Cognition and the Computational Power of Connectionist Networks

Robert F. Hadley
School of Computing Science
and Cognitive Science Program
Simon Fraser University
Burnaby, B.C., V5A 1S6
hadley@cs.sfu.ca

Abstract

This paper examines certain claims of "cognitive significance" which (wisely or not) have been based upon the theoretical powers of two distinct classes of connectionist networks, namely, the "universal function approximators", and recurrent finite-state simulation networks. Each class will be considered with respect to its potential in the realm of cognitive modeling. Regarding the first class, I argue that, contrary to the claims of some influential connectionists, feed-forward networks do *not* possess the theoretical capacity to approximate all *functions of interest to cognitive scientists.* By contrast, I argue that a certain class of recurrent networks (i.e., those which closely approximate *deterministic finite automata*, DFA) shows considerably greater promise in some domains. However, serious difficulties arise when we consider how the relevant recurrent networks (RNNs) could acquire the *weight vectors* needed to support DFA simulations.

Introduction.

Do connectionist networks have the computational power, in theory, to provide successful explanatory models for *high-level* human cognition? Many connectionists believe so, and their confidence stems from a number of formal results that have appeared in the last decade. These 'computability' results pertain to network architectures which, on the face of it, avoid the shortcomings of the oft-cited McCulloch & Pitts (1943) implementation of conventional 'logic gates'. It is now widely recognized that McCulloch-Pitts 'neural circuitry' designs are radically unlike the kinds of neural structures found in actual brains, and offer no advantage over standard digital circuitry. By contrast, some network architectures involved in recent theoretical results bear at least a superficial resemblance to biological neural structures. This fact has inspired hope, in some quarters, that connectionist networks (abbreviated here as 'c-nets') may both (a) match the power of high-level, symbolic AI programs for explaining *higher cognitive functions* (such as abstract planning, mathematical reasoning, and theory formation), and (b) provide insight into how these higher functions could be instantiated in living brains. Among those who have appealed to the 'theoretical power' of multilayer c-nets (especially their purported 'universal' function approximation capacity) *within a cognitive context* are Churchland (1990), Elman et al (1996), and Niklasson & van Gelder (1994).

This paper will examine several frequently cited theoretical results with regard to their potential for satisfying both conditions (a) and (b) above. In particular, I focus upon whether the relevant proofs appeal to *processing models* which present a genuine alternative to conventional high-level symbolic processing. In addition, I explore whether the proposed models possess some measure of cognitive/biological plausibility. For even if a given c-net architecture essentially *implements* a classical machine, at the computational level, such an implementation would still be vastly important if it revealed how high-level classical processes could be realized in brain-like systems.

In what follows, I consider 'computability results' which fall into two major categories. These are: (1) 'universal' function approximators (emphatic scare quotes), (2) deterministic finite automata simulators. Papers belonging these categories are: **(1)** Hartman, Keeler, & Kowalski (1990); Hornik, Stinchcombe, & White (1989). **(2)** Casey, 1996; Cleerman, Servan-Schreiber, & McClelland (1989); Omlin & Giles (1996); Sontag (1995).

I shall argue that results in the first of the above categories *hold no promise* for explaining the abstract planning and reasoning capacities of humans. ('Universal' function approximators lack the requisite power – they are not universal.) By contrast, the second category (deterministic finite automata, DFA) offers some measure of hope provided the c-nets are designed in a fashion that ensures close mimicry of the state transitions of some DFA. However, even here, the existence of cognitively/biologically defensible training methods is highly problematic.

(Apart from the foregoing, a third class of computability results exists. This concerns *Turing-equivalent networks.* [See Siegelmann (1996); Siegelmann & Sontag, 1994] These are examined in detail in my extended technical report (Hadley, 1999), where I argue that Turing-equivalent networks *not only* require weights and nodes of infinite precision, but are at least as "brittle" and "hand-crafted" as classical symbolic algorithms. It should be noted that the "infinite precision" difficulties have also been explored by Sontag [1995]. Due to space constraints, I must here omit further discussion of

Turing-equivalent networks.)

2. Universal Function Approximators.

The thesis that multi-layer, feed-forward neural networks can be universal function approximators, derives primarily from two influential publications; Hornik, Stinchcombe & White (1989) and Hartman, Keeler & Kowalski (1990). Although the titles of both these publications contain the phrases 'universal approximators' and 'universal approximations', respectively, the text of each paper makes it clear that only a certain class of functions is being addressed. In both cases, only the class of measurable Borel functions is discussed, but this includes continuously valued functions. Hornik et al contend that this class covers 'virtually any function of interest'. Hartman et al do not address this issue, perhaps because they believed the cited work of Hornik et al had sufficiently explored the question. In any case, the difference between the proofs provided by these two groups of researchers concerns only the activation functions applied to hidden layer units, and does not concern the input-output mapping functions which are to be approximated. For this reason, my discussion will center upon Hornik et al, since theirs is the earlier work. My conclusions, however, apply equally well to any claim or 'proof' that finite multi-layer, feed-forward networks can be universal function approximators.

2.1 Problems with Universality.

Let us note at the outset that any given *measurable Borel function* is a set of ordered pairs, where each element of each ordered pair is a single *real* number. Since the output layer of any *actual* neural network can contain only a finite number of units, each having finite precision, any given output value produced for a given input can have only finite precision. Such an output value cannot *uniquely* approximate, in the general case, a specific real number, no matter how large the output layer may be. For, an infinity of real numbers will lie arbitrarily close to any fixed precision output value. In purely numerical domains, this may often be a negligible issue. However, when numerical output is being used to encode symbolic formulae or sentences, serious problems can arise. Admittedly, such difficulties can at times be obviated by a prudent choice of encoding schemes. Unfortunately, this is often not possible in domains associated with higher cognition.

For example, the realms of logic, mathematics, and linguistic theory each involve infinite sets of formulae or sentences. In addition, within these realms there are important functions which can map a given sentence (selected from an infinite set) onto another sentence (or even an infinite set of sentences). That is, such functions have infinite ranges and domains. Yet, the input/output layers of any physically realizable c-net can contain only a finite number of units, each having finite precision. Thus, these layers can accurately encode values spanning only a finite interval. If an infinite number of *formulae encodings* are to be 'packed' into a finite numerical span, there can be no lower bound on how close together a pair of encodings can be.[1]

Now, since a given output value will seldom *precisely* represent the target encoding, one will usually be forced to rely upon some *approximate* result to determine which formula was being output. However, this would commonly lead to disaster, since, in the present context, two numbers which encode distinct formulae should not be viewed as approximations of each other, no matter how numerically close the encodings are. For example, a given numerical code, C, may be very close to the encoding of some theorem, and yet C may encode some other formula which is logically invalid. This difficulty cannot be circumvented by taking some actually generated output value and searching for the *nearest* number which encodes a well-formed formula. For there simply is no "nearest formula encoding". Given any pair of formulae encodings, some other, distinct formula encoding would always lie between them. The brute fact is that formulae are discrete objects which can seldom be viewed as approximations of each other.

It might now be objected that the problem just described is not unique to neural networks. After all, no physically realizable computer could actually derive an infinity of symbolic theorems, or process arbitrarily long symbolic strings. However, this objection misses the *theoretical* point I am making. The point is that there are important 'functions of interest' which cannot be approximated by any feed-forward network, *even though* such networks can approximate many continuously valued functions. Functions which map formulae/sentences into other formulae/sentences are both interesting and important, as I explain below. Many of these functions cannot be approximated in any sense relevant to the foregoing discussion. Yet, as defined by automata theory, they are *computable* functions (i.e., recursive). We can write useful symbolic programs which *intensionally* embody these functions, and the programs will halt, though not always rapidly. This is true even though the ranges and domains of the computable functions may be infinite. An example of such a function, used by some automated theorem provers, is the clause-form conversion algorithm. It takes any sentence of first-order logic (FOL) and generates a set of *clauses*. This set can be written as a simple conjunction of formula.

Now, of course, a given computation involving a computable function can exceed a computer's memory resources. However, as Fodor and Pylyshyn observe (1988), one can always add more memory without having to alter the program, or the function being computed. This cannot be said, in general, of c-nets, where increasing the amount of memory (or nodes) will create a different weight configuration, thereby altering the function being computed.

Returning to the main thread, it might be objected that the foregoing discussion is misguided, since Hornik et al were addressing *measurable* nu-

[1] To see this, suppose there existed such a finite lower bound. Since there are an infinity of formulae encodings, and each pair of encodings is separated by at least this lower bound, then the entire spanned region would have to be infinitely broad.

merical functions. Functions which map symbolic expressions into symbolic expressions are not measurable in the relevant sense. In reply, I would emphasize two points. First, Hornik et al identify the class of functions they address with 'virtually any function of interest'. They even say that 'failures in application can be attributed to ... the presence of a stochastic rather than a deterministic relation between input and target'. Secondly, regardless of the original intent, Hornik et al's results have been construed by many cognitively motivated connectionists as a clear demonstration that, in principle, multilayer feed-forward c-nets can match the power of conventional AI programs. It is this construal (or misconstrual) which concerns me most.

My critique thus far has centered upon problems that arise when interpreting finite numerical output as approximate encodings of discrete symbolic strings. However, significantly deeper problems are encountered when we consider the kinds of (potentially non-halting) computations required by theorem provers in the realm of FOL and higher mathematics. For these realms, there exist working programs which prove interesting theorems. Moreover, standard AI textbooks present various means whereby theorem provers for FOL can be applied to practical problems such as planning, natural language interpretation, and even programming.

The theorem proving programs just alluded to are not guaranteed to halt (complete their computations) in general, though for many input values, they will halt and produce interesting output. However, these programs are usually deterministic and instantiate partial recursive functions (the relevant sets of theorems are recursively enumerable). Such functions comprise a major focus of recursive function theory. Furthermore, as far as we presently know, humans may at times employ high-level mental processes which are best modeled by non-halting programs of the kind being considered. One obvious example would involve mathematicians who discover complex proofs for existing conjectures. The most successful cognitive models in this area are high-level programs which employ both heuristic rules and subprograms that halt for some inputs but not others.

Now, programs which implement theorem provers for standard first-order logic and higher-order mathematical domains involve iterative processes. Their computational complexity is infinite, since no *a priori* limit can be set on the number of iterations involved. By contrast, non-recurrent, feed-forward networks have computational complexity which is linear. Such networks (implemented in parallel, as connectionist theory assumes) always complete their computations in time which is a linear function of the number of layers present. Moreover, unlike partial recursive functions (or their corresponding programs), these networks always produce an output for any given input. In light of this, it is extremely doubtful that multi-layer feed-forward networks could approximate the partial recursive functions under consideration. For example, consider the partial function which, given any suspected theorem of standard FOL, yields a formal proof for that sentence if and only if a proof exists. Certainly, no feed-forward network could reliably produce an 'approximate output' which could then be used to mechanically infer the desired theorem-proof, if one exists. For this supposition would entail that there exists an effective decision procedure for ascertaining theoremhood in the full first-order calculus. Such a decision procedure has been proven to be impossible (Church, 1956).

Moreover, nothing in Hornik et al (or in Hartman et al) would suggest that a feed-forward network could even *frequently* produce output that approximated the encoding of a target proof for a suspected theorem. Their proofs simply do not address the class of partial recursive functions. We know also, from the renowned work of Church (1956), Gödel (1931) and others, that a function which relates arbitrary sentences of FOL (or number theory) to their *supposed* proofs is not reducible to any arithmetic (or algebraic) equation. To be sure, the 'proof predicates' employed by Gödel in his famous incompleteness theorems correspond to algebraic formulae. However, these predicates apply (or 'hold') only in cases where a proof actually exists. Thus, these 'proof predicates' could not be embedded in a c-net's weight configuration in order to determine *whether* a suspected theorem had a proof. Any purported feed-forward net of this kind would produce output for theorems and non-theorems alike, and this output could not be used to distinguish the two cases.

It might now be objected that the difficulties just considered are innocuous, since it may appear that *partial recursive functions* (PRFs) are not *really* functions at all. After all, these "functions" are only partially defined; they fail to produce output for certain input values.

In reply, two points are relevant. First, as previously noted, PRFs form a major topic in recursive function theory. Researchers in this realm (including Alan Turing, Alonzo Church, and other giants) certainly have regarded PRFs as an important type of function, and recursive function theory is an essential field within both Mathematics and Computer Science. However, setting aside quibbles about the semantics of 'function', there remains the crucial point that theoretical proofs about "universal" function approximation have been cited by many connectionists as conclusive evidence that any mental *computation* could, in principle, be closely approximated by feed-forward networks. Here lies the crux of the matter. Now, certain of the PRFs cited above *have been used* to model high-level cognitive processes, and it is beyond dispute that computer programs embodying these PRFs actually perform computations. Moreover, it would be entirely question-begging for any connectionist to insist that the mental computations of interest could never be accurately modeled by PRFs. So, regardless of how narrowly we choose to construe the sense of 'function', the theoretical proofs in question simply have not established that all *mental computations of interest* can be approximated by feed-forward networks.

3. Recurrent Neural Networks and Deterministic Finite Automata.

In a 1989 paper, Cleermans et al offered a demon-

stration that simple recurrent networks "can learn to mimic closely a finite-state automaton (FSA) both in its behavior and in its state representations". Their demonstration was experimental, rather than formal. Using backpropagation, they successfully trained recurrent networks to induce comparatively simple deterministic finite automata (DFA). Experimental evidence of this nature could not, of course, establish any general equivalence between recurrent neural networks (RNNs) and DFA. Indeed, Elman's work with RNNs (1990, 1993) illustrated that simple recurrent nets, trained via backpropagation, may provide only limited approximations to DFA, in that network performance can significantly degrade when even moderately deep recursion is present within input strings. On a more encouraging note, the capacity of RNNs to simulate the *general class* of DFAs has been formally proven (see Sontag, 1995; Casey, 1996; Omlin & Giles, 1996). These proofs are significant because many powerful computational processes can be modeled by DFAs. (Whether high-level human cognition can, in general, be modeled by DFAs is less clear, however. We shall return to that issue below.)

Now, it is noteworthy that RNNs and DFAs are not equivalent classes. Given a particular DFA, there does exist some RNN whose I/O behaviour is equivalent to the DFA. However, there are many RNNs which fail to approximate any DFA to a degree sufficient to enable successful automaton simulation on long input strings. Omlin & Giles (1996) are emphatic on this point. They also provide an algorithm which partially *pre-determines* a network's weights, prior to training, in order to ensure that subsequent learning yields weight vectors that guarantee successful simulation. The resulting networks can achieve very close simulations of DFA, but the question naturally arises, do these simulations possess any advantage over classical DFA? For example, will the network possess a tolerance to 'noise' which would be absent in an entirely *precise* DFA simulation?

In principle, some degree of noise tolerance could be present. Indeed, Casey (1996) has proven that RNN simulations of DFA can, in general, be noise tolerant (within narrow bounds) provided the requisite weight vectors are assumed to be present. However, Casey does not offer an algorithm for generating the required weights. His concerns were of a different order.

The overriding goal of Casey's work (1996) was demonstrated in his first theorem, which states that "if an RNN performs an FSM [finite state machine] computation, then it must organize itself so that it models the minimal DFA which performs the same FSM computation." The minimal DFA is one that achieves the given task with the least number of states (and state nodes). Elsewhere, Casey asserts that the RNN "mimics" the organization of the minimal DFA.

It is important to appreciate the generality of Casey's results. For we now know that *every* deterministic RNN which successfully matches the input-output behaviour of a given DFA must *implement* some particular DFA, viz., the minimal DFA that performs the given computation. To be sure, the limited noise tolerance of the RNN implementation may bring advantages in some domains. Nevertheless, a precise correspondence will exist between the RNN state representations and those of the minimal DFA in question. Any attempt to expand the noise tolerance of the RNN can introduce errors on each iterative state transition. Such errors are rapidly compounded and lead to increasingly degraded performance.

Now, on the face of it, Casey's results conflict with an observation of Cleermans, et al, namely that "representations [within the simple recurrent network] correspond to nodes in the FSA only when resources are severely constrained." The conflict dissolves, however, when we recall that Cleermans et al are thinking of the FSA which one is ostensibly modeling, whereas Casey's proof refers to the minimal DFA.

The upshot of the foregoing discussion is that successful RNN simulations of DFA can be viewed as close approximations of DFA. Moreover, every DFA implements some classical algorithm. The question naturally arises, then, whether RNN implementations of DFA can provide important insights into how high-level algorithmic processes could be implemented in the brain. As I will now argue, the answer largely depends upon the cognitive plausibility of several assumptions which are crucial to the computability results cited above.

3.1 Genesis of Weight Vectors.

A key premise, found both in (Casey, 1996) and (Sontag, 1995) is that weights vectors, having suitable topological properties, may be assumed to be present in the RNN. Admittedly, Casey shows concern for how weights are to be acquired. For this reason, his proofs are designed with "robust" RNNs in mind (the weight vectors need fall only within certain tolerance ranges). Nevertheless, neither he nor Sontag proves that the requisite weights can be generated by any feasible method. Let us consider, therefore, how the appropriate weight vectors might come to be present.

There appear to be just three salient possibilities. (A) Some or all of the required weights are innately present. (B) The weights are induced by supervised training, e.g., via the backpropagation algorithm. (C) The weights are induced by unsupervised learning methods.

For each of these three possibilities, there may be cognitive realms where considerable plausibility exists. However, the realm of high-level cognition (abstract reasoning, planning, theory formation) seems to present some difficulty for all three possibilities.

Consider first the innateness hypothesis. Given that most forms of abstract reasoning and theory formation rely, in part, on a variety of specialized mental skills (e.g., the ability to apply acquired verbal rules, the ability to find analogies with existing theories/methods; the ability to mentally entertain a list of alternative plans; etc.), it seems highly likely that some prior learning, at least, is required to support these powerful cognitive processes (cf. Hadley, in press). So, the supposition that we employ an innate, fully weight-configured

RNN to achieve all such cognitive processing is problematic at best. Nevertheless, there remains the apparent possibility that we possess certain modular NNs which are trained *by experience*, but that we also possess a high-level, innately-configured, general problem solving RNN, which matches the power of some DFA. This general-purpose RNN might invoke the empirically-trained modules to achieve certain sub-tasks, but the high-level RNN could still be viewed as an innately programmed network.

Unfortunately, a serious difficulty remains. Specifically, the *innate weight-configuration* hypothesis would find little support among some prominent neuro-psychologists and connectionists. For example, Elman, Bates, et al (1996) argue forcefully that detailed, pre-specified weights (supporting specific representations) are *not* innately present in the cerebral cortex. While this proposition is not embraced by all developmentalists, it is worrisome that its dissenters have not produced physiological evidence to the contrary. Note also, that the *partial* weight prespecification strategy of Omlin & Giles (1996) is seemingly undermined by Elman et al's arguments. For, though Omlin & Giles offer an algorithm for pre-specifying weights, this algorithm is not driven by experiential training. Thus, their *a-priori* weight pre-specification should be viewed as comparable to innate structuring.

Turning now to (B), the supervised learning hypothesis, we are again confronted with a serious obstacle. For, all known supervised learning algorithms require a representative sampling of target output values to be available during the training process. Yet, in domains such as abstract reasoning, planning, and especially theory formation, the overwhelming majority of "output values" are not available beforehand. Rather, they have yet to be discovered by the agents who are undergoing training. Also, in this domain, "interpolation" between known output values often fails to work (as we saw in the case of 'universal' function approximation). Moreover, it would be ludicrous to suppose that humans learn how to devise theories or to prove theorems by a simple form of stimulus-response training. Part of the learning process, at the very least, involves being taught general principles, and then applying these principles in novel combinations (cf. Hadley, in press).

Apart from the above, we must bear in mind that abstract reasoning, and theory formation are highly complex processes. Any *computationally complete and effective* DFA model of these processes will involve hundreds or thousands of distinct states and state transitions, at the least.[2] In light of this, any RNN, corresponding to such a DFA, will involve complex and lengthy recursive processes. Now, as Omlin & Giles have emphasized, there are no known supervised learning algorithms which reliably induce weight configurations in RNNs so as to achieve *successful* simulation of DFAs in the face of lengthy recursive processes. Yet, humans certainly engage in lengthy internal processing in cases where they are devising a proof strategy or, say, planning their next move in a chess game. So, even ignoring the difficulty of 'unavailable target outputs', we presently have no reason to believe that supervised learning methods could induce the requisite weight vectors.

Admittedly, it is *possible* that an appropriate, supervised learning method will someday be discovered. However, from this it does not follow that a purely agnostic attitude on the issue is reasonable. Any successful, supervised learning algorithm (in this domain) must have highly specific mathematical properties, just as a proof for a mathematical proposition must have very special properties. If one arbitrarily pinpoints some extremely complex mathematical proposition and asserts, 'It is just as likely that this proposition is provable as that it is not', one may expect a (figurative) barrage of rotten tomatoes before an audience of mathematicians. Likewise, it would be imprudent to claim that the existence of the required supervised learning algorithm is just as likely as its non-existence.

There remains, of course, possibility (C) – unsupervised learning methods. Unfortunately, as many experienced connectionists can testify, it is even more difficult to induce appropriate weight vectors in a large, complex RNN via unsupervised methods than by supervised training. This explains why the overwhelming majority of connectionist models published (in various Cognitive Science journals and proceedings) employ backpropagation (or its near kin) rather than unsupervised algorithms.

To be sure, it is widely agreed that much human learning occurs in an unsupervised fashion. Moreover, unsupervised competitive and/or Hebbian learning appears to be the foundation for most or all "weight adjustment" (via synaptic modification) that occurs in the cerebral cortex (cf. Elman et al, 1996). Presumably then, unsupervised learning can achieve astounding results in the cognitive realm. Given this, should we not remain open-minded regarding hypothesis (C)?

Unfortunately, matters are not so simple. Although I happily embrace the view that unsupervised learning may achieve wonderful feats, it is far from clear that it does so via *training RNNs to behave like DFA*. I have argued elsewhere (Hadley, in press) that much of the power of high-level cognition derives from architectures that arise through novel interactions of modules. Unsupervised learning, in conjunction with other forms of plasticity, probably plays a large role in the formation of these modules. However, it is unclear, at best, whether these modules can be modeled as DFA. Among other difficulties, DFA do not even possess sufficient power to parse context-sensitive grammars.

In summary, we have now seen that significant doubts arise, for all three possibilities {(A), (B), and (C)}, with respect to how suitable weight vectors, needed to support DFA simulation, could occur within brain-based RNNs. The innateness hypothesis presents fewer difficulties, at first glance. However, this hypothesis finds little favor in some connectionist quarters, due to the dearth of supporting physiological evidence. Of the learning-based pos-

[2]The skeptical reader might attempt to construct an *complete* DFA model for SOAR (Laird, et al, 1987), a well known candidate for a 'general cognitive model'.

sibilities, only supervised algorithms (in particular, variants of backpropagation) have successfully induced even approximately correct weights in comparatively simple RNNs. It is noteworthy, moreover, that backpropagation, and its refinements, have thus far resisted any *reduction* to biologically based weight modification methods. This aspect further clouds the prospect that *trained* RNNs may reveal the nature of *cognitively plausible* implementations of DFA in actual brains.

Conclusions and Summary.

In the foregoing, I have examined claims made for the computational powers of two distinct classes of connectionist networks, namely, the (putative) "universal function approximators", and recurrent finite-state simulation networks. Each class was considered with respect to its potential in the realm of cognitive modeling. Regarding the first class, I argued that, contrary to the claims of some influential connectionists, feed-forward networks do *not* possess the capacity to approximate all functions of interest to cognitive scientists. They clearly do not possess the ability to approximate partial recursive (non-halting) functions, though the latter may very well provide good models of some high-level cognitive processes. Moreover, we saw that feed-forward networks cannot approximate certain important, simple recursive (halting) functions which map symbolic strings onto other symbolic strings.

By contrast, we saw that a particular class of recurrent networks (namely, RNNs that closely approximate DFAs) shows considerably greater promise in some domains. For, many computable functions, which map symbolic strings of arbitrary length onto similar strings can, in principle, be modeled by DFAs and their connectionist simulations. However, serious difficulties emerged when we considered how the relevant recurrent networks could acquire the *weight vectors* needed to support DFA simulations. Indeed, the most widely used method of inducing weight vectors (supervised learning) was seen to be implausible in the realm of several high-level cognitive functions. Furthermore, hypotheses founded upon innate-wiring and self-organizing learning likewise presented serious obstacles. This is not to say that a set of separate RNN modules would present similar obstacles. However, a *set* of RNN modules is not an RNN in the sense assumed by the various theoretical proofs we have considered here. Moreover, as previously mentioned, I offer theoretical arguments in (Hadley, in press) to show that a modular connectionist architecture inevitably leads to *classical* patterns of inter-modular processing.

References.

Casey (1996) The Dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction, *Neural computation*, 8, 1135-1178.

Church, A. (1956) *Introduction to mathematical logic*, Princeton, NJ: Princeton University Press.

Churchland, P. (1990) Cognitive activity in artificial neural networks. In Osherson, D.N. & Smith, E.E. (eds.), *An invitation to cognitive science: Thinking (Vol. 3)*, Cambridge, MA: MIT Press.

Cleeremans, A., Servan-Schreiber, D. & McClelland, J.L. (1989) Finite state automata and simple recurrent networks, *Neural computation*, 1, 372-381.

Elman, J.L. (1993) Learning and development in neural networks: The importance of starting small, *Cognition*, 48, 71-99.

Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D., Plunkett, K. (1996), *Rethinking Innateness: A Connectionist Perspective on Development*, Cambridge, MA: MIT Press.

Gödel, K. (1931) Über formal unentscheidbare Sätze der Principia mathematica und verwandter System I, *Monatschefte für Mathematik und Physik*, 38, 173-198 (Translated in van Heijenoort, 1967).

Fodor, J.A. and Pylyshyn, Z.W. (1988), Connectionism and Cognitive Architecture: A Critical Analysis, *Cognition*, Vol. 28, 3-71.

Hadley, R.F. (1999) Cognition and the Computational Power of Connectionist Networks, Technical Report, SFU CMPT TR 1999-01, School of Computing Science, Simon Fraser University, Burnaby, B.C., V5A 1S6.

Hadley, R.F. (in press) Connectionism and novel combinations of skills: implications for cognitive architecture, *Minds and Machines*, pp. (to appear).

Hartman, E.J., Keeler, J.D. & Kowalski, J.M. (1990) Layered neural networks with Gaussian hidden units as universal approximations, *Neural computation*, 2, 210-215.

Hornik, K., Stinchcombe, M., & White, H. (1989) Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, 359-366.

Laird, J.E., Newell, A., Rosenbloom, P.S. (1987) SOAR, An architecture for general intelligence, *Artificial Intelligence*, 33, 1-64.

Niklasson, L.F. and van Gelder, T. 1994: On Being Systematically Connectionist. *Mind and Language*, 9, 288-302.

Omlin, C.W. & Giles, C.L. (1996) Constructing deterministic finite-state automata in recurrent neural networks, *Journal of the ACM*, 43, 937-972.

Siegelmann, H.T. (1996) The simple dynamics of super Turing theories, *Theoretical Computer Science*, 168, 461-472.

Siegelmann, H.T. & Sontag, E.D. (1994) Analog computation via neural networks, *Theoretical Computer Science*, 131, 331-360.

Sontag, E.D. (1995) Automata and neural networks, *The Handbook of brain theory and neural networks*, (ed.) Arbib, M.A., Cambridge, MA: MIT Press.