normally be randomized. Failure to do this can result in anomalous representations being formed. A single pattern presentation includes the following stages;

a) The elements $a_1 \ldots a_n$ of the particular input pattern are transferred to the input layer nodes. Each of the hidden layer node activation values b_i is then calculated using the equation

$$b_i = f\left(\sum_{h=1}^{n} a_h v_{hi} + \theta_i\right) \qquad EQ1$$

EQ1 assigns to each of the nodes in the hidden layer the sum of its weighted inputs from the input layer plus its current threshold value, which is an independent term with an important function in the error backpropagation process, the whole being constrained by the sigmoid threshold function $f(x) = (1 + e^{-x})^{-1}$. f(x) is sometimes known as a "squashing" function because it maps an arbitrary domain of input values into the range [0,1]. Some such function is needed to stabilize the network's performance by preventing inputs which produce high activation levels from dominating the representations formed.

b) Once the hidden layer node activation values have been calculated, the process

is repeated to calculate the output layer node activations, c_j, using the equation

$$c_j = f\left(\sum_{i=1}^{p} b_i w_{ij} + \gamma_j\right) \qquad EQ2$$

The values thus arrived at constitute the calculated output pattern which, to begin with at least, will bear no relation to the desired output because it is produced using random connection strengths and thresholds.

c) The third step is to calculate the discrepancy between the desired and the computed output for each of the output nodes c_j. This is done using the equation

$$d_j = c_j(1 - c_j)(c_j^k - c_j) \qquad EQ3$$

The c_j are the computed values and the c_j^k are the desired values. The term $c_j(1 - c_j)$ ensures that the computed discrepancy is larger relative to the absolute discrepancy when the c_j lie near the middle of the interval [0,1] than when they lie near either extreme. This means that the rate of change decreases as the c_j approach the asymptotic values of 0 and 1. The d_j values constitute the feedback which is used as the basis for the adjustment of node thresholds and connection strengths.

d) Given the d_j the next step is to calculate the error of each of the hidden layer nodes b_i relative to each of the d_j. This is done with the equation

$$e_i = b_i(1 - b_i) \sum_{j=1}^{q} w_{ij} d_j \qquad EQ4$$

EQ4 shows that the error at each of the hidden layer nodes contains a contribution from each of the d_j proportional to the strength of the connection w_{ij}. Like the d_j, the rate of change of the e_i decreases as they approach their asymptotic values.

e) At this stage modifications to the w_{ij}, i.e. the connection strengths between the hidden layer nodes and the output nodes, can be made. The change is defined by

$$\Delta w_{ij} = \alpha b_i d_j \qquad EQ5$$

where α is a positive constant controlling the learning rate. α would normally be

small relative to the activation values b_i. Large values for α produce more change, but tend to make the learning less stable. EQ5 shows that the change to a given weight w_{ij} is proportional to the product of the errors of the appropriate hidden layer and output nodes b_i and c_j.

f) The output layer node thresholds are then altered using the same learning rate parameter α. The change is given by

$$\Delta\gamma_j = \alpha d_j \qquad EQ6$$

g) The input to hidden layer node connections v_{hi} are then modified using

$$\Delta v_{hi} = \beta a_h e_i \qquad EQ7$$

where β is a positive constant controlling the learning rate. a_h is the value of the h'th element of the input vector.

h) Finally the hidden layer node thresholds are altered according to

$$\Delta\theta_i = \beta e_i \qquad EQ8$$

At this stage, one training pattern has been presented and the appropriate action taken to modify the responses of the network in the direction of the required association. Steps a) to h) are repeated until the d_{ij} are zero or have reached some acceptably low value.

The performance of the network as a whole is achieved by statistical juggling of the node threshold values and the connection strengths between nodes. Connectionists sometimes talk of a network learning to satisfy multiple simultaneous constraints. The most obvious place where this can be seen is in EQ4. The term $\Sigma w_{ij}d_j$ shows that the error calculation for a particular hidden layer node involves all of the output layer discrepancies and all of the connections from that hidden layer node to the output layer. Clearly some of these contributions will be in conflict with each other. What happens is that the network minimizes the overall error by finding similarities in the (A_k,C_k) pairs, such that the conflicting requirements on hidden

layer nodes can be reduced.

The important questions for present purposes are how to characterize the functionality of feedforward networks and what their performance limitations are. The first point is the question of capacity. Assuming the cardinality of the input and output layers of the network to be the same, and assuming that both input and output vectors are binary, both of which are common assumptions, then there are 2^N input patterns and 2^N output patterns definable over layers with N nodes. Thus the input and output vectors constitute a finite alphabet of patterns. The question of the maximum number of input-output mappings which a network can store with respect to a given performance criterion is more difficult to determine because multi-layer feedforward networks are resistant to analysis in this respect, Amit (1989, p.271). However, although the connection strengths between nodes which serve as the system's memory for patterns are continuous, and although the storage of patterns is distributed over the nodes and connections among them, there is every reason to suppose that network capacity is finite. For simpler but similar, fully connected network topologies using distributed storage, provable capacity limitations are known, Amit (1989, Chapter 6) which show that the number of patterns which can be stored and successfully retrieved is proportional to the number of nodes in the network. There is no doubt that some similar limitation applies to feedforward nets.

The second question is what sort of functions a given feedforward network can learn to compute with this finite capacity. Despite the impressive and thought provoking performances of a variety of feedforward network applications, the class of functions which simple feedforward networks can learn to compute is a subset of the class of functions computable by finite state machines. Once one ceases to be bewitched by the fascinating learning process, it is clear that a feedforward network learns to approximate a function which is defined by a look-up table consisting of the set of patterns (A_k,C_k). "Simple function approximation is one level above a look-up table in computational complexity; functions can at least attempt to interpolate between examples, and generalize to examples that are not in the learning

data set. Learning is still *fairly* simple, although already the subtleties of probability and statistics begin to complicate the matter. However, simple function approximation has less computational capability than a finite state machine. At present, there are no good learning algorithms for finite state machines. Without counting, conditional looping, etc., many problems will simply remain insoluble." Farmer (1990, p.183)

Informally, the exlanation for the limitations on the performance of a feedforward net can be couched in terms of its possible responses to its inputs. Consider first a feedforward net without error correction by backpropagation. The net has some fixed set of connection strengths and some fixed set of node thresholds chosen at random. When the net is given an input, its response is produced by applying EQ1 to produce activation of the hidden layer nodes and EQ2 to produce activation of the output layer nodes which produce the elements c_j of the output vector. Because the equations EQ1 and EQ2 are fixed, the response of the network for any particular input is also fixed. The crucial point to note is that although the activation, and hence the response, of the network varies from input A_i to input A_j, for a given input A_i the response of the network at different times t_n, t_{n+k} will be the same. The network's state is not altered by the processing of inputs. For this reason, it is appropriate to describe the network as computing a function from input to output which can be computed by a one-state finite automaton defined for the purpose, and that the network constitutes an implementation of the machine table for that function.

Now consider a network which, having learned to compute a given function from a set of training examples using backpropagation, is being tested on a set of test inputs. By definition the network is no longer learning, i.e. backpropagation has been turned off. For the reasons outlined in the preceding paragraph, it is clear that the network will produce a single fixed output for each of its inputs. The only difference between this case and the case above is that the function computed is one chosen by the experimenter and taught to the machine rather than one arrived at by chance as a result of the random assignment of connection strengths and node

thresholds. Thus, a fully trained, correctly functioning feedforward net, is also appropriately described as an implementation of a single state finite automaton. What the training has done is to make systematic, structural modifications to the network to change the function it computes, i.e. it has changed it from a network which computes an arbitrary function $\phi(A_k)$ to a network which computes a desired function $\psi(A_k)$. It is worth repeating that this is not a trivial achievement but no purpose is served by overestimating what such networks are capable of. The reason, therefore, why simple function approximation by feedforward nets using backpropagation has "less computational capability than a finite state machine" is that such nets can approximate only those functions which can be computed by single state finite automata.

If that were the end of the story, it would eventually spell the demise of interest in connectionist networks for much the same reasons as interest in single layer perceptrons declined after the critique of Minsky & Papert (1969). However, the situation is much more promising than that. One of the major problems with simple feedforward nets has been finding ways of making them sensitive to temporal sequences. It is easy to see, in the light of the discussion above, why there should be difficulties of this sort. In order for the temporality of a sequence to be discriminated it is essential that a given input occurring at one point in the sequence be discriminable from the same input appearing at another point. In other words, it is essential for more than one response to be associable with a given input. This, as described in Chapter 2 requires multiple states and that is precisely what is lacking in a simple feedforward network. One solution has been to simulate temporal sequence with spatial position. In this way, a vector of n bits can be treated as a "window" on the state of a single bit at n successive moments of time. The most obvious limitation on this technique is that the size of "window" is limited by the size of the vector onto which it is mapped.

A much more promising technique is the adoption of a network topology which includes recurrence. McCulloch & Pitts (1943) showed that nets with "circles" were capable of more complex behaviour than nets without circles, because the

former, but not the latter could "make reference to past events of an indefinite degree of remoteness." McCulloch & Pitts (1943,p.33). This point has an interpretation in terms of architectural differences between connectionist networks. A circle is an internal loop, so the distinction between nets with and without circles is essentially a distinction between nets which have internal processing loops and those which do not. A distinction of this kind exists between simple feedforward networks, and networks with recurrent connections such as those of Jordan (1986) and Elman (1990).

Work reported by Servan-Schreiber, Cleeremans & McClelland (1989) and Cleeremans (1993) using Elman's network architecture has made a significant advance in understanding the processing of temporal sequences. In addition to input, output and hidden unit layers, Elman's recurrent network has a layer of context units which take their input from the hidden unit layer and feed their output back to it. The input to the context units is a copy of the activation vector of the hidden units at each time step *t*. The context units store but do not alter this vector which is fed back unchanged to the hidden units as part of their input at time *t+1*. This means that at all times other than t_0, the hidden unit layer input consists of an input feature vector plus a copy of its own activation pattern from the previous time step. Cleeremans (1993) has demonstrated how this recurrent context layer input to the hidden units can be made functionally equivalent to a modification of the base state of the network, thus allowing it to learn to behave as a multi-state finite automaton, and become sensitive to long distance dependencies in its input sequences.

This is an important result for a number of reasons. First it suggests that connectionist networks with recurrence are sufficient to implement the finite state control structures which Turing's analysis suggests are what is required of the neural part of the cognitive system. Second it shows that this functionality can be learned given appropriate starting structures and adequately distinct input sequences, although Cleeremans reports some difficulties with the backpropagation algorithm. Third, Cleeremans reports that the representations developed over the hidden units,

given sufficient numbers of those units, can manifest sensitivity to sequences of inputs, provided these are of fixed length, as well as achieving the functionality of a finite state machine. This increased functionality has led Cleeremans and his colleagues to describe suitably trained recurrent networks as "graded state" machines. In so far as people often remember something of how they arrived at their current cognitive state as well as what their current options are this is a promising characteristic for a network to possess. It is interesting that such a marked increase in functionality should come about as a consequence of a simple loop which copies activation in a set of units at a given time out to a simple store and then back to those units at a later time. The ubiquity of reciprocal and re-entrant pathways in the nervous system which might perhaps indicate functionality of this kind is a major feature of its organization which is widely held to be of great significance, Damasio (1989); Edelman (1989); Zeki (1993).

The previous paragraph should not be read as claiming that the cognitive system can be explained as a single recurrent network. It seems highly probable that the cognitive system will turn out to involve a large number of modules of varying degrees of autonomy. If that is so, then the recurrent network model may turn out to be a useful basis for understanding the inner workings of modules. What then remains to be done to complete the picture from the point of view of the ETH, given the idea of a connectionist net as the implementation of a finite automaton, is to understand what is involved in implementing the same kind of relationship between the network and its environment as exists between the finite state control of a Turing machine and its tape. At this stage, work in robotics, particularly of the kind carried out by Brooks, discussed in the previous chapter may provide a useful link.

If the description of cognitive architecture argued for throughout the thesis is correct, then we can think of the cognitive system as realizing a Turing machine, with the embodied brain as the control and aspects of the environment as the symbolic resources. It seems plausible to view the brain and the structures required for cognition as capable of being modelled by a modular system of communicating,

recurrent connectionist networks, each implementing a specific cognitive function. This suggestion appears to support the view argued for earlier, that there is no a priori requirement for internal structured, symbolic expressions. The control states of the finite automata implemented in the brain interact directly with external symbols of a variety of kinds. One consequence of this view is that the knowledge of the system is implicit in its structure.

7.2. The ETH and representational redescription.

The question which is considered in the final part of this chapter is whether, in addition to the primary architecture, there are grounds for postulating the existence of a secondary system in which part of the knowledge of the system is made more explicit. There seem to be two lines of thought leading in this direction. The first is the capacity which humans develop for making explicit both to themselves and to others at least part of what they know. Just how this capacity should be related to the structure of the cognitive architecture is far from clear, but it is, perhaps, a point in favour of the generic theory, that the postulation of explicit internal symbolic representations can account for this capacity. Humans can make explicit to themselves what they know to the extent to which they have access to their own knowledge structures. Access of this kind is harder to explain if the ETH is correct because knowledge on the ETH account is implicit in the large scale structures of the system. It should be noted, of course, that consciousness is an important element of the kind of direct access which the generic theory appears to give a reasonably natural account of, but the generic theory has no more need of consciousness to explain its processing than does the ETH.

A second, perhaps more compelling reason for thinking that at least part of the knowledge of the human may become more explicit than the primary architecture of the ETH suggests, is the growing body of experimental evidence which shows that children have differing types of access to what they know at different points during their development. Annette Karmiloff-Smith has argued, in a series of papers and a recent book, Karmiloff-Smith (1992) for a model of development which incorporates what she calls a process of *representational redescription*. "The

RR model attempts to account for the way in which children's representations become progressively more manipulable and flexible, for the emergence of conscious access to knowledge, and for children's theory building. It involves a cyclical process by which information already present in the organism's independently functioning, special-purpose representations is made progressively available, via redescriptive processes, to other parts of the cognitive system. In other words, representational redescription is a process by which implicit information *in* the mind subsequently becomes explicit knowledge *to* the mind, first within a domain and then sometimes across domains." Karmiloff-Smith (1992, pp.17-18).

According to the RR model, development involves three recurrent phases. During the first phase, the child focuses predominantly on input from the external environment to create "representational adjunctions" which are more or less independent additions to the behavioural repertoire. Phase 1 culminates in what Karmiloff-Smith calls "behavioural mastery" which is the ability to perform with consistent success in a given cognitive micro-domain, e.g. balancing blocks on a narrow support. During the second phase "system-internal dynamics take over such that internal representations become the focus of change." Karmiloff-Smith (1992, p.19). These internal representations are hypothesized to pre-dominate over direct input from the environment, and may lead to a decrease in successful behaviour, thus giving rise to a U-shaped developmental curve such as has famously been observed with respect to forming the past tenses of verbs. Finally in phase three "internal representations and external data are reconciled, and a balance is achieved between the quests for internal and external control." Karmiloff-Smith (1992, p.20). The three phases work at different times and at different rates within different cognitive domains. Four different levels of knowledge representation are hypothesized to underlie the three phases of development. These are termed Implicit (I), Explicit-1 (E1), Explicit-2 (E2) and Explicit-3 (E3). Level I representations are in the form of procedures for analysing and responding to stimuli in the external environment. They provide "the ability to compute specific inputs in preferential ways and to respond rapidly and effectively to the environment. But

the behaviour generated from level-I representations is relatively inflexible." Karmiloff-Smith (1992, pp.20-21). Level E1 representations are compressed, abstract redescriptions of the level I representations. They are simpler, in the sense of losing inessential detail, but they are more cognitively flexible because they are detachable from the specifics of the context in which they were generated. They exist alongside, but do not replace, the level-I representations which remain available for operations requiring speed and automaticity. Although E1 representations are available as data to the system they are not necessarily available to conscious access or to verbal report. Level E2 representations are available to consciousness but not to verbal report, while level E3 representations are "recoded into a cross-system code. This common format is hypothesized to be close enough to natural language for easy translation into statable, communicable form." Karmiloff-Smith (1992, p.23). Karmiloff-Smith draws a distinction between the process of representational redescription which she takes to be domain general, and the RR model which gives a specific account of the process in operation in a variety of cognitive micro-domains. This distinction is made so as to allow the process to survive the refutation of any particular model incorporating it.

An impressively wide variety of developmental evidence with a general bearing on the model and experiments designed to test specific aspects of it are presented in the book, but little detail is provided of the mechanisms which might implement the process of representational redescription. This is largely deliberate. "In my view, soft-core modeling often leads to a broader intuitive understanding of general principles of change, whereas both the information-processing use of the flow chart and the symbolic approach to computer simulation run the serious risk of reifying into one or more boxes or single-named operators what is in fact the product of a highly interactive system." Karmiloff-Smith (1992, pp.175-6). What does seem clear is that level-I representations and the notion of behavioural mastery, which together form the foundation on which the representational redescription process operates, have a very natural interpretation in terms of connectionist models. "...the intuition underlying the notion of behavioral mastery maps rather well onto the connectionist notion of a network's having settled into a stable state." Karmiloff-

Smith (1992, p.182). "It seems plausible that connectionist models can indeed lend precision to an account of what I have called phase-1 learning - the phase that results in behavioral mastery (i.e. the period of rich interaction with environment during which level-I representations are built and consolidated)." Karmiloff-Smith (1992, p.189).

What is much less clear is what sort of mechanism might be involved in producing the E1 level representations and what sort of form they might take. Clark & Karmiloff-Smith (1990) and Clark (1993) refer to the technique of skeletonization, Mozer & Smolensky (1989) as a possible candidate mechanism. Skeletonization computes a measure of functionality or *relevance* over the input and hidden layers of a network and uses this measure to prune the least relevant nodes, but it is not clear that this technique effects the kind of transformation of representational format which seems to be central to the representational redescription hypothesis.

The proposal put forward here is that the Turing computable process of self-description, originally described by Lee (1963), might serve to effect the transformation from level-I to level-E1 representations. Further, given the way in which this process might work in the context of the kind of architecture proposed by the ETH, an additional advantage is that it provides at least a partial solution to the symbol-grounding problem, Harnad (1990). The symbol-grounding problem is the problem of understanding how the semantics of a formal representational system can be made intrinsic to the system rather than parasitic on the interpretation assigned by an outside observer, as is the case, for example, with the symbol systems manipulated by computers, cf. Rosenschein (1985) and the discussion of his work in Chapter 6.3.2.

There are three additional reasons why computable self-description might be a satisfactory candidate process for the level-I to level-E1 transformation. The first is that it executes precisely the type of function which Karmiloff-Smith requires, i.e. the transformation of an implicit structure into an explicit representation of that structure. In the case of a self-describing Turing machine, the process can be

understood informally as follows. We think of the machine's control as a black box whose machine table is unavailable to an observer. The tape of the machine is blank at the start of the computation. Thus the observer has no clue as to the structures and functioning of the machine. The machine is put into operation and begins to write symbols on its tape. When it halts, the sequence of symbols left on the tape, is a copy, in some standard format, of its own machine table, which is now available to the observer. The second reason why computable self-description might be a candidate for the first step in the process of representational redescription is that self-description is highly constrained and requires very specific structures and sequencing to operate correctly. It should, therefore, be amenable to experimental testing. Third, it is a purely automatic process. This is an important characteristic for a process which is assumed to provide the basis for explicit knowledge of a domain. A process which required consciousness or explicit access to its knowledge in order to construct a representation would clearly be unsatisfactory. Skeletonization, which uses a measure of relevance as the basis for altering network structure is a dubious candidate for this reason.

To understand the proposal, the process of self-description needs to be set out in some detail. It has its origins in von Neumann's investigation of the logic of self-reproduction. von Neumann asked whether it was possible to construct a machine which, given a reservoir of suitable elementary components, would be able to construct an identical copy of itself. He answered the question in the affirmative and outlined the sequence of steps which such a machine would need to carry out. He noted that the self-reproduction capacity was strongly analogous to Turing's notion of computational universality. von Neumann's work stimulated logicians to tackle the associated problem of self-describing machines. The question was, could a Turing machine be constructed which, when started on a blank tape, would eventually halt and print out a description of the structure of its finite state control in some standard format? Lee (1963) provided the first description of such a machine and Thatcher (1963), motivated by Lee's work, produced a considerably simpler machine. The point to notice about such a machine process in the context of representational redescription is that it does two of the things which Karmiloff-

Smith requires. First it produces an explicit symbolic description of the structure of its control, and second it does this without altering that structure. The latter point is important because Karmiloff-Smith is quite explicit about the co-existence of level-I and level-E1 representations.

The way in which the self-describing process works is best understood, initially, in the context of a high level programming language. The examples are written in Pascal. The blank tape constraint is replaced by the constraint that a program may not read its own source code from an external file. If this were allowed, the task would be trivial. With the blank tape constraint, or its equivalent, the task is rather more demanding. At first sight it appears impossible because it seems to demand an infinitely long program. Consider the simple program below;

```
program simple(output);
begin
        write('Hello')
end.
```

This program outputs the single word "Hello", but does not describe itself. Now consider a program to describe program simple, i.e. a program whose function is to output the source code of program simple. This is easily done as follows;

```
program describe_simple(output);
begin
     writeln('program simple(output);');
        writeln('begin');
        writeln('     write(''Hello'')');
        writeln('end.')
end.
```

The description is achieved by encoding program simple as a sequence of strings which are then printed. There is a syntactic complication in the line "writeln(' write(''Hello'')');", which requires a repeated quote mark to indicate that it is itself part of the string to be printed and not one of the string's delimiters. Even so, the program is straightforward, and, in general, there is no problem in writing one program which describes another. However, the same technique applied to the task of constructing a self describing program, quickly runs into insoluble difficulties.

```
program self_describe(output);
begin
     writeln('program self_describe(output);');
     writeln('begin');
     writeln('writeln(''program self_describe(output);'');');
     writeln('writeln(''begin'');');
     writeln('writeln(''writeln('''''program  self_describe(output);''''');''');');
        writeln('writeln(''writeln('''''begin''''');''');');
```

Clearly, a finite self-describing program cannot be written in this way. The problem with the method is that the whole structure grows inexorably, because each line of the program requires another line to describe it. A solution to the problem is shown below.

```
program self_describe(output);
var
        store : array[1..12] of string[80];
        i : integer;
begin
        store[1] := 'program self_describe(output);';
        store[2] := 'var';
        store[3] := '    store : array[1..12] of string[80];';
        store[4] := '    i : integer;';
        store[5] := 'begin';
        store[6] := '    store[';
        store[7] := '] := ';
        store[8] := '    for i := 1 to 5 do writeln(store[i]);';
        store[9] := '    for i := 1 to 12 do';
        store[10] := '           writeln(store[6],i,store[7],chr(39),store[i],chr(39),chr(59));';
        store[11] := '  for i := 8 to 12 do writeln(store[i])';
        store[12] := 'end.';
        for i := 1 to 5 do writeln(store[i]);
        for i := 1 to 12 do
                writeln(store[6],i,store[7],chr(39),store[i],chr(39),chr(59));
        for i := 8 to 12 do writeln(store[i])
end.
```

The key to understanding the solution is to see that the program is divided into two sections. The first is the construction of a reservoir of strings of program code. This is the array called "store" defined in line 3. It is analogous to von Neumann's reservoir of elementary parts for a self-reproducing machine. Lines 6 to 17 stock the array with the required components. The second part of the program, lines 18 to 21 uses the contents of the array to construct a program description. The first five and last five lines of the program are constructed simply by copying out the contents of the appropriate locations of the store. The construction of the twelve lines describing the stocking of the store is slightly more complicated. In order to get round the problem of inexorable growth, these lines are put together using subcomponents as well as components and require a syntactic fiddle. The subcomponents are found in locations 6 and 7 of the store and the syntactic fiddle is seen in line 20 of the program. The fiddle involves two different ways of

referring to characters. One is via direct quotation. The other is via the Pascal construction "chr(n)" which represents the ASCII character whose underlying value is the byte sized binary pattern representing decimal number n. Thus "chr(59)" represents the semi-colon because the semi-colon is mapped onto the binary pattern 00111011. The importance of the fiddle is that it makes it possible to refer to the single quote character without having to double it up. The line "write(chr(39))" is equivalent to "write('''')" and outputs a single quote mark.

Some important points about the general form of the process of self description can be made by observing two ways in which the program might fail to operate correctly. First there is the possibility that one or more of the assignments to the storage reservoir is incorrect. If, for example, store[5] contained the string "start" rather than the string "begin", the program would run but would not accurately describe itself. Second there is the possibility that the construction part of the program incorrectly manipulates the description contained in the array of strings. If, for example the final "for" statement read "for i := 8 to 11 do ..." the terminating "end." statement would be omitted from the self description. These two sources of error are quite independent of each other. The assignment of contents to the store is independent of the way in which the store is later used, and there is nothing in the construction part of the program which is sensitive to the contents of the store. The match between the parts is a contingent one.

As it stands, the self-describing program is a curiosity, the sort of tricky exercise that gets assigned to students learning a programming language. From the point of view of a possible model of the process underlying representational redescription, the program needs to be taken one step further to show that a program which does something apart from describing itself can also be made self-describing. One of the features of the first self-describing Turing machine constructed by Lee (1963) was that it was universal, which demonstrated that any program could, in principle, be made self-describing. For present purposes, a simple demonstration suffices. Consider the following program which prompts the user for a pair of numbers and returns their sum.

```
program add(output);
var
   i,j : integer;
begin
     write('Enter a number >> ');
     readln(i);
     write('Enter another number >> ');
     readln(j);
     writeln('The sum is ',i+j)
end.
```

This function can be incorporated in a self-describing program by using exactly the technique adopted for program self_describe as shown below.

```
program add_and_self_describe(output);
var
     store : array[1..26] of string[80];
     i,j : integer;
begin
     write('Enter a number >> ');
     readln(i);
     write('Enter another number >> ');
     readln(j);
     writeln('The sum is ',i+j);
     store[1] := 'program add_and_self_describe(output);';
     store[2] := 'var';
     store[3] := '     store : array[1..26] of string[80];';
     store[4] := '     i,j : integer;';
     store[5] := 'begin';
     store[6] := '     write(';
     store[7] := 'Enter a number >> ';
     store[8] := 'Enter another number >> ';
     store[9] := ');';
     store[10] := '     readln(i);';
     store[11] := '     readln(j);';
     store[12] := '     writeln(';.
     store[13] := 'The sum is ';
     store[14] := 'i+j);';
     store[15] := '     store[';
     store[16] := '] := ';
     store[17] := '     for i := 1 to 5 do writeln(store[i])';
     store[18] := '     writeln(store[6],chr(39),store[7],chr(39),store[9])';
     store[19] := '     writeln(store[10])';
     store[20] := '     writeln(store[6],chr(39),store[8],chr(39),store[9])';
     store[21] := '     writeln(store[11])';
     store[22] := '     writeln(store[12],chr(39),store[13],chr(39),store[14])';
     store[23] := '     for i := 1 to 26 do';
     store[24] := '        writeln(store[15],i,store[16],chr(39),store[i],chr(39),chr(59));';
     store[25] := '     for i := 17 to 26 do writeln(store[i])';
     store[26] := 'end.';
     for i := 1 to 5 do writeln(store[i]);
     writeln(store[6],chr(39),store[7],chr(39),store[9]);
     writeln(store[10]);
     writeln(store[6],chr(39),store[8],chr(39),store[9]);
     writeln(store[11]);
     writeln(store[12],chr(39),store[13],chr(39),store[14]);
     for i := 1 to 26 do
        writeln(store[15],i,store[16],chr(39),store[i],chr(39),chr(59));
     for i := 17 to 26 do writeln(store[i])
end.
```

The program has a number of features which make it interesting as a candidate process for representational redescription, when one thinks of it in terms of the hypothesized shift from level-I to level-E1. Suppose a child to be capable of asking for two numbers and adding them together. This level-I competence might be

modelled in terms of a compiled version of program add. The fact that the source code of a program is generally not recoverable from the machine code into which it is compiled serves to reinforce the point that the child's representations at that stage are implicit. The stand alone version of program add represents behavioral mastery but no more. Once behavioral mastery has been achieved, or perhaps while it is being achieved, one can hypothesize a process of breaking down and storing the components of the task performing mechanism in individual, non-efficacious chunks. This process is modelled by the part of program add_and_describe in which contents are assigned to the store. One might speculate that these somewhat fragmented contents could model what Karmiloff-Smith (1992, p.18) calls "representational adjunctions". The final stage in the shift from I to E1 is completed by the internal generation of processes, modelled by the construction part of the program, which operate on the store to produce an explicit version of what was initially an implicit process. Finally, if one further hypothesizes that the output of the redescriptional process is to an area of the brain independent of the part subserving task performance, it is possible to see how the implicit and explicit versions of the program could co-exist. It is also clear that the explicit representation thus arrived at might be neither conscious nor available for verbal report thus remaining consistent with the postulation of further development at levels E2 and E3 of Karmiloff-Smith's model.

Two other points are worthy of note. The first is the specificity of the task and the generality of the process. Karmiloff-Smith hypothesizes that the RR process is domain-general but can be applied to a variety of specific cognitive capacities. This distinction is maintained in add_and_describe. The second point is the clear separation of task performance from the re-descriptional process. Empirical data suggest that representational redescription takes place only when behavioral mastery has been achieved. Such a position is consistent with the program although not entailed by it.

An important question is why a self-describing process is more appropriate for a treatment of representational redescription than a modified program of the same type

as program describe_simple. Consider how program add might be treated so as to provide both functionality and a description.

```
program add_and_describe_add(output);
var
   i,j : integer;
begin
     write('Enter a number >> ');
     readln(i);
     write('Enter another number >> ');
     readln(j);
     writeln('The sum is ',i+j);
     write('program add(output);');
     write('var');
     write('i,j : integer;');
     write('begin');
     write('     write(''Enter a number >> '');');
     write('     readln(i);');
     write('     write(''Enter another number >> '');');
     write('     readln(j);');
     write('     writeln(''The sum is '',i+j)');
     write('end.')
end.
```

It might be suggested that program add_and_describe_add is a better basis for a model of representational redescription because it provides the functionality of program add and also outputs an explicit description of it without all the complication of add_and_self_describe which might, in any case, be thought psychologically unrealistic because it describes its own inner structure completely which does not seem to be characteristic of the human cognitive system.

Perhaps the simplest answer to this suggestion is that we do not know what limits exist within the cognitive system on the extent to which redescriptive processes can make previously implicit structure available as explicit data. Complete self-description is therefore an idealization of a capacity which we appear to have to a considerable extent, but it is no more arbitrary than the idealization which suggests that the brain implements a Turing machine even though it is clearly finite in its capacities. It may turn out, of course, that the idealization to complete self-description is flawed in a way comparable to the way in which this thesis suggests that the model of the brain as a Turing machine is flawed, but that does not make it an arbitrary or un-useful idealization. A second point is that if the cognitive system consists of a set of autonomous mechanisms implementing specific capacities, i.e. if the cognitive system is modular, it is possible that the functioning of individual modules may be made completely explicit via self-description. This

can be hypothesized without supposing that every module and hence the cognitive system as a whole must at some stage be made explicit.

Given the program models above as illustrative examples, it is now appropriate to turn to the notion of a self-describing Turing machine. The distinction between a store which holds the elements of a description and the constructor which puts together the description from the elements in the store, is a feature of the Turing machine model as well as of the Pascal program versions. It appears that these characteristics are invariant features of any completely self-describing system.

The original self-describing universal machine of Lee(1963) and a simpler non-universal machine described by Thatcher (1963) were both based on Wang's (1957) program model of computation rather than directly on Turing's machine model. Minsky (1967, p.287) outlines the way to construct a self-describing Turing machine. The first step is to construct a machine T, which converts an arbitrary sequence of symbols $s_{i1},s_{i2},\ldots,s_{in}$ from a fixed alphabet $\{s_1,s_2,\ldots,s_r\}$ to the sequence $q_0,B,q_1,s_{i1},1,X,q_1,B,q_2,s_{i2},1,X,\ldots,q_{n-1},\ B,q_n,s_{in},1,X,s_{i1},s_{i2},\ldots,s_{in}$ where q_i is the integer representation of state i and B is the representation for a blank square. When the sequence $s_{i1},s_{i2},\ldots,s_{in}$ is a representation of the machine T itself, the long sequence is a description in quintuple form of a self describing machine.

Somewhat less formally, a self-describing Turing machine requires a subset of its states to implement a machine M_1, which prints a string of symbols, S_1, on its tape. S_1 is interpretable as the description of a machine M_2 which manipulates strings of symbols. The dynamics of M_1 are extremely simple. It consists of a long chain of states each of which prints a single symbol. There are no loops or branches and the machine moves in one direction only. Bearing in mind that the tape of the machine is blank by definition at the start of the self-describing computation, the configurations of the machine M_1 are all of the form (q_i,B). Once the string S_1 has been printed, a new subset of states implementing machine M_2 takes over. M_2 has much more complex dynamics. It takes each element $S_{1,j}$ of S_1 in turn and includes it as part of a string which is interpretable as an instruction of M_1, specifically the

instruction to print $S_{1,j}$. M_2 is constructed so that the first such string it prints is interpretable as the first instruction of M_1 and so forth. Thus, when M_2 has processed each element of S_1 in turn, it will have printed another string S_2 which is interpretable as the description of M_1. If M_2 includes instructions to print the original string S_1 at the end of S_2, then the concatenated string S_2S_1 can be interpreted as a description of the combined machine M_1M_2. Since the machine which prints the concatenated string is precisely the combined machine M_1M_2, M_1M_2 is self-describing.

The mapping between the parts of such a Turing machine and the parts of program self_describe, while not exact, is indicative of marked functional similarity. The string S_1 is roughly analogous to the contents of the store in the program, and hence the sub-machine M_1 can be thought of as analogous to the parts of the program which define the store and assign contents to it. The sub-machine M_2 which operates on string S_1 is roughly analogous to lines 18-21 of the program which construct the full description from the elements of the store. A further point of similarity is that functionality can be added to a self-describing Turing machine by incorporating a set of states which carry out the desired function, in much the same way that functionality was added to pure self-description with the move from program self_describe to program add_and_self_describe.

The links between the ETH, self-description and representational redescription are hypothesized to occur as follows. First, it is hypothesized that the cognitive system as a whole is modular or functionally specialized, i.e. that the control automaton of the system, while amenable to description in terms of a set of global states, consists of a set $M = \{m_i \mid 1 \leq i \leq n\}$ of n interacting mechanisms much like the system of mechanisms (kmp, ce(α), etc.) realizing the control of Turing's universal machine. Each of the m_i implements the internal control system for a particular behavioural capacity such as the capacity to draw a human figure, for example. Thus each of the m_i is hypothesized to consist of a set of k states $m_{i1}...m_{ik}$. An individual control system m_i interacts with a set of external media such as paper and pencil to produce its characteristic symbolic output. Thus the configurations of the

sub-system m_i are of the kind hypothesized by the ETH, i.e. they combine an internal control state with external media and the sub-system as a whole can be thought of as a Turing machine. Behavioural mastery is reached when externally specified performance criteria can be met successfully. The machine table of the sub-system m_i is not available to the cognitive system as data at this stage because the external media are external and the internal states m_{ij} are accessible only in the course of execution of the behaviour. It is structures of this kind which are assumed to constitute Level-I representations.

The way in which self-description is hypothesized to work to produce level E1 representations given the basis described above is as follows; it is assumed that some part of the brain's storage capacity is available to the growing organism, and we might think of this, to a first approximation, as an additional, internal, tape of fixed, but possibly quite large capacity. It is hypothesized that this internal tape receives the output of the self-descriptive process as it is applied to specific behaviour producing sub-machines m_i. The suggestion is that this auxiliary tape acts like the store array of the self-describing programs and becomes stocked with behavioural fragments in the course of the organism's commerce with the external world. Some of these fragments will be sensory impressions, others the reactions of the organism to those sensory impressions and so forth. As the control system of the brain develops it acquires the self-descriptive functionality which in due course enables it to process the behavioural and sensory fragments in the store into explicit accounts of the activities which produced them. What is particularly significant is the way in which the process of self-description interacts with the components of the architecture as hypothesized by the ETH. At the stage of behavioural mastery, the symbols with which the system works are explicit but external. The child is assumed to be able to interact with them but not to reason about them because internal representations of them have not been formed. At the same time, the states of the sub-machine m_i are internal but purely implicit. Thus again, the child is able to use them but not to reason about them. By hypothesizing an internal memory resource as the locus of the output of the self-descriptive procedure, the external symbols involved in a capacity become internalized and

remain explicit while the internal states remain internal but are made explicit and are specifically linked to the internal descriptions of the symbols. Thus a description of the whole process becomes available to the system. The reason why such a process represents a partial solution to the symbol-grounding problem is that the symbols which form the explicit representational accounts produced from the store are based on the experiential interaction of the organism with aspects of the external environment.

To take the treatment in the direction of biological plausibility and build a bridge between the linear tape of a Turing machine and the neural substrate of the brain it is possible to describe some of the basic processes of laying down fragments in terms of a canonical network scheme described in Conrad (1985). This scheme represents state transitions in a standard form in a simple network and may be thought to enhance the plausibility of the suggestions made above. Consider the state transition of the simple machine shown in Figure 7.2 when it is in state q0 and receives input 0. It makes a transition to state q0.

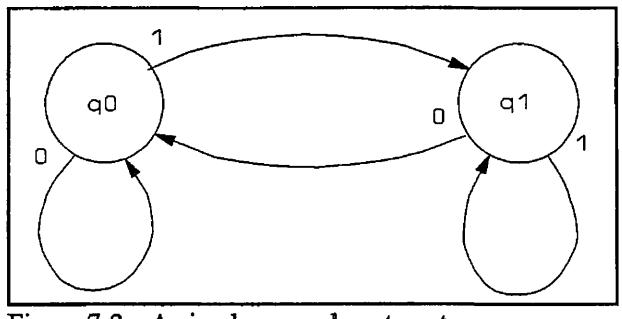

Figure 7.2. A simple example automaton.

Machinery to realize this state transition is shown in the network of Figure 7.3. The network has two columns of nodes representing the states q0 and q1 of the automaton, and two input lines, carrying the inputs 0 and 1 respectively. The nodes are threshold elements which will fire if the sum of their inputs equals or exceeds the value shown inside. All the nodes in Figure 7.3 have a threshold of 2.

A state is instantiated when either of the nodes in its column fires. A state transition, q_i,s_j,q_{ij} is instantiated in the network by making a connection from each of the nodes in the i'th column to the i'th node in the j'th column. In this way, when one of the nodes in a column representing a state and an input line both fire at time t, the appropriate node for the i,j'th state will fire at time t+1. Figure 7.4

shows the connectivity needed to realize the state transition q0,1,q1.

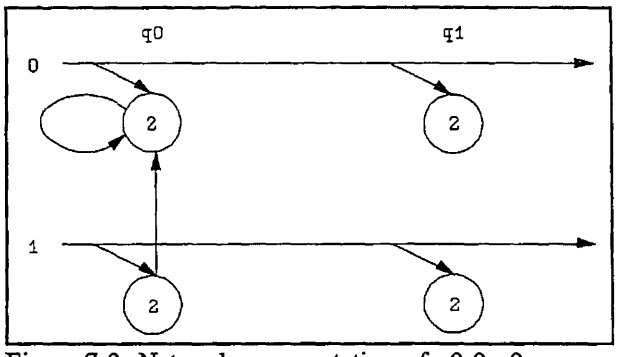

Figure 7.3. Network representation of q0,0,q0.

The networks of Figures 7.3 and 7.4 can be combined as in Figure 7.5 to give the network representation of q0,0,q0 and q0,1,q1. Inspection of Figure 7.5 shows that combining state transition instructions in a single network of the kind shown does not affect the network's ability to distinguish the two instructions. It makes the appropriate state transition for all combinations of state node and input line when the current state is q0. This performance generalizes to the situation where the instructions involving state q1 are also introduced into the network. The full system is shown in Figure 7.6.

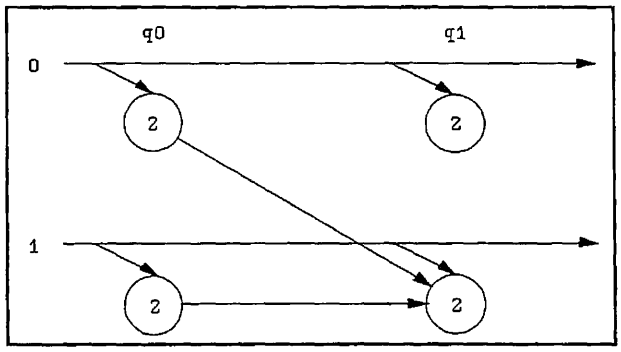

Figure 7.4. Network representation of q0,1,q1.

What this canonical network capability suggests, thinking in terms of the Turing machine model of self-description is that the human equivalent of machine M_1 might be thought of as laying down instructions as state transitions in an area of neural space, by modifying its basic connectivity. Thus, rather than printing instructions sequentially on a linear tape, the human M_1 superimposes successive instructions, which remain functionally separable, on a suitably large area of neural space. A feature of S_1 which makes the above treatment plausible is that it is divisible in a variety of ways. At the finest grain, each element of S_1 is a symbol. Thirty of these symbols form the description of a machine instruction. One to four instructions constitute the description of a state (depending on the possible inputs to the system in a given state), combinations of states form submachines at various

levels of complexity, and the whole of S_1 constitutes the description of a machine. Thus S_1 manifests organization at the levels of symbol, instruction component, instruction, state, sub-machine and machine. The finest grain analysis is not likely to be psychologically significant, because it is unlikely that the nervous system uses binary addresses. The levels of instruction components and instructions, however, are much more appropriate. Setting aside for the present the questions of output signal and movement relative to the tape, an instruction determines the transition to a new state associated with a particular current state and current input. An instruction is an autonomous unit because the set of instructions defining a target machine can appear in any order on the tape of a machine simulating it without altering the functional performance of the simulator. The order of the instructions in S_1, defining the machine M_2, is, in this sense, arbitrary. An instruction component is not autonomous in the same way as an instruction. Thus an instruction represents an indivisible unit of Turing machine performance, less than a complete state but more than a component.

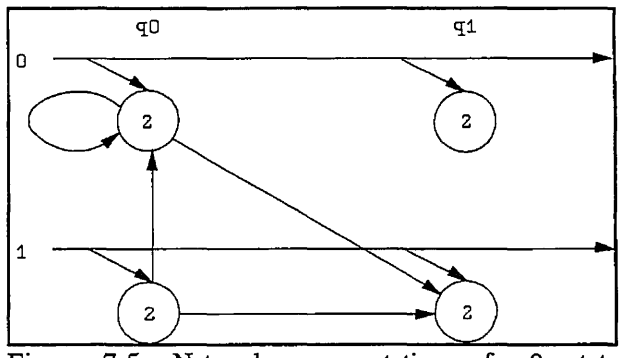

Figure 7.5. Network representation of q0 state transitions.

The hypothesis thus stated requires that experience comes in instruction sized chunks if it is to be realized as patterns of connectivity in a neural space analogous to the tape of a Turing machine. Is this a reasonable supposition? In view of the limited attention span of the human infant, the suggestion has at least some plausibility. It is also clear that patterns of connectivity can be established in networks as a result of repeated experience. Perhaps then, the operations of M_1 of the self-describing Turing machine are not completely implausible as a rather abstract account of the acquisition of an internal record of experience of the world. The external tape of the human provides a source for the structured material in S_1. As the organism grows and its sensory systems become better tuned, these internalized action sequences become more differentiated and precise.

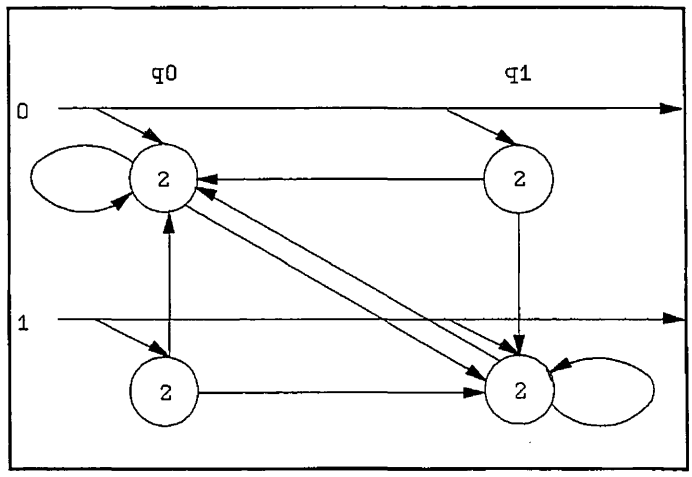

Figure 7.6. Network representation for all q0 and q1 state transitions.

The operations of the M_2 part of the self-describing Turing machine provide a hypothesis about the way in which thinking and self awareness might be developed. The important function of the submachine M_2 is that it organizes the S_1 input into a temporal sequence. This it does by embedding S_1 instruction components, as output symbols, in an ordered succession of new instructions. The proposal is to think of the human equivalent of M_2, not so much as a machine which lays down instructions, but as a machine which generates instruction execution sequences. M_2 as constituted in the self describing scheme, constructs an execution sequence of states which implements the machine M_1, and then appends its own description to complete the M_1M_2 concatenation, but it need not do only this.

M_1 and M_2 are, in principle, autonomous machines. This provides an interesting possibility. It may make sense, developmentally, to suppose that the machine M_1, which has the simpler dynamics, represents the organism at a relatively early stage of representational processing during which it does little except to register input. This fits with Karmiloff-Smith's suggestion that the early representational focus is predominantly on information from the external environment. But the sensory registration process might very well also alter the structure of the total system of which it forms a part. Thus, it might be hypothesized that machine M_2 develops as its description is being laid down. The sort of process that might be involved here could be the gradual differentiation of say, exogenous and endogenous sources of stimulation. Redescription will begin as soon as M_2 is sufficiently developed to be able to access the record of sensory experience which shaped its own construction. This suggests a) that redescription is consequent upon sufficient appropriate

experience, and b) that this early form of redescription is a precursor of self awareness because M_2, which controls the generation of the state execution sequences which constitute redescription, uses action sequences laid down as part of its own development. Perfect self-description will never be possible for this kind of machine, because the dynamics of its operations constantly modify its structure. Thus its self descriptive capacities remain one step behind its experience, which is, perhaps, how it should work out.

This chapter has been speculative, particularly in the latter part but it has served to bring the ETH into contact both with connectionism and with the cognitive scientific study of development, and to provide a picture of how it might illuminate work in both of these areas.

Chapter 8. Summary, Conclusions and Prospects.

In the previous seven chapters an argument has been constructed to support a new view of the nature of the human cognitive architecture considered as a computational system. The argument is based on Turing's analysis of computation rather than on comparisons with the structures and processes of digital computers. It has been suggested that both symbol processing approaches to cognitive computation, referred to collectively as the generic theory, and connectionist approaches are in need of revision.

8.1 The Generic Theory.

The generic theory is founded on the idea that there is a structural isomorphism between the architecture of general purpose digital computers and the architecture of the cognitive system which licenses the use of computational concepts in the theoretical description of cognitive structures and processes. Justification for this idea comes from two main sources. First there is the belief that the designs for early stored program computers were modelled on what was known about the human nervous system. Second there is the theoretically more interesting suggestion that the link between cognitive and computer architectures is a consequence of the fact that both are isomorphic to the abstract Turing machine architecture. Digital computers are essentially practical versions of Turing machines, and Turing derived the concept of the Turing machine from consideration of a person engaged in a routine numerical calculation. Hence, it is claimed, by transitivity of isomorphism, that digital computers and human cognitive architecture share fundamental structural features.

The assumption of transitivity of isomorphism is controversial because Turing machines are formal, idealized and abstract isomorphs of minds. In particular, Turing machines are idealized to have an unbounded memory in the form of an indefinitely extendible tape, but unboundedness cannot be true of human memory which is supervenient on the brain. The arguments used to resolve this difficulty lead to the heart of the issues with which the thesis is concerned. At the most

general level of analysis there are two strategies to be found in the literature. The first, which leads to various versions of the generic theory is to argue that the idealization is fruitful and provides support for distinctions between competence and performance.

8.1.1. The generic theory and the language of thought.

The arguments offered in support of the generic theory fall into two main classes. The first is characterized by the view that the idealization to unbounded memory is justified by the productivity of thought which is hypothesized to be unbounded in principle. The best known and most influential exponent of this view is Jerry Fodor, and the locus classicus of the approach is Fodor (1975). In that work Fodor argued that to treat the mind as a computational system entailed the attribution of a representational medium. Further he claimed that the medium must be implemented in such a way as to allow, in principle, for "an infinity of distinct representations." Fodor (1975, p.31). The most plausible way to think of such a system, he argued, was in terms of a set of conceptual primitives, like the symbol alphabet of a Turing machine, plus a working space, like the tape of a Turing machine, in which complex representations could be composed out of the conceptual primitives. Because such a system resembled a natural language in its combinatorial characteristics, it was appropriate to call it the language of thought. Thus Fodor argued that it is appropriate to think of the mind as organized like a Turing machine, even though the mind is finite, partly because of the generality of the Turing machine concept, but mainly because the symbol manipulating capacities of the Turing machine offer an explanation of the productivity (in principle) of thought, while practical limitations on the sizes and structures of the computational devices subserving thought constrain the idealized theoretical capacity to produce observed performance.

8.1.2. The generic theory and the physical symbol systems hypothesis.

The second influential line of thought which argues for the propriety of the Turing machine as a model of cognitive architecture is the physical symbol systems hypothesis. The best known and most influential exponents of the foundations of

this view have been Allen Newell and Herbert Simon, in a series of books and papers published over a period of more than twenty years. Symbol systems theorists differ from Fodor and other LOT theorists in the basic argument which is used to justify the idealization to unbounded memory. The symbol systems hypothesis is based on the claim that computational universality rather than the unbounded productivity of thought is the essential criterion for cognitive architecture. This claim entails that programs as well as data must be expressible as explicit, symbolic expressions in "a neutral, stable medium that is capable of registering variety" Newell (1990,p.61). This is a stronger claim than the Fodorean claim. However since both the productivity of thought and computational universality arguments require potentially unbounded symbolic resources the net result is much the same as far as the basic architecture is concerned.

In summary, proponents of the generic theory argue that the Turing machine and the digital computer are appropriate bases for models of cognitive architecture because they view the idealization to unbounded memory as principled and the practical constraints on finite realization as a fertile source of empirical hypotheses.

8.2. Connectionism.

The second major strategy mentioned above, which is found in much connectionist theorizing, but has a wider constituency, suggests that the requirement for unbounded memory whose contents consist of symbolic expressions, shows the Turing machine concept to be at best irrelevant and at worst actively misleading about the organization of memory and the nature of cognitive processes. Clark (1993), for example, has argued that the generic theory, particularly in its Fodorean form, is overly text based and anti-developmental. Other arguments on various grounds such as neural implausibility and operational fragility have been extensively discussed. Connectionist research, by and large, looks to neuroscience or to physics rather than to formal methods for its inspiration. It has even been suggested that connectionism may "challenge the strong construal of Church's thesis as the claim that the class of well-defined computations is exhausted by those of Turing

machines." Smolensky (1988,p.3).

As yet there is no clear resolution to the debate. Connectionist systems are generally better at the "softer" kinds of tasks like pattern completion and generalization to novel stimuli, while symbol systems are generally better at tasks requiring temporally extended sequences of processing and/or highly structured representations. Various projects have looked to hybrid symbolic/connectionist architectures as a means of resolving some of the difficulties. Unfortunately, many such proposals have a distinctly ad hoc flavour or run into difficulties in specifying the nature of the interface between the components, although Cooper & Franks (1993) have described some of the principles governing the specification of hybrid systems.

8.3. Turing's analysis of computation.

The suggestion developed in the thesis is that the nature of the linkages between the Turing machine concept and architectural hypotheses arising from both the generic theory and connectionism can be clarified by re-examining Turing's seminal work, Turing (1936-37), in which the concept of a universal machine was first introduced. The major conclusion reached is that Turing's analysis of computation does not support the generic theory's model of cognitive architecture. The crux of the argument is the claim that the principles of cognitive architecture on which the generic theory is based, model human memory in a way which contradicts a principled argument used by Turing to develop his formal machine model. The primary point is that in Turing's scheme the tape of the machine corresponded to the paper on which an individual worked a calculation and not to the human calculator's memory. It was a distinct, external resource which supplemented the human memory and Turing had good arguments for separating the two. In digital computers, however, symbolic, random access memory serves the same function as the tape of the Turing machine and it is this form of memory which is used to model human memory in the generic theory. The mapping from computer memory to human memory thus directly contradicts Turing's analysis and is neither compatible with it nor entailed by it. This point has substantial consequences for

theories of cognitive architecture based on Turing's analysis. Perhaps the most surprising is the way in which connectionist theorizing can be seen to be entirely consonant with Turing's analysis when connectionist networks are thought of as implementations of the finite state control automata of Turing machines. A secondary point is that the generic theory makes use of an unnecessarily impoverished concept of what a symbol is which is an accidental consequence of the focus of Turing's interest, rather than an essential feature of the definition of an effective procedure. In view of the many problems faced by the generic theory with respect to issues as diverse as its dubious neural plausibility and the difficulties encountered in finding a defensible philosophical approach to its account of the nature of mental content, removing the theoretical support of Turing's analysis of computation must lead to a reappraisal of its tenability.

8.4. The External Tape Hypothesis.

The ETH was developed to address the issues raised above. The starting point is the fact that the Turing machine was not, and was not intended to be, a model of the internal workings of the mind. It was a model of a system of states of mind interacting with a set of external symbols. At the time when he developed the model Turing was interested specifically in the nature of routine computation and his analysis is restricted to dealing with numerical symbols and processes but it is proposed that the idea of a symbol for computation can be extended beyond conventional alpha-numeric symbols to embrace a variety of objects in the world, provided, at a minimum that they satisfy the type identity criterion.

The critical notion for the ETH is that of a computational "configuration" which is a pair consisting of a "state of mind" and an external symbol. The ETH proposes that cognitive states should be thought of as configurations of Turing machines. Thus they are distinct from mental states. Mental states are considered to be supervenient on the brain but cognitive states are not because they have external components. Thus the same mental state can, in principle, form part of different cognitive states by being paired with different external symbols.

8.5. Current areas of weakness.

No doubt there are many more weaknesses and shortcomings in the approach than are recognized and acknowledged here. The author is perhaps more than usually indebted to others for pointing out problems with the approach and the process of critical exchange is very illuminating. From the point of view of the author, however, perhaps the principal omission at present is a detailed account of how the ETH might make contact with the approach of evolutionary psychology, cf. Barkow, Cosmides & Tooby (1992). Adaptationists, as these authors sometimes call themselves and others of a similar persuasion, argue against the picture of cognitive architecture as a content independent, general purpose, programmable, computational system of the kind proposed by generic theorists (see especially Newell 1990, pp.54-65), and in favour of a picture of cognitive architecture as a collection of special purpose, content dependent mechanisms. This approach to cognitive architecture is entirely consonant with Turing's analysis of computation and with the ETH. What is particularly important is that experimental predictions are being derived from the adaptationist stance and applied to a variety of cognitive topics. The work of Cosmides (1989) on forms of content in the Wason selection task is particularly interesting.

Additional areas of weakness are the underdeveloped state of the theories of internal states, external symbols and configurations identified as important needs in Chapter 6. A better understanding of the forms and possibilities of control memory will be a particularly important improvement to the theory of internal states.

8.6. Areas for future research.

Apart from making good the omissions noted in the previous section, there are many areas which offer prospects for further development. The application of the ETH to the modelling of representational redescription as outlined in Chapter 7 is a compelling topic as is the question of how to build recurrence into models of internal states. Of particular interest is the need to incorporate accounts of attention, motivation and emotion into the theory of internal states. The extent to which these issues can be tackled in a way which yields tractable models and

testable hypotheses will be a measure of the extent to which the ETH will be a valuable approach for psychologists.

References.

Aho, A.V. & Ullman, J.D. (1977). *Principles of Compiler Design.* Reading, MA: Addison-Wesley.

Amit, D.J. (1989). *Modeling Brain Function. The world of attractor neural networks.* Cambridge: Cambridge University Press.

Anderson, J.R. (1990). *Cognitive Psychology and Its Implications.* 3rd Edition. New York: W.H. Freeman and Company.

Andler, D. (1990). Untitled paper, read at the University of Sussex, conference on Concepts and Categorization, 6th-8th April 1990.

Arbib, M. (1987). *Brains, Machines, and Mathematics.* Second Edition. New York: Springer-Verlag.

Arterberry, M.E. & Yonas, A. (1988). Infants' sensitivity to kinetic information for three-dimensional object shape. *Perception and Psychophysics*, **44**, 1-6.

Ashcraft, M.H. (1989). *Human Memory and Cognition.* Glenview Illinois: Scott, Foresman and Company.

Aslin, R.N., Pisoni, D.B., & Jusczyk, P.W. (1983). Auditory development and speech perception in infancy. In M.M. Haith & J.J. Campos (Eds.) *Handbook of child psychology; Volume 2. Infancy and developmental psychology.* 4th. Edition. New York: Wiley, 573-687.

Aspray, W. & Burks, A. (Eds.) (1987). *Papers of John von Neumann on Computing and Computer Theory.* Vol. 12 in the Charles Babbage Institute Reprint Series for the History of Computing. Cambridge, MA: MIT Press.

Backus, J. (1978). Can Programming be Liberated from the von Neumann style? A Functional Style and Its Algebra of Programs. *Communications of the Association for Computing Machinery*, **21(8)**, 613-641.

Baddeley, A. (1986). *Working Memory.* Oxford: Oxford University Press.

Barbacci, M.R. & Uehara, T. (1985). Computer Hardware Description Languages: The Bridge Between Software and Hardware. *Computer*, **18(2)**, 6-8.

Barkow, J.H., Cosmides, L., & Tooby, J. (1992). *The Adapted Mind. Evolutionary Psychology and the Generation of Culture.* Oxford: Oxford University Press.

Barr, M.L. & Kiernan, J.A. (1988). *The Human Nervous System. An Anatomical Viewpoint.* Fifth Edition, Philadelphia PA: J.B. Lippincott Company.

Bartee, T.C. (1981). *Digital Computer Fundamentals.* Fifth Edition. McGraw-Hill Inc.

Barwise, J. (1975). *Admissible Sets and Structures. An Approach to Definability Theory.* Berlin: Springer-Verlag.

Barwise, J. & Etchemendy, J. (1986). *Turing's World: A Computer-based Introduction to Computability Theory.* Ventura CA : Kinko's Academic Courseware Exchange.

Bechtel, W. (1988). Connectionism and Rules and Representation Systems: are they compatible? *Philosophical Psychology*, **1(1)**, 5-16.

Bechtel, W. & Abrahamsen, A. (1991). *Connectionism and the Mind. An Introduction to Parallel Processing in Networks.* Oxford: Basil Blackwell.

Berkeley, E.C. (1949). *Giant Brains or Machines that Think.* London: Chapman & Hall.

Block, N. (1980). Troubles with functionalism. In N. Block (Ed.) *Readings in Philosophy of Psychology. Volume 1.* London: Methuen, 268-305.

Block, N. Ed. (1981). *Readings in Philosophy of Psychology: Volume 2.* London: Methuen.

Block, N. & Fodor, J.A. (1972). What psychological states are not. *Philosophical Review*, **81**, 159-181.

Boden, M.A. (1988). *Computer Models of Mind.* Cambridge: Cambridge University Press.

Boden, M.A. (Ed.) (1990). *The Philosophy of Artificial Intelligence.* Oxford: OUP. Oxford Readings in Philosophy.

Bolliet, L. (1968). Compiler Writing Techniques. In Genuys, F. (Ed.) *Programming Languages.* London: Academic Press, 113-289.

Boolos, G.S. & Jeffrey, R.C. (1989). *Computability and Logic.* Third Edition, Cambridge: Cambridge University Press.

Borland (1985). *Turbo Pascal Version 3.0. Reference Manual.* Scotts Valley, CA : Borland International Inc.

Borland (1991). *Turbo Assembler Version 2.5 User's Guide.* Scotts Valley, CA: Borland International Inc.

Bowden, B.V. (Ed.) (1953). *Faster than Thought. A symposium on Digital Computing Machines.* London: Pitman.

Brand, M. & Harnish, R.M. (1986). *The Representation of Knowledge and Belief.* Tucson: The University of Arizona Press.

Broadbent, D. (1985). A Question of Levels: Comment on McClelland and Rumelhart. *Journal of Experimental Psychology: General*, **114(2)**, 189-192.

Brooks, R.A. (1990). Elephants Don't Play Chess. *Robotics and Autonomous Systems*, **6**, 3-15.

Brooks, R.A. (1991). Intelligence without representation. *Artificial Intelligence*, **47**, 139-159.

Bundy, A. Ed. (1984). *Catalogue of Artificial Intelligence Tools.* Berlin: Springer-Verlag.

Burks, A.W. (1966). Editor's Introduction to Von Neumann (1966).

Burks, A.W., Goldstine, H.H., & von Neumann, J. (1947). Preliminary Discussion of the Logical Design of an Electronic Computing Instrument. Reprinted in Aspray, W. & Burks, A. (Eds.) (1987).

Carpenter, B.E. & Doran, R.W. (1977). The Other Turing Machine. *Computer Journal*, **20(3)**, 269-279.

Carpenter, B.E. & Doran, R.W. (Eds.) (1986). *A.M. Turing's ACE Report of 1946 and Other Papers.* Vol. 10 in the Charles Babbage Institute Reprint Series for the History of Computing. Cambridge, MA: MIT Press.

Changeux, J-P., & Dehaene, S. (1989). Neuronal models of cognitive functions. *Cognition*, **33**, 63-109.

Chater, N. (1991). Learning to respond to structure in time. Paper given at the British Psychological Society Annual Conference, Bournemouth, April 1991.

Church, A. (1936). An Unsolvable Problem of Elementary Number Theory. *The American Journal of Mathematics*, **58**, 345-363. Reprinted in Davis (1965), pps. 89-107

Churchland, P.M. (1989). *A Neurocomputational Perspective. The Nature of Mind*

and the Structure of Science. Cambridge, MA: MIT Press.

Churchland, P.S. (1986). *Neurophilosophy : Toward a Unified Science of the Mind-Brain.* Cambridge Massachusetts : The MIT Press, A Bradford Book.

Clark, A. (1989). *Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing.* Cambridge, MA: MIT Press. A Bradford Book.

Clark, A. (1993). *Associative Engines. Connectionism, Concepts, and Representational Change.* Cambridge, MA: MIT Press.

Clark, A. & Karmiloff-Smith, A. (1990). The Cognizer's Innards: A psychological and philosophical perspective on the development of thought. Manuscript.

Cleeremans, A. (1993). *Mechanisms of Implicit Learning. Connectionist Models of Sequence Processing.* Cambridge, MA: MIT Press.

Clocksin, W.F. & Mellish, C.S. (1984). *Programming in Prolog.* Second Edition. Springer-Verlag.

Conrad, M. (1974). Molecular Information Processing in the Central Nervous System. Part I: Selection Circuits in the Brain. In Conrad, M., Güttinger, W., & Dal Cin, M. (Eds.) *Physics and Mathematics of the Nervous System.* Berlin: Springer-Verlag, 82-107.

Conrad, M. (1985). On design principles for a molecular computer. *Communications of the Association for Computing Machinery*, **28(5)**, 464-480.

Conrad, M. (1988). The Price of Programmability. In Herken, R. (Ed.) *The Universal Turing Machine. A Half Century Survey.* Oxford: Oxford University Press, 285-307.

Cooper, R. & Franks, B. (1993). Interruptibility as a constraint on hybrid systems. *Minds and Machines*, **3**, 73-96.

Cosmides, L. (1989). The logic of social exchange: Has natural selection shaped how humans reason? Studies with the Wason selection task. *Cognition*, **31**, 187-276.

Cotman, C.W. & Lynch, G.S. (1989). The neurobiology of learning and memory. *Cognition*, **33**, 201-241.

Cotterill, R.M.J. (Ed.) (1988). *Computer simulation in brain science.* Cambridge, Cambridge University Press.

Craik, F.I.M., & Lockhart, R.S. (1972). Levels of processing: A framework for memory research. *Journal of Verbal Learning and Verbal Behavior*, **11**, 671-684.

Craik, K.J.W. (1943). *The Nature of Explanation.* Cambridge: Cambridge University Press.

Craik, K.J.W. (1966). *The Nature of Psychology. Writings by the Late Kenneth Craik.* Ed. S.L. Sherwood. Cambridge: Cambridge University Press.

Crawford, J.H. & Gelsinger, P.P. (1987). *Programming the 80386.* Alameda CA: Sybex Inc.

Cripps, M. (1977). *An Introduction to Computer Hardware.* London: Edward Arnold.

Cummins, R. (1989). *Meaning and Mental Representation.* Cambridge MA : MIT Press, A Bradford Book.

Damasio, A.R. (1989). Time-locked mulitregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. *Cognition*, **33**,

25-62.
Davidson, D. (1981). The Material Mind. Chapter 12 in Haugeland (1981).
Davies, M. Connectionism, modularity, and tacit knowledge. *British Journal for the Philosophy of Science*, **40**, 541-555.
Davis, M. (1965). *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*. New York: Raven Press Books Ltd.
Davis, M. (1988a). Mathematical Logic and the Origin of Modern Computers. In Herken (1988), pp.149-174.
Davis, M. (1988b). Influences of Mathematical Logic on Computer Science. In Herken (1988), pp.315-326.
Dennett, D.C. (1978). *Brainstorms: Philosophical Essays on Mind and Psychology*. Hassocks, Sussex : Harvester Press.
Dennett, D.C. (1984). Cognitive wheels: the frame problem of AI. In Hookway, C. (Ed.). *Minds, Machines and Evolution*. Cambridge, CUP. pps. 129-151.
Dennett, D.C. (1986). The Logical Geography of Computational Approaches: A View from the East Pole. In Brand & Harnish, Eds. (1986), pp. 59-79.
Dennett, D.C. (1987). *The Intentional Stance*. Cambridge, MA: MIT Press. A Bradford Book.
Dijkstra, E.W. (1987). The Humble Programmer. 1972 ACM Turing Award Lecture. Reprinted in *ACM Turing Award Lectures. The First Twenty Years*, pp.17-32. Addison-Wesley Publishing Company.
Douglas, R.J., & Martin, K.A.C., (1990). Neocortex. In G.M. Shepherd (Ed.) *The Synaptic Organization of the Brain*. Third Edition, Oxford: Oxford University Press, 389-438.
Eckmiller, R. & von der Malsburg, C. (Eds.) (1989). *Neural Computers*. Berlin: Springer-Verlag.
Edelman, G.M. (1989). *Neural Darwinism. THe Theory of Neuronal Group Selection*. Oxford: Oxford University Press.
Eimas, P.D., Miller, J.L., & Jusczyk, P.W. (1987). On infant speech perception and the acquisition of language. In S. Harnad (Ed.) *Categorical Perception. The groundwork of cognition*. Cambridge: Cambridge University Press, 161-195.
Elman, J.E. (1990). Finding Structure in Time. *Cognitive Science*, **14(2)**, 179-211.
Fahlman, S.E. (1979). *NETL: A System for Representing and Using Real-World Knowledge*. Cambridge, MA: MIT Press.
Farmer, J.D. (1990). A Rosetta Stone for Connectionism. *Physica D*, **42**, 153-187.
Feigenbaum, E.A., & Feldman, J. (Eds.) (1963). *Computers and Thought*. New York: McGraw-Hill Book Company.
Feldman, J.A. & Ballard, D.H. (1982). Connectionist Models and Their Properties. *Cognitive Science*, **6**, 205-254.
Fodor, J.A. (1975). *The Language of Thought*. Hassocks, Sussex: The Harvester Press.
Fodor, J.A. (1980). Methodological Solipsism Considered as a Research Strategy

in Cognitive Psychology. *The Behavioral and Brain Sciences*, **3**, 63-109. Reprinted in Haugeland (1981), pp. 307-338.

Fodor, J.A. (1981). Some notes on What Linguistics is About. Introduction to Part Three of Block (1981).

Fodor, J.A. (1983). *The Modularity of Mind. An Essay on Faculty Psychology.* Cambridge, MA: MIT Press. A Bradford Book.

Fodor, J.A. (1985). Fodor's Guide to Mental Representation: The Intelligent Auntie's Vade-Mecum. *Mind*, **94**, 76-100. Reprinted in Fodor, J.A. (1990). *A Theory of Content and Other Essays.* Cambridge, MA: MIT Press, 3-29.

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

Fodor, J.A. & Pylyshyn, Z.W. (1981). How direct is visual perception?: Some reflections on Gibson's "Ecological Approach". *Cognition*, **9**, 139-196.

Fodor, J.A. & Pylyshyn, Z.W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, **28**, 3-71.

Foster, C.L. (1990). *Algorithms, Abstraction and Implementation: A Massively Multilevel Theory of Strong Equivalence of Complex Systems.* Ph.D. University of Edinburgh.

Fox, G.C. & Messina, P.A. (1987). Advanced Computer Architectures. *Scientific American*, **257(4)**, 45-52.

Freyd, J.J. (1983). Representing the dynamics of a static form. *Memory and Cognition*, **11(4)**, 342-346.

Freyd, J.J. (1987). Dynamic Mental Representations. *Psychological Review*, **94(4)**, 427-438.

Frost, R.A. (1986). *Introduction to Knowledge Base Systems.* London: Collins.

Gandy, R. (1980). Church's Thesis and Principles for Mechanisms. In Barwise, J., Keisler, H.J., & Kunen, K. (Eds), *The Kleene Symposium.* Amsterdam: North-Holland Publishing Company, 123-148.

Gandy, R. (1988). The Confluence of Ideas in 1936. In Herken (1988), pp. 55-111.

Gardner, H. (1985). *The Mind's New Science. A History of the Cognitive Revolution.* New York: Basic Books. Issued in paperback with an epilogue by the author, 1987.

Gelernter, D. (1987). Programming for Advanced Computing. *Scientific American*, **257(4)**, 65-71.

Gibson, E.J., & Walk, R.D. (1960). The "visual cliff". *Scientific American*, **202**, 64-71.

Gibson, J.J. (1986/1979). *The Ecological Approach to Visual Perception.* Hillsdale, NJ: Lawrence Erlbaum Associates. Originally published in 1979.

Godden, D.R., & Baddeley, A.D. (1975). Context-dependent memory in two natural environments: On land and under water. *British Journal of Psychology*, **66**, 325-331.

Goldstine, H.H. (1972). *The Computer from Pascal to von Neumann.* Princeton, NJ: Princeton University Press.

Greenbaum, A. (1989). Synchronization costs on multiprocessors. *Parallel Computing*, **10**, 3-14.

Grossberg, S. (1987). Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, **11**, 23-63. Reprinted in Grossberg (1988, pp.213-250).

Grossberg, S. (1988). Ed. *Neural Networks and Natural Intelligence*. Cambridge, MA: MIT Press.

Hack, J.J. (1989). On the promise of general-purpose parallel computing. *Parallel Computing*, **10**, 261-275.

Hanson, S.J. & Burr, D.J. (1990). What connectionist models learn: Learning and representation in connectionist networks. *Behavioral and Brain Sciences*, **13(3)**, 471-518.

Harnad, S. (Ed.) (1987). *Categorical Perception. The Groundwork of Cognition.* Cambridge: CUP.

Harnad, S. (1990). The Symbol Grounding Problem. *Physica D*, **42**, 335-346.

Haugeland, J. Ed. (1981). *Mind Design: Philosophy, Psychology, Artifical Intelligence.* Cambridge MA: MIT Press, A Bradford Book.

Haugeland, J. (1985). *Artificial Intelligence: The Very Idea.* Cambridge, MA: MIT Press, A Bradford Book.

Hawthorne, J. (1989). On the Compatibility of Connectionist and Classical Models. *Philosophical Psychology*, **2(1)**, 5-15.

Hendriks-Jansen, H. (1994). Brain-Models, Mind-Modesl and Models of Situated Behaviour. *AISB Quarterly*, **87**, 29-35.

Herken, R. Ed. (1988). *The Universal Turing Machine A Half-Century Survey.* Oxford: Oxford University Press.

Hillis, W.D. (1985). *The Connection Machine.* Cambridge, MA: MIT Press.

Hinton, G.E. & Sejnowski, T.J. (1986). Learning and Relearning in Boltzmann Machines. In Rumelhart, McClelland & the PDP Research Group (1986), Ch.7.

Hitachi (1984). *Hitachi 8-Bit - 16-Bit Microprocessor and Peripheral Data Book.* Catalog No. D88/16 MC 8404. Hitachi (UK) Ltd.

Hitachi (no date). *Hitachi IC Memory Data Book.* DBICM/8404. Hitachi (UK) Ltd.

Hoare, C.A.R. (1985). *Communicating Sequential Processes.* London: Prentice-Hall International, UK, Ltd.

Hochberg, J. (1968). In the Mind's Eye. In Haber, R.N. (Ed.) *Contemporary Theory and Research in Visual Perception.* London: Holt, Rinehart & Winston.

Hockney, R.W. & Jesshope, C.R. (1988). *Parallel Computers 2. Architecture, Programming and Algorithms.* Bristol, Adam Hilger.

Hodges, A. (1983). Alan Turing: The Enigma. London: Vintage Books.

Hodges, A. (1988). Alan Turing and the Turing Machine. In Herken (1988), Part 1, pp.3-15.

Holland, J.H., Holyoak, K.J., Nisbett, R.E. & Thagard, P.R. (1987). *Induction: Processes of Inference, Learning, and Discovery.* Cambridge, MA: MIT Press.

Horel, J.A. (1979). Lost Maps and Memories. Commentary on O'Keefe & Nadel: Hippocampus as cognitive map. *Behavioral and Brain Sciences*, **2(4)**, 506-507.

Hughes, G.E. & Cresswell, M.J. (1968). *An Introduction to Modal Logic.* London: Methuen.

Jackendoff, R. (1976). Toward an Explanatory Semantic Representation. *Linguistic Inquiry*, **7(1)**, 89-150.

Jackendoff, R. (1978). Grammar as Evidence for Conceptual Structure. In Halle, M., Bresnan, J., & Miller, G.A. (Eds.) *Linguistic Theory and Psychological Reality*. Cambridge, MA: MIT Press, 201-228.

Jackendoff, R. (1983). *Semantics and Cognition*. Cambridge, MA: MIT Press.

Jackendoff, R. (1987). *Consciousness and the Computational Mind*. Cambridge Massachusetts : The MIT Press, A Bradford Book.

Jensen, K. & Wirth, N. (1974). *PASCAL: User Manual and Report*. Second Edition. Berlin: Springer-Verlag.

Johnson, M.H., & Morton, J. (1991). *Biology and cognitive development: The case of face recognition*. Oxford: Basil Blackwell.

Johnson, S.C. (1975). YACC - yet another compiler compiler. *Computing Science Technical Report 32*, Murray Hill, NJ: AT&T Bell Laboratories.

Johnson-Laird, P.N. (1983). *Mental Models : Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge : Cambridge University Press.

Jordan, M.I. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kaelbling, L.P., & Rosenschein, S.J. (1990). Action and Planning in Embedded Agents. *Robots and Autonomous Systems*, **6**, 35-48.

Karmiloff-Smith, A. (1992). *Beyond Modularity. A Developmental Perspective on Cognitive Science*. Cambridge, MA: MIT Press.

Kelly, P. (1989). *Functional Programming for Loosely-Coupled Multiprocessors*. Research Monographs in Parallel and Distributed Computing. London: Pitman.

Kirk, R. (1986). Mental Machinery and Gödel. *Synthese*, **66**, 437-452.

Knuth, D.E. (1973). *The Art of Computer Programming. Volume One: Fundamental Algorithms*. 2nd. Edition. Reading MA : Addison- Wesley.

Knuth, D.E. & Pardo, L.T. (1980). The Early Development of Programming Languages. In Metropolis, Howlett, & Rota (Eds.) (1980).

Kohonen, T. (1988). *Self-Organization and Associative Memory*. Second Edition. Berlin: Springer-Verlag.

Krellenstein, M. (1987). A Reply to "Parallel Computation and the Mind-Body Problem". *Cognitive Science*, **11**, 155-157.

Kuffler, S.W., Nicholls, J.G. & Martin, A.R. (1984). *From Neuron to Brain. A Cellular Approach to the Function of the Nervous System*. 2nd. Edition. Sunderland, MA: Sinauer Associates Inc.

Larson, J., Wong, D., & Lynch, G. (1986). Patterned stimulation at the theta frequency is optimal for induction of long-term potentation. *Brain Research*, **368**, 7-35.

Lee, C.Y. (1963). A Turing Machine which prints its own code script. *Proceedings of the Symposium on Mathematical Theory of Automata*, Brooklyn, NY: Polytechnic Press of the Polytechnic Institute of Brooklyn,

155-164

Leslie, A.M. (1987). Pretense and Representation: The Origins of "Theory of Mind". *Psychological Review*, **94(4)**, 412-426.

Leventhal, L.A. (1979). *6502 Assembly Language Programming*. Berkeley, CA : Osborne/McGraw-Hill.

Lewis, H.R. & Papadimitriou, C.H. (1981). *Elements of the Theory of Computation*. Englewood Cliffs, NJ : Prentice-Hall Inc.

Lipschutz, S. (1976). *Schaum's Outline of Theory and Problems of Discrete Mathematics*. London: McGraw-Hill Book Company.

Lockwood, M. (1989). *Mind, Brain and the Quantum. The Compound 'I'*. Oxford: Basil Blackwell.

Longuet-Higgins, H.C. (1987). *Mental Processes. Studies in Cognitive Science*. Cambridge, MA: MIT Press. A Bradford Book.

Mahowald, M.A. & Mead, C. (1991). The Silicon Retina. *Scientific American*, **264(5)**, 40-46.

Mandler, J.M., & Bauer, P.J. (1988). The cradle of categorization: Is the basic level basic? *Cognitive Development*, **3**, 247-264.

Marr, D. (1982). *Vision : A Computational Investigation into the Human Representation and Processing of Visual Information*. San Francisco : W.H. Freeman & Co.

Maurer, D. (1985). Infants' perception of facedness. In T. Fields & N. Fox, (Eds.), *Social Perception in infants*. Norwood, NJ: Ablex, 73-100.

McClelland, J.L., Rumelhart, D.E. & the PDP Research Group, (1986). *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge, MA: MIT Press. A Bradford Book.

McCormick, D.A. (1990). Membrane Properties and Neurotransmitter Actions. In G.M. Shepherd, (Ed.), *The Synaptic Organization of the Brain. Third Edition*. Oxford: Oxford University Press.

McCulloch, W.S. (1965). *Embodiments of Mind*. Cambridge, MA: MIT Press.

McCulloch, W.S. & Pitts, W.H. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, **(5)**, 115-133. Reprinted in Boden (1990), pp.22-39.

Mead, C. (1989). *Analog VLSI and Neural Systems*. Reading MA: Addison-Wesley Publishing Company.

Mead, G.H. (1934). *Mind, Self, and Society from the Standpoint of a Social Behaviorist*. Edited with an introduction by Charles W. Morris, Chicago: University of Chicago Press.

Mehler, J. & Fox, R. (Eds.) (1985). *Neonate Cognition: Beyond the Blooming Buzzing Confusion*. Hillsdale, NJ : Lawrence Erlbaum Associates.

Metropolis, N., Howlett, J., & Rota, Gian-Carlo (Eds.) (1980). *A History of Computing in the Twentieth Century. A collection of essays*. New York: Academic Press.

Meyer, B. (1988). *Object-Oriented Software Construction*. London: Prentice-Hall International (UK) Ltd.

Miller, G.A., Galanter, E., & Pribram, K.H. (1960). *Plans and the Structure of Behavior*. Holt, Rinehart and Winston, Inc.

Milner, R. (1989). *Communication and Concurrency*. London: Prentice Hall.

Minsky, M.L. (1967). *Computation: Finite and Infinite Machines*. New Jersey: Prentice-Hall Inc.

Minsky, M.L. & Papert, S.A. (1969,1988). *Perceptrons. An Introduction to Computational Geometry*. First Edition 1969, Expanded Edition 1988. Cambridge, MA : MIT Press.

Moore, W.R. (1989). Conventional Fault-Tolerance and Neural Computers. In Eckmiller & von der Malsburg (1989) pp.29-37.

Morse, S.P. (1982). *The 8086/8088 Primer. An Introduction to Their Architecture, System Design, and Programming*. Second Edition. Rochelle Park, NJ: Hayden Book Co. Inc.

Morton, J., & Johnson, M.H. (1991). CONSPEC and CONLERN: A two-process theory of infant face recognition. *Psychological Review*, **98**, 164-181.

Mozer, M.C. & Smolensky, P. (1989). Using Relevance to Reduce Network Size Automatically. *Connection Science*, **1(1)**, 3-16.

Nadel, L., Cooper, L.A., Culicover, P., & Harnish, R.M. (Eds.) (1989). *Neural Connections, Mental Computation*. Cambridge, MA: MIT Press. A Bradford Book.

Nadel, L., Willner, J., & Kurz, E. (1986). The Neurobiology of Mental Representation. In M. Brand & R.M. Harnish, Eds. (1986), *The Representation of Knowledge and Belief*. Tucson: The University of Arizona Press.pp. 219-257.

Neisser, U. (Ed.) (1987). *Concepts and conceptual development: Ecological and intellectual factors in categorization*. Cambridge: Cambridge University Press.

Nelson, R.J. (1987). Machine models for Cognitive Science. *Philosophy of Science*, **54**, 391-408.

Nelson, R.J. (1989). *The Logic of Mind*. 2nd. Edition. Amsterdam: Kluwer Academic Publishers.

Newell, A. (1980). Physical Symbol Systems. *Cognitive Science*, **4**, 135-183.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, **18**, 87-127.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Newell, A., Shaw, J.C., & Simon, H.A. (1958). Elements of a Theory of Human Problem Solving. *Psychological Review,*, **65(3)**, 151-166.

Newell, A. & Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Newell, A. & Simon, H.A. (1976). Computer Science as Empirical Inquiry: Symbols and Search. *Communications of the Association for Computing Machinery*, **19**, 113-126. Reprinted in Haugeland (1981, pp.35-66).

Nilsson, N.J. (1982). *Principles of Artificial Intelligence*. Berlin : Springer-Verlag.

Norman, D.A. (1986). Reflections on Cognition and Parallel Distributed Processing. In J.L. McClelland, D.E. Rumelhart & the PDP Research Group, (1986). *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge, MA: MIT Press, 531-546.

Norris, D. (1991). The constraints on connectionism. *The Psychologist*, **4(7)**, 293-296.

O'Keefe, J. (1989). Computations the hippocampus might perform. In Nadel, L., Cooper, L.A., Culicover, P., & Harnish, R.M. (Eds.) *Neural Connections, Mental Computation*. Cambridge, MA: MIT Press, 225-284.

O'Keefe, J. & Nadel, L (1979). Multiple book review of J. O'Keefe and L. Nadel. The Hippocampus as a cognitive map. *The Behavioral and Brain Sciences*, **2(4)**, 487-533.

Patton, P.C. (1985). Multiprocessors: Architecture and Applications. *Computer*, **18(6)**, 29-40.

Pellionisz, A.J. (1988). Vistas from tensor network theory: a horizon from reductionalist neurophilosophy to the geometry of multi-unit recordings. In Cotterill (1988), pp.44-73.

Penrose, R. (1990). *The Emperor's New Mind. Concerning Computers, Minds and the Laws of Physics*. Oxford: OUP 1989. Vintage paperback edition 1990.

Pinker, S. (1994). *The Language Instinct. The New Science of Language and Mind*. London: Allen Lane, The Penguin Press.

Pinker, S., & Mehler, J. (1988). Guest Editors' Introduction. *Cognition*, **28**, 1-2.

Pinker, S. & Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, **28**, 73-193.

Posner, M. (Ed.) (1989). *Foundations of Cognitive Science*. Cambridge, MA: MIT Press. A Bradford Book.

Post, E.L. (1936). Finite Combinatory Processes. Formulation I. *Journal of Symbolic Logic*, **(1)**, 103-105. Reprinted in Davis (1965,pp.289-291).

Post, E.L. (1947). Recursive Unsolvability of a Problem of Thue. *Journal of Symbolic Logic*, **(12)**, 1-11. Reprinted in Davis (1965, pp.293-303).

Pratt, T.W. (1984). *Programming Languages: Design and Implementation*. Second Edition. Englewood Cliffs, NJ: Prentice-Hall Inc.

Premack, D. (1991). The Infant's Theory of Self-Propelled Objects. In D. Frye & C. Moore (Eds.) *Children's Theories of Mind. Mental States and Social Understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates, 39-48.

Putnam, H. (1960). Minds and Machines. In Putnam (1975) pp.362-385.

Putnam, H. (1967). The Mental Life of Some Machines. In Putnam (1975), pp.408-428.

Putnam, H. (1973). Philosophy and our Mental Life. In Putnam (1975), pp.291-303.

Putnam, H. (1975). *Mind, Language and Reality. Philosophical Papers. Volume 2*. Cambridge: Cambridge University Press.

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

Pylyshyn, Z.W. (1980). Computation and cognition: Issues in the foundation of cognitive science. *The Behavioral and Brain Sciences*, **3(1)**, 111-169.

Pylyshyn, Z.W. (1984). *Computation and Cognition: Toward a Foundation for Cognitive Science*. Cambridge, MA. : MIT Press.

Pylyshyn, Z.W. (1989). Computing in Cognitive Science. Chapter 2 in Posner (Ed.) (1989).

Ramsey, W.M. (1989). Parallelism and Functionalism. *Cognitive Science*, **13**,

139-144.

Randell, B. (Ed.) (1973). *The Origins of Digital Computers. Selected Papers.* New York, Springer-Verlag.

Renwick, W. (1950). The E.D.S.A.C. Demonstration. *Report of a Conference on High Speed Automatic Calculating Machines.* Cambridge, University Mathematical Laboratory. Reprinted in Williams & Campbell-Kelly (Eds.) (1989), pps. 21-26.

Rosen, S. (1969). Electronic Computers: A Historical Survey. *Computing Surveys*, **1(1)**, 7-36.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* Washington: Spartan Books.

Rosenschein, S.J. (1985). Formal Theories of Knowledge in AI and Robotics. *New Generation Computing*, **3(4)**, 345-357.

Rumelhart, D.E., Hinton, G.E. & McClelland, J.L. (1986). A General Framework for Parallel Distributed Processing. Chapter 2 in Rumelhart, McClelland et al. (1986).

Rumelhart, D.E., Hinton, G.E. & Williams R.J. (1986) Learning Internal Representations by Error Propagation. Chapter 8 in Rumelhart, McClelland et. al. (1986).

Rumelhart, D.E. & McClelland, J.L. (1985). Levels Indeed! A Response to Broadbent. *Journal of Experimental Psychology: General*, **114(2)**, 193-197.

Rumelhart, D.E., McClelland, J.L., & the PDP Research Group (1986). *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press, A Bradford Book.

Sammet, J.E. (1969). *Programming Languages: History and Fundamentals.* Englewood Cliffs, NJ: Prentice-Hall Inc.

Schnelle, H. (1988). Turing Naturalized: Von Neumann's Unfinished Project. In Herken (1988), pp. 539-559.

Servan-Schreiber, D., Cleeremans, A., & McClelland, J.L. (1989). Learning sequential structure in simple recurrent networks. In D.S. Touretzky (Ed.) *Advances in Neural Information Processing Systems 1.* San Mateo, CA: Morgan Kaufman, 643-652.

Shallice, T. (1988). *From Neuropsychology to Mental Structure.* Cambridge: Cambridge University Press.

Sharp, J.A. (1985). *Data Flow Computing.* Chichester: Ellis Horwood Ltd.

Shepard, R.N. (1989). Internal Representation of Universal Regularities: A Challenge for Connectionism. In Nadel et. al. (Eds.) (1989), pps. 104-134.

Shepherdson, J.C. (1975). Computation over Abstract Structures: Serial and Parallel Procedures and Friedman's Effective Definitional Schemes. *Logic Colloquium '73 (Eds. H.E. Rose, & J.C. Shepherdson.* Amsterdam: North-Holland Publishing Co., 445-513.

Shepherdson, J.C. (1988). Mechanisms for Computing over Arbitrary Structures. In Herken, R. (Ed.) *The Universal Turing Machine. A Half Century Survey.* Oxford, OUP., 581-601.

Siewiorek, D.P., Bell, C.G. & Newell, A. (1982). *Computer Structures: Principles and Examples.* International Edition, McGraw-Hill Book Company.

Simpson, P.K. (1990). *Artificial Neural Systems. Foundations, Paradigms,*

Applications, and Implementations. New York: Pergamon Press.

Smith, E.E., & Medin, D.L. (1981). *Categories and Concepts*. Cambridge, MA: Harvard University Press.

Smith, S.M., Glenberg, A., & Bjork, R.A. (1978). Environmental context and human memory. *Memory and Cognition*, **6**, 342-353.

Smolensky, P. (1986). Neural and Conceptual Interpretation of PDP Models. In J.L. McClelland, D.E. Rumelhart, and the PDP Research Group, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge, MA: MIT Press, 390-431.

Smolensky, P. (1988). On the proper treatment of connectionism. *The Behavioral and Brain Sciences*, **11**, 1-74.

Smolensky, P. (1989). Connectionist Modeling: Neural Computation/Mental Connections. In Nadel et. al. (1989), pp.49-67.

Smolensky, P. (1991). Connectionism, Constituency, and the Language of Thought. In B. Loewer, & G. Rey (Eds) *Meaning in Mind. Fodor and his Critics*. Oxford: Basil Blackwell, 201-227.

Spelke, E.S. (1985). Perception of Unity, Persistence, and Identity: Thoughts on Infants' Conceptions of Objects. In Mehler & Fox (1985), Chapter 6, pp.89-113.

Spelke, E.S. (1990). Principles of object perception. *Cognitive Science*, **14**, 29-56.

Squire, L.R. (1979). The hippocampus, space, and human amnesia. Commentary on O'Keefe & Nadel: Hippocampus as cognitive map. *The Behavioral and Brain Sciences*, **2(4)**, 514-515.

Steele, G.L. Jnr. (1984). *Common Lisp. The Language*. Digital Press.

Stoffregen, T.A. & Riccio, G.E. (1988). An Ecological Theory of Orientation and the Vestibular System. *Psychological Review*, **95(1)**, 3-14.

Tatar, D.G. (1987). *A Programmer's Guide to COMMON LISP*. Bedford MA: Digital Press.

Taylor, J.G. (1989). Living neural nets. In Taylor & Mannion (1989), pp.31-52.

Taylor, J.G. & Mannion, C.L.T. (Eds.) (1989). *New Developments in Neural Computing*. Bristol: Adam Hilger, IOP Publishing Ltd.

Taylor, S. (1989). *Parallel Logic Programming Techniques*. Englewood Cliffs, NJ: Prentice-Hall International Inc.

Thagard, P. (1986). Parallel Computation and the Mind-Body Problem. *Cognitive Science*, **10**, 301-318.

Thagard, P. (1987). Reply to Krellenstein on Parallel Computation. *Cognitive Science*, **11**, 159-161.

Thatcher, J.W. (1963). The Construction of a Self-Describing Turing Machine. *Proceedings of the Symposium on Mathematical Theory of Automata*, Brooklyn, NY: Polytechnic Press of the Polytechnic Institute of Brooklyn, 165-171.

Thompson, J.M.T. & Stewart, H.B. (1986). *Nonlinear Dynamics and Chaos. Geometrical Methods for Engineers and Scientists*. Chichester : John Wiley and Sons.

Tulving, E. (1983). *Elements of episodic memory*. Oxford: Oxford University Press.

Turing, A.M. (1936-7). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, **ser. 2, vol. 42**, 230-265. Reprinted in Davis (1965) pp. 116-154.

Turing, A.M. (1939). Systems of Logic Based On Ordinals. *Proceedings of the London Mathematical Society*, **ser.2, vol.45,** 161-228. Reprinted in Davis (1965), pp.155-222.

Turing, A.M. (1946). Proposal for Development in the Mathematics Division of an Automatic Computing Engine (ACE). Reprinted in Carpenter & Doran (Eds.) (1986), pps. 20-105.

Turing, A.M. (1947). Lecture to the London Mathematical Society on 20 February 1947. Reprinted in Carpenter & Doran (Eds) (1986), pps.106-124.

Turing, A.M. (1950). Computing Machinery and Intelligence. *Mind*, **59**, 433-460. Reprinted in Feigenbaum and Feldman (1963), pp.11-35, and in Boden (1990), pps. 40-66.

von Foerster, H. (Ed.) (1952). *Cybernetics. Circular Causal and Feedback Mechanisms in Biological and Social Systems*. Josiah Macy, Jr. Foundation.

von Neumann, J. (1945). First Draft of a Report on the EDVAC. Reprinted in Aspray & Burks (1987). Sections 1 to 5 reprinted in Randell (1973), pp.355-364.

von Neumann, J. (1951). The General and Logical Theory of Automata. *Collected Works, Volume 5*, pp.288-328. Ed. A.H. Taub, (1963), New York : MacMillan. Reprinted in von Neumann (1987), pps. 391-431.

von Neumann, J. (1956). Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components. *Collected Works, Volume 5*, pp.329-378. Ed. A.H. Taub, (1963), New York : MacMillan.

von Neumann, J. (1958). *The Computer and the Brain.* New Haven: Yale University Press.

von Neumann, J. (1966). *Theory of Self-Reproducing Automata.* Edited and completed by Arthur W. Burks. University of Illinois Press.

von Neumann, J. (1987). *Papers of John von Neumann on Computing and Computer Theory*. (Eds.) Aspray, W. & Burks, A. Cambridge, MA: MIT Press.

Vosniadou, S., & Ortony, A. (Eds.) (1989). *Similarity and Analogical Reasoning*. Cambridge, Cambridge University Press.

Walsh, K.W. (1978). *Neuropsychology. A clinical approach.* Edinburgh: Churchill Livingstone.

Wang, H. (1957). A Variant to Turing's Theory of Computing Machines. *Journal of the Association for Computing Machinery*, **4**, 63-92.

Wells, A.J. (1993). Parallel Architectures and Mental Computation. *British Journal for the Philosophy of Science*, **44**, 531-542.

Wiener, N. (1961). *Cybernetics: or Control and Communication in the Animal and the Machine*. Second Edition. Cambridge, MA: MIT Press.

Wilensky, R. (1986). *Common LISPcraft*. New York: W.W. Norton & Company.

Wilkes, M.V. (1951). The Best Way to Design and Automatic Calculating Machine. *Manchester University Computer. Inaugural Conference*. Reprinted in Williams & Campbell-Kelly (1989), pps. 182-184.

Wilkes, M.V. & Renwick, W. (1950). The EDSAC. *Report of a Conference on*

High Speed Automatic Calculating Machines. Cambridge, University Mathematical Laboratory. Reprinted in Williams & Campbell-Kelly (Eds.) (1989), pps. 16-20.

Wilks, Y. (1975). Putnam and Clarke and Mind and Body. *British Journal for the Philosophy of Science*, **26**, 213-225.

Williams, M.R. & Campbell-Kelly, M. (Eds.) (1989). *The Early British Computer Conferences*. Vol. 14 in the Charles Babbage Institute Reprint Series for the History of Computing. Cambridge, MA: MIT Press.

Woodger, M. (1958). The History and Present Use of Digital Computers at the National Physical Laboratory. Reprinted in Carpenter & Doran (Eds.) (1986), pps. 125-140.

Zeki, S. (1993). *A Vision of the Brain*. Oxford: Blackwell Scientific Publications.

Zipser, D. (1986). Biologically Plausible Models of Place Recognition and Goal Location. In McClelland, J.L., Rumelhart, D.E., & the PDP Research Group. *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge, MA: MIT Press, 432-470.