Symbol Processing Systems, Connectionist Networks, and Generalized Connectionist Networks

Vasant Honavar
Department of Computer Science
Iowa State University

Leonard Uhr
Computer Sciences Department
University of Wisconsin-Madison

Technical Report #90-23, December 1990
Department of Computer Science
Iowa State University, Ames, IA 50011

Symbol Processing Systems, Connectionist Networks, and Generalized Connectionist Networks

Vasant Honavar
Department of Computer Science
Iowa State University

Leonard Uhr
Computer Sciences Department
University of Wisconsin-Madison

Abstract

Many authors have suggested that SP (symbol processing) and CN (connectionist network) models offer radically, or even fundamentally, different paradigms for modeling intelligent behavior (see Schneider, 1987) and the design of intelligent systems. Others have argued that CN models have little to contribute to our efforts to understand intelligence (Fodor & Pylyshyn, 1988). A critical examination of the popular characterizations of SP and CN models suggests that neither of these extreme positions is justified. There are many advantages to be gained by a synthesis of the best of both SP and CN approaches in the design of intelligent systems. The Generalized connectionist networks (GCN) (alternately called generalized neuromorphic systems (GNS)) introduced in this paper provide a framework for such a synthesis.

1. Symbol Processing Architectures for Intelligent Systems

The symbol processing approach to the design of intelligent systems was summarized by Newell (1980) and Newell & Simon (1972) in terms of what they called the *physical symbol systems* and by Fodor (1976) in terms of what he called the *language of thought*. In this framework, perception and cognition are tantamount to acquiring and manipulating symbolic representations. It is suggested that the symbol structures that constitute such representations have a counterpart in the physical structure of the brain and/or the brain's internal states. Models of intelligent systems developed within this framework typically are (but do not have to be) based on the von Neumann serial stored program model of computation. Popular interpretations of this definition are often overly restrictive, and appear to exclude (for no good reason) systems that perceive, learn, and reason with non-symbolic (e.g., iconic or analogic) representations, or using numerically-encoded probabilistic or fuzzy inference structures. The following sections critically examine popular conceptions and major strengths and weaknesses of SP models of intelligent systems.

This is a draft of a paper that is currently under review. Comments and suggestions for improvement will be appreciated. This work was partially supported by the University of Wisconsin-Madison Graduate School and the Iowa State University College of Liberal Arts and Sciences.

Knowledge Encoded in Symbol-Structures and Accessed by Programs

Knowledge in SP models is encoded in complex data structures (e.g., lists, trees, etc.) and (often recursive) procedures that are stored in a central memory, and is typically accessed by a program through the use of addresses and indices. This is clearly not an essential property of SP systems: for example, data-flow machines and finite state automata do not need memory-access mechanisms, since the memory is locally available to each processor.

The knowledge representations in SP models typically use highly abstract symbolic descriptions. This in fact allows such systems to attack *high-level* cognitive tasks while circumventing the difficulties involved in arriving at the knowledge representations starting with raw sensory data. Naturally, such systems fail miserably in domains in which it is difficult to encapsulate the relevant knowledge into a *knowledge base* in the form of a manageable set of rules and symbolic descriptions at the necessary level of precision.

Deterministic, Serial, Explicitly Rule-Driven Inference

The computations in SP systems typically involve deterministic and explicitly rule-driven symbol manipulation or serial chains of inference of the type if **x,** *then* **y** of the sort used in *production systems* e.g., SOAR (Laird, Newell, & Rosenbloom, 1987) or *logic programming* (Kowalski, 1979). The control and coordination of such logical inference chains and the access of stored knowledge structures is often very rigid and centralized. But there is no need to limit such production systems to using symbols and/or serial processing with centralized control: several rules may be activated in parallel and can have interactive effects on the final outcome (as in parallel production systems (e.g., Uhr, 1979)).

There is no compelling reason for using explicit rules in SP systems; the rules can be implicit in the physical operation of the hardware that supports the computations involved.

Apparently non-deterministic behavior can result from the complex interactions among purely deterministic processes (as in *deterministic chaos* (Pagels, 1989)).

Focus on Knowledge Engineering

Most of the work based on this approach has concentrated on the development of complex domain-specific, knowledge intensive programs (e.g., expert systems). The establishment of a knowledge-base in such systems is through a process of *knowledge engineering* (Waterman, 1985) - which is a complex task that entails making explicit the knowledge and working methods of the human expert, which are often implicit and difficult to characterize. That is, the human expert must have learned, and be able to make explicit all the knowledge needed, rather than having the system learn. Knowledge engineering is a difficult and tedious enterprise because experts are often unable to translate the mental processes that they use in solving problems in their domains of expertise into a sufficiently detailed, perfectly-defined, set of rules.

Learning by Modifying Symbol-Structures

The difficulties of knowledge engineering have in recent years stimulated research in machine learning. Learning in such systems involves additions and changes to the stored data structures and procedures. We will not review the current work in machine learning within the SP paradigm (see Carbonell, Michalski, & Mitchell, 1983; Dietterich & Michalski, 1983; Michalski & Kodratoff, 1990 for such reviews). We simply note that many learning techniques based on symbolic representations are extremely sensitive to noisy data (primarily because of the use of rigid and inflexible categorization and inference strategies). The past few years have seen some tentative moves toward using probablistic or fuzzy inference strategies to alleviate the problem of brittleness and noise-sensitivity in machine learning systems (Michalski & Kodratoff, 1990).

Another serious difficulty that plagues machine learning is the choice of inappropriate abstract knowledge representations (e.g., a feature vector representation of data available in a road map where an *iconic* (i.e., picture-like) representation would have been much more appropriate), in order to make use of the available learning algorithms. It is well-known that the knowledge representation chosen influences the ease and the *computational complexity* of learning: e.g., Boolean concepts expressed in k-DNF (*disjunctive normal form* with at most k literals per disjunct) are not feasibly learnable whereas the same concepts expressed in k-CNF (*conjunctive normal form* with at most k literals per conjunct) are (Valiant, 1984). Badly needed are a range of learning algorithms and architectures that exploit the strengths of available representations and/or transform from one representation to another as necessary. Algorithms and architectures for learning from iconic representations (e.g., 2-dimensional images) - e.g., *recognition cones* (Honavar & Uhr, 1987; 1988; 1989a) are examples of steps in this direction.

2. Connectionist or Neural Network Architectures for Intelligent Systems

Smolensky (1988) has characterized the connectionist networks (CN) approach (Rosenblatt, 1958; Feldman & Ballard, 1982; Rumelhart, Hinton, & McClelland, 1986) as the sub-symbolic approach to the design of intelligent systems. In connectionist models, perception and cognition emerge from the collective behavior of a large number of interacting simple processing units, often reminiscent (but often only vaguely) of neurons. CN are *universal* computing structures (McCulloch & Pitts, 1943; Pollack, 1987) in the sense that there exist such (sufficiently large) networks that can compute any *Turing-computable* function (Turing, 1936) or *recursive* function (Kleene, 1936) or *grammatically computable* function (Chomsky, 1959). Thus CN formulations obviously need not be restricted to computations over sub-symbolic structures.

Massively Parallel Computation

CN are characterized by massive fine-grained parallelism of computation. Thus, they offer a natural framework for the development of highly-parallel algorithms and computer architectures for intelligent systems. Such massively-parallel computations are necessary for real-time perception, as shown by the estimates based on the known speeds of neural transmission and the time it takes for human beings to perceive (Uhr, 1980; Feldman & Ballard, 1982). But clearly, this need for massively parallel computation does not necessitate a commitment to sub-symbolic representations: Virtually all parallel computers - e.g., the *Actor* system (Hewett, 1977), the connection machine (Hillis, 1985), and a variety of algorithmically-structured computer networks (Uhr, 1984) handle massively-parallel symbol processing as well as numeric computations.

Associative Access of Stored Knowledge

In contrast with the address-and-index-based memory access that typically characterizes today’s symbol processing systems, many CN exhibit associative recall (i.e., the ability to reconstruct or complete patterns based on partial cues), since they basically perform inexact or fuzzy template-match or correlation over the entire network in parallel. But associative recall can be (and often is, e.g., in hierarchical *cache* memories used in digital computers) implemented in SP systems.

Distributed Knowledge Representations and Fault/Noise Tolerance

Knowledge representations in CN can range from those that are highly localized (with one node representing each possible concept) to those that are highly distributed (with each concept represented by patterns of activity over many nodes in the network). To some extent, this is a matter of interpretation, since a locally encoded concept can in fact be a micro-feature that is part of a distributed encoding of a more complex concept. Distributed representations are not special in and of themselves and provide no new representational abilities that other representations used in computer science and artificial intelligence systems do not have (Hanson, 1990). Distributed representation and processing, together with a modest amount of redundancy, can yield substantial fault tolerance and damage resistance (e.g., the malfunctioning of a few units does not cripple the system) - properties essential for successfully coping with real world tasks. But such distributed encoding and redundancy can be used in symbol-processing systems as well: digital representation of symbols using binary code is clearly distributed, and fault tolerance can be accomplished by storing multiple copies of the data, using multiple processors to replicate the same computation in parallel, and using some form of a *majority rule* to pick the result (Von Neumann, 1958).

Statistical, Fuzzy, or Evidential Inference

CN computations have a large probabilistic, statistical, fuzzy, or evidential component that is typically handled numerically. This makes possible inferences that are much more robust in the presence of noisy or incomplete information than the typical logic-based inference techniques used in many SP models. Conceptually, the form of inference in CN is very much akin to that based on *fuzzy logic* (Zadeh, 1975) as demonstrated by (Oden, 1988). But fuzzy logic operates on symbolic representations, and probabilistic or fuzzy inference are widely used in symbol-processing systems as well - e.g., in the form of *heuristic evaluation functions* (Samuel, 1959) or probabilistic reasoning (Pearl, 1988).

Centrality of Learning

Learning plays a central role in CN, and typically involves small experience-induced changes to the links and nodes in the network. In fact, learning (especially inductive learning) is the preferred mode of knowledge acquisition in CN (contrast this with the carefully engineered complex representations typically built into most SP systems, which have no capability to learn). The micro-modularity of CN is particularly conducive to learning, since it enables the CN (given suitable adaptation and learning structures and processes), to change a little bit at a time, potentially in ways that carry it toward adequate behavior. Furthermore, the internal representations of the environment that are acquired by the CN through learning are *semantically grounded* (Harnad, 1990) via causal links with the external environment provided by the sensors and effectors - a property lacked by arbitrary symbol structures of the sort built by knowledge-engineering (although one that can easily be incorporated into many symbol-processing systems - e.g., Samuel’s checkers learner (Samuel, 1959), and Uhr-Vossler’s perceptual learning system (Uhr & Vossler, 1961). Learning is an integral part of the knowledge representation system in CN (Hanson, 1990). But there is no reason not to incorporate such an integration of knowledge representation and learning into massively parallel micro-modular SP systems as well.

Loosely Brain-Like Computing Structures

CN are (loosely) brain-like, with large numbers of simple (loosely) neuron-like processing elements. The complexity of neurons and neural circuits is far greater than that of the processing elements and networks that are used in today's CN (Shepherd, 1989). However, CN have the potential of being made far more brain-like, by making their processing elements more neuron-like and also by incorporating a variety of brain-like topological structures and processes (Honavar & Uhr, 1989a), microcircuits (Uhr, 1990b; Honavar & Uhr, 1990c), and functional modules (Uhr, 1990b). Similarly, an SP system can be made micro-modular and brain-like, using micro-circuits and primitive hardwired processes and processors (see below).

Distributed and Non-Rigid Control

CN control and coordination is typically expected to emerge out of the collective interaction of large numbers of simple units, and is highly distributed. The primitive nature of CN control structures is probably one of the most important reasons why CN cannot yet handle non-trivial high-level cognitive tasks (e.g., complex reasoning, natural language understanding, problem solving). But a range of powerful control structures and processes can, potentially, be incorporated into CN, e.g., to synchronize cooperating processes etc (see below) (Honavar & Uhr, 1990a).

Symbol Structures Encoded As Patterns

Today's CN have been generally lacking in their ability to effectively handle inferences on symbol structures, although attempts have been made to develop CN implementations of conventional symbol processing operations (Touretzky & Hinton, 1988). But there is no reason to restrict ourselves to the direct realization of conventional symbol processing in CN; entirely new processes that encode symbol-structures into patterns in the CN and processes that decode such patterns exploiting the massive fine-grained parallelism of CN can be used instead (see below).

3. Connectionist Networks (Narrowly) Defined & Described

A connectionist network (Rosenblatt, 1958; Feldman & Ballard, 1982; Rumelhart, Hinton, & McClelland, 1986) consist of a directed graph whose nodes compute relatively simple functions over the inputs they receive from other nodes in the network via their weighted input links (or from the external environment) and transmit the results to other nodes (or the environment) via their weighted output links. The network is typically specified in terms of the function computed by the individual nodes, the topology of the graph linking the nodes, and (if the network is adaptive), the algorithm used to modify the weights on the links. The network structures that implement the learning algorithm, the control structures that are necessary to perform a variety of functions (e.g., synchronizing the nodes, to switch on the learning processes, etc.) are generally left unspecified.

Each Node Computes a Simple Function

Each node computes a single scalar output value s that is some simple function of its (numeric-valued) inputs. A node n_j with K_j inputs has a K-dimensional *weight vector*

$$\mathbf{W}_j = [w_{j,1},\ w_{j,2},\ \cdots,\ w_{j,K_j}]$$

One weight is associated with each input link. Let the input to the node n_j be

$$\mathbf{I}_j = [i_1,\ i_2,\ \ldots,\ i_{K_j}]$$

Each node n_j has associated with it, a *node function* $\mathbf{T}_j$ which represents one or more computational steps involved in the calculation of the output of the node s_j. Some examples of such calculations are shown in figure 1-A and 1-B.

Some form of nonlinearity is necessary for making decisions necessary for categorization (Nilsson, 1965). The threshold function is a discrete decision-making device. It classifies all its input patterns into two sharply distinguished sets. The logistic function is a continuous version of the threshold function. Its graded response is attractive for a variety of purposes - including noise tolerance, contrast enhancement, and the existence of a derivative over the entire range of the function (a property required by some learning algorithms e.g., *backprop* (Rumelhart, Hinton, & Williams, 1986) for error back-propagation).

Often it is necessary to compare an input pattern with a stored *template*. Various versions of the *match* function are occasionally used to accomplish this task. A particular class of such functions is shown in figure 1-B. Such functions are attractive for a variety of reasons: The output is maximum when there is a perfect match between the stored weight vector and the input and monotonically decreases with increasing mismatch. The rate of decrease in response is governed by the functional form of f_j and can be tuned by varying α_j. The explicit measure of mismatch between $\mathbf{I}_j$ and $\mathbf{W}_j$ can be used to design a variety of learning algorithms (Honavar & Uhr, 1990f).

Three-layer feed-forward networks (i.e., 1 *hidden* layer between the input and output layers) of logistic nodes or gaussian match nodes have been shown to be universal function approximators (Hornick, Stinchcombe & White, 1988; Cybenko, 1989; Girosi & Poggio, 1989; Hartman, Keeler & Kowalski, 1990): There exists such a CN (constructed from a sufficient number of such nodes) which can approximate any smooth real-valued function to any desired accuracy.

Let $w_{j,0}$ be the *bias* associated with the node n_j. It is customary to treat the bias as if it is a weight associated with a constant input $i_0 = -1$.

$$u_j = \sum_{k=0}^{K_j} w_{j,k} \times i_k$$

The output of the node n_j is given by

$$s_j = f_j(u_j)$$

Some commonly used f_j define the linear, threshold and logistic node functions:

$$f_j(u_j) = u_j \qquad \textit{linear function}$$

$$f_j(u_j) = \begin{cases} 0 & u_j < 0 \\ 1 & \textit{otherwise} \end{cases} \qquad \textit{threshold function}$$

$$f_j(u_j) = \frac{1}{1 + e^{-u_j}} \qquad \textit{logistic function}$$

Figure 1-A: Linear, Threshold, and Logistic Functions

The output of the node n_j is given by

$$s_j = f_j(u_j)$$

Some typical choices for u_j are:

$$u_j = \frac{\left(\sum_{k=0}^{K_j} |(w_{j,k} - i_k)|^r \right)^{1/r}}{\alpha_j \times \sigma_j^{\,r}}$$

where r is a positive integer, $\sigma_j^{\,r}$ is a suitable normalization term, and α_j is a tunable parameter. It is straightforward to specialize this expression to yield *normalized euclidian distance* between the vectors $\mathbf{I}_j$ and $\mathbf{W}_j$ or the *Hamming distance* if the vectors are binary. Some typical choices for f_j are:

$$f_j(u_j) = e^{-u_j^{\,q}}$$

where q is a positive integer, or, alternatively,

$$f_j(u_j) = \frac{1}{1+u_j}$$

These functions belong to the family of *radial basis functions*.

Figure 1-B: Various Match Functions

Learning Modifies the Weights on the Links

Learning in CN typically involves the modification of weights on the links. Several schemes for modifying the weights are available (see Hinton, 1989, for a review). Some are based on correlations in the activation values associated with connected nodes - these are essentially variations on the mechanism proposed by Hebb (1949). Others use various estimates of error between the desired and actual network outputs. Error estimates may be based on extremely specific feedback e.g., the desired network output provided by the teacher for each input pattern which is used in error backpropagation (Rumelhart, Hinton & Williams, 1986; Werbos, 1974; Le Cun, 1987) and some of its faster variants (Fahlman, 1988). Alternatively, they may be obtained from not-so-specific feedback e.g., a reward/punishment signal used in reinforcement learning (Barto & Anandan, 1985), or they may be internally derived, e.g., based on an estimate of the output necessary from a node to produce the overall desired behavior from the network, e.g., in competitive learning (Grossberg, 1987).

Consider a CN node n_j which receives one of it's inputs i_k, from the node n_k. The feedback to the node is t_j, an estimate of the desired output of the node n_k. The current output of the node is s_j. A typical rule for modifying the weight $w_{j,k}$ takes the form:

$$\Delta w_{j,k} = \eta \times \lambda(i_{k,} w_{j,k}, s_{j,} t_j)$$

where η is a constant of proportionality called the learning rate, and λ is the function used to compute the weight modification as a function of the error between the feedback and current output. Some of the most popular weight modification schemes of this form are the Perceptron algorithm for a network with one layer of modifiable links (Rosenblatt, 1958) and its generalization to a network with multiple layers of modifiable links (Werbos, 1974; Le Cun, 1987; Rumelhart, Hinton, & Williams, 1986).

4. CN Broadly Defined: Steps Toward Generalized Connectionist Networks (GCN)

The following rather general definition (adapted from Uhr, 1990a; Honavar, 1990) makes clear that a large variety of CN architectures (other than the ones commonly used today) can be constructed; it also makes precise comparisons possible.

A GCN is a graph (of linked nodes) with a particular topology Γ. The total graph can be partitioned into three functional sub-graphs - $\Gamma_{\mathbf{B}}$ (the *behave/act* sub-graph), Γ_{Λ} (the *evolve/learn* sub-graph), and $\Gamma_{\mathbf{K}}$ (the *coordinate/control* sub-graph). The motivation for distinguishing among these three functions will become clear later. The nodes in a GCN compute one or more different type/s of functions: **B** (*behave/act,*); Λ (*evolve/learn,*); and **K** (*coordinate/control*).

$$\text{GCN} = \{\Gamma, \mathbf{B}, \Lambda, \mathbf{K}\}$$

Today's CN are specified (typically only partially) as follows:

* The overall graph structure, $\Gamma_{\mathbf{B}}$ of the sub-net that behaves (today much of the total graph, including the entire sub-graphs needed to handle learning and control - is usually left unspecified). A complete description of the entire graph Γ is necessary to completely specify CN realizations of such architectures.
* The node function $\mathbf{T}_k$ and the weight vector $\mathbf{W}_j$ associated with each node n_j that define the functions computed during the behave cycle. Typically, the same node function is used with each of the nodes in the network.
* The single function λ that is used to compute the changes to the weight vectors during the learning cycle - but not the actual sub-net structures that are needed to actually compute and make these changes.

A network's behavior is typically initiated by sending values to the input sensing nodes. Its resulting performance is the set of values sent by its output acting nodes. The net's behavior is completely determined by its topology, the values originally associated with its links, and the functions computed by its nodes and links - plus any modifications made to these by learning.

Several potentially powerful extensions to today's CN - including more powerful structures and processes for behaving, control, and learning - suggest themselves, leading up to systems that can be characterized as *generalized connectionist networks* (GCN) or *generalized neuromorphic systems* (GNS) (Honavar, 1990).

4.1. Behave-Act Functions In GCN

In today's CN, the processes that are used during the behave cycle **B**, culminating in output performance, usually add, or in other ways integrate, their inputs, then threshold or transform the results with a logistic or other squashing function.

There is no compelling reason to limit ourselves to the simple functions that the individual nodes used in today's CN compute. That would be tantamount to insisting that all computers be built using only NOR gates, or all programs be expressions built using only NOT-AND, or a minimal set of Turing machine instructions. A variety of node functions (e.g., match, difference, etc) may be used wherever appropriate, along with learning functions that are tailored for use with such functions. More generally, a large variety of analog, digital, or hybrid analog-digital microcircuits suggested by the brain or analog and digital computer designs (Uhr, 1990b; Shepherd, 1989, 1990; Honavar & Uhr, 1990b) can be used wherever appropriate.

4.2. Evolve-Learn Functions In GCN

Learning in today's CN is almost always handled by processes that change the weight-vector $\mathbf{W}_k$ associated with each node n_k so as to reduce the error between the desired output from the node and its actual output. There is no compelling reason to restrict ourselves to this set of learning processes. There are a variety of potentially more powerful alternatives:

[1] Learning that modifies the node functions $\mathbf{T}_k$ associated with the processing elements in the GCN: Learning might alter the steepness of the sigmoid function (Tawel, 1989); equivalently, if a threshold element is used, learning might involve systematically ranging through a number of alternative node functions (e.g., from logical OR to logical AND) by changing that threshold.

[2] Learning that modifies the *weight matrices* or *local templates* associated with each processing element in the GCN: This might involve the use of one of the

several weight-modification strategies currently used in CN (e.g., Widrow & Hoff, 1960) or other forms of weight modification that may be appropriate for the corresponding node functions used (e.g., if a node that matches its input with a stored weight matrix and produces a measure of match - e.g., a gaussian match node - is used, a suitable weight modification strategy might be to blur the weight matrix by adding a small fraction of a sufficiently well-matched input).

[3] Learning that modifies the connectivity of the network Γ (i.e., by the addition (and when indicated, or necessary to make space, deletion) of links and nodes) viz., *generative learning* (Honavar & Uhr, 1988; 1989a; 1989b 1990f; Honavar, 1990) and related approaches (Ash, 1989; Diederich, 1989; Fahlman & Lebiere, 1990; Hanson, 1990b; Gallant, 1990; Nadal, 1989; Rujan & Marchand, 1989). See (Honavar & Uhr, 1990f; Honavar, 1990) for a discussion of alternate generative learning strategies for GCN. Given a suitable set of generative learning mechanisms, a GCN can adaptively search for and assume whatever connectivity is appropriate for particular tasks (possibly under certain predefined topological constraints - e.g., those imposed by locally-linked multi-layer converging-diverging networks (Honavar & Uhr, 1988; 1989a).

[4] Learning that modifies the control structures and processes **K** that are used in GCN: In particular, learning might alter the controls that regulate particular types of learning (e.g., parameters such as the learning rate used in weight modification); or the initiation and termination of *plasticity*, or the manner in which different sorts of learning (e.g., weight modification and generative learning) are coordinated.

[5] Learning that modifies the learning structures and processes (Λ) themselves, e.g., changes in weight modification strategies, changes in node generation strategies, and so on.

4.3. Coordinate-Control Functions in GCN

Several important aspects of control are usually (implicitly) assumed in today's CNs. All nodes fire in parallel, at discrete instants in time with the behave and learning cycles alternating. Gating networks control and coordinate the interaction between different functional modules in CN implementations of production systems (Touretzky & Hinton, 1988). A variety of powerful distributed coordination and control structures and processes suggested by the examination of a wide range of natural and artificial systems can be incorporated into CN (see Honavar & Uhr, 1990a for details). These include structures suggested by the brain (e.g., oscillators, diffuse networks of neuromodulators; attentional networks; feedback regulation); the developmental processes that guide fetal development (e.g., networks of markers; structures that encode information necessary to generate other network structures); evolutionary processes that operate on large populations of organisms (e.g., competitive interactions); the processes of communication and coordination among living cells; the great variety of partially competing partially cooperating controls found in small groups, organizations, and societies; and the control structures used in computers, computer programs, and computer networks. Interacting networks of such control structures operating over a range of spatial and temporal scales can be used to serve a number of functions - e.g., regulation of the interaction among a variety of learning processes (Honavar & Uhr, 1988; 1989a; 1990b; 1990f; Honavar, 1990).

4.4. Structures Used to Link Individual Nodes into Overall Topologies

Any graph topology might be chosen, but only a few have actually been used to date: one-layer nets where each node that receives inputs from the external environment links directly to all, or to a sub-set (typically random), of nodes that output to the external environment (e.g., perceptrons (Rosenblatt, 1958)); multi-layer feed-forward nets where each layer's nodes link into all, or some, of the next layer's nodes (e.g., the error back-propagation architecture (Rumelhart, Hinton, & Williams, 1986)); one or two-layer nets with cycles (e.g., Pineda, 1987; Pearlmutter, 1988). Other topologies may be used, including near-neighbor converging tree-like connectivity from one layer to the next (Honavar & Uhr, 1987; 1988; 1989a; 1989b; Le Cun, 1989; Le Cun et al., 1990), structures that expedite the particular functions computed, e.g., Hough transforms (Ballard, 1984), and structures (e.g., arrays, trees, pipelines) that are built in because they appear to be appropriate for the particular type of problem being attacked. But a large variety of topological structures that link behave, learn, and control subnetworks remain to be explored (see Uhr, 1984, Honavar & Uhr, 1990a, 1990d for some examples of such structures).

4.5. Symbol Processing in GCN

A *symbol* is an identifiable entity that points to, or signifies, or encodes some structure of information. Each symbol is represented by an encoding (e.g., the binary encoding used in digital computers). Symbol processing in GCN can be handled simply by using *code books* which are implicitly handled by encoders (at the source) and decoders (at the destination) of a channel of symbol transmission (Uhr, 1990c). Thus, symbols are encoded into *patterns of activity* in GCN (much like the patterns of bits transmitted on wires in a conventional computer). For related work on inferences using symbol-structures, see (Shastri & Ajjanagadde, 1989; Dolan & Smolensky, 1989).

The major difference between this way of handling symbols in GCN and the way symbols are typically handled in other symbol processing systems (e.g., the serial computer) is that the code book in the latter is stored in a central memory and accessed in a serial fashion by the central processor. In a parallel network, there is no need for a centralized repository for the code book. Each node needs to have only the code book entries for symbols for which those nodes directly connected to it have entries. Thus, the code book is distributed among many nodes and the encoder-decoder microcircuits associated with them. Furthermore, each GCN node or microcircuit can develop its own code book as a function of learning.

Encoding and decoding of symbols can be made efficient by the use of parallel-hierarchical encoder and decoder microcircuits. Redundancy and error-correction mechanisms can be introduced into the encoding and decoding processes to make them robust, noise-tolerant, and capable of handling variations.

The structures and processes outlined above, together with a range of control structures (e.g., for synchronization, gating, etc) enable GCN to perform the entire range of symbol processing operations while supporting what many (e.g., Fodor & Pylyshin, 1988) consider to be the essential properties of cognition: *productivity*, *systematicity*, and *compositionality* of *mental representations*. Each of these properties essentially amounts to a recognition of constituent semantic structure (e.g., spatial relationships such as *left of* that are implicit in a visual image) of representations. Hierarchies of iconic or picture-like representations of the sort used in recognition cone networks (Honavar & Uhr, 1988; 1989a) enable the recognition of such semantic structure in GCN.

Attentional control processes in GCN can assist in the formation of transient node assemblies that represent compositions of multiple symbols or symbol-structures in GCN through the activation of rapidly modifiable links in a dynamic fashion (Honavar & Uhr, 1990a). Transient node-assemblies also offer a solution to the *dynamic variable binding problem* (i.e., the association of different *values* with the same symbol over time) in GCN (Shastri & Ajjanagadde, 1989; Honavar & Uhr, 1990a).

4.6. Hierarchies of Structures in Space and Time

GCN support hierarchies of topological structures which organize the network in space (in ways that reflect the physics of the environment, e.g., local interactions in visual perception - see Honavar & Uhr, 1989a for details) and hierarchies of temporal processes instantiated by behave/act, evolve/learn, and coordinate/control that operate and interact over a range of time-scales (e.g., processes that regulate the interaction among different learning processes - see Honavar, 1990; Honavar & Uhr, 1990f for details). Such hierarchies yield complex non-linear spatio-temporal dynamics which may be essential to realizing the full range of intelligent behavior. They also offer a means of reducing the space-time complexity of the constituent computations (Grossberg, 1980).

4.7. GCN and the Modeling of Neural Systems

The brain clearly uses mixtures of analog/subsymbolic and digital/symbolic processes: the flow of neurotransmitter molecules and of photons into receptors is quantal (digital); the depolarization and hyperpolarization of neuron membranes is analog; the transmission of pulses is digital; global interactions mediated by neurotransmitters and slow waves appear to be both analog and digital. Analog processes are most economically and naturally captured by continuous numeric processing of the sort typically used in CN. But interactions at the level of individual molecules and receptors are best modeled by the symbol processing mechanisms (e.g., distributed code-books) supported by GCN (See Cooper, 1990 for examples of such models of some neural subsystems e.g., calcium oscillator, burst generator, etc.).

Brains exhibit a highly modular (Mountcastle, 1978), often hierarchical architecture (Zeki & Shipp, 1988) that can be modeled by the topological structures (layers, clusters of nodes, trees) of GCN. Perception, learning and control in the brain appear to utilize processes over multiple spatial and temporal scales (Grossberg, 1980). Learning also appears to change not only the strength of interaction among neurons (which is loosely analogous to learning in CN) but also dynamic changes in network connectivity (Greenough, 1986; Greenough, Black, & Wallace, 1987; Greenough & Bailey, 1988) which can be modeled by generative learning (see Honavar, 1990; Honavar & Uhr, 1990f for details). GCN provide a framework for modeling such neural systems that is far richer than the one provided by either symbol processing systems or by today's CN.

5. Toward a Synthesis of Symbol Processing and Sub-Symbolic Paradigms

The discussion of the symbol processing approach and the sub-symbolic (CN) approach to the design of intelligent systems in previous sections suggest that there is really no fundamental incompatibility between SP and CN approaches to the modeling and design of intelligent systems. Generalized connectionist networks (GCN) introduced above incorporate a number of powerful extensions to CN including a variety of structures and processes that determine behavior, learning, coordination/control, and the network architecture. These extensions go a long way toward alleviating some of the limitations of today's sub-symbolic (CN) and symbol processing architectures for intelligent systems while retaining the strengths of both approaches.

It has been argued that CN *just provide an implementation theory* for cognitive models formulated within the symbol processing paradigm (Fodor & Pylyshyn, 1988). But the implementation medium offers powerful constraints on the designs that can be effectively realized in practice. Cognitive models should simultaneously satisfy the design constraints over multiple levels (ranging from the psychological to the neurochemical). Efficient, computationally tractable (in terms of memory and processing time requirements), implementations of processes that support perception, learning, cognition, and action offer the key to powerful, practical, and robust designs for intelligent systems. GCN provide a framework within which we can explore the design/performance tradeoffs among: parallel versus serial processing; localized versus distributed processing (in space as well as over time), representation, and control; symbolic versus sub-symbolic structures and processes; and implicit versus explicit rule-based behavior. An understanding of such tradeoffs is crucial in our quest for realistic models of intelligent systems.

References

Ash, T. (1989). Dynamic node creation in backpropagation networks. *Connection Science - Journal of Neural Computing, Artificial Intelligence and Cognitive Research* 1 365-375.

Ballard, D. H. (1984). Parameter nets. *Artificial Intelligence* 22 235-267.

Barto, A. G. & Anandan, P. (1985). Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics* 15 360-375.

Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.) *Machine Learning - An Artificial Intelligence Approach.* Palo Alto, CA: Tioga.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control* 2 137-167.

Cooper, E. D. (1990). An object-oriented interpreter for symbol train processing. (Preprint).

Cybenko, G. (1989). Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals, and Systems* 2 (In press).

Diederich, J. (1988). *Knowledge-intensive recruitment learning.* Technical report TR-88-010. Berkeley, CA: International Computer Science Institute.

Dietterich, T. G., & Michalski, R. S. (1983). A comparative review of selected methods for learning from examples. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.) *Machine Learning - An Artificial Intelligence Approach.* Palo Alto, CA: Tioga.

Dolan, C. P., & Smolensky, P. (1989). Tensor product production system: a modular architecture and representation. *Connection Science - Journal of Neural Computing, Artificial Intelligence and Cognitive Research* 1 53-58.

Fahlman, S. E. (1988). Faster-learning variations on back-propagation. In G. E. Hinton, T. J. Sejnowski, & D. S. Touretzky (Eds.) *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems (Vol. 2).* D. S. Touretzky (Ed.) San Mateo, CA: Morgan Kaufmann.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science* 6, 205-264.

Fodor, J. (1976). The Language of Thought. Boston, MA: Harvard Univ. Press.

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

Gallant, S. I. (1990). Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks.* 1 179-.

Girosi, F. & Poggio, T. (1989). *Networks and the Best Approximation Property.* A.I. Memo 1164. Cambridge, MA: MIT AI laboratory.

Greenough, W. T. (1986). Thoughts on the basis of experience-sensitive synaptic plasticity. In *Developmental Psychobiology* New York: Academic Press.

Greenough, W. T., Black, J. E., & Wallace, C. S. (1987). Experience and brain development. *Child Development* 58 539-559.

Greenough, W. T., & Bailey, C. H. (1988). The anatomy of a memory: convergence of results across a diversity of tests *Trends in Neuroscience*, 11, 142-147.

Grossberg, S. (1980). How does the brain build a cognitive code? *Psychological Review* 1 1-51.

Grossberg, S. (1987). Competitive learning: from interactive activation to adaptive resonance. *Cognitive Science* 11 22-63.

Hanson, S. J. (1990). What connectionist models learn: learning and representation in connectionist networks. *Behavioral and Brain Sciences* (In press).

Harnad, S. (1990). The symbol grounding problem. *Physica* (In Press).

Hartman, E., Keeler, K., Kowalski, J. M. (1990). Layered neural networks with gaussian hidden units are universal approximators. *Neural Computation* 2 210-215.

Hebb, D. O. (1949). *The Organization of Behavior.* New York, NY: Wiley.

Hewett, C. (1977). Viewing control structures as patterns of passing messages. *Artificial Intelligence* 8 232-364.

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

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence* 40 185-234.

Honavar, V., & Uhr, L. (1987). *Recognition Cones: A Neuronal Architecture for Perception and Learning.* Technical report 717. Madison, WI: Computer Sciences Dept.

Honavar, V., & Uhr, L. (1988). A network of neuron-like units that learns to perceive by generation as well as reweighting of its links. In G. E. Hinton, T. J. Sejnowski, & D. S. Touretzky (Eds.) *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

Honavar, V., & Uhr, L. (1989a). Brain-Structured connectionist networks that perceive and learn. *Connection Science - Journal of Neural Computing, Artificial Intelligence and Cognitive Research* 1 139-160.

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.

Honavar, V. (1989). *Perceptual development and learning: From behavioral, neurophysiological, and morphological evidence to computational models.* Technical report 818. Madison, WI: University of Wisconsin, Computer Sciences Dept.

Honavar, V., & Uhr, L. (1990a). Coordination and control structures and processes: possibilities for connectionist networks. *Journal of Experimental and Theoretical Artificial Intelligence* (In Press).

Honavar, V., & Uhr, L. (1990b). Efficient learning using multi-resolution representations of spatial, temporal, and spatio-temporal patterns. In *Proceedings of the 1990 SMC Conference* (To appear).

Honavar, V., & Uhr, L. (1990c). Analog and hybrid analog-digital microcircuits for

connectionist networks. (paper in preparation).
Honavar, V., & Uhr, L. (1990d). On the role of reciprocal links in neural networks. (Paper in preparation).
Honavar, V. (1990). Generative Learning Structures for Generalized Connectionist Networks. Ph.D Thesis, University of Wisconsin, Madison.
Hornik, K., Stinchcombe, M. & White, H. (1988). Multilayer feedforward networks are universal approximators. *Neural Networks 1.*
Kleene, S. C. (1936). General recursive functions of natural numbers. *Mathematische Annalen* 112 727-742.
Kowalski, R. (1979). *Logic for Problem Solving.* Amsterdam: North Holland.
Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence* 33.
Le Cun, Y. (1987). *Modeles Connexionnistes le l'Apprentissage.* Ph.D Thesis, Universite Pierre et Marie Curie, Paris.
Le Cun, Y. (1989). Generalization and Network Design Strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.) *Connectionism in Perspective.* Zurich, Switzerland: Elsevier.
Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In D. S. Touretzky (Ed.) *Neural Information Processing Systems (Vol. 2).* San Mateo, CA: Morgan Kaufmann.
McCulloch, W. S., & Pitts, W. H. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics,* 5, 115-133.
Mountcastle, V. (1978). An organizing principle for cerebral function: the unit module and the distributed system. In G. M. Edelman & V. B. Mountcastle (Eds.) *The Mindful Brain.* Cambridge, MA: MIT Press.
Nadal, J. P. (1989). Study of a growth algorithm for a feedforward network. *International journal of neural systems.* 1 55-.
Newell, A., & Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, NJ: Prentice-Hall.
Newell, A. (1980). Physical symbol systems. *Cognitive Science* 4 135-183.
Nilsson, N. J. (1965). *The Mathematical Foundations of Learning Machines.* New York: McGraw-Hill.
Oden, G. C. (1988). Fuzzyprop: a symbolic superstrate for connectionist models. In *Proceedings of the 2nd IEEE Conference on Neural Networks.* San Diego, CA.
Pagels, H. R. (1989). *The Dreams of Reason - The Computer and the Rise of the Sciences of Complexity.* New York: Bantam books.
Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems.* San Mateo, CA: Morgan Kaufmann.
Pearlmutter, B. A. (1988). Learning state space trajectories in recurrent neural networks. In G. E. Hinton, T. J. Sejnowski, & D. S. Touretzky (Eds.) *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.
Pineda, F. (1987). Generalization of back-propagation to recurrent neural networks.

Physical Review Letters 19(59) 2229-2232.
Pollack, J. (1987). Technical report MCCS-87-100. Las Cruces, NM: New Mexico State University Computing Research Laboratories.
Rosenblatt, F. (1958). The Perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* 65 386-408.
Rujan, P., & Marchand, M. (1989). Learning by activating neurons: a new approach to learning in neural networks. *Complex Systems* 3 229-.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation, In *Parallel Distributed Processing - Explorations into the Microstructure of Cognition (Vol. 1: Foundations).* Cambridge, MA: MIT Press.
Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. *Parallel Distributed Processing - Explorations into the Microstructure of Cognition (Vol. 1: Foundations).* Cambridge, MA: MIT Press.
Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* 3 211-229.
Schneider, W. (1987). Connectionism: is it a paradigm shift for psychology? *Behavior Research Methods, Instruments, & Computers* 19 73-83.
Shastri, L. & Ajjanagadde, V. (1989). *A Connectionist System for Rule Based Reasoning with Multi-Place Predicates and Variables.* Technical report MS-CIS-8906. Philadelphia, PA: University of Pennsylvania Computer and Information Science Department.
Shepherd, G. M. (1989). The significance of real neuron architectures for neural network simulations. In E. Schwartz (Ed.) *Computational Neuroscience.* Cambridge, MA: MIT Press.
Shepherd, G. M. (Ed.) (1990). *The Synaptic Organization of the Brain.* New York: Oxford University Press.
Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11, 1-23.
Touretzky, D. S. & Hinton, G. E. (1988). A distributed connectionist production system. *Cognitive Science* 12 423-466.
Turing, A. M. (1936). On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society.*
Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM* 1134-1142.
Uhr, L. & Vossler, C. (1961). A pattern recognition program that generates, evaluates, and adjusts its own operators. In *Proceedings of Western Joint Computer Conference* 19 555-569.
Uhr, L. (1979). Parallel-serial production systems with many working memories. In *Proceedings of the International Joint Conference on Artificial Intelligence.*
Uhr, L. (1984). *Algorithm-Structured Computer Arrays and Networks.* New York: Academic Press.
Uhr, L. (1990a). Connectionist networks defined generally, to show where power can be increased. *Connection Science - Journal of Neural Computing, Artificial*

Intelligence and Cognitive Research (In press).
Uhr, L. (1990b). Specifying useful sub-net and micro-circuit structures for connectionist networks. (Paper in preparation).
Uhr, L. (1990c). Handling lists and structures of lists in connectionist networks. (Paper in preparation).
Von Neumann, J. (1958). *The Computer and the Brain.* New Haven, CT: Yale University Press.
Waterman, D. A. (1985). *A Guide to Expert Systems.* Reading, MA: Addison-Wesley.
Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences.* Ph.D Thesis, Harvard University.
Zadeh, L. A. (1975). Fuzzy logic and approximate reasoning. *Synthese* 30 407-28.
Zeki, S. & Shipp, S. (1988). The functional logic of cortical connections. *Nature* 335 311-317.