Symbolic Artificial Intelligence, Connectionist Networks and Beyond

TR94-16

Vasant Honavar and Leonard Uhr

August 18, 1994

Iowa State University of Science and Technology
Department of Computer Science
226 Atanasoff
Ames, IA 50011

1

SYMBOLIC ARTIFICIAL INTELLIGENCE, CONNECTIONIST NETWORKS, AND BEYOND

Vasant Honavar* and Leonard Uhr**

**Dept. of Computer Science, Iowa State University, Ames, Iowa.*

***Computer Sciences Dept., University of Wisconsin, Madison, Wisconsin.*

1 INTRODUCTION

The goal of Artificial Intelligence, broadly defined, is to understand and engineer intelligent systems. This entails building theories and models of embodied minds and brains – both natural as well as artificial. The advent of digital computers and the parallel development of the theory of computation since the 1950s provided a new set of tools with which to approach this problem – through analysis, design, and evaluation of computers and programs that exhibit aspects of intelligent behavior – such as the ability to recognize and classify patterns; to reason from premises to logical conclusions; and to learn from experience.

The early years of artificial intelligence saw some people writing programs that they executed on serial stored–program computers (e.g., Newell, Shaw and Simon, 1963; Feigenbaum, 1963); Others (e.g., Rashevsky, 1960; McCulloch and Pitts, 1943; Selfridge and Neisser, 1963; Uhr and Vossler, 1963) worked on more or less precise specifications of more parallel, brain–like networks of simple processors (reminiscent of today's connectionist networks) for modelling minds/brains; and a few took the middle ground (Uhr, 1973; Holland, 1975; Minsky, 1963; Arbib, 1972; Grossberg, 1982; Klir, 1985).

It is often suggested that two major approaches have emerged – symbolic artificial intelligence (SAI) and artificial neural networks or connectionist networks (CN) and some (Norman, 1986; Schneider, 1987) have even suggested that they are fundamentally and perhaps irreconcilably different. Others have argued that CN models have little to contribute to our efforts to understand cognitive processes (Fodor and Pylyshyn, 1988). A critical examination of the popular conceptions of SAI and CN models suggests that neither of these extreme positions is justified (Boden, 1994; Honavar and Uhr, 1990a; Honavar, 1994b; Uhr and Honavar, 1994). Recent attempts at reconciling SAI and CN approaches to modelling cognition and engineering intelligent systems (Honavar and Uhr, 1994; Sun and Bookman, 1994; Levine and Aparicioiv, 1994; Goonatilake and Khebbal, 1994; Medsker, 1994) are strongly suggestive of the potential benefits of exploring computational models that judiciously integrate aspects of both. The rich and interesting space of designs that combine concepts, constructs, techniques and technologies drawn from both SAI and CN invite systematic theoretical as well as experimental exploration in the context of a broad range of problems in perception, knowledge representation and inference, robotics, language, and learning, and ultimately, integrated systems that display what might be considered human–like general intelligence. This chapter examines how today's CN models can be extended to provide a framework for such an exploration.

2 A CRITICAL LOOK AT SAI AND CN

This section critically examines the fundamental philosophical and theoretical assumptions as well as what appear to be popular conceptions of SAI and CN (Honavar and Uhr, 1990a; Uhr and Honavar, 1994; Honavar, 1990; 1994b). This examination clearly demonstrates that despite assertions by many to the contrary, the differences between them are less than what they might seem at first glance; and to the extent they differ, such differences are far from being in any reasonable sense of the term, fundamental; and that the purported weaknesses of each can potentially be overcome through a judicious integration of techniques and tools selected from the other (Honavar, 1990; 1994b; Honavar and Uhr, 1990a; Uhr and Honavar, 1994; Uhr, 1990; Boden, 1994).

2.1 SAI and CN approaches to modelling intelligence share the same working hypothesis

The fundamental working hypothesis that has guided most of the research in artificial intelligence as well as the information–processing school of psychology is rather simply stated: *Cognition, or thought processes can, at some level, be modelled by computation.* This has led to the *functional* view of intelligence which is shared explicitly or implicitly by almost all of the work in SAI. Newell's *physical symbol system hypothesis* (Newell, 1980), Fodor's *language of thought* (Fodor, 1976) are specific examples of this view. In this framework, perception and cognition are tantamount to acquiring and manipulating symbolic *representations*. Representations, in short, are caricatures of an agent's environment that are operationally useful to the agent: operations on representations can be used to predict the consequences of performing the corresponding physical actions on the environment. (See Newell, 1990; Honavar, 1994b; Chandrasekaran and Josephson, 1994 for more detailed discussion of the nature of representation and its role in the functional view of intelligence). Exactly the same functional view of intelligence is at the heart of current approaches modelling intelligence within the CN paradigm, as well as the attempts to understand brain function using the techniques of computational neuroscience and neural modelling. This is clearly demonstrated by the earliest work on neural networks by Rashevsky (1960), McCulloch and Pitts (1943) and Rosenblatt (1962) – from which many of today's CN models (McClelland, Rumelhart et al., 1986; Kung, 1993; Haykin, 1994; Zeidenberg, 1989) as well as the increasing emphasis on *computational models* in contemporary neuroscience (Churchland and Sejnowski, 1992).

Thus, CN models or theories of intelligence are stated in terms of abstract computational mechanisms just as their SAI counterparts. The abstract descriptions in both cases are usually stated in sufficiently general languages. One thing we know for certain is that such languages are all equivalent (see below). This provides absolute assurance that particular physical implementations of systems exhibiting mind/brain–like behavior can be described using the language of SAI or CN irrespective of the physical medium (biological or silicon or some other) that is used in such an implementation. And the choice of the language should (as it usually is, in science in general) be dictated by pragmatic considerations.

2.2 SAI and CN Rely on Equivalent Models of Computation

One of the most fundamental results of computer science is the *Turing–equivalence* of various formal models of computation – including Turing's specification of the general–purpose serial stored program computer with potentially infinite memory (the *Turing machine*), Church and Rosser's *lambda calculus*, Post's *production systems*, McCulloch and Pitts' *neural networks* (among others). Turing was among the first to formalize the common–sense notion of computation in terms of execution of what he called – an *effective procedure* or an algorithm. In the process, he invented a hypothetical computer – the *Turing machine*. The behavior of the Turing machine is governed by an algorithm which is realized in terms of a *program* or a finite sequence of instructions. Turing also showed that there exists a *universal* Turing machine (essentially a general purpose stored program computer with potentially infinite memory) – one that can *compute* anything that any other Turing machine could possibly compute – given the necessary program as well as the data and a means for interpreting its programs. The various formal models of computation mentioned above (given potentially infinite memory) were proved exactly equivalent to the Turing Machine. That is, any computation that can be described by a finite program can be programmed in any general purpose language or on any Turing–equivalent computer (Cohen, 1986). (However, a program for the same computation may be much more compact when written in one language than in some other; or it may execute much faster on one computer than some other). But the provable equivalence of all general purpose computers and languages assures us that *any* computation – be it numeric or symbolic – can be realized, in principle, by both SAI as well as CN systems.

Given the reliance of both SAI and CN on equivalent formal models of computation, the questions of interest have to do with the identification of particular subsets of Turing–computable functions that model various aspects of intelligent behavior given the various design and performance constraints imposed by the physical implementation media at our disposal.

2.3 Problem Solving as State Space Search

The dominant paradigm for problem solving in SAI is *state space search* (Winston, 1992; Ginsberg, 1993). States represent snap–shots of the problem at various stages of its solution. Operators enable transforming one state into another. Typically, the states are represented using structures of symbols (e.g., lists). Operators transform one symbol structure (e.g., list, or a set of logical expressions) into another. The system's task is to find a path between two specified states in the state–space (e.g., the initial state and a specified goal, the puzzle and its solution, the axioms and a theorem to be proved, etc.).

In almost any non–trivial problem, a blind exhaustive search for a path will be impossibly slow, and there will be no known algorithm or a procedure for directly computing that path without resorting to search. However, search can be guided by the knowledge that is at the disposal of the problem solver. If the system is highly specialized, the necessary knowledge is usually built into the search procedure (in the form of criteria for choosing among alternative paths, heuristic functions to be used, etc.). However, general purpose problem solvers also need to be able to retrieve problem–specific and perhaps even situation–specific knowledge to be used to guide the search during problem–solving. Indeed, such retrieval might itself entail search (albeit in a different space). Efficient, and flexible representations of such knowledge as well as mechanisms for their retrieval as needed during problem solving are, (although typically overlooked because most current AI systems are designed for very specialized, narrowly defined tasks), extremely important.

State–space search in SAI systems is typically conducted using serial programs or production systems which are typically executed on a serial Von Neumann computer. However, there is no reason to not use parallel search algorithms or parallel production systems (more on this later).

The CN system (a network of relatively simple processing elements, neurons, or nodes) is typically presented with an input pattern or initialized in a given starting state encoded in the form of a state vector each of whose elements corresponds to the state of a neuron in the network). It is designed or trained to output the correct response to each input pattern it receives (perhaps after undergoing a series of state updates determined by the rules governing its dynamic behavior). The input–output behavior of the network is a function of the network architecture, the functions computed by the individual nodes and parameters such as the weights.

For example, the solution of an optimization problem (traditionally solved using search) can be formulated as a problem of arriving at a state of a suitably designed network that corresponds to one of minimum energy – which is defined to correspond in some natural way to the optimality of the solution being sought (Hopfield, 1982). Ideally, the network dynamics are set up so as to accomplish this without additional explicit control. However, in practice, state updates in CN systems are often controlled in a manner that is not much different from explicit control (as in sequential update of neurons in Hopfield networks (Hopfield, 1982) where only one neuron is allowed to change its state on any update cycle) to guarantee certain desired emergent behaviors). Indeed, a range of cognitive tasks do require selective processing of information that often necessitates the use of a variety of (albeit flexible and distributed) networks of controls that is presently lacking in most CN models (Honavar and Uhr, 1990b). Many such control structures and processes are suggested by an examination of computers, brains, immune systems, and evolutionary processes.

In short, in both SAI and CN systems, problem–solving involves state–space search; and although most current implementations tend to fall at one end of the spectrum or the other, it should be clear that there exists a space of designs that can use a mix of different state representations and processing methods. The choice of a particular design for a particular class of problems should primarily be governed by performance, cost, and reliability considerations for artificial intelligence applications and psychological and neurobiological plausibility for cognitive modelling.

2.4 Symbols, Symbol Structures, Symbolic Processes

Knowledge representation in SAI systems involves the use of symbols at some level. The standard notion of a symbol is that it *stands for* something and when a symbol token appears within a symbolic expression carries the interpretation that the symbol stands for something within the context that is specified by its place in the expression. In general, a symbol serves as a surrogate for a body of knowledge that may need to be accessed and used in processing the symbol. And ultimately, this knowledge includes semantics or meaning of the symbol in the context in which it appears, including that provided by the direct or indirect grounding of the symbol structure in the external environment (Harnad, 1990).

Symbolic processes are essentially transformations that operate on symbol structures to produce other symbol structures. Memory holds symbol structures that contain symbol tokens that can be modified by such processes. This memory can take several forms based on the time scales at which such modifications are allowed. Some symbol structures might have the property of determining choice and the order of application of transformations to be applied on other symbol structures. These are essentially the programs. Programs when executed – typically through the conventional process of compilation and interpretation and eventually – when they operate on symbols that are linked through grounding to particular effectors – produce behavior. Working memory holds symbol structures as they are being processed. Long–term memory, generally speaking, is the repository of programs and can be changed by addition, deletion, or modification of symbol structures that it holds. The reader is refered to (Newell, 1990) for a detailed treatment of symbol systems of this sort.

Such a symbol system can compute any Turing–computable function provided it has sufficiently large memory and its primitive set of transformations are adequate for the composition of arbitrarily symbol structures (programs) and the interpreter is capable of interpreting any possible symbol structure. This also means that any particular set of symbolic processes can be carried out by a CN – provided it has potentially infinite memory, or finds a way to use its transducers and effectors to use the external physical environment to augment its memory (just as humans have in their use of stone tablets, papyrus, and books through the ages).

Knowledge in SAI systems is typically embedded in complex symbol structures such as lists (Norvig, 1992), logical databases (Genesereth and Nilsson, 1987), semantic networks (Quillian, 1968), frames (Minsky, 1975), schemas (Arbib, 1972; 1994), and manipulated by (often serial) procedures or inferences (e.g., list processing, application of production rules (Waterman, 1985), or execution of logic programs (Kowalski, 1977) carried out by a central processor that accesses and changes data in memory using addresses and indices.

It is often claimed that the CN systems predominantly perform numeric processing in contrast to SAI systems which manipulate symbol structures. But as already pointed out, CN systems represent problem states using (typically binary) state vectors which are manipulated in a network of processors using (typically) numeric operations (e.g., weighted sums and thresholds). It is not hard to see that the numeric state vectors and transformations employed in such networks play an essential symbolic role although the rules of transformation may now be an emergent property of a large number of nodes acting in concert. In short, the formal equivalence of the various computational models guarantees that CN can support arbitrary symbolic processes. It is not therefore surprising that several alternative mechanisms for variable binding and logical reasoning using CN have been discovered in recent years. Some of these require explicit use of symbols (Shastri and Ajjanagadde, 1989); others resort to quasi–symbols that have some properties of symbols while not being actually symbols in their true sense (Pollack, 1990; Maclennan, 1994); still others use pattern vectors to encode symbols (Dolan and Smolensky, 1989; Smolensky, 1990; Sun, 1994; Chen and Honavar, 1994). The latter approach to symbol processing is often said to use sub–symbolic encoding of a symbol as a pattern vectors each of whose components is insufficient in and of itself to identify the symbol in question (see the discussion on distributed representations below). In any case, most, if not all, of these proposals are implemented and simulated on general purpose digital computers, so none of the functions that they compute are outside the Turing framework.

2.5 Numeric Processing

Numeric processing, as the name suggests, involves computations with numbers. On the surface it appears that most CN perform essentially numeric processing. After all, the formal neuron of McCulloch and Pitts computes weighted sum of its numeric inputs. And the *neurons* in most CN models perform similar numerical computations. On the other hand, SAI systems predominantly compute functions over structures of symbols. But numbers are in fact symbols for quantities; and any computable function over numbers can be computed by symbolic processes. In fact, general purpose digital computers have been performing both symbolic as well as numeric processing ever since they were invented.

2.6 Analog Processing

It is often claimed that CN perform *analog computation*. Analog computation generally implies the use of dynamical systems describable using continuous differential equations. They operate in continuous time, generally with physical entities such as voltages and currents, which serve as physical *analogs* of the quantities of interest. Thus soap bubbles, servomechanisms, and cell membranes can all be regarded as analog computers (Rajaraman, 1981).

Whether physically realizable systems are truly analog or whether analog system is simply a mathematical idealization of (an extremely fine–grained) discrete system is a question that borders on the philosophical – are time, space, and matter continuous or discrete?. However, some things are fairly clear. Most CN are simulated on digital computers and compute in discrete steps and hence are clearly not analog. The few CN models can be regarded as analog devices – e.g., the analog VLSI circuits designed and built by Carver Mead and colleagues (Mead, 1989) – are incapable of discrete symbolic computations (because of their inability to make all–or–none or discrete choices) (Maclennan, 1994) although they can approximate such computations. (For example, the stable states or attractors of such systems can be interpreted as identifiable discrete states).

Analog systems can be, and often are simulated on digital computers at the desired level of precision. However, this might involve a time-consuming iterative calculation to produce a result that could potentially be obtained almost instantaneously (and transduced using appropriate transducers) given the right analog device. Thus analog processing appears to be potentially quite useful in many applications (especially those that involve perceptual and motor behavior). It is possible that evolution has equipped living systems with just the right repertoire of analog devices that help them process information in this fashion. However, it is somewhat misleading to call such processing *computation* (in the sense defined by Turing) because it lacks the discrete combinatorial structure that is characteristic of all Turing-equivalent models of computation (Maclennan, 1994).

Whether analog processes play a fundamental role (beyond being part of grounding of representations) in intelligent systems remains very much an open question. It is also worth pointing out that digital computers can, and in fact do, make use of essentially analog devices such as transistors but they use only a few discrete states to support computation (in other words, the actual analog value is irrelevant so long as it lies within a range that is distinguishable from some other range). And when embedded in physical environments, both SAI and CN systems do encounter analog processes through sensors and effectors.

2.7 Compositionality and Systematicity of Representation

It has been argued by many e.g., Fodor and Pylyshyn, 1988) that *compositionality* and *systematicity* (structure sensitivity) of representation are essential for explaining mind. In their view, CN are inadequate models of mind because CN representations lack these essential properties. Compositionality is the property that demands that representations must possess an internal *syntactic structure* as a consequence of a particular method for composing complex symbol structures from simpler components. Systematicity requires the existence of processes that are sensitive to the syntactic structure. As argued by Sharkey and Jackson (1994), lack of compositionality is demonstrably true only for a limited class of CN representations; and compositionality and systematicity in and of themselves are inadequate to account for cognition (primarily for lack of grounding or semantics). Van Gelder and Port (1994) have shown that several forms of compositionality can be found in CN representations.

2.8 Grounding and Semantics

Many in the artificial intelligence and cognitive science research community agree on the need for *grounding* of symbolic representations through sensory (e.g., visual, auditory, tactile) transducers and motor effectors in the external environment on the one hand and the internal environment of needs, drives, and emotions of the organism (or robot) in order for such representations (which are otherwise devoid of any intrinsic meaning to the organism or robot) to become imbued with meaning or semantics (Harnad, 1990). Some have argued that CN systems provide the necessary apparatus for grounding (Harnad, Hanson, and Lubin, 1994). It is important to realize that CN as computational models do not provide physical grounding (as opposed to grounding in a simulated world of virtual reality) for representations any more than their SAI counterparts. It is only the physical systems with their physical substrate on which the representations reside that are capable of providing such grounding in physical reality when equipped with the necessary transducers and effectors. This is true irrespective of whether the system in question is a prototypical SAI system, or a prototypical CN system, or a hybrid or integrated system.

2.9 Serial versus Parallel Processing

As pointed out earlier, most of today's SAI systems are serial programs that are executed on serial von Neumann computers. However, serial symbol manipulation is more an artifact of most current implementations of SAI systems than a necessary property of SAI. In parallel and distributed computers, memory is often locally available to the processors and even can be almost eliminated in data flow machines which model functional or applicative programs where data is transformed as it flows through processors or functions. Search in SAI systems can be, and often is, parallelized by mapping the search algorithm onto a suitable network of computers (Uhr, 1984; 1987b; Hewitt, 1977; Hillis, 1985) with varying degrees of centralized or distributed control. Many search problems that arise in applications such as temporal reasoning, resource allocation, scheduling, vision, language understanding and logic programming can be formulated as constraint satisfaction problems which often lend themselves to solution using a mix of serial and parallel processing (Tsang, 1993).

Similarly, SAI systems using production rules can be made parallel by enabling many rules to be matched simultaneously in a data flow fashion – as in RETE pattern matching networks (Forgy, 1982). Multiple matched rules may be allowed to fire and change the working memory in parallel as in parallel production systems (Uhr, 1979) and classifier systems (Holland, 1975) – so long as whenever two or more rules demand conflicting actions, arbitration mechanisms are provided to choose among the alternatives or resolve such conflicts at the sensory–motor interface. Such arbitration mechanisms can themselves be realized using serial, parallel (e.g., winner–take–all mechanism), or serial–parallel (e.g., pyramid–like hierarchies of decision mechanisms) networks of processes.

CN systems with their potential for massive fine–grained parallelism of computation offer a natural and attractive framework for the development of highly parallel architectures and algorithms for problem solving and inference. Such systems are considered necessary by many researchers (Uhr, 1980; Feldman and Ballard, 1982) for tasks such as real–time perception. But SAI systems doing symbolic inference can be, and often are, parallelized, and certain inherently sequential tasks need to be executed serially. On any given class of problems, the choice of decomposition of the computations to be performed into a parallel–serial network of processes and their mapping onto a particular network of processors has to be made taking the cost and performance tradeoffs into consideration.

2.10 Knowledge Engineering Versus Knowledge Acquisition Through Learning

The emphasis in some SAI systems – especially the so–called knowledge–based expert systems (Waterman, 1985) – on knowledge engineering has led some to claim that SAI systems are, unlike their CN counterparts, incapable of learning from experience. This is clearly absurd as even a cursory look at the current research in machine learning (Shavlik and Dietterich, 1990; Buchanan and Wilkins, 1993) and much early work in pattern recognition (Uhr, 1973; Fu, 1982; Miclet, 1986) shows. Research in SAI and closely related systems indeed have provided a wide range of techniques for deductive (analytical) and inductive (synthetic) learning. Learning by acquisition and modification of symbol structures almost certainly plays a major role in knowledge acquisition in humans who learn and communicate in a wide variety of natural languages (e.g., English) as well as artificial ones (e.g., formal logic, programming languages). While CN systems with their micro–modular architecture offer a range of interesting possibilities for learning, for the most part, only the simplest parameter or weight modification algorithms have been explored to date (McClelland, Rumelhart et al., 1986; Kung, 1993; Gallant, 1993; Haykin, 1994). In fact, learning by weight modification alone appears to be inadequate in and of itself to model rapid and irreversible learning that is observed in many animals. Algorithms that modify networks through structural changes that involve the recruitment of neurons (Honavar, 1989; 1990; Honavar and Uhr, 1989a; 1989b;

1992a; 1993; Kung, 1993; Grossberg, 1982) appear promising in this regard (see below).

Most forms of learning can be understood and implemented in terms of structures and processes for representing and reasoning with knowledge (broadly interpreted) and for memorizing the results of such inference in a form that lends itself to retrieval and use at a later time (Michalski, 1993). Thus any CN or SAI or some hybrid architecture that is capable of performing inference and has memory for storing the results of inference for retrieval and use on demand can be equipped with the ability to learn. The interested reader is referred to (Honavar, 1994) for a detailed discussion of systems that learn using multiple strategies and representations. In short, SAI systems offer powerful mechanisms for manipulation of highly expressive structured symbolic representations while CN offer the potential for robustness, and the ability to fine–tune their use as a function of experience (primarily due to the use of tunable numeric weights and statistics). This suggests a number of interesting and potentially beneficial ways to integrate SAI and CN approaches to learning. The reader is refered to (Uhr, 1973; Holland, 1975; Honavar, 1992b; 1994; Honavar and Uhr, 1993; Carpenter and Grossberg, 1994; Shavlik, 1994; Gallant, 1993; Goldfarb and Nigam, 1994; Booker, Riolo, and Holland, 1994) for some examples of such systems.

2.11 Associative as Opposed to Address–Based Storage and Recall

An often cited distinction between SAI and CN systems is that the latter employ associative (i.e., content–addressable) as opposed to the address–and–index based storage and recall of patterns in memory typically used by the former. This is a misconception for several reasons: Address–and–index based memory storage and retrieval can be used to simulate content–addressable memory and vice versa and therefore unless one had access to the detailed internal design and operation of such systems, their behavior can be indistinguishable from each other. Many SAI systems conventional computers use associative memories in some form or another (e.g., hierarchical cache memories). While associative recall may be better for certain tasks, address (or location–based) recall (or a combination of both) may be more appropriate for others. Indeed, many computational problems that arise in symbolic inference (pattern matching and unification in rule–based production systems or logic programming) can take advantage of associative memories for efficient processing (Chen and Honavar, 1994).

In prototypical CN models, associative recall is based on some relatively simple measure of proximity or closeness (usually measured by Hamming distance in the case of binary patterns) to the stored patterns. While this may be appropriate in domains in which related items have patterns or codes that are close to each other, it would be absurd to blindly employ such a simple content–addressed memory model in domains where symbols are arbitrarily coded for storage (which would make hamming distance or a similar proximity measure useless in recalling the associations that are really of interest). Establishing (possibly context–sensitive) associations between otherwise arbitrary symbol structures based on their meanings and retrieving such associations efficiently requires complex networks of learned associations more reminiscent of associative knowledge networks, semantic networks (Quillian, 1968), frames (Minsky, 1975), conceptual structures (Sowa, 1984), schemas (Arbib, 1994), agents (Minsky, 1986) and object–oriented programs of SAI (Norvig, 1992) than today's simple CN associative memory models. This is not to suggest that such structures cannot be implemented using suitable CN building blocks – see (Arbib, 1994; Dyer, 1994; Miikkulainen, 1994; Bookman, 1994; Barnden, 1994) for some examples of such implementations. Indeed, such CN implementations of complex symbol structures and symbolic processes can offer many potential advantages (e.g., robustness, parallelism) for SAI.

2.12 Distributed Storage, Processing, and Control

Distributed storage, processing, and control are often claimed to be some of the major advantages of CN systems over their SAI counterparts. It is far from clear as to what is generally meant by the term *distributed* when used in this context (Oden, 1994).

Perhaps it is most natural to think of an item as distributed when it is coded (say as a pattern vector) whose components by themselves are neither sufficient to identify the item nor have any useful semantic content. Thus, the binary code for a letter of the alphabet is distributed. Any item thus distributed eventually has to be reconstructed from the pieces of its code. This form of distribution may be in space, time, or both. Thus the binary code for a letter of the alphabet may be transmitted serially (distributed in time) over a single link that can carry 1 bit of information at a time or in parallel (distributed in space) using a multi–wire bus. If a system employs such a mechanism for transmission or storage of data, it also needs decoding mechanisms for reconstructing the coded item at the time of retrieval. It is easy to see that this is not a defining property of CN systems as it is found in even the serial von Neumann computers. In any event, both CN as well as SAI systems can use such distributed coding of symbols. And, as pointed out by Hanson and Burr (1990), distributed coding in and of itself, offers no representational capabilities that are not realizable using a non–distributed coding.

In the context of CN, the term *distributed* is often used to refer to storage of parts of an item in a unit where parts of other items also stored (for example, by superposition). Thus, each unit participates in storage of multiple items and each item is distributed over multiple units. (There is something disconcerting about this particular use of the term distributed in a technical sense: Clearly, one can invent a new name for whatever it is that a unit stores – e.g., a number whose binary representation has a '1' in its second place. Does the system cease to be distributed as a result?). It is not hard to imagine an analogous notion of distribution in time instead of space but it is also fraught with similar semantic difficulty.

The term *distributed* when used in the context of *parallel and distributed processing*, generally refers to the decomposition of a computational task into more or less independent pieces that are executed on different processors with little or no inter–processor communication (Uhr, 1984; 1987b; Almasi and Gottlieb, 1989). Thus many processors may perform the same computation on pieces of the data (as in single–instruction–multiple–data or SIMD computer architectures) or each processor may perform a different computation on the same data e.g., computation of various intrinsic properties of an image (as in multiple–instruction–single–data or MISD computer architectures), or a combination of both (as in multiple–instruction–multiple–data or MIMD computer architectures). Clearly, both CN and SAI systems can take advantage of such parallel and distributed processing. The reader is referred to (Almasi and Gottlieb, 1989; Uhr, 1984; 1987b) for examples.

Today's homogeneous CN models can be viewed as MIMD computers if the weights or parameters associated with the nodes are viewed as data – which is a reasonable characterization of what occurs during learning. On the other hand, the weights are stored locally in the processors and can be viewed as part of the instruction or the program executed by the nodes (especially if the weights are not modified) in which case, such CN can be thought of as SIMD computers in which each processor executes the same instruction (typically to compute the threshold or sigmoid function).

The control of execution of programs in CN is normally thought of as being distributed with little or no centralized control. While this appears to be true of *some* CN models (e.g., Hopfield networks operating with asynchronous parallel update), it is an arguable point in the case of *most* CN models. In most cases, the elaborate control structures that are necessary in many cases (e.g., the sequential update of Hopfield networks, synchronization of neurons in each layer of a multi–layer back–propagation network during processing and learning phases) are generally left unspecified. In any event, a wide range of centralized or (to various degrees distributed) controls can be embedded in both SAI as well as CN systems (Honavar and Uhr, 1990b).

2.13 Redundancy and Fault Tolerance

Often the term *distributed* is used more or less synonymously with *redundant* and hence fault–tolerant in the CN literature. This is misleading because there are many ways to ensure redundancy of representation, processing and control. One of the simplest involves storing multiple copies of items and/or using multiple processors to replicate the same computation in parallel, and using a simple majority vote or more sophisticated statistical evidence combination processes to pick the result. Redundancy and distributivity are orthogonal properties of representations. And clearly, SAI as well as CN systems can be made redundant and fault–tolerant using the same techniques.

2.14 Statistical, Fuzzy, or Evidential Inference

Many SAI systems represent and reason with knowledge using (typically first–order) logic. If sound inference procedures are used, such reasoning is guaranteed to be truth–preserving. In contrast, It is often claimed that CN models provide noise–tolerant and robust inference because of the probabilistic, fuzzy, or evidential nature of the inference mechanisms used. This is largely due to combination and weighting of evidence from multiple sources through the use of numerical weights or probabilities. It is possible to establish the formal equivalence inference in certain classes of CN models with probabilistic or fuzzy rules of reasoning. But fuzzy logic (Zadeh, 1975; Yager and Zadeh, 1994) operates (as its very name suggests), with *logical* (hence symbolic) representations. Probabilistic reasoning is an important and active area of research in SAI as well (See Pearl, 1988 for details). Heuristic evaluation functions that are widely used in many SAI systems provide additional examples of approximate, that is, not strictly truth–preserving inference in SAI systems. At the same time, it is relatively straightforward to design CN implementations of logical inference (at least in restricted subsets of first–order logic). The reader is refered to (Sun, 1994; Pinkas, 1994) for examples of such implementations.

In many SAI systems, the requirements of soundness and completeness of inference procedures are often sacrificed in exchange for efficiency. In such cases, additional mechanisms are used to (after the fact) verify and if necessary, override the results of inference if they are found to conflict with other evidence. Much research on human reasoning indicates that people occasionally draw inferences that are logically unsound (Johnson–Laird and Byrne, 1991). This suggests that although people may be capable of applying sound inference procedures, they probably take shortcuts when faced with limited computational or memory resources. Approximate reasoning under uncertainty is clearly an important tool that both SAI and CN systems can potentially employ to effectively make rapid, usually reliable and useful, but occasionally fallible inferences in real time.

2.15 SAI and CN As Models of Minds/Brains

Some of the SAI research draws its inspiration from (rather superficial) analogies with the mind and mental phenomena and in turn contributes hypotheses and models to the study of minds; Similarly, many CN models draw their inspiration from (albeit superficial) analogies with the brain and neural phenomena and in turn contribute models that occasionally shed light on some aspects of brain function (Churchland and Sejnowski, 1992).

Today's CN models are at best, extremely simplified caricatures of biological neural networks (Shepherd, 1989; 1990; McKenna, 1994). They lack the highly structured modular organization displayed by brains. The brain appears to perform symbolic, numeric, as well as analog processing. The pulses transmitted by neurons are digital; the membrane voltages are analog (continuous); The molecular level phenomena that involve closing and opening of channels appears to be digital; The diffuse influence of neurotransmitters and hormones appear to be both analog and digital.

Changes in learning appear to involve both gradual changes of the sort modeled by the parameter changing or weight modification algorithms of today's CN as well as major structural changes involving the recruitment of neurons and changes in network topology (Greenough and Bailey, 1988; Honavar, 1989). Also missing from most of today's CN models are elaborate control structures and processes of the sort found in brains including networks of oscillators that control timing. Perception, learning and control in brains appear to utilize events at multiple spatial and temporal scales (Grossberg, 1982). Additional processes not currently modelled by CN systems include processes that include networks of markers that guide neural development, structures and processes that carry information that might be used to generate other network structures, and so on (Honavar and Uhr, 1990).

Clearly, living minds/brains are among the few examples of truly versatile intelligent systems that we have today. They are our existence proof that such systems are indeed possible. So even those whose primary interests are in constructing artificial intelligence systems can ill afford to ignore the insights offered by a study of biological intelligence (McKenna, 1994). (This does not of course mean that such an effort cannot exploit alternative technologies to accomplish the same functions, perhaps even better than their natural counterparts). But it is a misconception to assume that today's CN model brains any more than today's SAI programs model minds. In short, the processes of the minds appear to be far less rigidly structured and far more flexible than today's SAI systems and the brains appear to have a lot more structure, organization, and control than today's homogeneous networks of simple processing elements that we call CN. A rich space of designs that combine aspects of both within a well—designed architecture for intelligence remains to be explored.

3 INTEGRATION OF SAI AND CN

It must be clear from the discussion in the previous sections that at least on the surface it looks like SAI and CN are each appropriate, and possibly even necessary for certain problems, and grossly inappropriate, almost impossible, for others. But of course each can do anything that the other can. The issues are ones of performance, efficiency, and elegance, and not theoretical capabilities as computational models.

This is a common problem in computing. One computer or programming language may be extremely well–suited for some problems but awful for others, while a second computer or language may be the opposite. This suggests several engineering possibilities (Uhr and Honavar, 1994; Honavar, 1994b), including:

1. Try to re–formulate and re–code the problem to better fit the computer or language. This may, for example, entail execution of parallelized CN

equivalents or near–equivalents (possibly improvements) serial SAI programs.

2. Use one computer or language for some parts of the process and the other for others. Thus, one may embed black–boxes that execute certain desired functions (e.g., list processing) within a larger network that includes CN modules for certain other functions (e.g., associative storage and recall of patterns).

3. Build a new computer or language that contains constructs from each, and use these as appropriate.

4. Try to find as elegant as possible a set of primitives that underlie both computers or languages, and use these to build a new system.

The term *hybrid* is beginning to be used for systems that in some way try to combine SAI and CN. If any of the above is called a hybrid probably all of the others should also. But usually hybrid refers to systems of type [2] or [3]. Types [3] and [4] would appear to be better than [2] (although harder to realize), since they would probably be more efficient and more elegant. Thus the capabilities of both SAI and CN should be combined by tearing them apart to the essential components of their underlying processes and integrating these as closely as possible. Then the problem should be re–formulated and re–coded to fit this new system as well as possible. This restates a general principle most people are coming to agree on with respect to the design of multi–computer networks and parallel and distributed algorithms: the algorithm and the architecture should be designed to fit together as well as possible, giving algorithm–structured architectures and architecture–structured algorithms (Uhr, 1984; 1987b; Almasi and Gottlieb, 1989). The remainder of this chapter explores one approach to the design of such architectures for intelligent systems – by generalizing the rather overly restrictive definition of most of today's CN models while retaining their essential advantages.

4 CONNECTIONIST NETWORKS (CN) NARROWLY DEFINED

Connectionist networks or *artificial neural networks* are massively parallel, shallowly serial, highly interconnected networks of relatively simple computing elements or *neurons* (Gallant, 1993; Kung, 1993; Haykin, 1994; Zeidenberg, 1989; McClelland, Rumelhart et al., 1986). More precisely, a CN can be thought of as a directed graph whose nodes apply relatively simple numerical transformations to the (numerical) inputs that they receive via their input links and transmit the resulting output via their output links.

4.1 Nodes in a CN Perform Simple Numerical Computations

The input to an n–input node or neuron n_j in a CN is a pattern vector $\mathbf{X}_j \in \Re^n$ or in the case of binary patterns, by a binary vector $\mathbf{X}_j \in [0,1]^n$. Each neuron computes a relatively simple function of its inputs and transmits outputs to other neurons to which it is connected via its output links. A variety of neuron functions are used in practice. The most commonly used are the *linear*, the *threshold*, and the *sigmoid*. Each neuron has associated with it a set of parameters which are modifiable through learning. The most commonly used parameters are the so–called *weights*. The weights associated with an n–input neuron n_j are represented by an n–dimensional weight vector $\mathbf{W}_j \in \Re^n$. In the case of a linear neuron, the output o_j in response to an input pattern $\mathbf{X}_j$ on its input links is given by the vector dot product $\mathbf{W}_j \cdot \mathbf{X}_j$. In the case of a threshold neuron, $o_j = 1$ if $\mathbf{W}_j \cdot \mathbf{X}_j > 0$ and $o_j = 0$ otherwise. For a sigmoid neuron, $o_j = 1/(1 + e^{-\mathbf{W}_i \cdot \mathbf{X}_j})$. In each of these cases, usually, one of the n inputs is held constant and the corresponding weight is called the threshold.

The functions listed above by no means exhaust the possible choices for neuron functions. A wide range of other possibilities exist – including replacing the simple neurons with digital and analog microcircuits that perform the needed symbolic as well as numeric computations (Shepherd, 1989; Uhr, 1994). More on this later.

4.2 Representation and Computation in CN

The representational and computational power of such networks depends on the functions computed by the individual neurons as well as the architecture of the network (e.g., the number of neurons and how the neurons are connected). In general, a layered *feed–forward* network with n inputs and m outputs can represent some subset of the possible mappings from $\Re^n$ to $\Re^m$. Such networks find application in data compression, feature extraction, pattern classification and function approximation. A great deal is known about the representational power of different classes of feed-forward networks. For example, a 2–layer feed–forward network (not counting the input neurons that simply transmit the components of the input vector) with a *sufficiently large* (finite) number of sigmoid neurons in the first layer and linear neurons in the output layer can approximate to arbitrary accuracy, arbitrary continuous functions on bounded closed subsets of $\Re^n$. When feedback loops are included (thereby allowing the network's output at a given time step to be influenced by its output at a previous time step), it can produce complex temporal dynamics. Such networks find use in applications such as robot motion control and approximation of regular grammars from examples. Connectionist networks become Turing–equivalent in their computational power if we allow the networks to grow arbitrarily large.

Only a small subset of possible network topologies have been studied to date. A much broader range of alternatives exist – including those suggested by the brain and contemporary parallel computer architectures (see below).

4.3 CN Learn by Modifying Parameters

Much of the research on learning in connectionist networks has tended to focus on algorithms for changing the modifiable parameters (weights) in networks with a certain a–priori chosen network architecture using samples of the function to be approximated or patterns to be classified. In other words, the task of the learning algorithms is to find a set of weights that yields a satisfactory approximation of the unknown function on the given samples (The expectation is that the network will generalize well on samples not seen during training – more on this later). This is fundamentally a search or optimization problem and so a variety of linear and non–linear optimization methods (gradient–descent, simulated annealing, etc.) can be used in this context. For details of such algorithms, the reader is referred to (Kung, 1993; Haykin, 1994).

Many connectionist learning algorithms use some form of error–guided search (e.g., changing each modifiable parameter in the direction of the negative gradient of a suitably defined error measure with respect to the parameter of interest. A commonly used error measure in function approximation applications is the *mean squared error* between the desired and actual network outputs. For pattern classification tasks, a number of different error measures motivated by statistics and information theory (e.g., loss functions, cross–entropy) are possible candidates. For data compression or feature extraction, other error measures can be used to map high-dimensional input patterns to feature vectors in a lower dimensional space with minimal loss of information. Common examples of such methods include *competitive learning networks* and *principal component analysis*.

Almost all such learning algorithms involve searching for a set of weights (a solution to the pattern classification problem) in a high–dimensional *weight space* that meet the desired performance criteria (e.g., classification accuracy on a set of training patterns). In order for this approach to succeed, such a solution must in fact lie within the space being searched and the search procedure (e.g., gradient descent) must in fact be able to find it. This means that unless adequate a–priori problem–specific knowledge can be brought to bear upon the problem of choosing an adequate network topology, it is reduced to a process of trial and error. This strongly argues for the need for more powerful learning algorithms – including those that incrementally construct the necessary network structures (see below).

4.4 CN are Typically Only Partially Specified

A CN is typically specified in terms of the transformations performed by the individual nodes, the topology of the graph linking the nodes, and (if the network is designed to learn) the algorithm used to modify the parameters (typically weights) associated with the nodes. It is important to note that the additional network structures necessary to synchronize the nodes in the network, and to switch between behaving (producing outputs as a function of input) and learning (modifying the parameters as dictated by the learning algorithm) are generally left unspecified. The functions of such network structures are typically performed by programs that simulate CN on a traditional stored program computer. It is not hard to see that if all such functions were to be performed by the CN itself, it would need to be augmented with adequate coordination and control structures as well as learning structures to carry out the corresponding tasks (Honavar and Uhr, 1990b).

The next section explores an evolving framework for a broader (and more complete) specification of CN (Honavar, 1990; Honavar and Uhr, 1990; Uhr, 1990) that not only makes explicit the various structures and functions that need to be built into today's CN but also suggests a broad range of potentially promising alternatives for the design of integrated SAI/CN architectures for versatile, powerful, robust, and adaptive intelligent systems.

5 CONNECTIONIST NETWORKS BROADLY DEFINED

The following rather general definition (Uhr, 1990; Honavar and Uhr, 1990) makes clear that a large variety of CN architectures (other than the ones commonly used today) can be constructed. This definition is not meant to be exhaustive or complete in every respect. However, it is intended to facilitate the exploration of a vast space of designs for intelligent systems.

A CN is a graph (of linked nodes) with a particular topology $\mathbf{\Gamma}$. The total graph can be partitioned into three functional sub–graphs – $\mathbf{\Gamma_B}$ (the *behave/act* sub–graph), $\mathbf{\Gamma_\Lambda}$ (the *evolve/learn* sub–graph), and $\mathbf{\Gamma_K}$ (the *coordinate/control* sub–graph). (The motivation for distinguishing among these three functions will become clear later. It primarily has to do with making explicit everything that is necessary to completely specify the structure and behavior of of a CN).

The nodes in a CN compute one or more different type/s of functions: $\mathbf{B}$ (behave/act); $\mathbf{\Lambda}$ (evolve/learn); and $\mathbf{K}$ (coordinate/control). Thus we have a CN $\mathbf{\Omega} = (\mathbf{\Gamma}, \mathbf{B}, \mathbf{\Lambda}, \mathbf{K})$; where $\mathbf{\Gamma} = \mathbf{\Gamma_B} \cup \mathbf{\Gamma_\Lambda} \cup \mathbf{\Gamma_K}$. Each behave, learn, or control function has associated with it, a suitable subgraph of nodes on which it is executed (to avoid cluttering the notation, this assignment is not explicitly specified in the definition).

It should be immediately clear that today's CN are specified (typically only partially) as follows:

- The topology, $\mathbf{\Gamma_B}$ of the sub–network that behaves. (And much of the total graph, including the entire sub–graphs needed to handle learning and control – is usually left unspecified).
- The behave functions computed by each node n_j in $\mathbf{\Gamma_B}$. (These usually take the form of simple (numerical) transformations $\beta_j(\mathbf{W}_j, \mathbf{X}_j)$ where $\mathbf{W}_j$ is the weight vector associated with node n_j and $\mathbf{X}_j$ is the input vector to node n_j. And common choices are, as already pointed out, threshold, sigmoid, linear, and radial basis functions. Typically, the same node function is used with each of the nodes in the network, or at least, for each node in a given layer of a multi–layer network).
- A learning algorithm that translates to a learning function λ_j that modifies the parameters (typically the weight vector $\mathbf{W}_j$ at each node n_j in $\mathbf{\Gamma_B}$. (The actual sub–networks that are needed to actually compute and

make these changes or to switch between learning and behaving are left unspecified).

The broad definition of CN given above when superimposed with the critical examination of SAI and CN systems given in section 2 above suggests several potentially powerful extensions to today's CN – including more powerful structures and processes for behaving, control, and learning – in other words, a rich space of integrated SAI/CN architectures for intelligent systems.

6 INTEGRATED SAI/CN DESIGNS FOR INTELLIGENT SYSTEMS

The broad definition of CN outlined above suggests a general approach to the design of hybrid or integrated architectures for intelligent systems – one that essentially entails extending the range of behave, learn, and control structures and processes so that the capabilities typically associated with SAI systems (e.g., list processing, deductive inference, etc.) can be incorporated in relatively natural and well–integrated ways into such systems. This section outlines a range of such potentially useful extensions to today's CN.

6.1 More Powerful Behave Structures and Functions

As already pointed out, there is no compelling reason to limit ourselves to the simple behave functions (e.g., threshold, linear, sigmoid) computed by the nodes in today's CN. That would be tantamount to arguing that all computer programs must be written using a minimal set of Turing–machine instructions. A much broader range of behave structures and functions may be used as appropriate (along with learning and control structures and functions that are designed to work hand–in–hand with such functions). Many such analog, digital, as well as hybrid analog–digital behave structures and functions are suggested by digital and analog computers, micro–circuits found in brains, and the design of algorithms and computer programs as well as some of the hybrid SAI/CN architectures that have been explored to date. The reader is refered to (Shepherd, 1989; Uhr, 1994) for a detailed discussion of several such micro–circuits. The short list that follows is suggestive of the wide range of possibilities that can be beneficially incorporated into CN:

- Micro–circuits for comparison of numeric, symbolic, or iconic patterns to determine similarities amd differences between two or more patterns (including stored templates).
- A broad range of flexible pattern matching functions – including those for matching highly structured patterns of symbols – e.g., strings, lists, labeled trees and graphs – with stored patterns. (A variety of metrics may be used to compute the degree of match – including weighted *edit distances* (Goldfarb, 1994; Honavar, 1994) of the sort used in structural pattern recognition systems).
- Micro–circuits for computing minimal generalizations over two or more patterns or minimal specialization of a stored template to prevent match with a given pattern.
- Automata for parsing symbol structures – preferably fault–tolerant CN implementations (Chen and Honavar, 1994).
- Modifiable temporary memories – including CN implementations of such structures using recurrent networks.
- Encoders/decoders for transforming symbol–structures for efficient, fault–tolerant transmission between spatially separated modules of CN
- Oscillators, clocks, and variable delay circuits.
- Specialized micro–circuits for specific perceptual tasks – e.g., spot and oriented edge detectors of the sort found in the mammalian visual system; circuits for computing different intrinsic image properties such as depth, shape, and motion from visual images.
- Shifters for alignment and comparison of multiple sources of input – e.g., from two spatially separated visual sensors.
- Building blocks for components of semantic memories, associative networks, frames, and schema.
- Expandable fault–tolerant content–addressible memories.
- Serial–to–parallel (or temporal to spatial) mapping networks and their generalizations.
- Winner–take–all networks to facilitate choice among several competing alternatives.

- Micro–circuits for performing dynamical variable binding and unification necessary for matching complex symbolic expressions, performing logical inference, and simulating production systems.
- Dynamically configurable micro–circuits for solving optimization problems through the minimization of suitably defined energy functions.
- Micro–circuits for performing operations on lists such as insertion, deletion, and substitution of elements – analogous to the operations supported by a conventional symbol processing language like LISP.
- Micro–circuits that peform simple set operations such as union, intersection, and difference.
- Micro–circuits that perform simple spatial and temporal inferences on input spatial patterns or temporal event sequences (e.g., determining whether a particular object is contained within another object in a visual scene; or deciding if a particuar event preceded another in time).
- Micro–circuits that perform simple statistical computations – e.g., computing histograms, estimating relative frequencies and probability density functions, statistical means, etc.

6.2 More Appropriate, and When Indicated, Modular Network Structures

As already pointed out, CN can use virtually any graph topology but only a small subset have been used to date. These include simple one–layer or multi–layer feed–forward networks with complete connectivity between layers of nodes (typically used for function approximation, associative storage, recall, or classification of pattern vectors, and data compression); recurrent networks with feedback connections from higher layers to lower layers (typically used for temporal pattern recognition and sequential prediction tasks). However, a wide range of other graph topologies may be used as dictated by considerations of efficiency and design constraints imposed by the technology (e.g., VLSI) used for the physical realization of such networks. A variety of such network topologies are suggested by contemporary developments in the design of parallel and distributed architecures and algorithms (Uhr, 1984; Almasi and Gottlieb, 1989). A few such possibilities are enumerated below:

- Near–neighbor connectivity of the sort used in two–dimensional arrays or N–dimensional hypercubes of processors (such topologies have been especially useful in performing a variety of intrinsic image (shape, motion, texture, depth, etc. from gray scale images) computations in image processing and computer vision (Ballard and Brown, 1982; Uhr, 1987a; 1987b; Wechsler, 1990)).

- Near–neighbor tree–or–pyramid–like converging connectivity from one layer to the next of the sort used in several computer vision architectures (Honavar and Uhr, 1989a; 1989b; Uhr, 1987a; Tanimoto and Klinger, 1980).
- Network topologies that are specifically designed to expedite certain computations (e.g., Hough transforms that detect lines or parameterized curves in images (Ballard and Brown, 1982).
- Dynamically configurable network topologies that facilitate certain computations on demand – e.g., structures used to construct logic proofs (Pinkas, 1994).
- Specialized networks that carry context sensitive control and coordination signals – e.g., to focus or control attention.
- Modular network structures that reflect natural decomposition of tasks (e.g., object identification and object location in a visual image) – including decompositions discovered through learning or evolutionary processes.

6.3 More Powerful Learning Structures and Functions

As already pointed out, the only major forms of learning that have been studied in any detail in the context of CN are rote learning (which essentially involves storing patterns for future retrieval as in associative memories) and inductive learning (or learning from examples). A variety of other powerful forms learning including deductive, abductive, and discovery techniques developed in the context of SAI systems do not have any CN counterparts at present. In today's CN is almost always limited to processes that change modifiable parameters (typically the weight vectors) within an otherwise a–priori fixed network topology. The list that follows is meant to be merely suggestive of a few of the much wider range of potentially powerful alternatives that exist:

- Learning that modifies the behave functions β_j associated with each of the nodes n_j in the behave sub–graph $\mathbf{\Gamma_B}$ of a CN or selects among a set of candidate functions based on the characteristics of the problem at hand.

- Learning that modifies the control functions κ_j associated with each of the nodes n_j in the control sub–graph $\mathbf{\Gamma_K}$ of a CN. For example, such mdifications might involve control of the locus, rate, and type of learning.
- Learning that modifies the learning functions λ_j associated with each of the nodes n_j in the learning sub–graph $\mathbf{\Gamma_\Lambda}$ of a CN or selects from among a set of candidate learning rules.
- Learning that involves modification of the topology of behave, learn, or control sub–graphs by addition and deletion of individual nodes, links, micro–circuits, layers, or more generally necessary sub–graphs. Given suitable constructive learning mechanisms, CN can adaptively search for and assume whatever connectivity is appropriate for the tasks at hand within the specific design constraints (e.g., local connectivity for VLSI implementation). Only some of the simplest constructive or generative algorithms that dynamically modify the network topology by recruiting nodes as necessary for a particular pattern classification or function approximation task have been examined to date (Honavar and Uhr, 1989a; 1989b; 1993, Gallant, 1993; Kung, 1993). Such algorithms extend the search for solution to (a suitably constrained) space of network topologies. In this context efficient randomized search techniques such as genetic algorithms are worth exploring (Balakrishnan and Honavar, 1994). Complementary processes of deletion of nodes and links are also of interest and a few such mechanisms have been explored to date (Kung, 1993).
- Learning that fine–tunes the symbolic representations such as rules and grammars used in SAI systems using the parameter–modification processes of CN systems (Gallant, 1993; Shavlik, 1994; Honavar, 1992b; Goldfarb and Nigam, 1994).
- Learning that uses forms of inference such as deduction and abduction which are rarely used in today's CN systems which rely primarily on induction.

6.4 More Powerful Control Structures and Functions

As already pointed out, despite assertions to the contrary, some subtle control mechanisms are used (without ever being made explicit – because they are handled by the programs that simulate such CN). On the other hand, perhaps one of the most important limitations of today's CN is their relatively impoverished repertoire of control structures and processes. Many such mechanisms are suggested by an examination of computers and programs, biological organisms, and brains (Honavar and Uhr, 1990b). The list that follows is suggestive of the wide variery of such potentially powerful control structures and processes that may be beneficially incorporated into CN:

- Control can be introduced by building in particular structures and processes as needed – as in specifying particular micro–circuits like winner–take–all nets or decision–trees that make choices and selectively transmit the appropriate information.
- Partial control can be built in globally – e.g., through the use of global clocks that synchronize subsets of processors.
- The topology of the network, sub–net, and local micro-circuits can exercise major control functions – as when a tree of processors successively transforms, combines, and reduces information, or a pipeline of arrays sequentially applies a series of transformations to the input.
- Complete (and, if desired, rigid) control of the sort found in conventional computers and multi–computers (e.g., execution of instructions in order; controlled sequences of state transitions) can be built in either centrally or locally at each node or small sets of nodes in the CN.
- A variety of coordination and control structures (e.g., message passing, blackboard structures for messages, instruction broadcasting, multiplexing, conflict resolution) used in multicomputer networks can be built into the CN.
- A host of control mechanisms of the sort found in conventional programming language constructs (conditional execution, loops, etc.) can be built into local or global microcircuits embedded in the CN.
- CNs may be provided with compact (gene–like) encodings of their structural and functional properties (e.g., the sizes of receptive fields, general topological constraints on connectivity, etc.). Such encodings may be transformed through genetic operators (e.g., crossover, mutation) to yield variant CN specifications. Environmental rewards and punishments may be used as means of guiding whole CN populations to evolve so as to perform better at tasks presented to them by the environment.
- DNA–like encodings can be incorporated, along with the capability to make copies, linking networks over which these copies can be sent, and decoders to transform these encodings into specifications for network structures and processes to be realized.

- Gene–like information can be used to dynamically specify different types of functional units in CN (analogous to the mechanisms of cellular differentiation). Controls can activate or suppress the expression of different functional properties in CN nodes or node ensembles.
- Local interactions of the sort found on the surface of cells that are bound by cell adhesion molecules might be used to determine how to build single units and larger micro–circuits into successively larger structures (e.g., through a specification of how the CN microcircuits are to be assembled together).
- The immune system's rapid, evolution–like adaptations suggest the possibility of triggering massive proliferations of a range of variant nodes or micro–circuits to serve a variety of control functions under specific environmental conditions.
- Mechanisms analogous to chemical markers and pilot cells can – guided by information of the sort found in chemical gradients and lock–in–key–type templates – combine to build complex network structures.
- The complex interactions between chains of enzymes suggest the possibility of mechanisms that can modify, build, and recycle network structures in a self–sustaining manner.
- Multi–messenger pathways analogous to those supported by intra– as well as inter–cellular communication (e.g., axonal transport mechanisms, membrane proteins) can be built into and among individual CN units and micro–circuits of units.
- Several different types of global controls sub–networks can be used, tailored to and serving different purposes – much like the brain's neurotransmitters, global electrical waves, and chemical messenger systems. Such systems can be embedded in the CN using token–passing networks or distributed encoder–decoder structures.
- Neuromodulator–like influences can be achieved by incorporating linking sub-networks that contain links with different amounts of delays, transmit information to changes thresholds, to alter network plasticity.
- Modulatory networks analogous to hormones can now have relatively diffuse, slow–acting effects on such global processes as memory storage, memory modification or selective recall of stored memories.
- Regulatory subnetworks can also be used to initiate, modulate, and terminate plasticity of specific CN modules in a controlled fashion during learning.

- Specific control subnetworks can be embedded into the CN to instantiate processes like attention – selectively enhancing or attenuating the relative contributions of different aspects of the environmental stimuli.
- A variety of simple controls that regulate various aspects of learning (including the form and content of the learned representations), alter vigilance, and choose between different learning strategies can be incorporated into CN
- A variety of contextually driven switching mechanisms can be built into the CN to alter, in a dynamic fashion, the functions of different CN modules or the interactions among modules.
- Oscillators can be used to handle a variety of important problems – e.g., to switch between processes; to decide when to initiate, or to terminate, a process; to sample the environmental input at a desired rate.
- Clocks or networks of clocks can execute the equivalent of the *while* loop of a conventional programming languages in CN. For example, information can cycle through several interior layers until a decision network is triggered – which in turn fires into nodes, switching them on so that they in turn fire out in synchrony to other regions of the network.
- Synchronized sampling of environmental stimuli in different sensory modalities can be used as a means of multi–sensory integration.
- Network structures that initiate, transmit and terminate global wave–like signals to large regions of the network can instantiate arousal–like processes or help prepare entire network modules to better process the incoming signals.
- Multi–level shifter-like structures embedded in the CN can be used for a variety of control functions (e.g., to register inputs from two visual sensors, to compensate for the motion of objects in the scene, to dynamically control the degree of smoothing to suppress the noise in the input).
- Feedback pathways can be built into CN to subserve a number of subtle control functions (e.g., selective modulation of the sensory signals as a function of some assessments made at the higher levels, selective attention to specific aspects of the environmental stimuli ordered by their salience).
- Control structures that dynamically link subnetworks that are activated by different features in the environmental stimuli into transient network assemblies can serve a host of useful functions (e.g., dynamic variable binding necessary for complex inferences) in CN.

- Specific subnetworks can be built into the CN that ensure a proper balance in the allocation of different network resources (e.g., nodes, links, long–range communication networks) among different functions as the network learns and evolves.
- Network controls can dynamically alter the incentives for CN units, micro-circuits or functional modules to cooperate (as opposed to compete) with each other.

7 SUMMARY

SAI and CN are generally regarded as two disparate and perhaps fundamentally different approaches to modelling cognitive processes and engineering intelligent systems. A closer examination of the two approaches clearly demonstrates that there can be no fundamental incompatibility between them. They essentially offer two different (but general–purpose) description languages for modelling systems in general, and intelligent systems in particular.

The choice between the two – much like deciding between two general–purpose programming languages (or equivalently, virtual computer architectures) is primarily a matter ofo convenience, efficiency and elegance – given a set of design and performance constraints. SAI and CN each demonstrate at least one way of performing certain tasks naturally and thus pose the problem to the other of doing the same thing perhaps more elegantly, efficiently, or robustly than the other.

Living minds/brains offer an existence proof of at least one architecture for general intelligence. SAI and CN paradigms together offer a wide range of architectural choices. Each architectural choice brings with it some obvious (and some not so obvious) advantages as well as disadvantages in the solution of specific problems using specific algorithms, given certain performance demands and design constraints imposed by the available choices of physical realizations of the architecture. Together, the cross—product of the space of architectures, algorithms, and physical realizations constitutes a large and interesting space of possible designs for intelligent systems. Examples of systems resulting from a judicious integration of concepts, constructs, techniques and technologies drawn from both traditional artificial intelligence systems and artificial neural networks clearly demonstrate the potential benefits of exploring this space. And, perhaps more importantly, the rather severe practical limitations of today's SAI and CN systems strongly argues for the need for a systematic exploration of such design space.

This suggests that it might be fruitful to approach the choice of architectures, implementations, and their physical realizations using the entire armamentarium of tools drawn from the theory and practice of computer science — including the design of programming languages (and hence virtual architectures), computers, algorithms, and programs. Our primary task is to identify subsets of Turing—computable functions necessary for general intelligence, an appropriate mix of architectures for supporting specific subsets of these functions, as well as appropriate realizations of such architectures in physical devices. In the short–term, this entails the design and analysis of *hybrid* systems that use SAI and CN modules to perform different but well–coordinated sets of functions in specific applications. In the long—term, a coherent theoretical framework for analysis and synthesis of such systems needs to be developed. One way to approach this task is to place both SAI and CN systems within a common framework and identify the various significant dimensions that characterize the resulting space of designs for intelligent systems. The attempt to define CN (and SAI) systems in broader than usual terms in this chapter is an attempt in this direction. This makes explicit some of the dimensions of the design space for intelligent systems that CN (broadly defined) offer – in terms of a variety of behave, control, and learning structures and functions as well as the attendant design/performance tradeoffs among among: parallel versus serial processing; localized versus distributed representation, processing, and control (in space as well as over time); symbolic, numeric, and analog representation and processing/inference structures and processes; and related issues. It also suggests a number of ways to extend today's CN models – by providing them with more powerful building blocks and functional micro–circuits, more appropriate modular topologies, more powerful learning structures and processes, and a rich variety of control structures.

Acknowledgements

The authors would like to thank Professors Suran Goonatilake and Sukhdev Khebbal for their invitation to contribute this chapter to their book. This work was partially supported by the National Science Foundation grant (IRI–9409580) to Vasant Honavar. Vasant Honavar would like to thank his students (especially Rajesh Parekh, Chun–Hsieh Chen and Karthik Balakrishnan) and

Professors Bill Robinson and Gregg Oden for many useful discussions on the topics addressed in this chapter.

REFERENCES

[1] Almasi, G. S. & Gottlieb, A. (1989). *Highly Parallel Computing.* New York: Benjamin–Cummings.

[2] Arbib, M. A. (1972). *The Metaphorical Brain.* New York: Wiley–Interscience.

[3] Arbib, M. A. (1994). Schema Theory: Cooperative Computation for Brain Theory and Distributed AI. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. and Uhr, L. (Ed.) San Diego, CA: Academic Press.

[4] Balakrishnan, K. & Honavar, V. (1994). Evolutionary Approaches to the Exploration of the Design Space of Artificial Neural Networks. In preparation.

[5] Ballard, D. & Brown, C. (1982) *Computer Vision.* Englewood Cliffs, NJ: Prentice Hall.

[6] Barnden, J. A. (1994). How Might Connectionist Networks Represent Propositional Attitudes? In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[7] Boden, M. (1994). Horses of a Different Color? In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[8] Booker, L. E., Riolo, R. L., & Holland, J. H. (1994). Learning and Representation in Classifier Systems. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. and Uhr, L. (Ed.) San Diego, CA: Academic Press.

[9] Bookman, L. (1994). A Framework for Integrating Relational and Associational Knowledge for Comprehension. In: *Computational Architectures Integrating Symbolic and Neural Processes.* Sun, R. & Bookman, L. (Ed.) Boston: Kluwer.

[10] Buchanan, B. G. & Wilkins, D. C. (1993). *Readings in Knowledge Acquisition and Learning.* San Mateo, CA: Morgan Kaufmann.

[11] Carpenter, G. & Grossberg, S. (1994). Integrating Symbolic and Neural Processes in a Self–Organizing Architecture for Pattern Recognition and Prediction. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[12] Chandrasekaran, B. Josephson, S. G. (1994). Architecture of Intelligence: The Problems and Current Approaches to Solutions. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[13] Chen, C., & Honavar, V. (1994). Neural Networks for Inference. In preparation.

[14] Churchland, P. S. & Sejnowski, T. J. (1992). *The Computational Brain.* Boston, MA: MIT Press.

[15] Cohen, D. (1986). *Introduction to Computer Theory.* New York: Wiley.

[16] Dolan, C. P. & Smolensky, P. (1989). Tensor product production system: a modular architecture and representation. *Connection Science*, 1:53–58.

[17] Dyer, M. (1994). Grounding Language in Perception. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[18] Feigenbaum, E. A. (1963). In: *Computers and Thought.* Feigenbaum, E. A. & Feldman, J. (Ed.) New York: McGraw–Hill.

[19] Feldmann, J. A. & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science*, 6:205–264.

[20] Fodor, J. (1976). *The Language of Thought.* Boston, MA: Harvard University Press.

[21] Fodor, J. & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: a critical analysis. In: *Connections and Symbols.* Pinker, S. & Mehler, J. (Ed.) Cambridge, MA: MIT Press.

[22] Forgy, C. L. (1982). RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligence*, 19:17–37.

[23] Fu, K. S. (1982). *Syntactic Pattern Recognition and Applications.* Englewood Cliffs, NJ: Prentice Hall.

[24] Gallant, S. (1993). *Neural Networks and Expert Systems.* Cambridge, MA: MIT Press.

[25] Genesereth, M. R., & Nilsson, N. J. (1987). *Logical Foundations of Artificial Intelligence.* Palo Alto, CA: Morgan Kaufmann.

[26] Ginsberg, M. (1993). *Essentials of Artificial Intelligence.* San Mateo, CA: Morgan Kaufmann.

[27] Goldfarb, L. & Nigam, S. (1994). The Unified Learning Paradigm: A Foundation for AI. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[28] Goonatilake, S. & Khebbal, S. (1994). (Ed.) *Hybrid Intelligent Systems.* London: Wiley.

[29] Greenough, W. T. & Bailey, C. H. (1988). The Anatomy of Memory: Convergence of Results Across a Diversity of Tests. *Trends in Neuroscience* 11, 142–147.

[30] Grossberg, S. (1982). *Studies of Mind and Brain.* Boston, MA: Reidel.

[31] Hanson, S. J. & Burr, D. (1990). What Connectionist Models Learn: Learning and Representation in Connectionist Networks. *Behavior and Brain Sciences*, 13:1–54.

[32] Harnad, S. (1990). The Symbol Grounding Problem. *Physica D*, 42:335–346.

[33] Harnad, S., Hanson, S. J., & Lubin, J. (1994). Learned Categorical Perception in Neural Nets: Implications for Symbol Grounding. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[34] Haykin, S. (1994). *Neural Networks.* New York: Macmillan.

[35] Hewitt, C. (1977). Viewing Control Structures as Patterns of Passing Messages. *Artificial Intelligence*, 8:232–364.

[36] Hillis, D. (1985). *The Connection Machine.* Cambridge, MA: MIT Press.

[37] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

[38] Hopfield, J. J. (1982). Neural Networks and Physical Systems With Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558.

[39] Honavar, V. (1989). Perceptual Development and Learning: From Behavioral, Neurophysiological and Morphological Evidence to Computational Models. Tech. Rep. 818. Computer Sciences Dept., University of Wisconsin, Madison, Wisconsin.

[40] Honavar, V. (1990). *Generative Learning Structures for Generalized Connectionist Networks*. Ph.D. Dissertation. University of Wisconsin, Madison, Wisconsin.

[41] Honavar, V. (1992a). Some Biases For Efficient Learning of Spatial, Temporal, and Spatio–Temporal Patterns. In: *Proceedings of the International Joint Conference on Neural Networks*. Beijing, China.

[42] Honavar, V. (1992b). Inductive Learning Using Generalized Distance Measures. In: *Proceedings of the SPIE Conference on Adaptive and Learning Systems*. Orlando, Florida.

[43] Honavar, V. (1994a). Toward Learning Systems That Use Multiple Strategies and Representations. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. Honavar, V., & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[44] Honavar, V. (1994b). Symbolic Artificial Intelligence and Numerical Artificial Neural Networks: Towards a Resolution of the Dichotomy. In: *Computational Architectures Integrating Symbolic and Neural Processes*. Sun, R., & Bookman, L. (Ed.) New York: Kluwer.

[45] Honavar, V. & Uhr, L. (1989a). Brain–Structured Connectionist Networks that Perceive and Learn. *Connection Science*, 1:139–159.

[46] Honavar, V. & Uhr, L. (1989b). Generation, Local Receptive Fields, and Global Convergence Improve Perceptual Learning in Connectionist Networks. In: *Proceedings of the 1989 International Joint Conference on Artificial Intelligence*, San Mateo, CA: Morgan Kaufmann.

[47] Honavar, V. & Uhr, L. (1990a). Symbol Processing Systems, Connectionist Networks, and Generalized Connectionist Networks. Tech. Rep. 90–23. Department of Computer Science, Iowa State University, Ames, Iowa.

[48] Honavar, V. & Uhr, L. (1990b). Coordination and Control Structures and Processes: Possibilities for Connectionist Networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:277–302.

[49] Honavar, V. & Uhr, L. (1993) Generative Learning Structures and Processes for Generalized Connectionist Networks, *Information Sciences*, 70:75–108.

[50] Honavar, V. & Uhr, L. (1994a). (Ed.) *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* San Diego, CA.: Academic Press.

[51] Johnson–Laird, P. & Byrne, J. (1991). *Deduction.* New York: Lawrence Erlbaum.

[52] Klir, G. J. (1985). *Architecture of Systems Problem Solving.* New York: Plenum.

[53] Kowalski, R. A. (1977). *Predicate Logic as a Programming Language.* Amsterdam: North–Holland.

[54] Kung, S. Y. (1993). *Digital Neural Networks.* New York: Prentice Hall.

[55] Levine, D. S. Aparicioiv, M. (1994). (Ed.) *Neural Networks for Knowledge Representation.* New York: Lawrence Erlbaum.

[56] Maclennan, B. J. (1994). Image and Symbol – Continuous Computation and the Emergence of the Discrete. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[57] McCulloch, W. S. and Pitts, W. (1943). A Logical Calculus of the Ideas Immanent in Neural Activity. *Bulletin of Mathematical Biophysics*, 5:115–137.

[58] McClelland J., Rumelhart, D. et al. (1986). (Ed.) *Parallel Distributed Processing.* Boston, MA: MIT Press.

[59] McKenna, T. (1994). The Role of Inter–Disciplinary Research Involving Neuroscience in the Design of Intelligent Systems. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[60] Mead, C. (1989). *Analog VLSI and Neural Systems.* Reading, MA: Addison–Wesley.

[61] Medsker, L. (1994). *Hybrid Neural Network and Expert Systems.* Boston, MA: Kluwer.

[62] Miikkulainen, R. (1994). Integrated Connectionist Models: Building AI Systems on Subsymbolic Foundations. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[63] Michalski, R. S. (1993). Toward a Unified Theory of Learning: Multi–Strategy Task–Adaptive Learning. In: *Readings in Knowledge Acquisition and Learning.* Buchanan, B. G., & Wilkins, D. C. (Ed.) San Mateo, CA: Morgan Kaufmann.

[64] Miclet, L. (1986). *Structural Methods in Pattern Recognition.* New York: Springer–Verlag.

[65] Minsky, M. (1963). Steps Toward Artificial Intelligence. In: *Computers and Thought.* Feigenbaum, E. A. & Feldman, J. (Ed.) New York: McGraw–Hill.

[66] Minsky, M. (1975). A Framework for Representing Knowledge. In: *The Psychology of Computer Vision.* Winston, P. H. (Ed). New York: McGraw–Hill.

[67] Minsky, M. (1986). *Society of Mind.* New York: Basic Books.

[68] Newell, A. (1980). Symbol Systems. *Cognitive Science*, 4:135–183.

[69] Newell, A. (1990). *Unified Theories of Cognition.* Cambridge, MA: Harvard University Press.

[70] Newell, A., Shaw, J. C., & Simon, H. (1963). In: *Computers and Thought.* Feigenbaum, E. A., & Feldman, J. (Ed.) New York: McGraw–Hill.

[71] Norman, D. A. (1986). Reflections on Cognition and Parallel Distributed Processing. In: *Parallel Distributed Processing.* McClelland, J., Rumelhart. D. et al. (Ed.). Cambridge, MA: MIT Press.

[72] Norvig, P. (1992). *Paradigms in Artificial Intelligence Programming.* Palo Alto, CA: Morgan Kaufmann.

[73] Oden, G. C. (1994). Why the Difference Between Connectionism and Anything Else is More Than You Might Think But Less Than You Might Hope. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[74] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* Palo Alto, CA: Morgan Kaufmann.

[75] Pinkas, G. (1994). A Fault–Tolerant Connectionist Architecture for the Construction of Logic Proofs. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[76] Pollack, J. (1990). Recursive Distributed Representations. *Artificial Intelligence*, 46:77–105.

[77] Quillian, M. R. (1968). Semantic Memory. In: *Semantic Information Processing.* Minsky, M. (Ed.) Cambridge, MA: MIT Press.

[78] Rajaraman, V. (1981). *Analog Computation and Simulation.* Englewood Cliffs: Prentice Hall.

[79] Rashevsky, N. (1960). *Mathematical Biophysics.* New York: Dover.

[80] Rosenblatt, F. (1962). *Principles of Neurodynamics.* Washington, DC: Spartan.

[81] Schneider, W. (1987). Connectionism: is it a paradigm shift for psychology? *Behavior Research Methods, Instruments, and Computers*, 19:73–83.

[82] Selfridge, O. G. & Neisser, U. (1963). Pattern Recognition by Machine. In: *Computers and Thought.* Feigenbaum, E. A. & Feldman, J. (Ed.) New York: McGraw–Hill.

[83] Sharkey, N. & Jackson, S. J. (1994). Three Horns of a Representational Dilemma. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[84] Shastri, L. & Ajjanagadde, V. (1989). Connectionist System for Rule Based Reasoning with Multi–Place Predicates and Variables. Tech. Rep. MS–CIS–8906. Computer and Information Science Dept. University of Pennsylvania, Philadelphia, PA.

[85] Shavlik, J. W., & Dietterich, T. G. (1990). (Ed). *Readings in Machine Learning.* San Mateo, California: Morgan Kaufmann.

[86] Shavlik, J. W. (1994). A Framework for Combining Symbolic and Neural Learning. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[87] Shepherd, G. M. (1989). The significance of real neuron architectures for neural network simulations. In: *computational Neuroscience.* Schwartz, E. (Ed.) Cambridge, MA: MIT Press.

[88] Shepherd, G. M. (Ed.) (1990). *Synaptic Organization of the Brain.* New York, NY: Oxford University Press.

[89] Smolensky, P. (1990). Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. *Artificial Intelligence*, 46:159–216.

[90] Sowa, J. F. (1984). Conceptual Structures: Information Processing in Mind and Machine. Reading, Massachusetts: Addison–Wesley.

[91] Sun, R. (1994). Logic and Variables in Connectionist Models: A Brief Overview. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[92] Sun, R. & Bookman, L. (1994). (Ed.) *Computational Architectures Integrating Symbolic and Neural Processes.* New York: Kluwer.

[93] Tanimoto, S. L. & Klinger, A. (1980). *Structured Computer Vision: Machine Perception Through Hierarchical Computation Structures.* New York: Academic Press.

[94] Tsang, E. (1993). *Foundations of Constraint Satisfaction.* New York: Academic Press.

[95] Uhr, L. & Vossler, C. (1963). A Pattern Recognition Program that Generates, Evaluates, and Adjusts its Own operators. In: *Computers and Thought.* Feigenbaum, E. & Feldman, J. (Ed). New York: McGraw–Hill.

[96] Uhr, L. (1973). *Pattern Recognition, Learning, and Thought.* New York: Prentice–Hall.

[97] Uhr, L. (1979). Parallel–serial production systems with many working memories. In: *Proceedings of the Fifth International Joint Conference on Artificial Intelligence.*

[98] Uhr, L. (1980). In: *Structured Computer Vision.* Tanimoto, S., and Klinger, A. (Ed.) New York: Academic Press.

[99] Uhr, L. (1984). *Algorithm–Structured Computer Arrays and Networks.* New York: Academic Press.

[100] Uhr, L. (1987a). Parallel Computer Vision. New York: Academic Press.

[101] Uhr, L. (1987b). *Multi–computer Architectures for Artificial Intelligence.* New York: Wiley.

[102] Uhr, L. (1990). Increasing the Power of Connectionist Networks by Improving Structures, Processes, Learning. *Connection Science*, 2:179–193.

[103] Uhr, L. (1994). Digital and Analog Sub–Net Structures for Connectionist Networks. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[104] Uhr, L. & Honavar, V. (1994). Artificial Intelligence and Neural Networks: Steps Toward Principled Integration. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[105] Van Gelder & Port (1994). Beyond Symbolic: Prolegomena to a Kama–Sutra of Compositionality. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. & Uhr, L. (Ed.) San Diego, CA: Academic Press.

[106] Waterman, D. A. (1985). *A Guide to Expert Systems.* Reading, MA: Addison–Wesley.

[107] Wechsler, H. (1990). *Computational Vision.* New York: Academic Press.

[108] Winston, P. H. (1992). *Artificial Intelligence.* Boston, MA: Addison–Wesley.

[109] Yager, R. R. & Zadeh, L. A. (1994). (Ed.) *Fuzzy Sets, Neural Networks, and Soft Computing.* New York: Van Nostrand Reinhold.

[110] Zadeh, L. A. (1975). Fuzzy Logic and Approximate Reasoning. *Synthese*, 30:407–28.

[111] Zeidenberg, M. (1989). *Neural Networks in Artificial Intelligence.* New York: Wiley.

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

Tech Report: TR94-16
Submission Date: August 18, 1994