Preprint of Paper 'Connectionist Inference Models' to appear in *Neural Networks*

Antony Browne†, and Ron Sun ‡

† *School of Computing, Information Systems and Mathematics, London Guildhall University, UK*
‡ *CECS Department, University of Missouri-Columbia, USA*
Reprint requests should be sent to: Antony Browne, School of Computing, Information Systems and Mathematics, London Guildhall University, London EC3N 1JY, UK, Tel: (+44) 0207 320 1307 Fax: 0207 320 1717 E-mail: abrowne@lgu.ac.uk
Running title: Connectionist Inference

Abstract

The performance of symbolic inference tasks has long been a challenge to connectionists.In this paper, we present an extended survey of this area. Existing connectionist inference systems are reviewed, with particular reference to how they perform variable binding and rule-based reasoning and whether they involve distributed or localist representations. The benefits and disadvantages of different representations and systems are outlined, and conclusions drawn regarding the capabilities of connectionist inference systems when compared with symbolic inference systems or when used for cognitive modelling.

Key words: Symbolic inference, resolution, variable binding, localist representations, distributed representations.

1 Introduction

An essential question in connectionist inference systems research is:'Why should we bother attempting to perform symbolic inference using connectionist networks?'. It could be argued that symbolic inference should be left to classical symbolic Artificial Intelligence (AI) systems (such as theorem provers or production systems), whilst the application of connectionist systems should be restricted to tasks that they perform best (such as learning function mappings of noisy or incomplete data). Another approach could be to just 'bolt' together a symbolic inference system and a connectionist network into a hybrid system which although being a non-uniform solution to a problem, solved that problem nonetheless. However, there are many important reasons for developing uniform connectionist solutions for performing symbolic inference, including:

- The human capacities for algorithmic reasoning and abstraction suggest the importance of symbol-based processing (Buchheit, 1999). Indeed, recent experiments (Marcus et al., 1999) suggest that even human infants may perform rule-based reasoning.
- To demonstrate the power of connectionism as an alternative paradigm for AI, it must be demonstrated that connectionist models are fully capable of performing symbolic reasoning. Therefore, attempts must be made to develop these capabilities within entirely connectionist systems.
- Science seeks to unify theories and explanations, because in such a process a deeper understanding of the nature of things can be achieved. In AI, a great variety of models, techniques, and paradigms have been developed, and it may be fruitful to unify these different approaches.
- Serial symbolic systems are much too slow to account for the speed and style of reflexive and direct reasoning processes. There is no adequate parallel symbolic system available for handling these types of reasoning. Massively parallel connectionist models are more suitable for carrying out such reasoning.
- In many hybrid systems (Sun, 1995c; Chan and Franklin, 1998; Ghalwash, 1998), while the connectionist component enjoys being fault tolerant and generalizable, the symbolic component is brittle and rigid and becomes the 'Achilles' heel' of the system.

To construct solutions for the points outlined above, many different connectionist systems have been developed. In the next section an attempt is made to construct broad taxonomies of these systems.

2 Classification of connectionist inference systems

It is impossible to precisely classify the wide variety of existing connectionist inference systems into specific groupings, but a rough attempt can be made depending on whether or not the systems implement variable binding and whether they employ localist or distributed representations. In the following discussion the syntax of the Prolog programming language has been adopted, where variables are in upper case (such as X) and constants or functors are in lower case (such as f).

2.1 The variable binding problem

Some connectionist rule-based reasoning systems avoid implementing variable binding (see section 3). However, many researchers have argued that the ability to perform variable binding is essential if connectionist systems are ever to perform complex reasoning tasks (but see (Barnden and Srinivas, 1996)). There are several main points to this argument:

- Variable binding is essential if dynamic structures are to be created in connectionist systems (Feldman and Ballard, 1992; Sun, 1992). As an example, consider a system that can reason with Prolog structures representing vehicles such as '*blue cars*' and '*red motorcycles*', the system must have a method of binding the color *blue* with the vehicle *car* and *red* with the vehicle *motorcycle*, as in:

$$colour(X, blue), vehicle(X, car) \ and$$
$$colour(X, red), vehicle(X, motorcycle)$$

- Variable binding is needed to achieve greater computational power in connectionist systems (Sun, 1992), as without variable binding there must be many specific rules rather than one general rule. For example, with variable binding there can be one general rule such as:

$$dog(X) \longrightarrow \ hairy(X), barks(X)$$

· representing that X is a dog if X is hairy and X barks. Without variable binding we must have a specific rule for each eventuality that the system may encounter, such as:

$$dog(fido) \longrightarrow hairy(fido), barks(fido) \ and$$
$$dog(rex) \longrightarrow hairy(rex), barks(rex) \ and...etc$$

- Variable binding is essential in modelling human cognition. For example, it has been argued that it is necessary for modelling the human language faculty (Pinker and Prince, 1988) as it imposes constraints on rule matching needed when modelling certain aspects of past-tense learning.

One problem that arises is how to handle variables that are used in rule-based reasoning, that is, variables as arguments to a predicate in predicate logic or as parameters in production rules, in a connectionist fashion. Difficulties exist in:

(1) How to represent the values of a variable, which can be changed at any time during reasoning processes due to the application of a rule.
(2) How to transfer such values from a known (stored) fact to a rule, or from conditions of a rule to the conclusion of a rule, or from a rule to a fact to be stored (all these types of transfers are necessary to ensure that correct inference is performed when variables are used).

There are many solutions to the variable binding problem (see (Browne and Sun, 1999) for a review) and some are discussed in sections 4 and 5.

2.2 Localist and distributed representations

Those models using localist representational schemes (often known as structured or spreading activation networks) can be broadly divided into two types:

(1) Those using a fully localist representation, which is characterized by representing each concept (in a particular task domain) with a separate node in a network, implying *one node for one concept* (i.e., there is an one-to-one mapping between nodes and concepts).
(2) Those using a distributed localist (or modularly distributed) representation, which uses a set of nodes for one concept, each of which does the same processing. This implies *a set of nodes for one concept* (i.e., there is an one-to-one mapping between sets of nodes and concepts).

In models using these schemes, every different item to be represented is mapped onto its own distinctive unit or units in the network. These models tend to represent knowledge in structures similar to the semantic network of classical AI, in which concepts are represented by individual neurons or units and relations between concepts are encoded by weighted connections between those units. The activation level on each unit generally represents the amount of evidence available for its concept within a given context. These models address important questions, such as how connectionist models can deal with discrete, symbolic, and step-by-step reasoning as well as performing the continuous mapping, associative retrieval, and constraint satisfaction characteristics commonly associated with connectionist models. These models are capable of:

(1) Carrying out symbolic rule-based reasoning, including handling the variable binding problem.
(2) Reasoning in a computationally efficient manner (when implemented on parallel hardware).
(3) Replacing hybrid models in which a symbolic component is coupled with a connectionist component.
(4) Showing promise in better addressing some symbolic processing tasks (Sun, 1995b).

Despite the accomplishments of localist connectionist models, they have a number of (alleged) shortcomings, including:

- They are incapable of generalization and therefore suffer from the same rigidity problem as traditional symbolic AI systems (Sun, 1995b), whereas in most models using distributed representations, the representations are a result of the organization of statistical input which provides a natural means to capture semantic similarity (Smolensky, 1995).
- It is difficult to apply connectionist learning algorithms to develop representation automatically (although some learning algorithms do exist, such as recruitment learning (Diederich, 1988; Diederich, 1991)).
- Localist models are not robust to noise or damage, unlike those using distributed representations (Churchland and Sejnowski, 1992).
- Localist models entail a high representational complexity as every possible concept need to be represented explicitly as an individual node, leading to combinatorial explosion. Models using distributed representation make more efficient use of representational resources.
- Available evidence seems to indicate that the representations used in biological neural systems are distributed.

Although some of these problems may not be entirely true (for example, the problems with noise, generalization and fault tolerance may not be as clear cut as described above, for a further discussion see (Page, 2000)), they are still indicative of the need to develop distributed alternatives to localist models. Localist connectionist models which perform rule-based reasoning are discussed in sections 3 and 4.

Connectionists using distributed representations maintain that the correct level at which to model intelligent systems (including the human mind) lies *below* the level of the symbol (see the *subsymbolic hypothesis* (Smolensky, 1990)) and reject the Physical Symbol System Hypothesis of symbolic AI (Newell, 1980; Newell, 1986). There have been many attempts to define distribution in connectionist representations, such as *microfeatures* (Hinton et al., 1986; Hinton, 1990), and *coarse coding* (Rosenfeld and Touretzky, 1988). Perhaps the most formal notion of distribution has been given by van Gelder (van Gelder, 1991) who described distributed representations with respect to their *extendedness* and *superposition*. For a representation to be extended the things being represented must be represented over many units, or more generally over some relatively extended proportion of the available resources in the system (such as the units in a neural network). A representation can be said to be superpositional if it represents many items using the same resources. Representations in a standard feedforward network can be both extended and superposed (Sharkey, 1992), as the representation of each input may be distributed over many hidden units, and each hidden unit may be representing several inputs. An in-depth discussion of the various definitions of distribution in connectionist representations would make this paper too lengthy, for a more extensive elaboration of these definitions readers are referred to the previous references or (Browne and Sun, 1999). One could argue that, as both neural networks and Von-Neumann type physical symbol systems are both universal Turing machines (Franklin and Garzon, 1990), at some level of abstraction there is no distinction between them. However, the key issue is what constitutes the primitive representation used by these different systems. This is a pertinent issue, both in modelling human cognition and in building intelligent systems. Models of rule-based reasoning using distributed representations are described in Section 5.

2.3 Other classification schemes, for hybrid systems

In another approach to system classification (Sun and Bookman, 1994) architectures can be divided up into two broad categories: single-module architectures and multi-module architectures. In both, it is easier to incorporate prior knowledge into models using localist representations since their structures can be made to directly correspond to that of symbolic knowledge. In multi-module systems, there can be different combinations of different types of constituent modules. For example, a system can be a combination of localist modules and distributed modules.

Other classification schemes have been proposed. Medskers (1994) classification scheme is based on the degree of coupling between neural and symbolic components (where coupling is the degree of communication activity between modules in the hybrid system) and makes no attempt to describe a hierarchy of modules. The classification scheme proposed by Hilario (1997 and 2000) classifies systems into unified and hybrid approaches where the unified approach attempts to endow neural networks with symbolic capabilities so that no distinct symbolic component is required. The hybrid approach integrates separate symbolic and neural elements using four distinct integration techniques based upon the flow of data between the modules and has two degrees of coupling (loosely and tightly coupled).

Another scheme (McGarry et al., 1999a) proposes that a classification scheme can be made with three groups:

(1) *Unified hybrid* systems are those that have all processing activities implemented by neural network elements.
(2) *Transformational hybrid* systems can transform a symbolic representation into a neural one and vice versa.
(3) *Modular hybrid* systems are comprised of several neural network and symbolic modules which can have different degrees of coupling and integration. The hierachy of module configuration can allow sequential flow of information between modules (i.e. one process must be completed before being passed on) or parallel flow. In this scheme module coupling can take three forms: *passive coupling*, when the symbolic and neural components communicate only through a shared data file, so after one component completes its task the results are placed in a file to be read by the other component; *active coupling*, which involves having memory/data structures shared between the neural and symbolic modules and communication may be bi-directional allowing feedback to occur between the different modules; *interleaved coupling* involves the neural and symbolic components interacting at a very fine-grained level (such as at the level of function calls) and an external observer would be unable to differentiate between them.

There is currently a large amount of work being performed on hybrid systems, and a comprehensive overview is too large to encompass here. Interested readers are referred to Sun and Alexandre (1997) and Wermter and Sun (2000).

3 Connectionist inference systems without variable binding.

The idea of implementing propositional logic in connectionist networks has been explored early on in the history of connectionism. For example, McCullough and Pitts (1943) studied the encoding of simple logical operations in neural networks, and Collins and Loftus (1975) explored the idea of spreading activation within networks. Without variable binding models can consist of extremely simple production systems containing no parameters or propositional logic without arguments to predicates. Suppose there is a set of rules for determining the classification of certain objects:

$$\begin{aligned} table &\longrightarrow phy\_obj \\ chair &\longrightarrow phy\_obj \end{aligned}$$

$phy\_obj \longrightarrow thing$

The set of rules together form a particular structure determined by the connection imposed by the rules between various concepts (from conditions to conclusions). The structure stays the same irrespective of any particular order in which rules are presented, and of the semantics of the rules and the concepts used in the rules. In order to implement such a rule set, the connectivity pattern of the network should reflect the structure of the rule set by:

(1) Representing each concept mentioned in the rule set with an individual node in the network.
(2) Implementing a rule by using a link to directly connect each node representing a concept in the condition of a rule and the node representing a concept in the conclusion of the rule.
(3) If two rules reach the same conclusion two intermediate nodes should be created, each of which is used in place of the conclusion of one rule.
(4) If there is a negated concept involved it is represented by the same node that represents the positive form of that concept.
(5) If a condition is positive and the conclusion is negative, the link between them should have a negative weight.
(6) If a condition is negative and the conclusion is negative, the link between them should have a positive weight.
(7) If a condition is negative and the conclusion is positive, the link between them should have a negative weight.

In this way, there is a direct, one-to-one isomorphic mapping between the structure of the rule set and that of the network.

(insert figure 1 here)

For the set of rules listed above, a connectionist network implementation is depicted in Figure 1. Reasoning in such a network is carried out by spreading of activation from nodes representing the conditions of rules to nodes representing the conclusions of rules. This process starts from externally activated and clamped nodes (representing initially known information) and then (if some of them are conditions of a rule) activation will be propagated to the nodes representing the conclusion of the rule. Furthermore, if some of these newly activated nodes in turn are the conditions of other rules, the activation will be further propagated to simulate application of these rules. This process continues until no more nodes can be activated. If weighted-sum computations are used in each node of the network for determining the activation of the node based on inputs received, the resulting activation in each node can be viewed as an accumulation of evidence to determine the confidence in the conclusion which the node represents. This is accomplished through first weighing each piece of evidence (with respect to each concept in the condition of a rule) by multiplying it with a numerical weight for each input that represents the confidence (in relation to the corresponding concept) and then summing all the products:

$$a_l(t) = \sum_j w_j * i_j(t)$$

where l is any node in the network, a denotes the activation, and i_j's are inputs received. Thus the most commonly used weighted-sum computation in connectionist models is a simple way for performing evidential reasoning. Other commonly encountered node computations can also be used, such as radial basis functions for implementing fuzzy logic rules (Peterson and Sun, 1998).

In certain situations logical reasoning may need to be carried out precisely, without involving evidential or fuzzy reasoning. Precise (propositional) logic rules can be implemented using the approach outlined above. For example, propositional Horn clause logic consists of single concepts, such as p, and rules such as $p_1\ p_2....p_n \longrightarrow q$. The concept in the conditions of a rule can be in negated form (e.g. $\neg p_1$), but the conclusion must be in positive form. To implement such logic, weights are assigned from (all the nodes representing) all the conditions of a rule to (the node representing) the conclusion of the rule in a way that ensures the total weight (the sum of the absolute values of all the weights) is 1, and the threshold of (the node representing) the conclusion of the rule is assigned the value of 1. If a condition is positive, then the corresponding weight is positive, otherwise a negative weight is assigned. In this way, the activation of a node will be either 1 or -1, depending on whether the weighted-sum of the inputs is greater than or equal to the threshold. For example, supposing there is the following rule:

$$a\ b\ c \longrightarrow d$$

a connectionist implementation will be as shown in Figure 2.

(insert figure 2 here)

Problems with this approach include how to handle reasoning involving multiple instances of the same rules and how to handle distinct instances that should be treated differently. One approach is to set up multiple copies of the rule, each of which takes care of one particular instantiation. In a rule set that contains multiple rules and where the conclusion of one rule is the condition of another, care must be taken in handling the chaining of rules, where the conclusion reached from one rule is the condition in another rule. With multiple instantiation of rules there is a need to implement each valid pair of first rules and second rules, in order to enable all valid inferences to be drawn. The implementation is shown in Figure 3. Maintaining separate chains is necessary if multiple objects can be present at the same time and if there is a need to keep track of each of them.

(insert figure 3 here)

It is possible to calculate the computational complexity of this implementation. Supposing there are a_1 instantiations of the condition of the first rule, c_1 instantiations of the conclusions of the first rule, a_2 instantiations of the condition of the second rule, and c_2 instantiations of the conclusion of the second rule. Then there will be $a_1 * c_1$ instantiations of the first rule, and $a_2 * c_2$ instantiations of the second rule. The total number of instantiations can be as high as $a_1 * a_2 * c_1 * c_2$. The example above is based on a simple chain of two rules, but is suggestive of the kind of complexity inherent when there are a large number of rules, some of which may form long chains. Connectionist inference systems implemented without variables also lack expressive power. For example, it is difficult to express relations, especially high-order relations (or n-ary relations in general). Without an adequate representation of relations, there is no way to form transient associations and structures dynamically (which are needed in most reasoning situations). An example of this would be the handling of a complex structured object or the remembering of a sentence with its original structures intact. Without the use of variables it is also impossible to avoid interference (i.e., cross-talk) of multiple conjunctive concepts (Feldman and Ballard, 1992). Despite the inherent problems with localist models, many researchers have used them for performing inference. Models include one by Derthick (1988) who translated logical constraints into energy functions and used them to implement a subset of the language *KL-ONE*. A modified Hopfield network was used by Narazaki and Ralescu (1992), with inference realized by a minimization of the energy function using a relaxation method which avoids local minima by having pertubation which is automatically triggered when the solution has fallen into a local minimum. Pinkas (1995) developed a connectionist inference engine capable of representing and learning propositional knowledge. Using densely connected networks (such as Hopfield networks and Boltzmann machines) Pinkas performed symbolic constraint satisfaction to perform nonmonotonic reasoning, coping with inconsistent facts and unreliable knowledge sources. A radial basis function network has been used by Raghuvanshi and Kumar (1997) in which inductive learning takes place using an error based heuristic adaptation rule. Their model can perform inference with conflicting knowledge and perform abduction and deduction.

4 Localist connectionist inference systems with variable binding

The following systems are example of early localist implementations of rule-based reasoning using variable binding, which demonstrate the feasibility of this approach:

- Parallel logical inference (Ballard, 1986) has been performed to implement a restricted form of resolution (see section 5.2) using a parallel relaxation algorithm and simple terms consisting of unitary constants or variables. All possible variable substitutions were prewired using hand-coded weights, with incompatible bindings being represented by units being connected with inhibitory links. This need for prewiring restricts the applicability of the model severely, as if arbitrary function symbols are allowed the set of possible substitutions becomes infinite and so cannot be prewired. Another localist connectionist system for performing resolution has been constructed by Lima (Lima, 1992).
- A connectionist unification and inference system has been developed by Hölldobler (1990a and 1990b) which can perform Horn clause logic on logical terms of fixed width and having a fixed depth of nesting of arguments.
- The Role Binding and Inferencing Network (*ROBIN*) (Lange and Dyer, 1989) is a hybrid localist spreading-activation inference system which handles dynamic variable bindings using a mechanism similar to marker-passing. In this model the terms used are just simple constants and variables. Each constant has a unit in the network that has a uniquely identifiable value called its signature, bindings are formed by passing these signatures around the network.
- Shastri and Ajjanaggadde (Shastri and Ajjanagadde, 1990) perform backward-chaining rule-based reasoning with first-order predicate logic (see section 4.2).
- An analysis is presented of how to perform generic rule-based reasoning in connectionist models by Sun (Sun, 1989), and the work addresses a number of important but often neglected issues in performing such a task. The analysis has been extended in *CONSYDERR* (Sun, 1992), an architecture integrating rule-based and similarity-based reasoning. This system can accommodate variable binding and inference with functional terms, and the author suggests how some of the localist nodes could be represented in their own distributed representational space by implementing them using conventional three layer feedforward networks (see section 4.1).
- Barnden (1989) takes a different tack on the problem of performing rule-based reasoning. Instead of having rules wired-in ahead of the time and in fixed forms, his system can dynamically construct representation during the reasoning process. Any particular representation used at a particular moment is transient, constructed on the basis of the resource constraints and other system-dependent considerations.

- Net-Clause Language (*NCL*) (Markov, 1991) is a neural network tool which integrates some connectionist and some classical symbolic processing features in a unified computational environment. It cannot be considered to be a purely connectionist system as the processing elements do not perform simple numeric calculations. In addition the network connections, rather than propagating activation values, propagate complex data structures. This system is best seen as a symbolic system which uses some connectionist ideas to organize its computation in a more flexible way. Similarly, *INFANT* (Buchheit, 1999) is a symbolic-like system where nodes contain propositional fragments, that also organizes some of its computations in a connectionist-like way.

In the following sections two of these models are investigated in greater depth, the sign propagation model of Sun (1992) and a synchronous activation model (Shastri and Ajjanagadde, 1990).

4.1 Sign propagation

There must be some way of assigning values to variables dynamically during the reasoning process and passing such values from one variable to another. There are a variety of ways (for an overview see (Browne and Sun, 1999)), the simplest of which is *sign propagation*. This was first proposed in (Lange and Dyer, 1989) and (Sun, 1989), and further developed in (Sun and Waltz, 1991) and (Sun, 1992). A separate node is allocated for each variable associated with each concept. For example, in first-order predicate logic, each argument of a predicate is allocated a node as its representation. A value is allocated to represent each particular object (i.e., a constant in first-order logic) and thus is a sign of the object which it represents. A node may take on an activation value as a sign in the same way as in conventional connectionist models. However this activation value represents a particular object and is merely a pointer. This sign can be propagated from one node to other nodes, when the same object which the sign represents is being bound to other variables from the application of a rule. Consider an example (adapted from (Shastri and Ajjanagadde, 1990)):

$$\begin{aligned} X, Y, Z\ [give(X, Y, Z) &\longrightarrow own(Y, Z)] \\ X, Y\ [buy(X, Y) &\longrightarrow own(X, Y)] \\ X, Y\ [own(X, Y) &\longrightarrow can\_sell(X, Y)] \end{aligned}$$

A network can be constructed for representing these rules and reasoning with them, as shown in Figure 4.

(insert figure 4 here)

For each predicate in the rule set, an *assembly* (of nodes) is constructed. The assembly contains $k+1$ nodes if the corresponding predicate contains k arguments. Each node in the assembly represents one particular argument of the predicate (the *argument nodes*), except one node, which represents the predicate as a whole (*the predicate node*). The argument nodes handle the binding of the corresponding arguments, whilst the predicate node computes as its activation the confidence (certainty) measure of an entire instantiated predicate. If there is a rule containing a condition (such as $give(X, Y, Z)$) and a conclusion (such as $own(Y, Z)$), corresponding nodes are connected in the assembly representing the condition and in the assembly representing the conclusion. In this process, the corresponding predicate nodes are always linked, so that the second predicate node can calculate the confidence for the second assembly based on the confidence of the first assembly (its logical antecedent) represented by the activation of the first predicate node. For the argument nodes, an argument node in the first assembly is linked to those argument nodes in the second assembly that have the same name as the originating argument node in the rule being implemented. If there are multiple conditions in a rule, the predicate nodes of all the assemblies that represent the conditions of the rule are linked to the predicate node in the assembly representing the conclusion. A weighted-sum can be used in the receiving predicate node for evidential combination. To perform forward-chaining inference the assemblies that represent known facts are activated, then activations from these assemblies will propagate to other assemblies which the initial assemblies are connected to. Further propagation will occur when those newly activated assemblies pass on their activations to other assemblies downstream. This process will continue until all the assemblies that are reachable from the initial assemblies are activated. There are two kinds of activation that are propagated. One kind is the confidence measure (evidential support) of a corresponding predicate (which is passed between predicate nodes), and the other is the sign (a pointer) used for representing an individual object (which is passed between argument nodes). Both types can be represented as a real number within a pre-defined range (such as between 1 and -1). These two kinds of activation are completely different, and thus have separate pathways, implying there are two different types of weights. One type is for weighing evidence, and the other is for mapping (or passing around information about) an object. However, these two types of activations and two types of weights are treated in the same way in such networks, thus ensuring the simplicity and uniformity of the models. For backward chaining a hypothesis must be proven based on known facts. For example, in answering a query, the query is treated as a hypothesis and an attempt is made to prove it with known rules and facts. The process is as follows: Initially an attempt is made to match the hypothesis with the conclusions of existing rules; if a match is found the conditions of the matching rule are used as a new hypothesis; if this new hypothesis can be proven, the original hypothesis is also proven. This process repeats until all of the hypotheses are proven. For example, to prove that *mary* can sell a particular *book*: $can\_sell(mary, book1)$, the predicate is matched with the conclusion of the rule: $own(X,Y) \longrightarrow can\_sell(X,Y)$. With this match, an attempt is made to prove a new hypothesis: $own(mary, book1)$, because if the latter can be proven the former follows immediately. Assuming that the fact that Mary owns *book1* is already known: $own(mary, book1)$, the new hypothesis is matched exactly and proven, consequently the original hypothesis can be proven. To implement backward chaining with assemblies, in addition to the predicate node another node is needed in an assembly for indicating whether a node is being considered as a hypothesis: the *hypothesis node*. To simplify this discussion binary (true/false) cases are considered initially. To generate hypotheses backwards, the direction of the link between two hypothesis nodes across two assemblies should be reversed (i.e., the opposite of the direction of the corresponding rule) but the direction of the link between two predicate nodes across two assemblies should remain the same (the direction of the rule). The purpose of hypothesis nodes and links connecting them is to generate new hypotheses backwards, whilst the purpose of predicate nodes is to prove hypotheses generated in the forward direction (and thus in the opposite direction of hypotheses generation). To start backward chaining inference, the predicate nodes are activated of all the assemblies representing known conditions (but they do not propagate activation). Then the hypothesis node of the assembly representing the hypothesis to be proved is activated. This node will propagate activation through backward links to generate new hypotheses. If a hypothesis node is activated in an assembly where the predicate node is already activated (i.e., the assembly represents a known fact), the (backward) activation flow of the hypothesis node is stopped and the activation flow of the predicate node is started (in a forward direction) to activate the predicate nodes of the assemblies where the activation to the current hypothesis node is from. This forward activation flow continues to activate the predicate nodes of the assemblies on the path from the original hypothesis to the hypothesis that matches the known fact, in the opposite direction of hypothesis generating activation flow. Thus there are two passes involved, a backward pass and the other a forward pass. The system should be constructed as follows:

(1) A hypothesis node can propagate activation to other assemblies in a backward direction, as long as there is no activated predicate node in these other assemblies.
(2) A predicate node can propagate activation to other assemblies if and only if these other assemblies have activated hypothesis nodes.

To implement this policy, some gating or the use of conjunctive links is necessary (see Figure 4). Where rules have multiple conditions, when the conclusion of a rule is matched with a hypothesis multiple new hypotheses corresponding to all of the conditions are generated simultaneously with backward propagation. Later (during forward propagation) the activation from all of these conditions together activates the predicate node of the conclusion (weighted-sum computation can accomplish this as described above). It is also possible to carry out both forward and backward chaining inference simultaneously in one system. In this case the known facts are activated and the network run to obtain all plausible conclusions. Rules are treated as specifying associations between conditions and conclusions, simultaneously in both directions. Notice that here rules are not implemented in a strict logical sense as circular reasoning is not excluded. To perform forward and backward chaining simultaneously, instead of a hypothesis node and a predicate node in an assembly, two predicate nodes are used (a *forward predicate node* for forward chaining and a *backward predicate node* for backward chaining). Once a forward predicate node in an assembly is activated, the backward predicate node is activated. Once a backward predicate node is activated, the corresponding forward predicate node is activated. The two nodes contain the same confidence measure for the predicate they represent, but if both are activated from separate sources the larger activation value prevails. These two nodes are used in different ways:

- Forward predicate nodes are used to propagate activation in a forward direction from a condition of a rule to the conclusion.
- Backward predicate nodes are used to propagate activation in a backward direction from the conclusion of a rule to a condition.

The propagation along both directions occurs at the same time, from the initial activation of known facts. One problem with such a network is that some of the activated nodes may quickly rise to saturation level, because of mutual (bi-directional) reinforcement. One way to control activation is to use a global inhibition node that measures at each moment in time the overall activity level of the network (by receiving activations from all of the predicate nodes), and then inhibits all predicate nodes by an equal proportion. That is:

$$a_l(t) = \frac{\sum_j w_j i_j(t)}{a_g(t)}$$

where l is any predicate node in the network, g is the global inhibition node, and a denotes activation. Global inhibition allows the activation of the nodes in each assembly to reflect their true confidence measure, i.e. the evidential support received from input lines. This model is parallel at the knowledge level and allows any number of variables to be implemented, however its complexity is high. Based on the ideas explained above, a formal treatment of variable binding with sign propagation is presented in the Discrete Neuronal (*DN*) model formalism, first proposed in (Sun, 1989) and further developed in (Sun and Waltz, 1991; Sun, 1992). This model is a generalization of conventional connectionist models, aiming at resolving some difficulties inherent in these models (such as variable binding) by removing some unnecessary restrictions. The generalization in this formalism extends over several dimensions, such as internal states, differentiated outputs and temporal responses. However, there is insufficient space in this paper to explain this model in depth, for an in-depth treatment see (Browne and Sun, 1999).

4.2 Temporal synchrony

One way of dealing with variable binding in connectionist systems is to use temporal aspects of node activation, in addition to or substituting the use of instantaneous activation values. Phase synchronization can be used by allowing different phases in an activation cycle to represent different objects involved in reasoning, and representing variable binding by the in-phase firing of nodes. The connectionist inference system SHRUTI (Ajjanagadde and L.Shastri, 1989; Shastri and Ajjanagadde, 1990; Ajjanagadde and Shastri, 1991) can represent a restricted number of rules with multi-place predicates. There are three different types of nodes in SHRUTI (see Figure 5):

(insert figure 5 here)

- Circular nodes, which fire in a particular phase when they are activated in that phase.
- Triangular nodes, which fire in all of the phases of a cycle when they are activated in that phase.
- Pentagonal nodes, which fire in all of the phases of a cycle when they are activated in all of the phases of a previous cycle uninterruptedly.

The same inference as used in Section 4.1 is used to show backward chaining in this system. A predicate with k arguments is represented by a pair of pentagonal nodes and k circular nodes. One of the pentagonal nodes is equivalent to the hypothesis node in the sign propagation method discussed above, the other pentagonal node is equivalent to the predicate node. Each of the k circular nodes are equivalent to the argument nodes. The pentagonal node representing the hypothesis is used in the backward pass, whilst the pentagonal node representing the predicate is used in the forward pass. The hypothesis pentagonal nodes related by a rule are connected in a backward direction (the opposite direction of a rule) and the predicate pentagonal nodes are connected in a in forward direction (the direction of a rule). Circular nodes of the predicates involved are connected in a backward direction (the opposite direction of a rule) as in Figure 5. In this example, the known fact is $give(john, mary, book1)$. When the hypothesis nodes are activated by a query, activation flows backwards to those hypothesis nodes that represent the conditions of a rule that has the original hypothesis as its conclusion. These hypothesis nodes represent new hypotheses, which once proven can be used to prove the original hypothesis. Each circular node for an argument of a predicate that represents a known fact is gated by the node representing the object (constant) to which the circular node (representing the argument) is bound. If these nodes fire in the same phase, the gate will transmit activation. If such links pass on activation, the sum of these activations will activate the predicate pentagonal node of the assembly (showing that the hypothesis represented by the assembly is proven). Such an activated predicate pentagonal node will propagate activation in a forward direction to other predicate pentagonal nodes in other assemblies. In Figure 5, the original hypothesis is $can\_sell(mary, book1)$. The hypotheses $own(mary, book1)$ and $give(john, mary, book1)$ are then generated. After $give(john, mary, bookl)$ is proven with the gating system described above, it is possible to use the forward flow of activation between respective pentagonal nodes to prove those hypotheses that generate this hypothesis, namely: $own(mary, book1)$ *and* $can\_sell(mary, book1)$. The triangular nodes are used to implement constraints, such as the constraint when implementing $p(X) \longrightarrow q(X, a)$ that its second argument must be a. This can be achieved by a gate that will allow activation to pass if and only if both the circular node representing the argument and the circular node representing the constant a fire in the same phase.

More recently *SHRUTI* has been extended in a number of ways (Shastri and Wendelken, 1999; Shastri et al., 1999; Shastri, 1999). Other researchers (Lane and Henderson, 1998; Bailey et al., 1998; Cohen et al., 1996; Park et al., 1995; Park, 2000)) have used architectures similar to *SHRUTI* for modelling tasks such as human syntactic processing, language acquisition and attention shifting. Hummel and Holyoak have used a synchronous activation approach to model analogical inference (Hummel and Holyoak, 1998), whilst Ajjanagadde has worked on the problem of abductive reasoning (Ajjanagadde, 1991; Ajjanagadde, 1993) and also pursued an alternate set of representational mechanisms (Ajjanagadde, 1997).

4.3 Constructing localist rule representations dynamically

Due to their associative nature and the computational efficiency resulting from parallelism, the methods discussed in sections 4.1 and 4.2 may well capture reflexive non-deliberative reasoning in which the immediate association and instant response are required. However, one common shortcoming of these methods discussed is that they use a pre-wired static representation that cannot be changed dynamically. It is often necessary to be able to construct representation on the fly, and to modify the constructed representation whenever a change occurs. This is especially true for deliberative (non-reflexive) reasoning (Hadley, 1990). One way to overcome this is to build localist representation ‘on the fly’, without pre-assigning each node to represent a particular (fixed) concept. This can be called *transiently localist representation*, since each node in a network represents a particular object at a particular moment in time, but can be used to represent a different object at a different moment in time. There first method to attempt this was described by Barnden (1989). In this model, a working space is needed that will hold the representation constructed. One method to achieve this is to use a two-dimensional array of nodes, such as an array of ‘registers’. Each register is a connectionist network with a fixed set of representational nodes, which contains a set of *flags* (which can either be on or off) and a *symbol*. A symbol or a flag is an activation vector in the register network. However, it can simply be viewed view as a sign in a single node. A symbol can be used to denote an object in the domain, predicate or a logical connective (i.e., a ‘class’ as it is called in this model). A flag is used to denote the role of a particular symbol (which is stored in a register along with the flag) in the representation (whether it is an argument to a predicate or whether it is a symbol for a particular representation). Some particularly useful flags are *class* and *instance* flags, and those that denote arguments of a predicate: *arg1*, *arg2*, etc. Representation in this model is set up with two techniques. One is physical adjacency (of neighboring registers) to form *clumps*. The other is the use of the same symbol to link different clumps of representations together (if they are part of a representation for a structured piece of information). For example, to represent a predicate such as $give(john, mary, book1)$ a clump can be created by assigning a number of adjacent registers to represent each of the following: *give (class), W (instance), john (arg1), mary (arg2)*, and *book1 (arg3)*. Here in each pair the first word is a symbol, and the word in the parentheses is a flag associated with the symbol. (see Figure 6). The representation should be interpreted as: W is an instance of $give(john, mary, book1)$.

(insert figure 6 here)

However, the representation of a single item can be made up of several clumps. For example, to represent $\neg give(john, mary, book1)$ two clumps can be created by dividing the representation into two. One clump contains a set of registers for the following: *give (class), W (instance), john (arg1), mary (arg2)*, and *book1 (arg3)*, and the other clump contains the following: *not (class), Z (instance)*, and *W (arg1)*. In the same way as before, in each pair the first word is a symbol and the word in the parentheses is a flag associated with the symbol (see Figure 7). The representation can be interpreted as: W is an instance of $give(john, mary, book1)$, and Z is an instance of *not* W, which means that Z is an instance of $\neg give(john, mary, book1)$.

(insert figure 7 here)

The dynamic construction of such representation is accomplished by a hard-wired specialized network along with the help of the array of registers. The actual work of construction is performed through a sequence of 'command signals', which are initiated by the special network and are sent to all the registers in parallel. Each register can respond to the signals in specific ways based on its own flags and the flags of its neighboring registers. Each command signal can specify as part of the signal a number of conditions which each responding register has to satisfy, regarding its flags and/or its neighboring registers' flag. A command signal can also specify that only one of all the responding registers will be selected for a task. The selection can be accomplished by a *temporal winner-take-all* (Barnden and Srinivas, 1992). This is useful for selecting an area in the array to create a clump to represent a predicate. To create a clump, a command signal first chooses a set of free registers away from existing clumps in the array, sets up the main symbol to be represented (i.e., the predicate symbol), and then sets up its flags as appropriate. Subsequent command signals then set up symbols and flags for each adjacent register. The details of each register subnetwork and the command signal generator network will not be expanded, since they can be realized in various ways and the particular way they are implemented is not important.

5 Connectionist inference systems using distributed representations

Distributed representations can be divided into those which use modularly distributed representations (where each symbol may be represented within a group of units) or globally distributed (where the same representation space is used for all symbols).

5.1 A model using a modularly distributed representation

Some localist models (Barnden, 1989) can also be viewed as distributed, because each symbol in the system is an activation vector (a vector of the activation values of a particular set of nodes). These activation vectors are allocated to represent symbols when a representation is constructed and is propagated and matched with other representations (i.e. other activation vectors). Such finely modularly distributed models are very close to localist models in that a simple transformation is possible to turn one type into another by mapping an activation vector (a set of activation values of a number of nodes) in modularly distributed models into an activation value in localist models, and vice versa (see Figure 8).

(insert figure 8 here)

Referring to the localist connectionist rule-based reasoning model discussed in Section 4.1, it is possible to transform it into a modularly distributed model. Recall that in the model, assemblies containing a type of complex node (called *DN* nodes) are used, that can perform extensive processing (i.e. more that just weighted-sum computation). If each of these nodes can be turned into a small network of simple connectionist nodes (computing only weighted-sum or sigmoid functions), and if each input/output value of these complex nodes can be represented with an activation vector, this will be modularly distributed connectionist model. To achieve this:

(1) Consideration needs to be taken of a mapping between an input/output value of a complex node and an activation vector of a network of simple nodes.
(2) The question must be addressed of how to carry out input/output mapping (i.e., the action function of a *DN* node).
(3) Questions must be addressed of how to represent states in the *DN* formalism and then how to carry out the state transition function of a *DN* node.

It has been shown that a multi-layer recurrent connectionist network of conventional nodes can achieve this (Giles et al., 1992; Giles and Omlin, 1993a), where a group of nodes represents the current state and separate groups of nodes represent all input/output lines, and all the groups of nodes can propagate their activations through sets of links. A recurrent MLP maps the current state and current inputs to a new state (represented by another group of nodes) and outputs (represented by yet another group of nodes). The new state is fed back (through some delay elements) as the new current state to the network in the next cycle. This helps the network to decide (along with the current inputs) which state to enter next and what to output next. This cycle can completely implement the functionality of a *DN* node. With *DN* nodes replaced by a network of conventional nodes, it can be seen that assemblies are a set of these networks (interconnected as explained in Section 4.1), and a complete network for rule-based reasoning is a set of such assemblies, as shown in Figure 9.

(insert figure 9 here)

Figure 10 shows a network that implements the rules used in earlier examples.

(insert figure 10 here)

5.1.1 Compilation of rules into networks

In small examples such as that given above, it is easy to work out mappings used in predicate nodes and in argument nodes for carrying out variable binding. However, with large rule sets it is not easy to do this, what is needed is a systematic procedure for building up a network to carry out rule-based reasoning. A procedure has been constructed to build a network automatically when given a set of rules (each of which comes with weights). This procedure involves the *Compilation of Fuzzy Evidential Logic into DN Network (CFELDNN*) (Fuzzy Evidential Logic is explained in more depth in section 6). This procedure is restricted to forward-chaining reasoning, and to the distributed pattern propagation and sign propagation methods for variable binding. In this procedure, three different types of assemblies are used (see Figure 11):

- Ordinary assemblies, which compute weighted-sums of inputs and pass bindings for the variables.
- OR assemblies, which compute the maximum of inputs and select one set of bindings out of many using this maximum.
- Complex assemblies, which are similar to ordinary assemblies except that they perform checking and unification.

(insert figure 11 here)

As before, it can be assumed that each node has a set of intra-assembly inputs and a set of inter-assembly inputs. Likewise, each node has a set of intra-assembly outputs and a set of inter-assembly outputs. In addition, it is assumed that there is no recursion in the rule set (that is, the rule set is acyclic). It is also assumed that rules come with weights attached. The main bottom-up procedure for compilation of a set of rules into a network is as follows:

SET-UP-RULE-NET:

Initialize the rule set to include all the rules to be implemented.

Repeat until the set is empty:

- Choose a rule $(p_{\mathbf{1}}(X_{\mathbf{1}})\ p_{\mathbf{2}}(X_{\mathbf{2}}) ......p_{\mathbf{n}}(X_{\mathbf{n}}) \longrightarrow q(X)|\ w_{\mathbf{1}}, w_2, ..., w_{\mathbf{n}})$ to work on, if there is no rule in the set that has $p_{\mathbf{i}}$ as the conclusion.
- Identify all the existing assemblies for $\mathbf{p_i},\ \mathbf{i} = \mathbf{1}, \mathbf{2}, ..., \mathbf{n},$ if there are none (with the same binding requirement), make one.
- Identify all the existing assemblies for $\mathbf{q}$, if there are none (with the same binding requirement), make one.
- If there is only one assembly for each $\mathrm{p_i}$ and there is no in-link to $\mathbf{q}$, link each $\mathbf{p_i}$ to $\mathbf{q}$, assign weights to each link.
- If there are in-links to $\mathbf{q}$, link $\mathbf{p_i}$'s to an intermediate assembly, then or-link this assembly with the existing $\mathbf{q}$ to a new $\mathbf{q}$ assembly which assumes all the

out-links of **q**, and assign weights as appropriate.

· If there are multiple assemblies for $\mathbf{p_i}$, or-link all assemblies for the $\mathbf{p_i}$ to an intermediate assembly, and then link that intermediate assembly to **q**, and assign weights as appropriate.

Delete from the set the rule implemented.

The procedure for setting up each individual assembly is described in (Sun, 1992; Browne and Sun, 1999). A brief explanation is as follows:

- For an ordinary assembly, the predicate node receives intra-assembly inputs from argument nodes, computes weighted sums of inter-assembly inputs, and sends a simple signal to argument nodes. Its functionality can be fully determined by the mappings specified above which can be implemented as a MLP that maps strings to numerical values and $\perp$ to 0.
- For an OR assembly, the predicate node computes the maximum of inter-assembly inputs, and the computation can be carried out by a MLP. Each argument node chooses one binding out of many based on the maximum activation in the predicate node, this can also easily be implemented using a MLP. Intra-assembly communications can be taken care of along with the basic functionalities above.
- Complex assemblies perform constraint checking and/or unification (and other operations). From the discussion above, it is clear that they can be implemented using MLPs based on the mappings specified.

This model uses a modularly distributed representation, in which each variable has its own separate representational space and is a good way of avoiding interference and crosstalk (c.f. Shastri and Ajjanagadde's (1990) analysis of crosstalk in Touretzky and Hinton's (1988) model). However, the modularly distributed representation used in the model described above does not meet van Gelder's (van Gelder, 1991) criteria of '*superposition*' and '*extendedness*' of representation. Fully distributed connectionist inference systems such as those outlined in the rest of this section do meet these criteria. Touretzky and Hinton's DCPS (1988) uses a globally distributed representation (i.e. the same representational space for all symbols), but has some limitations. Only a limited form of production rules are allowed in the model, where there is only one variable and it appears in the first position of both the condition sides of the inference rule, and anywhere in any of the action side of that rule. The model is further restricted in that it assumes that at any one time only one hypothesis of a rule matches the contents of working memory. Although the representation of the symbols in the system is distributed, rules must be performed in a serial fashion (as they must first be 'extracted' by the settling of the Boltzmann Machine onto the 'symbol surface' and then executed). Because of this, the only kind of parallelism that exists in the system is below the level of rules (in the mechanism for implementing rules) limiting any potential gains available from parallelism. It is not clear how this model can be extended and enhanced to handle more complex rule representation issues, since it is already very complex. No suggestions have been made as to how the system may be modified to cope with variables appearing in other positions, or how the matching of variables present in rules against variables present in WM could be performed. Another model, RUBICON (Samad, 1992) is a connectionist rule-based system that does allow for a variable number of expressions in the left and right hand sides of each rule.

5.2 Performing resolution with a distributed representation

Resolution (Robinson, 1965b) is a powerful formal inference method. A system using resolution can infer new facts and relationships from given facts and relationships, hence resolution is more powerful than simpler reasoning approaches such as Herbrand instantiation (Shapiro, 1991). There are many examples of symbolic resolution-based inference systems such as the reasoning performed by the Prolog inference engine (which has a resolution-based theorem prover at its heart) and many practical applications of resolution, including the design and validation of logic circuits (Kabat and Wojcik, 1985). Resolution can be thought of as being a generalization of the *transitivity* rule 'from a implies b and b implies c, deduce that a implies c'. Logical implication $a \longrightarrow b$ is logically equivalent to the clause $\neg a \vee b$, so it can be shown from the two clauses $a \vee b$ and $\neg a \vee c$ that $b \vee c$. This implies that by using two clauses that contain a complementary pair of literals (where one is a negation of the other) a third clause can be deduced and the complementary pair disposed of. A mechanical process (Nilsson, 1971) can be used to reduce logical expressions that involve universal quantifiers, existential quantifiers and implication symbols to a set of clauses. A variable binding process called *unification* is used in tandem with resolution. For example, the resolution $p(X)$ and $p(\neg a)$ requires the binding $X = a$ as this would produce the desired contradiction. Resolution based theorem provers successively apply unification followed by the 'cut rule' (Hogger, 1990). This rule states that it is possible to remove an atom if it occurs at opposite sides of two clauses then merge the resultant clauses into a new one, i.e. from $p \longrightarrow q$ and $q \longrightarrow r$ it is possible to deduce $p \longrightarrow r$. A query to the theorem prover is formulated as a negative clause (i.e. when asking 'a' this is formulated as '$\neg a$'). By sucessively applying unification and the cut rule the theorem prover attempts to generate the empty clause.

The connectionist resolution model outlined here (Browne, 1998b) built on a previous model that performed unification of (fixed width) nested terms and included the occurs check (Browne and Pilkington, 1995; Browne and

Sun, 1999). There is insufficient space in this paper to discuss the unification system in detail, readers seeking further explanation are referred to (Browne and Sun, 1999) for a comprehensive explanation, and toWeber (1992 & 1993) for explanations of other methods of performing unification with distributed representations. In the unification system used (Figure 12) the symbolic representations of pre-unification term pairs and unification results (taken from Hölldobler, 1990a) were transformed into distributed representations on the hidden layers of autoassociators. Autoassociator A in Figure 12 produced distributed representations of input term pairs, whilst autoassociator B produced distributed representations of the unification results. Symbolic representations presented to the input autoassociator consisted of 9216 pairs of logical terms.

(insert figure 12 here)

A four layer feed-forward network (network C in Figure 12) mapped the distributed representations of pre-unification terms produced by input autoassociator A to the distributed representations of unification results produced by target autoassociator B. The distributed representation representing the unification results was read off the output layer of network C and presented as input to a second network for resolution step to be attempted (Figure 13). This network had two sections:

- A section that made a decision based on the similarity of two distributed patterns of activation that it was presented with.
- A section that performed the cut rule.

(insert figure 13 here)

This network took two distributed patterns of activation as inputs, the distributed pattern of activation representing the unification results read from the output layer of network C, and a distributed pattern of activation representing a term stored in a database. These database terms were identical to the representations of post-unification results taken from the hidden layer of network B. The outputs of this first section were connected via weighted connections to the second part of the network consisting of sigma-pi units (Rumelhart et al., 1986), connected by weighted connections to the units represented the distributed pattern of activation stored in the database prior to performance of the resolution step. The sigma-pi units were connected to an output layer representing the database contents after the resolution step had been performed. A form of gating was performed by the sigma-pi units, in that they allowed the first section of the network to gate the outputs of the network in such a way that the network could produce the following outputs:

- If the post-unification pattern of activation taken from the output layer of network C and the database pattern of activation were similar (i.e. where the distributed representations of a term and its complement were present) the first section of this network was trained to generate 0 at all of its outputs.

- If the post-unification pattern of activation taken from the output layer of network C and the database pattern of activation were not similar (i.e. they did not represent a term and its complement), the network was trained to generate 1 at all of its outputs to signify the cut rule should not be performed.
- If the two distributed patterns of activation were not judged to be similar the cut rule was not performed. The distributed pattern of activation present at the database inputs of the network was reproduced at its outputs, indicating that the term was still present in the database.

To check the performance of the resolution network the distributed patterns of activation generated at its outputs were decoded through the hidden-to-output layer weights of autoassociator B. Then the symbolic outputs generated were compared with their expected values to discover whether resolution had been performed correctly.

The unification network correctly unified 97% of the novel term pairs presented to it, whilst the resolution network correctly resolved 82% of the novel distributed representations passed to it from the output layer of network C. This was an acceptable level of generalization as the probability of the correct output being produced by either network at random was 1 in 2^{21} (as there are 21 binary output units representing the correctly decoded term). Less than 100% performance is typical of connectionist systems trained by gradient descent. Whether this is important depends on what the system is being used to model. As a replacement for a symbolic theorem prover this level of performance would not be acceptable, whereas from a cognitive modelling viewpoint it may be acceptable as Humans often give less than 100% correct performance. The performance of the resolution network was worse than the unification network, this can be explained as it was attempting resolution on the 'noisy' distributed representations generated on the output layer of network C. These (real numbered) representations will not be 100% accurate, and this inaccuracy inevitably affected the resolution process. Conceivably a distributed connectionist 'clean-up' network for transforming the distributed representation output by network C to the nearest representation formed on the hidden layer of network B could reduce these errors.

In its current form the model performs only a single resolution step, in most proofs many such steps are required for useful inference. To do this the system would have to incorporate a database containing the distributed representations of several terms (robust to the deletion of these terms). In addition, in a chain of inference often one of a series of resolution steps will fail, forcing the system to retrace its steps and then attempt a different series of resolutions. This entails the provision of a backtracking mechanism in the model. It is possible that some form of recursive auto-associative memory (Pollack,

1988) could provide the stacking facility required by backtracking. A form of overall control mechanism would have to be provided to control the series of resolution steps attempted. It is possible a finite-state machine could provide a solution (Giles et al., 1992; Giles and Omlin, 1993a). Many problems are computationally inefficient to solve by relying on a single inference rule such as resolution. However resolution has led to the development of inference rules that are effective in these situations, such as hyper-resolution (Robinson, 1965a), unit resolution (Henschen and Wos, 1974) and paramodulation (Robinson and Wos, 1981). In the future it would be interesting to construct a connectionist inference system using distributed representations to perform these inference rules.

5.3 Tensor product representations

The representation of variable-sized structures in connectionist networks has been a challenge to connectionists. Many connectionist systems (such as those described in section 5.2) have a limited set of inputs and can only cope with a limited level of recursion (i.e. nesting of term structures). Because of this, the set of input patterns (variables, constants and terms) that can be correctly generated and processed often need to be completely specified when the network is constructed. Recursive connectionist architectures using distributed representations, such as RAAMs (Pollack, 1988), XRAAMs (Lee et al., 1990), LRAAMS (Sperduti, 1995), SRAAMS (Callan and Palmer-Brown, 1997) and BRAAMS (Adamson and Damper, 1999) exist, however the width and depth of embedding of the structures represented within them are limited. Tensor product based systems are one way of avoiding this restriction, in this form of representation structures (such as logical terms) can be represented by a set of slots (roles or attributes) and fillers (values). The use of tensor products allows a set of variable-value pairs to be represented by accumulating activation in a fixed-size collection of units. For example, a set of variable-value pairs Var_1, val_1 and Var_2, val_2 is represented by:

$$Var\_1 \otimes val\_1 + Var\_2 \otimes val\_2$$

where $\otimes$ represents the tensor product operation. If a variable is represented by an m-dimensional vector and a value by an n-dimensional vector, the tensor product of this variable and value is a $m \times n$ matrix of units. New variable-value pairs are superimposed over this matrix of units, allowing recursive structures to be constructed 'on the fly' of a size only bounded by the graceful saturation properties of the network. The use of tensor products for variable binding was proposed by Smolensky (1990). A connectionist implementation of production systems using tensor products that extended the work of Touretzky and Hinton was discussed in Dolan and Smolensky (1988) (although this model still suffers from the restriction on variables found in DCPS and is serial at the rule level). Whilst this model used role-filler bindings, other tensor-based models have used symbol-argument-argument bindings (Wiles et al., 1992; Halford et al., 1994) to perform analogical inference. In these schemes multiple predicates can be stored and one component of a predicate (the variable or the value) can be retrieved given the remaining components. A predicate is represented by the tensor product of all its components, for example the representation of the predicate $brother\_of(fred, sam)$ would be:

$$brother\_of \otimes fred \otimes sam$$

Tensor product representations can represent certain properties of relations that do not seem to be possible for *SHRUTI*'s synchronous activation approach (Halford, 1993a). A relation $r(A, B, ...n)$ can be handled by a tensor product of rank $n+1$, and this tensor not only represents the predicate-argument bindings but also the interactions within the structure. For example, the tensor product representation of $r(A, B, C)$ represents the influence of C on $r(A, B)$, the influence of B on $r(A, C)$ and the influence of A on $r(B, C)$. The synchronous activation approach used by *SHRUTI* can handle slot-filler bindings but it does not appear to be able to represent these higher-order relations that are important to complex concepts. There is also evidence that tensor product based models show stronger systematicity of inference than conventional feed-forward or recurrent networks (Phillips and Halford, 1997; Phillips, 1998; Halford et al., 1998). However, one problem with tensor-product based systems is their space requirements, they require an exponentially increasing number of elements to represent bindings.

5.4 Inference systems using novel recursive representations

Novel representational schemes such as Holographic Reduced Representations (HRRs) (Plate, 1991; Plate, 1995; Plate, 2000), Binary Spatter Codes (Kanerva, 1998) and the Braid operator (Gayler, 1998) have been proposed as a way of avoiding the exponential space requirements of tensor-product based representational schemes, and have the ability to represent nested structures in fixed-width vectors. HRRs are constructed by using a form of vector multiplication called circular convolution and decoded using correlation (the approximate inverse of circular convolution). The convolution of two HRRs results in another HRR encoding the information in the two original HRRs. This third HRR is not similar to either of the two HRRs it was constructed from, although the original component HRRs can be regenerated by decoding the third HRR using correlation. A role-filler (variable-value) binding is generated by circular convolution ($*$), for example to bind the role *Lover* to the person *john* this would be $Lover * john$, and to bind the role Is_Loved to the person

$mary$ this would be $Is\_loved * mary$. Propositions are encoded by the superposition of role-filler bindings, for example the predicate $loves(john, mary)$ is represented by:

$$loves + Lover * john + Is\_Loved * mary$$

The representations of fundamental entities such as roles and labels are based on random activation patterns with zero mean and a Gaussian distribution. Representations of semantically similar entities (such as *mary* and *john*) are composed of a base type (such as *human*) and random identifying patterns that are unique. When decoding HRRs the resulting vectors are noisy, and are recognized using the dot product operation to find the nearest matching vector. When HRRs are convolved or superposed the dimensionality of the resulting dimension vector is the same as the original vectors, so they avoid an explosion in the size of representations as the structures represented by the system become more complex. This property, together with the fact that some of the information in the original HRRs is lost on encoding (i.e., the new HRR does not contain all the information present before encoding) allows these representations to be considered to be 'reduced' representations as defined by Hinton (Hinton, 1990). Connectionist systems using HRRs have been used to perform analogical inference (Plate, 2000; Eliasmith and Thagard, 2001). Other novel techniques for generating distributed representations (with insufficent space to detail it here) are *context dependent thinning* (Rachkovski and Kussul, 2001) and *linear relational embedding* (Paccanaro and Hinton, 2000).

6 Representing fuzzy inference in connectionist systems

Connectionist models appear to have some relationship with fuzzy logic as both deal with the inexactness of real world situations, using numerical values that can change in correspondence with the change in certainty of real world situations. The question is: 'How can the two be merged in order to unify them and to utilize the strengths of both?' One way to accomplish this aim is to map one paradigm onto another (preferably in a simple and direct way).This section presents a simple mapping of weighted-sum connectionist models to fuzzy logic, providing an interpretation of weighted-sum connectionist models in terms of fuzzy inferences. In the following subsections, a simple fuzzy logic *Fuzzy Evidential Logic* (*FEL*) is defined and then implemented it in connectionist networks, see (Sun, 1989; Sun & Waltz, 1991; Sun 1992.

Propositional *FEL* is a direct reformulation of the weighted-sum connectionist models in logical terms. Here are some definitions:

- A *fact* is an atom or its negation, represented by a letter (with or without a negation symbol) and having a value between a lower bound and an upper bound. The value of an atom is related to the value of its negation by a specific method, so that knowing the value of an atom results in immediately knowing the value of its negation, and vice versa. For example, a, $\neg a$, x, and m are all facts.
- A *rule* is a structure composed of two parts: A left-hand side (LHS), which consists of one or more facts, and a right-hand side (RHS), which consists of one fact. When facts in the LHS get assigned values, the fact in the RHS can be assigned a value according to a *weighting scheme*. When the value of the LHS of a fact is unknown, zero is assigned as its value. For example, $a\ b\ c \longrightarrow d$ is a rule, and a weighting scheme associated with it indicate the value of d once the values of a, b, and c are known.
- A *weighting scheme* is a systematic method for assigning a weight to each fact in the LHS of a rule, with the sum of the absolute values of all weights less than or equal to 1. It also determines the value of the fact in the RHS of a rule by a threshold (if thresholds are used) by using weighted-sum of the values of the facts in the LHS (i.e., an inner-product of a weight vector and a vector of values of the LHS facts). When the range of values is continuous, then the weighted-sum is passed on if its absolute value is greater than the threshold, or 0 if otherwise. When the range of values is binary (or bipolar), then the result will be one or the other depending on whether the weighted-sum (or the absolute value of it) is greater than the threshold (usually the result will be 1 if the weighted-sum is greater than the threshold, 0 or -1 if otherwise). For example, a weighting scheme for the rule above is: $w_1 = 0.3,\ \ w_2 = 0.3,\ \ w_3 = 0.4$. The value of d, which is between a pair of bounds $[l, u]$ (e.g., $[-1, 1]$ or $[0, 1]$, depending on the value ranges of a, b, and c) is calculated by their weighted-sum: i.e., $d = w_1 a + w_2 b + w_3 c$.
- A *conclusion* is a value associated with a fact, and is defined here recursively. In using this definition and the procedure for calculating conclusions implied by the definition it is possible to restrict the structure of theories allowed in order to avoid circular reasoning by doing the following:

(1) For each rule having that conclusion in its RHS, obtain conclusions of all facts in its LHS (if any fact is unobtainable, assume it to be zero). Then calculate the value of the conclusion in question using the weighting scheme.

(2) Take the MAX of all these values associated with that conclusion calculated from different rules or given in initial input.

Here are some further definitions:

- An *implementation* of *FEL* is a network of elements connected via links, where each element represents an atom and its negation (there is a one-to-one mapping between an atom, including its negation, and an element). There are links that represent rules, going from elements representing facts in the LHS of a rule to elements representing facts in the RHS of a rule.
- An *Element* is a structure that represents one and only one fact (including its negation) and has multiple sites, each of which receives a group of links that represents one single rule (i.e. links from facts in the LHS of the same rule).

This implementation of *FEL* is in fact a connectionist network where elements are nodes in a network, and rules are defined using links. There are links emanating from nodes representing conditions in a rule to nodes representing the conclusion in a rule, and weighted-sum computation is carried out within each site of a node for computing and propagating activations (i.e., for evidential combination in reaching conclusions based on given conditions). Activations from different sites of a node are MAXed, corresponding to the previous definition of *conclusions*.

In terms of dynamics, inferences in *FEL* can also be easily mapped to connectionist networks. In *FEL* no particular order is specified in which inferences should be performed, so any appropriate order can be used. The network implementation of *FEL* can be viewed as performing a parallel breadth-first search reasoning, with all the links that are applicable taking effect at the same time and activation flows propagating in all directions. Details of implementations are beyond the scope of this paper. For formal proofs of correctness and other implementation details, see (Sun, 1995a; Sun, 1995c).

FEL can handle some simple but important logics as special cases. Such generality is necessary to ascertain the usefulness of *FEL* and consequently the adequacy of connectionist networks in capturing logical inferences. Firstly, it is shown how *FEL* can simulate Horn clause logic (Sun, 1995a). A *Binary* FEL is a reduced version of *FEL*, in which values associated with facts are binary, the sum of weights (each of which is positive) in each rule is 1, and all thresholds are set to 1. The binary *FEL* is sound and complete with respect to Horn clause logic (Sun, 1995a; Sun, 1995c). In addition to Horn clause logic it can be shown that *FEL* can emulate one of its extensions – Shoham's Causal Theory (CT) (Sun, 1995c).

The connectionist model presented above can be viewed as an implementation of the propositional *FEL*, whilst the sign propagation method for variable binding presented in Section 4.1 can be viewed as a suitable implementation of predicate *FEL*. The logical correctness of the sign propagation variable binding method as an implementation of *FEL* can be established (Sun 1995a & 1995d).

Other researchers have investigated the relationship between fuzzy and connectionist modelling, including a fuzzy neural logic network which attempts to model a Prolog-like fuzzy inference system (Ding et al., 1996) and a model that can encode directly structured knowledge in a fuzzy neural network and perform incremental learning (Machado and da Rocha, 1997). Fuzzy rules have also been used in connectionist production systems (Kasabov and Shishkov, 1993; Kasabov, 1994). A thorough overview of fuzzy neurocomputing can be found in (Magdalena, 1997).

7 Extraction of inference rules

Many attempts have been made to extract symbolic *IF-THEN* rules from connectionist systems. Gallants connectionist expert systems (Gallant, 1988) and Matrix Controlled Inference Engine (*MACIE*) (Gallant and Hayashi, 1990) are two early models where expert system rules are extracted from a neural network. Many other rule extraction techniques followed, mostly applied to extracting rules from MLPs (Saito and Nakano, 1988; Shavlik and Towell, 1989; Baba et al., 1990; Bochereau and Boutgine, 1990; Goh, 1993; McMillan et al., 1993; Yeung and Fong, 1994; Yoon and Lacher, 1994; Sethi and Yoo, 1994; Fletcher and Hinde, 1995; Thrun, 1995; Benitez et al., 1997; Taha and Ghosh, 1997; Ampratwum et al., 1998; Ishikawa, 2000; Setiono, 2000) with a smaller number applied to Kohonen networks (Ultsch et al., 1993), recurrent networks (Giles and Omlin, 1993b) and radial basis function networks (McGarry et al., 1999b). The difference between the approaches to rule extraction can be categorised (Tickle et al., 2000) as that between 'decompositional' and 'pedagogical' approaches. Decompositional approaches involve analysing weights and links to extract rules, with some requiring specialised weight modification algorithms (Shavlik, 1994) or network architectures such as an extra hidden layer of units with staircase activation functions (Bologna, 2000). Pedagogical approaches treat the network as a 'black box' and extract rules by observing the relationship between its inputs and outputs, and because of this are general purpose in nature and can be applied to any feedforward network (Craven and Shavlik, 1997). Some of methods lead to the extraction of rules that involve fuzzy grades of membership calculation (Hayashi, 1991; Mitra and Hayashi, 2000).

A typical approach to rule extraction is that of Fu (1994) who proposed an exhaustive search based algorithm to extract conjunctive rules from MLPs. To find rules, the learner first searches for all the combinations of positive conditions that can lead to a conclusion. Then, with a previously found combination of positive conditions, the learner searches for negative conditions that should be added to guarantee the conclusion. In the case of three-layered networks, the learner can extract two separate sets of rules (one for each layer) and then integrate them by substitution.

An alternative form of rules exists, the *N-of-M* form. Rules in this form state: 'If N of the M conditions, $a_1, a_2, ..., a_m$, is true, then the conclusion b is true' It is argued (Towell and Shavlik, 1993) that some concepts can be better expressed in such a form, and they also help avoid the combinatorial explosion in tree size found with *IF-THEN* rules. A four-step procedure is used to extract such rules, by first grouping similarly weighted links, eliminating insignificant groups, and then forming rules from the remaining groups through an exhaustive search. The following steps are performed:

(1) With each output node, form groups of similarly-weighted links.
(2) Set link weights of all group members to the average of the group.
(3) Eliminate any group that do not significantly affect the output value.
(4) Optimize biases (thresholds) of all output nodes, using the backpropagation algorithm, while holding links weights constant.
(5) Form a rule for each output node, using the weights and threshold of the rule.
(6) If possible, create an *N-of-M* rule.

These rule extraction algorithms are meant to be applied at the end of the training of a network. Once extracted, the rules are fixed; there is no modification on the fly, unless the rules are re-extracted (starting anew) after further training of the network. On the other hand, in *CLARION* (Sun and Peterson, 1998) is an agent that can extract and modify rules dynamically. In this model connectionist reinforcement learning and rule learning work together simultaneously. Extracting and modifying rules dynamically is computationally less expensive because it minimizes the search necessary, and helps the agent adapt to changing environments by allowing the addition and the removal of rules at any time. In doing so, *Clarion* avoids examining the details of the network from which rules are extracted. Instead it focuses on the *behavior* of the network. Specifically, if some action decided by the bottom level is successful then a rule is extracted that corresponds to the decision and the rule is added to the rule network. In subsequent interactions with the world the extracted rule is verified by considering the outcome of applying the rule. If the outcome was not successful (as measured by a criterion), then the rule should be made more specific and exclusive of the current case. If the outcome was successful, an attempt is made to generalize the rule to make it more universal. This process is determined by an information gain criterion which measures the success of a rule application. As well as being a useful technique for understanding what a particular network has learnt rule extraction can be helpful when a link between the neural and symbolic components is required in hybrid architectures

(McGarry et al., 1999a; Wermter and Sun, 2000).

8 Comparisons between models using localist or distributed representations

Comparisons can be made between different representational schemes used to perform connectionist symbolic inference. Localist representations are popular with many researchers in this field, partially because they are easy to interpret as the activation value of a node in a localist connectionist system is semantically interpretable. One significant problem of distributed representations is that they can be difficult to interpret. Because of its superposed and extended nature, the distributed pattern of activation representing a concept can not be easily labelled. Techniques do exist which attempt to interpret distributed patterns of activation, including statistical techniques (Bullinaria, 1997), rule extraction techniques (Andrews et al., 1997) and generalized effect analysis (Browne, 1998a; Browne and Picton, 1999), but these are obviously more complex than simply recording the activation of a node in a localist connectionist representation. In addition, training times in distributed connectionist systems can be extensive, and it can often be difficult to achieve 100% correct performance.

As the representations in localist connectionist systems are localized they can be thought of as symbolic systems. However, in these models processing is typically carried out not by some form of in-built theorem prover as seen in classical AI but by the massively parallel flow of activation values between units.

In a classical symbolic AI system an entity is given an arbitrary label that distinguishes it from other entities. For example, nothing about an entity labelled *dog* makes this entity more closely related to the concept of '*dog*' than to the concept of '*tree*'. There is nothing about the entities themselves that indicates their meaning because they get their designated meaning only because of how the system interprets them. In a localist connectionist system an entity may be arbitrarily designated or it may be learnt using a learning algorithm such as recruitment learning (Diederich, 1988; Diederich, 1991). In contrast, a semantic entity in a distributed connectionist system is a distributed pattern of activity with a *microsemantics* (Dyer, 1990), because it has an internal pattern that systematically reflects the meaning of the entity. The entity representing the concept '*dog*' will in some way be closer to the internal representation of a concept such as '*cat*', and in some way more removed from a concept such as '*tree*'. In this way distributed connectionist representations carry their own meaning, and this gives them ability to generalize on encountering novel input. A novel example being presented to the network and fed through the appropriate weights and activation functions will be systematically given an appropriate representation. The internal structure of this representation will be relatively similar to the structure of representations of objects to which this example is semantically similar in relevant aspects, and quite different from the structure of semantically different objects. Symbolic systems and localist connectionist systems often generalize poorly on encountering novel input (Sun, 1995c) (but as a counter to this see (Page, 2000)). This generalization ability constitutes a significant advantages of distributed representations over symbolic or localist representations.

Although many connectionist models are not realistic implementations of biological neural networks (for a discussion see (Crick, 1989)), models using distributed representations are more neurobiologically realistic than localist connectionist or symbolic AI models (Smolensky, 1995).

9 Problems with connectionist inference systems

There are many problems with connectionist systems (Browne, 1997), but two major problems suffered by connectionist systems using localist or distributed representations to perform inference are described below. These problems are not suffered by symbolic AI systems.

9.1 Limits on the productivity and width of representations

The *productivity* of a system refers to the ability of that system to generate and correctly process items from an infinite set. With reference to inference, this refers to the processing of a (potentially) infinite number of variables and constants or functors (potentially) up to infinite length. Because of the possibility of nesting of arguments, computational structures such as terms with other terms nested inside them can be produced. The size of these structures cannot be pre-determined before the run-time of a system. The property of productivity is displayed by symbolic systems which can produce and process an (almost) infinite set of terms from an (almost) infinite set of symbols using recursive structures with an (almost) infinite level of embedding (only being limited by physical constraints such as memory size). Most connectionist systems have a limited set of inputs and can only cope with a limited level of recursion (i.e. nesting of term structures). Because of this, the set of input patterns that can be correctly generated and processed often need to be completely specified when the network is constructed. Recursive connectionist architectures using distributed representations, such as RAAMs (Pollack, 1988), XRAAMs (Lee et al., 1990), LRAAMS (Sperduti, 1995), SRAAMS (Callan and Palmer-

Brown, 1997) and BRAAMS (Adamson and Damper, 1999) exist, however the width and depth of embedding of structures represented within them are limited by the precision of implementation, and they cannot produce infinitely larger or deeper structures. The tensor product, HRR and other novel representations discussed in Sections 5.3 and 5.4 can be modified dynamically as the network is performing its processing. However, there are limitations that interfere with the representation of structures using vectors or real numbers. The precision with which real numbers can be implemented will affect the representation, together with the fact that the more deeply nested the components of recursive structures the more degradation by noise when encoding and decoding these structures. Whether this is a serious limitation on connectionist systems depends on the task they are being used for. If the task is to emulate the properties of a symbolic inference system (such as a theorem prover), this limitation is serious. However, if the task is that of modelling human cognition this may not be such a serious limitation. The representation of recursive structures is intimately related to the competence/performance issue in cognitive science (MacLennan, 1993). Cognitive science often distinguishes between an idealized theoretical *competence* (under which the embedding of structures is unlimited) from the observed *performance* (which is limited in the observed depth). There is evidence that humans exhibit a limit on the depth of recursion they can process (Dyer, 1995). This is indicated by the difficulty that humans have in understanding sentences containing more than a certain level of embedding. Other evidence relates to the number of elements in a representation (i.e. the *width* of the representation) that humans can successfully deal with, where the number of elements in a representation affects the level of relation that the representation can represent. The number of elements can be thought of as the number of facets of a situation that can be viewed simultaneously. There is considerable evidence (Halford, 1993b; Halford et al., 1998) that most humans can reason using representations with at most five elements, such as a predicate (the first element) with four arguments (the four other elements). The psychological existence of the processing of rank six representations (such as a predicate with five arguments) by humans is speculative, and if it exists it probably does so only for a minority of adults. This implies that insofar as matching human cognitive capacity is the goal, a network using distributed representations would only have to process a limited arity. In addition, the graceful degradation observed when inference systems using distributed representation have to process deeply embedded structures is much closer to observed human performance than the hard limit imposed by (for example) stack size in a symbolic AI system. Thus connectionist systems can provide more realistic models of observed human cognitive processing.

Structural systematicity leads to the ability of a system to correctly process inputs that it has never encountered before. There are a number of different definitions of structural systematicity. Three levels of systematicity from weak to strong were defined by Hadley (Hadley, 1992), but the most precise definition, using six levels of systematicity, has been given by Niklasson and van Gelder (Niklasson and van Gelder, 1994). These levels are:

(1) Where no novelty is present in the inputs presented to the system. Every input presented to the system has already been encountered (for example in the training phase of a connectionist system).
(2) Where novelty is present in the inputs, but all the individual arguments in these inputs have at some time appeared in the same positions in inputs previously encountered by the system.
(3) Where the inputs contain at least one example which has an argument in a position in which it has never before been encountered by the system. For example, a connectionist system could be trained on a selection of input patterns in which a particular symbol was never present in the first argument position of the patterns used for training, and then tested on a set of patterns where that symbol was present at that argument position.
(4) Where novel symbols appear in the new inputs presented to the system.
(5) Where there is novel complexity in the new inputs presented to the system. To display this form of complexity a connectionist system would have to be capable of being trained on patterns with n inputs in the training set and then correctly process patterns with $n + 1$ inputs in the test set (this is related to the arguments discussed in section 9.1).
(6) Where the test set contains both novel symbols and novel complexity.

Symbolic AI inference systems can display all these levels of systematicity as they are not dependent on the makeup of a training set to form their representations. Hence they can generate and process new items whilst only being restricted by physical constraints, such as the size of available memory resources. Localist connectionist systems are not usually dependent on the make-up of a training set to form their representations, and so easily cope with tasks involving up to the third level of systematicity described above, but because of the problems outlined in section 9.1, have problems displaying the fourth and higher levels because any recursive representation in the network has to be constructed to a pre-specified depth, unlike the (potentially almost infinite) recursive representation provided with symbolic systems..

Connectionist systems using distributed representations are heavily dependent on the make-up of the training set used to develop these representations, hence they often do not display systematicity beyond the second level described above. However, in performing simple logical operations with a system using distributed representations (Niklasson and van Gelder, 1994) systematic performance of a neural network both when exposed to novel inputs and when exposed to inputs appearing in novel positions has been demonstrated (i.e. up to the fourth level above). However, there is evidence (Phillips, 1998) that inference systems based on standard feed-forward and recurrent networks may never be able to reach the top level of systematicity described above. However, it may be that systems based on tensor products or HRRs will achieve this[1].

10 Conclusions

It has been argued that connectionist systems may well offer an opportunity to escape some of the problems of symbolic AI, only if ways can be found of instantiating the sources of power displayed by symbolic systems within connectionist systems (Smolensky, 1988; Sun, 1995b & 1995c). At least in the case of inference, connectionist systems have not as yet matched the power of symbolic AI inference systems. However, any attempt to match the capabilities of a symbolic theorem prover using a connectionist system may be a misguided attempt to tie connectionist systems into a 'symbolic straitjacket'. Connectionist systems may well give better models of the (limited) inference capabilities of the human mind. In addition, connectionist systems have unique advantages. For example, they generalize better to unexpected or noisy inputs than symbolic AI systems. For complex tasks some form of hybrid system (Sun and Bookman, 1994; Wermter and Sun, 2000) combining the capabilities of connectionist and symbolic AI components may be more appropriate.

New theories of representation are being developed. While some of these theories attempt to link symbolic AI and connectionist representations (Smolensky et al., 1992) in a description that applies to both, other theories try to subsume both types of representation within a higher level theory such as dynamical systems theory (Port and van Gelder, 1995). In the future we hope that from these theories will spring new types of more powerful and flexible representation for building better intelligent systems as well as for modeling human cognition.

[1] Hadley has achieved the top level of Niklasson and van Gelder's hierarchy and satisfied his own criteria of strong systematicity with a connectionist model (Hadley and Cardei, 1997), but this model has some symbolic components and so cannot be said to have achieved the highest levels of systematicity in a connectionist way.

Acknowledgments

We thank two anonymous referees who provided useful comments on the first draft of this paper. Ron Sun acknowledges the support from the ARI grant DASW01-00-K-0012 during the preparation of this article.

References

Adamson, M. J. and Damper, R. I. (1999). B-RAAM: A connectionist model which develops holistic internal representations of symbolic structures. *Connection Science*, 11:41–71.

Ajjanagadde, V. (1991). Abductive reasoning in connectionist networks: Incorporating variables, background knowledge, and structured explanada. Technical Report WSI 91-6, Wilhelm-Schickard Institute, University of Tubingen, Tubingen, Germany.

Ajjanagadde, V. (1993). Incorporating background knowledge and structured explanada in abductive reasoning: A framework. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:650–654.

Ajjanagadde, V. (1997). *Rule-Based Reasoning in Connectionist Networks*. PhD thesis, Department of Computer Science, University of Minnesota.

Ajjanagadde, V. and L.Shastri (1989). Efficient inference with multi-place predicates and variables in a connectionist system. In *Proceedings of the 11th Annual Cognitive Science Society Conference*, pages 396–403, Hillsdale, NJ. Lawrence Erlbaum.

Ajjanagadde, V. and Shastri, L. (1991). Rules and variables in neural nets. *Neural Computation*, 3:121–134.

Ampratwum, C. S., Picton, P. D., and Browne, A. (1998). Rule extraction from neural network models of chemical species in optical emission spectra. In *Proceedings of the Workshop on Recent Advances in Soft Computing*, pages 53–64.

Andrews, R., Tickle, A. B., Golea, M., and Diederich, J. (1997). Rule extraction from trained artificial neural networks. In Browne, A., editor, *Neural Network Analysis, Architectures and Algorithms*. Institute of Physics Press, Bristol, UK.

Baba, K., Enbutu, I., and Yoda, M. (1990). Explicit representation of knowledge aquired from plant historical data using neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 155–160.

Bailey, D., Chang, N., Feldman, J., and Narayanan, S. (1998). Extending embodied lexical development. In *Proceedings of the Twentieth Conference of the Cognitive Science Society*, pages 84–89.

Ballard, D. H. (1986). Parallel logical inference and energy minimisation.

In *Proceedings of the AAAI national conference on artificial intelligence*, pages 203–208.

Barnden, J. and Srinivas, K. (1992). Overcoming rule-based rigidity and connectionist limitations through massively parallel case-based reasoning. *International Journal of Man-Machine Studies*, 36:221–246.

Barnden, J. A. (1989). Neural net implementation of complex symbol processing in a mental model approach to syllogistic reasoning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 568–573.

Barnden, J. A. and Srinivas, K. (1996). Quantification without variables in connectionism. *Minds and Machines*, 6:173–201.

Benitez, J., Castro, J., and Requina, J. I. (1997). Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks*, 8(5):1156–1164.

Bochereau, L. and Boutgine, P. (1990). Extraction of semantic features and logical rules from multilayer neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 579–582, Washington, DC.

Bologna, G. (2000). Rule extraction from a multilayer perceptron with staircase activation functions. In *Proceedings of the International Joint Conference on Neural Networks*, pages 419–424, Como, Italy.

Browne, A. (1997). Challenges for neural computing. In Browne, A., editor, *Neural Network Perspectives on Cognition and Adaptive Robotics*, pages 3–19. Institute of Physics Press, Bristol, UK.

Browne, A. (1998a). Detecting systematic structure in distributed representations. *Neural Networks*, 11(5):815–824.

Browne, A. (1998b). Performing a symbolic inference step on distributed representations. *Neurocomputing*, 19:23–34.

Browne, A. and Picton, P. (1999). Two analysis techniques for feed-forward networks. *Behaviormetrika: Special Issue on Analysis of Knowledge Representations in Neural Network Models*, 26(1):75–87.

Browne, A. and Pilkington, J. (1995). Performing variable binding with a neural network. In Taylor, J., editor, *Neural Networks*, pages 71–84. Alfred Waller, Henley on Thames, UK.

Browne, A. and Sun, R. (1999). Connectionist variable binding. *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks*, 16(3):189–207.

Buchheit, P. (1999). A neuro-propositional model of language processing. *International journal of intelligent systems*, 14:585–601.

Bullinaria, J. (1997). Analyzing the internal representations of trained neural networks. In Browne, A., editor, *Neural Network Analysis, Architectures and Algorithms*, pages 3–26. Institute of Physics Press, Bristol, UK.

Callan, R. and Palmer-Brown, D. (1997). (S)RAAM: An analytical technique for fast and reliable derivation of connectionist symbol structure representations. *Connection Science*, 9(2):139–159.

Chan, S. W. K. and Franklin, J. (1998). Symbolic connectionism in natural language disambiguation. *IEEE Transactions on Neural Networks*, 9(5):739–755.

Churchland, P. S. and Sejnowski, T. (1992). *The Computational Brain*. MIT Press, Cambridge, MA.

Cohen, M. S., Freeman, J. T., and Wolf, S. (1996). Meta-recognition in time stressed decision making: Recognizing, critiquing, and correcting. *Human Factors*, 38(2):206–219.

Collins, A. and Loftus, J. (1975). Spreading activation theory of semantic processing. *Psychological Review*, 82:407–428.

Craven, M. W. and Shavlik, J. W. (1997). Understanding time series networks. *International Journal of Neural Systems*, 8(4):373–384.

Crick, F. (1989). The recent excitement about neural networks. *Nature*, 337:129–132.

Derthick, M. (1988). Mundane reasoning by parallel constraint satisfaction. Technical Report TR CMU-CS-88-182, Carnegie-Mellon University.

Diederich, J. (1988). Connectionist recruitment learning. In *Proceedings of the European Conference on Artificial Intelligence*, pages 351–356.

Diederich, J. (1991). Steps towards knowledge-intensive connectionist learning. In Barnden, J. A. and Pollack, J., editors, *Advances in Connectionist and Neural Computation Theory*, volume 1. Ablex, Norwood, NJ.

Ding, L., Teh, H. H., Wang, P., and Lui, H. C. (1996). A Prolog-like inference system based on neural logic: An attempt towards fuzzy neural logic programming. *Fuzzy Sets and Systems*, 82:235–251.

Dolan, C. P. and Smolensky, P. (1989). Tensor product production system – a modular architecture and representation. *Connection Science*, 1(1):53–68.

Dyer, M. G. (1990). Distributed symbol formation and processing in connectionist networks. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:215–239.

Dyer, M. G. (1995). Connectionist natural language processing: A status report. In Sun, R. and Bookman, L. A., editors, *Computational Architectures Integrating Neural and Symbolic Processes*, pages 389–429. Kluwer Academic Press, Boston, USA.

Eliasmith, C. and Thagard, P. (2001). Integrating structure and meaning: A distributed model of analogical mapping. *Cognitive Science*, 25, 245-286.

Feldman, J. A. and Ballard, D. H. (1992). Connectionist models and their properties. *Cognitive Science*, 6(3), 205–254.

Fletcher, G. and Hinde, C. (1995). Using neural networks as a tool for constructive rule based architectures. *Knowledge Based Systems*, 8(4):183–187.

Franklin, S. and Garzon, M. (1990). Neural computability. In Omidvar, O., editor, *Progress in Neural Networks*, volume 1. Ablex, NJ.

Fu, L. (1994). Rule generation from neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 24(8):1114–1124.

Gallant, S. I. (1988). Connectionist expert systems. *Communications of the ACM*, 31:152–169.

Gallant, S. I. and Hayashi, Y. (1990). A neural network expert system with confidence measurements. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 3–5. Paris, France.

Gayler, R. W. (1998). Multiplicative binding, representation operators and analogy. In Holyoak, K., Gentner, D., and Kokinov, B., editors, *Advances in analogy research: Integration of theory and data from the cognitive, computational and neural sciences*. New Bulgarian University, Sofia, Bulgaria (full text available at http://cogprints.soton.ac.uk).

Ghalwash, A. Z. (1998). A recency inference engine for connectionist knowledge bases. *Applied Intelligence*, 9:205–215.

Giles, C. and Omlin, C. (1993a). Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent networks. *Connection Science*, 5(3&4):307–328.

Giles, C. and Omlin, C. (1993b). Rule refinement with recurrent neural networks. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 801–806.

Giles, C. L., Miller, C. B., Chen, D., Chen, H. H., Sun, G. Z., and Lee, Y. C. (1992). Learning and extracting finite state automata with second order recurrent neural networks. *Neural Computation*, 4:393–405.

Goh, T. H. (1993). Semantic extraction using neural network modelling and sensitivity analysis. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1031–1034, Nagoya, Japan.

Hadley, R. (1990). Connectionism, rule following and symbolic manipulation. In *Proceedings of the American Association of Artificial Intelligence*, volume 2, pages 579–586.

Hadley, R. (1992). Compositionality and systematicity in connectionist language learning. In *Proceedings of the 14th Annual Conference of the Cognitive Science Society*, pages 659–664. Lawrence Erlbaum.

Hadley, R. F. and Cardei, V. C. (1997). Acquisition of the active-passive distinction from sparse input and no error feedback. Technical Report CSS-IS TR97-01, School of Computing Science, Simon Frazer University, Burnaby, B.C., Canada.

Halford, G., Wilson, W., and Phillips, S. (1998). Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Behavioral and Brain Sciences*, 21(6), 803–831.

Halford, G. S. (1993a). Commentary: Competing, or perhaps complementary approaches to the dynamic binding problem with similar capacity limitations. *Behavioral and Brain Sciences*, 16(3):461–462.

Halford, G. S. (1993b). Creativity and the capacity for representation: Why are humans so creative? *AISB Quarterly*, 85:32–41.

Halford, G. S., Wilson, W. H., Guo, J., Gayler, R. W., Wiles, J., and Stewart, J. E. M. (1994). *Connectionist implications for processing capacity limitations in analogies*, pages 363–415. Ablex, Norwood, NJ.

Hayashi, Y. (1991). A neural expert system with automated extraction of fuzzy if-then rules and its application to medical diagnosis. In Lippmann, R., Moody, J., and Touretzky, D., editors, *Advances in Neural Information Processing Systems*, volume 3. Morgan Kaufmann, San Mateo, CA.

Henschen, L. and Wos, L. (1974). Unit refutations and Horn sets. *Journal of the Association for Computing Machinery*, 21:590–605.

Hilario, M. (1997). An overview of strategies for neurosymbolic integration. In Sun, R. and Alexandre, F. (Eds.). *Connectionist symbolic integration*. Lawrence Erlabaum. Hillsdale, NJ.

Hilario, M. (2000). Architectures and techniques for knowledge-based neurocomputing. In Cloete, I. and Zurada, J. M., editors, *Knowledge-Based Neurocomputing*, pages 27–62. MIT Press, Cambridge, UK. Hybrid systems reference.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, (46):47–75.

Hinton, G. E., McClelleand, J. L., and Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing*, volume 1, pages 77–109. MIT Press, Cambridge, MA.

Hogger, C. J. (1990). *Essentials of Logic Programming*. McGraw-Hill, New York.

Hölldobler, S. (1990a). A connectionist unification algorithm. Technical Report TR-90-012, International Computer Science Institute, Berkeley, CA.

Hölldobler, S. (1990b). CHCL-a connectionist inference system for Horn logic based on the connection method. Technical Report TR-90-042, International Computer Science Institute, Berkeley, CA.

Hummel, J. E. and Holyoak, K. (1998). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104:427–466.

Ishikawa, M. (2000). Rule extraction by successive regularization. *Neural Networks*, 13:1171–1183.

Kabat, W. and Wojcik, A. (1985). Automated synthesis of combinatorial logic using theorem proving techniques. *IEEE Transactions on Computing*, C-34:610–628.

Kanerva, P. (1998). Encoding structure in boolean space. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 387–392.

Kasabov, N. K. (1994). Connectionist fuzzy production systems. In *Proceedings of the Fuzzy Logic in Artificial Intelligence IJCAI Workshop of LNAI*, volume 847, pages 114–127, Berlin. Springer-Verlag.

Kasabov, N. K. and Shishkov, S. I. (1993). A connectionist production system with partial match and its use for approximate reasoning. *Connection Science*, 5(3 and 4):275–305.

Lane, P. and Henderson, J. (1998). Simple synchrony networks: Learning to parse natural language with temporal synchrony variable binding. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 615–620.

Lange, T. and Dyer, M. G. (1989). High-level inferencing in a connectionist network. Technical Report UCLA-AI-89-12, UCLA, Los Angeles, USA.

Lee, G., Flowers, M., and Dyer, M. G. (1990). Learning distributed representations for conceptual knowledge and their application to script-based story processing. *Connection Science*, 2(4):313–345.

Lima, P. M. V. (1992). Logical abduction and prediction of unit clauses in symmetric hopfield networks. In Alexander, I. and Taylor, J., editors, *Artificial Neural Networks*. Elsevier, Amsterdam, The Netherlands.

Machado, R. J. and da Rocha, A. F. (1997). Inference, inquiry, evidence, censorship and explanation in connectionist expert systems. *IEEE Transactions on Fuzzy Systems*, 5(3):443–459.

MacLennan, B. (1993). Characteristics of connectionist knowledge representation. *Information Sciences*, 70:119–143.

Magdalena, L. (1997). A first approach to a taxonomy of fuzzy-neural systems. In Sun, R. and Alexandre, F. (Eds.). *Connectionist symbolic integration*. Lawrence Erlabaum. Hillsdale, NJ.

Marcus, G. F., Vijayan, S., Rao, S. B., and Vishton, P. (1999). Rule learning in seven month old infants. *Science*, 283:77–80.

Markov, Z. (1991). A tool for building connectionist-like networks based on term unification. In *Proceedings of the processing declarative knowledge international workshop*, pages 199–213.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

McGarry, K., Wermter, S., and MacIntyre, J. (1999a). Hybrid neural systems: From simple coupling to fully integrated neural networks. *Neural Computing Surveys*, 2(1):62–93.

McGarry, K., Wermter, S., and MacIntyre, J. (1999b). Knowledge extraction from radial basis function networks and multi layer perceptrons. In *Proceedings of the International Joint Conference on Neural Networks*. Washington, D. C.

McMillan, C., Mozer, M., and Smolensky, P. (1993). Dynamic conflict resolution in a connectionist rule-based system. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 1 and 2, pages 1366–1371.

Medsker, L. R. (1994). *Hybrid Neural Network and Expert Systems*. Kluwer Academic Publishers, Boston, MA.

Mitra, S. and Hayashi, Y. (2000). Neuro-fuzzy rule generation: Survey in a soft computing framework. *IEEE Transactions on Neural Networks*, 11(3):748–768.

Narazaki, H. and Ralescu, A. L. (1992). A connectionist approach for rule-based inference using an improved relaxation method. *IEEE transactions on Neural Networks*, 3(5):741–751.

Newell, A. (1980). Physical symbol systems. *Cognitive Science*, 4:135–183.

Newell, A. (1986). The symbol level and the knowledge level. In Pylyshyn, Z. W. and Demopoulos, W., editors, *Meaning and Cognitive Structure*, pages 31–39. Ablex Publishing Corp., Norwood, NJ, USA.

Niklasson, L. and van Gelder, T. (1994). Can connectionist models exhibit non-classical structure sensitivity? In *Proceedings of the Cognitive Science Society*, pages 664–669. Lawrence Erlbaum.

Nilsson, N. J. (1971). *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill, New York.

Paccanaro, A. and Hinton, G. (2000). Learning distributed representations of concepts using linear relational embedding. Technical Report GCNU TR 2000-002, University College London, London, UK.

Page, M. (2000). Connectionist modeling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, 23, 479–480.

Park, N. S. (2000). Connectionist symbolic rule encoding using a generalized phase-locking mechanism. *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks (Special Issue on Connectionist Symbol Processing)*, In Press.

Park, N. S., Robertson, D., and Stenning, K. (1995). Extension of the temporal synchrony approach to dynamic variable binding in a connectionist inference system. *Knowledge-Based Systems*, 8(6):345–357.

Peterson, T. and Sun, R. (1998). An RBF network alternative to a hybrid architecture. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 768–773.

Phillips, S. (1998). Are feedforward and recurrent networks systematic? analysis and implications for a connectionist cognitive architecture. *Connection Science*, 10(2):137–160.

Phillips, S. and Halford, G. S. (1997). Systematicity: Psychological evidence with connectionist implications. In *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, pages 614–619.

Pinkas, G. (1995). Reasoning, nonmonoticity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, 77:203–247.

Pinker, S and Prince, A. (1988) On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28(1-2), 73–193.

Plate, T. (1991). Holographic reduced representations. Technical Report CRG-TR-91-1, Department of Computer Science, University of Toronto, Ontario, CA.

Plate, T. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

Plate, T. (2000). Analogy retrieval and processing with distributed vector representations. *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks (Special Issue on Connectionist Symbol Processing)*, 17(1), 29–40.

Pollack, J. B. (1988). Recursive auto-associative memory – devising compositional distributed representations. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, pages 33–39.

Port, R. F. and van Gelder, T. (1995). *Mind as Motion: Explorations in the Dynamics of Cognition*, MIT Press, Cambridge, MA.

Rachkovski, D. A. and Kussul, E. M. (2001). Binding and normalisation of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13(2):411–452.

Raghuvanshi, P. S. and Kumar, S. (1997). Bipolar radial basis function inference networks. *Neurocomputing*, 14:195–204.

Robinson, G. and Wos, L. (1981). *Paramodulation and theorem proving in first-order theories with equality*, volume 4. Elsevier, Amsterdam, The Netherlands.

Robinson, J. A. (1965a). Automatic deduction with hyper-resolution. *International Journal of Computing and Mathematics*, 1:227–234.

Robinson, J. A. (1965b). A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12:23–41.

Rosenfeld, R. and Touretzky, D. (1988). Coarse coded symbol memories and their properties. *Complex Systems*, 2:463–484.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Learning internal representations by error propagation*, volume 1. MIT Press, Cambridge, MA.

Saito, K. and Nakano, R. (1988). Medical diagnostic expert system based on pdp model. In *Proceedings of IEEE International Conference on Neural Networks*, pages 255–262.

Samad, T. (1992). Hybrid distributed/localist architectures. In Kandel, A. and Langholz, G., editors, *Hybrid Architectures for Intelligent Systems*, chapter 10, pages 200–219. CRC Press, Boca Raton, FL. After Touretzky, Rubicon, a connectionist rule-based system that allows for a variable number of expressions in the left and right-hand of a rule. Similar approaches taken by Kasabov and Shishkov (1993) and Kasabov (1994).

Sethi, I. and Yoo, J. (1994). Symbolic approximation of feedforward networks. In Gesema, E. and Kanal, L., editors, *Pattern Recognition in Practice*, volume IV, pages 313–324. Elsevier, North-Holland.

Setiono, R. (2000). Extracting m-of-n rules from trained neural networks. *IEEE Transactions on Neural Networks*, 11(2):512–519.

Shapiro, E. (1991). *Encyclopedia of Artificial Intelligence*. MIT press, Cambridge, MA.

Sharkey, N. (1992). The ghost in the hybrid – a study of uniquely connectionist representations. *AISB Quarterly*, 79:10–16.

Shastri, L. (1999). Advances in SHRUTI: A neurally motivated model of relational knowledge representation and rapid inferencing using temporal synchrony. *Applied Intelligence*, 11(1):79–108.

Shastri, L. and Ajjanagadde, V. (1990). From simple associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings. Technical Report MS-CIS-90-05, University of Pennsylvania, Philadelphia, PA.

Shastri, L., Grannes, D. J., Narayanan, S., and Feldman, J. A. (1999). A connectionist encoding of schemas and reactive plans. In Kraetzschmar, G. K. and Palm, G., editors, *Hybrid Information Processing in Adaptive Autonomous vehicles: Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

Shastri, L. and Wendelken, C. (1999). Knowledge fusion in the large: Taking a cue from the brain. In *Proceedings of the Second International Conference on Information Fusion*, pages 1262–1269, Sunnyvale, CA.

Shavlik, J. (1994). Combining symbolic and neural learning. *Machine Learning*, 14(2):321–331.

Shavlik, J. and Towell, G. (1989). An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3):233–255.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:159–216.

Smolensky, P. (1995). *Computational models of mind*. Blackwell, Cambridge, MA.

Smolensky, P., Legendre, G., and Miyata, Y. (1992). Principles for an integrated connectionist and symbolic theory of higher cognition. Technical Report CU-CS-600-92, Computer Science Department, University of Colorado, Boulder, CO.

Sperduti, A. (1995). Stability properties of labeling recursive auto-associative memory. *IEEE Transactions on Neural Networks*, 6(6):1452–1460.

Sun, R. (1989). A discrete neural network model for conceptual representation and reasoning. In *Proceedings of the 11th Conference of the Cognitive Science Society*, pages 916–923, Hillsdale, NJ. Lawrence Erlbaum.

Sun, R. (1992). On variable binding in connectionist networks. *Connection Science*, 4:93–124.

Sun, R. (1995a). A new approach towards modelling causality in commonsense reasoning. *International Journal of Intelligent Systems*, 10:581–616.

Sun, R. (1995b). Robust reasoning: Integrating rule-based and similarity-based reasoning. *Artificial Intelligence*, 75(2):241–296.

Sun, R. (1995c). Schemas, logics and neural assemblies. *Applied Intelligence*, 5(2):83–102.

Sun, R. and Alexandre, F. (1997). Connectionist symbolic integration. Lawrence Erlabaum. Hillsdale, NJ.

Sun, R. and Bookman, L. A. (1994). *Computational Architectures Integrating Neural and Symbolic Processes: A Perspective on the State of the Art*. Kluwer, Boston.

Sun, R. and Peterson, T. (1998). Autonomous learning of sequential tasks: Experiments and analyses. *IEEE Transactions on Neural Networks*, 9(6):1217–1234.

Sun, R. and Waltz, D. (1991). Neurally inspired massively parallel model of rule-based reasoning. In Soucek, B., editor, *Neural and Intelligent System Integration*, pages 341–381. John Wiley and Sons, New York.

Taha, I. and Ghosh, J. (1997). Evaluating and ordering of rules extracted from feedforward networks. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 1, pages 408–413.

Thrun, S. (1995). Extracting rules from artificial neural networks with distributed representations. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, pages 505–512. MIT Press, San Mateo, CA.

Tickle, A., Maire, F., Bologna, G., Andrews, R., and Diederich, J. (2000). Lessons from past, current issues, and future research directions in extracting knowledge embedded in artificial neural networks. In Wermter, S. and Sun, R., editors, *Hybrid Neural Systems*. Springer-Verlag, Berlin.

Touretzky, D. S. and Hinton, G. E. (1988). A distributed connectionist production system. *Cognitive Science*, 12(3):423–466.

Towell, G. and Shavlik, J. W. (1993). The extraction of refined rules from knowledge based neural networks. *Machine Learning*, 31:71–101.

Ultsch, A., Mantyk, R., and Halmans, G. (1993). Connectionist knowledge aquisition tool: CONKAT. In Hand, J., editor, *Artificial Intelligence Frontiers in Statistics: AI and Statistics*, volume III, pages 256–263. Chapman and Hall, London, UK.

van Gelder, T. (1991). What is the 'D' in 'PDP'? A survey of the concept of distribution. In Ramsey, W., Stich, S., and Rumelhart, D. E., editors, *Philosophy and Connectionist Theory*, pages 33–60. Lawrence Erlbaum Associates, Hillsdale, NJ.

Weber, V. (1992). Connectionist unification with a distributed representation. In *Proceedings of the International Joint Conference on Neural Networks*, pages 555–560.

Weber, V. (1993). Unification in Prolog by connectionist models. In *Proceedings of the Fourth Australian Conference on Neural Networks*, pages A5–A8.

Wermter, S. and Sun, R. (2000). *Hybrid Neural Systems*. Springer, Heidelberg.

Wiles, J., Halford, G., Stewart, J. E. M., Humphreys, M. S., Bain, J. D., and Wilson, W. H. (1992). Tensor models: A creative basis for memory retrieval and analogical mapping. Technical Report 218, University of Queensland, Queensland, Australia.

Yeung, D. and Fong, H. (1994). Knowledge matrix: An explanation and knowledge refinement facility for a rule induced neural network. In *Proceedings of the 12th National Conference on Artificial Intelligence*, volume 2, pages 889–894.

Yoon, B. and Lacher, R. (1994). Extracting rules by destructive learning. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1771–1776, Orlando, FL.

Figure Captions

Figure 1. A Connectionist Implementation of Rules.

Figure 2. A Connectionist Implementation of Precise Logic Rules.

Figure 3. Implementing Rules in a Connectionist Network with Multiple Instantiations.

Figure 4. A Connectionist Network for Backward-Chaining Reasoning where '+' Indicates Enabling and '-' Indicates Blocking.

Figure 5. A Network Utilizing Phase Synchronization for Variable Binding.

Figure 6. Representing $give(John, Mary, book1)$ with adjacent registers for *give (class), W (instance), John (arg1), Mary (arg2)*, and *book1 (arg3)*.

Figure 7. Representing *not* give(John, Mary, book1)With Two Clumps of Adjacent Registers for *give (class), W (instance), John (arg1), Mary (arg2)*, and *book1 (arg3)*.

Figure 8. Mapping between modularly distributed representations and localist representations where a is (a1, a2....an), b is (b1, b2....bn), and c is (c1, c2....cn).

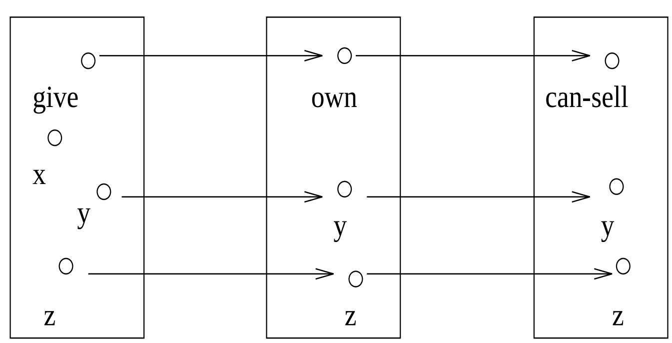
Figure 9. Implementing a DN node in Conventional Connectionist Networks.

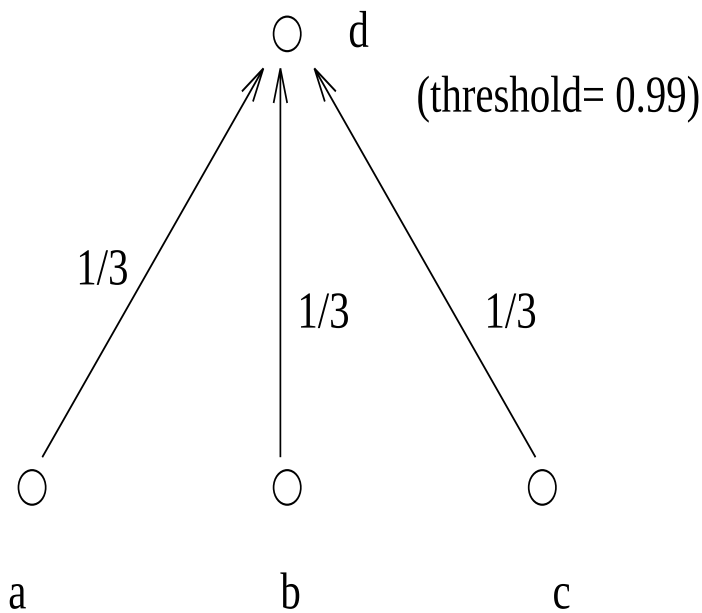
Figure 10. A Modularly Distributed Network for Rule-Based Reasoning.

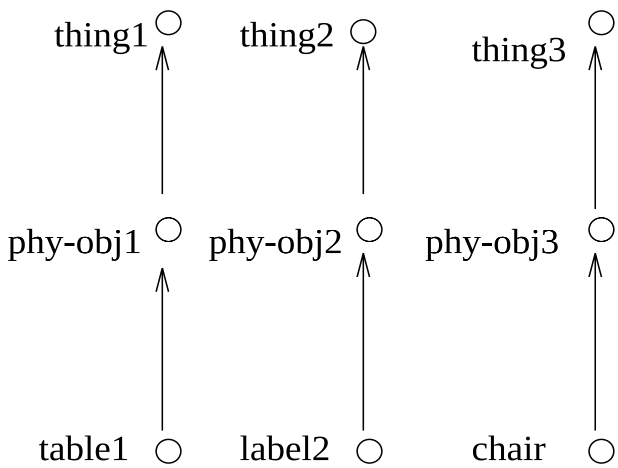
Figure 11. The Three Types of Assemblies, where C Denotes a Predicate Node and X Denotes an Argument Node: (1) an Ordinary Assembly, (2) an OR Assembly, and (3) a Complex Assembly (for Equality Checking and Other Tasks).

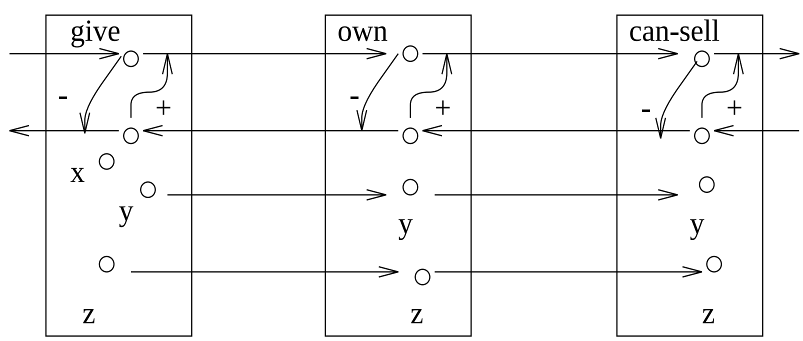
Figure 12. Complete Unification Network. Autoassociator A Produces Distributed Representations of two Input Terms. Autoassociator B Produces Distributed Representations of Unification Results. Network C Maps Distributed Representations from Hidden Layer of A to Hidden Layer of B.

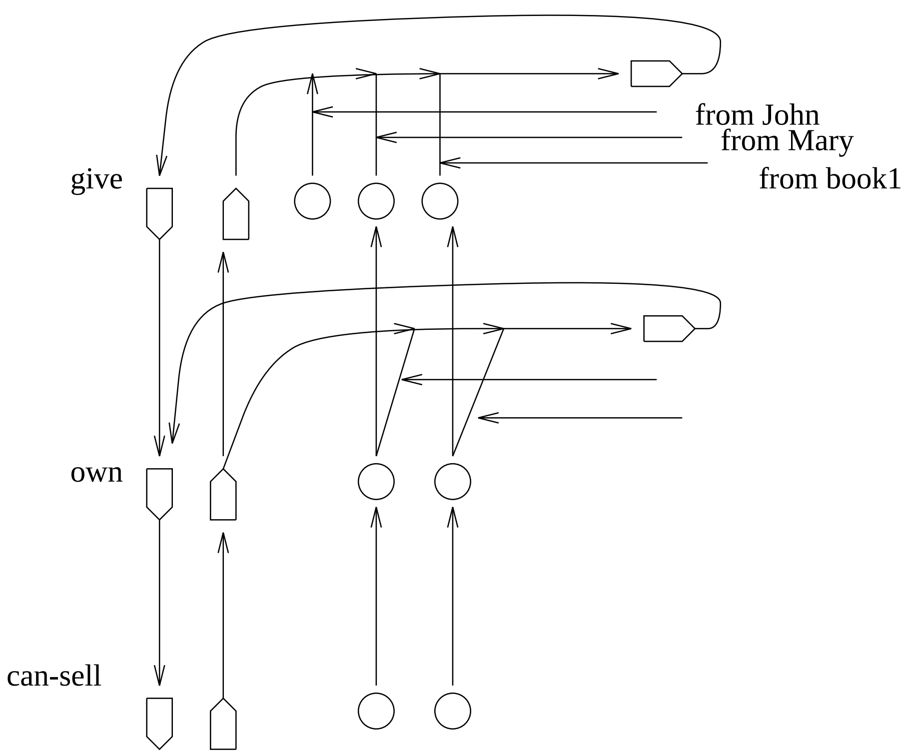
Figure 13. Cut Rule Network. Distributed Representation of Unification Results from Output Layer of Network C is Supplied as Input Together with Distributed Representation of Term from Database. Result of Feedforward Network Mapping Allows Sigma-Pi Units to Gate Database and Reproduce Representation of Term at Output if Resolution is Performed, or Set All Output Units to 0 Indicating Term has been Removed from Database.


give
x
y
z
own
y
z
can-sell
y
z


d
(threshold= 0.99)
1/3
1/3
1/3
a
b
c

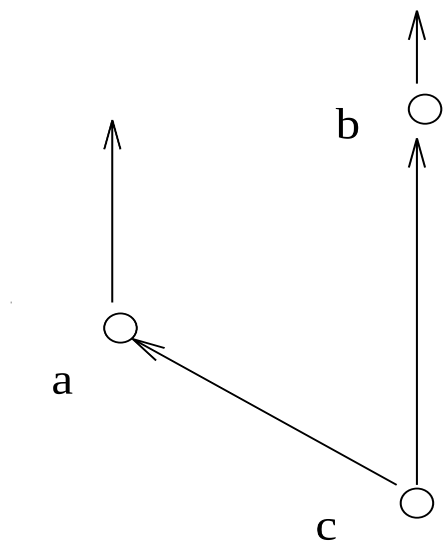

thing1
thing2
thing3
phy-obj1
phy-obj2
phy-obj3
table1
label2
chair


give
own
can-sell
-
+
-
+
-
+
x
y
y
y
z
z
z

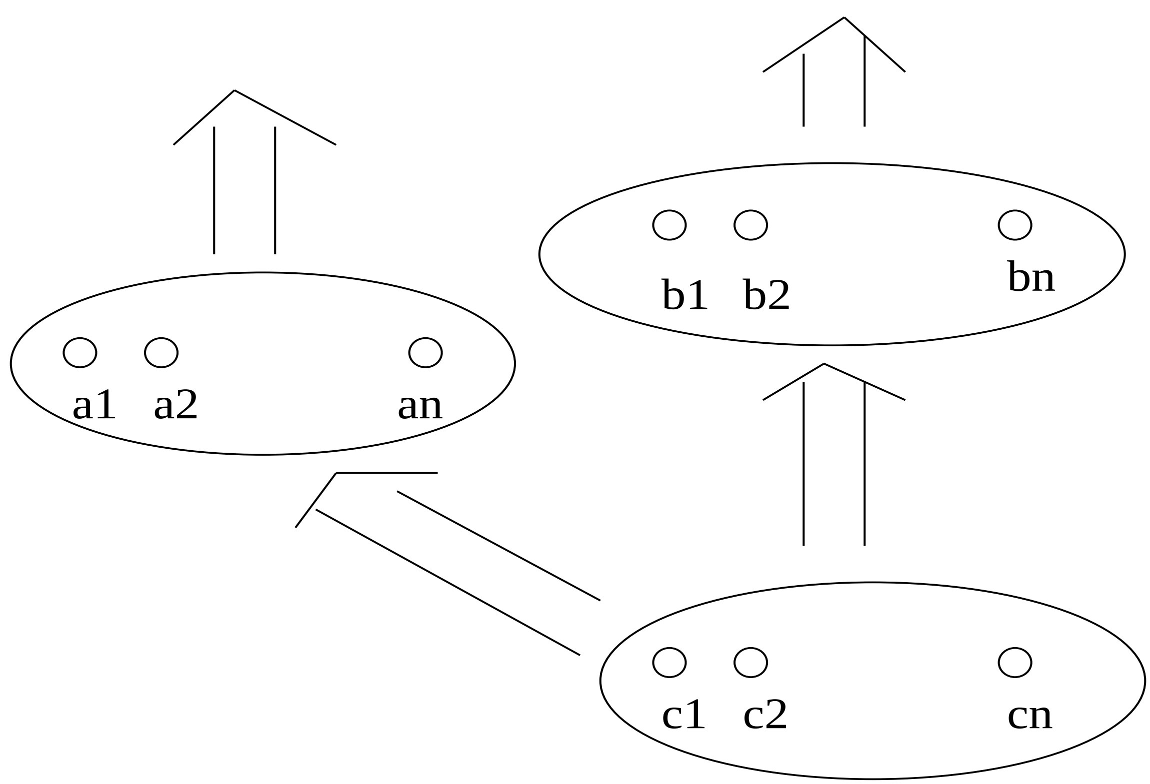

from John
from Mary
from book1
give
own
can-sell

				W (inst)	give (class)		
			John (arg1)	Mary (arg2)	book1 (arg3)		

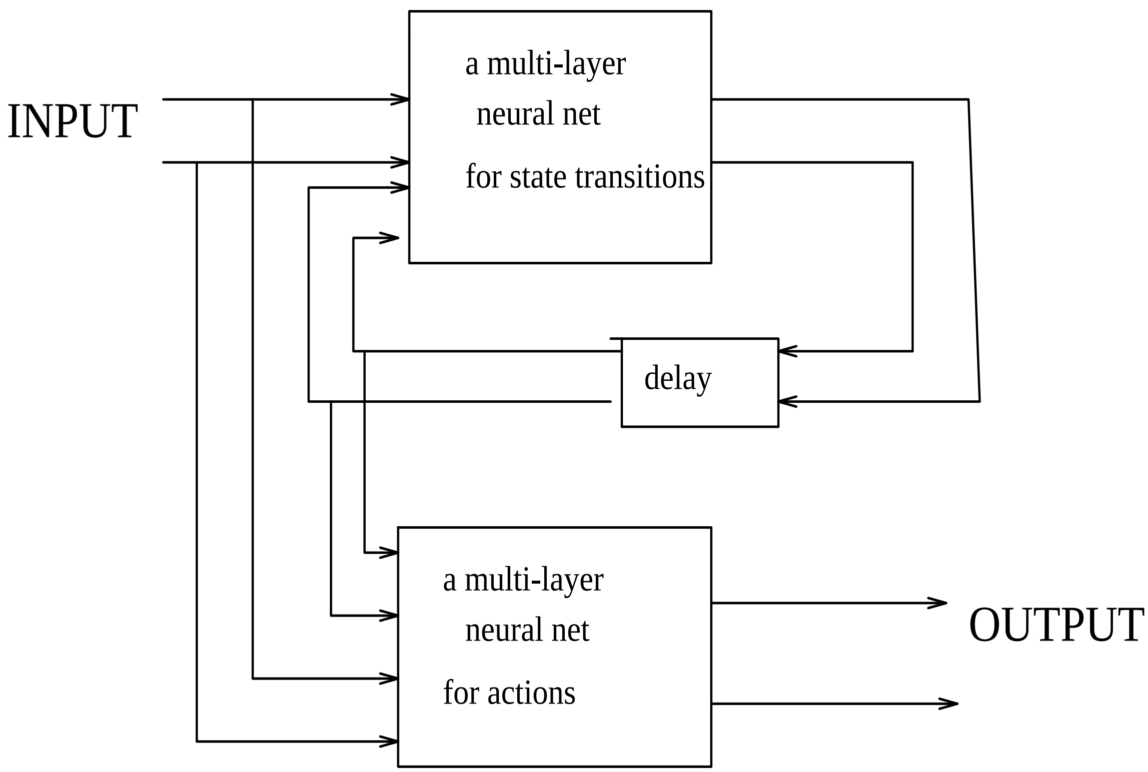
	Z (inst)	not (class)					
	W (arg1)						
				W (inst)	give (class)		
			John (arg1)	Mary (arg2)	book1 (arg3)		

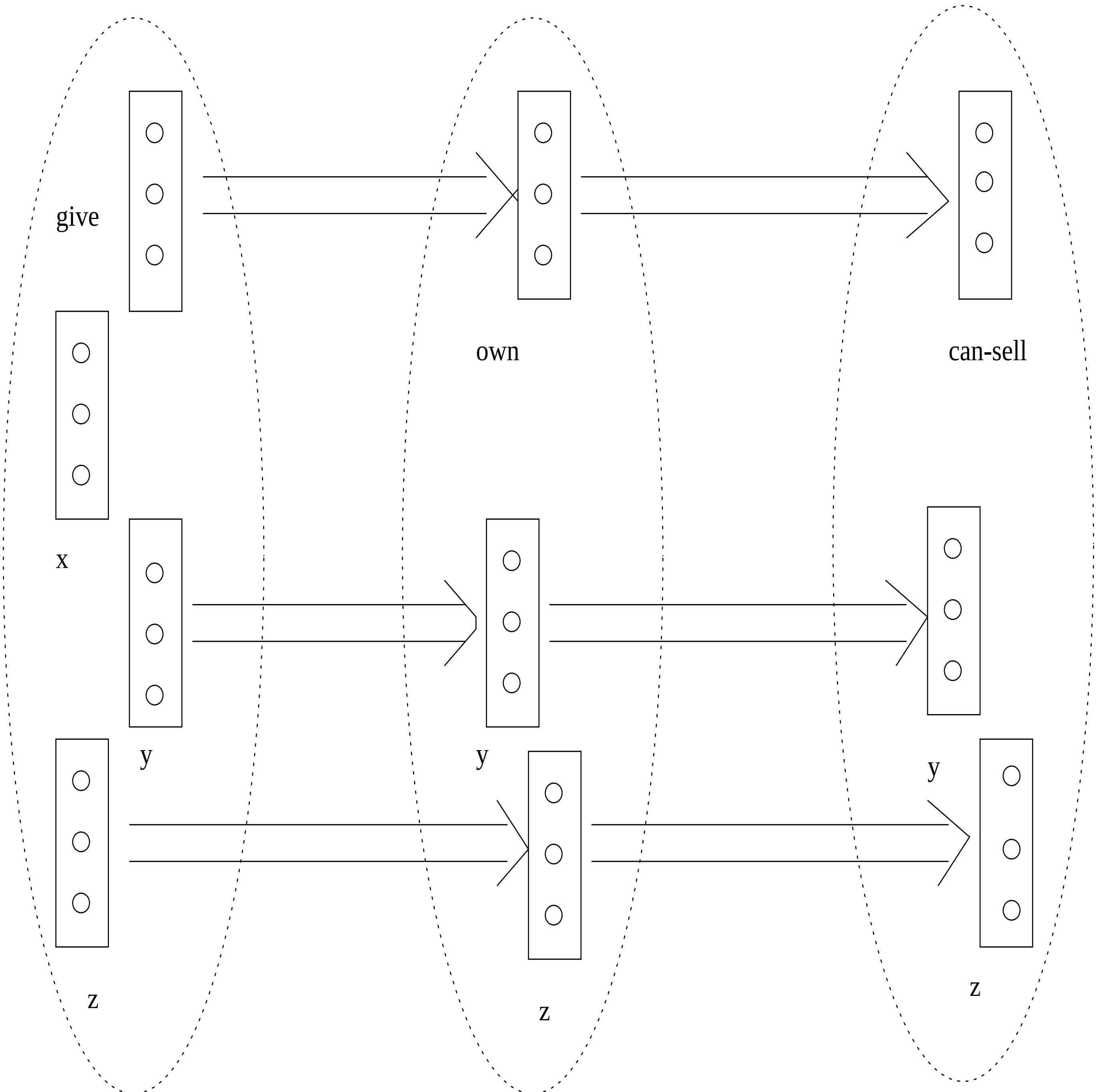


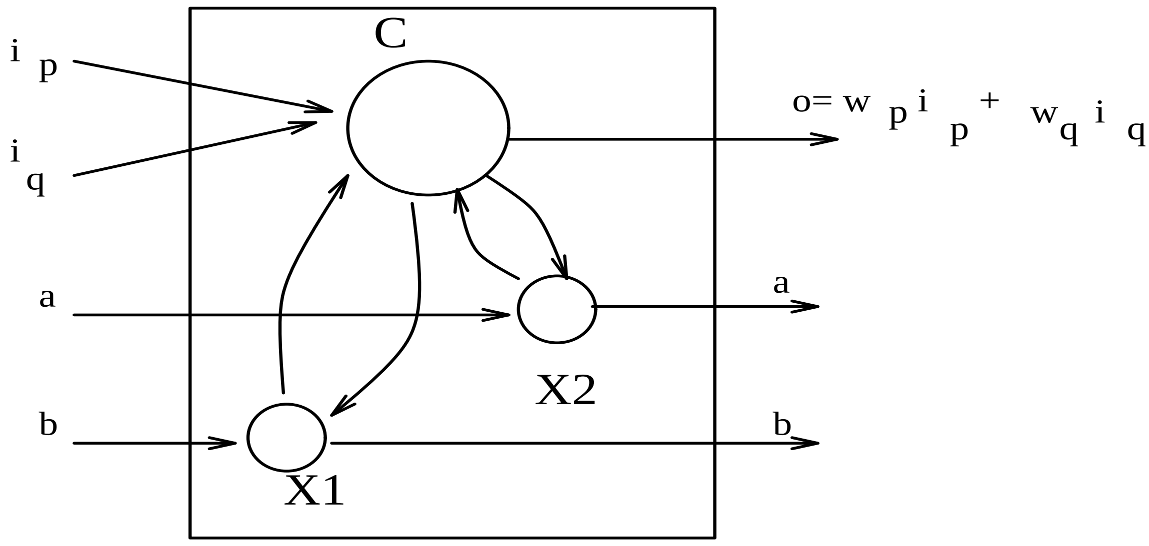
(1)



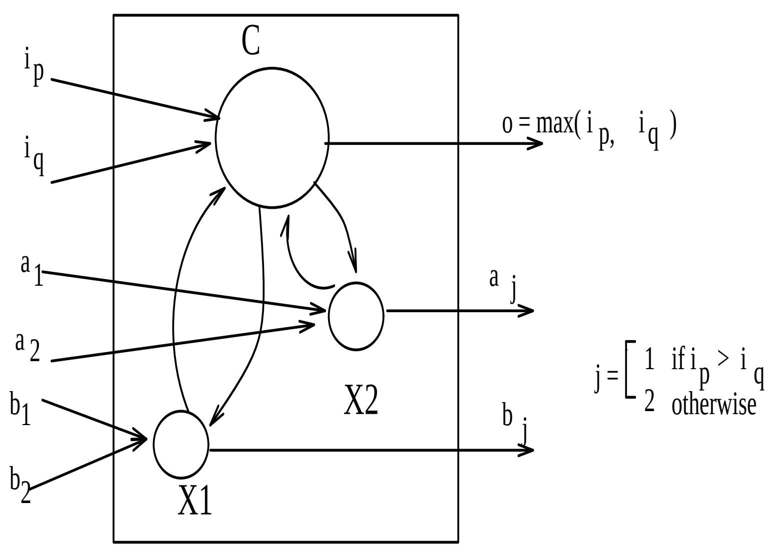
(2)


INPUT
a multi-layer
neural net
for state transitions
delay
a multi-layer
neural net
for actions
OUTPUT

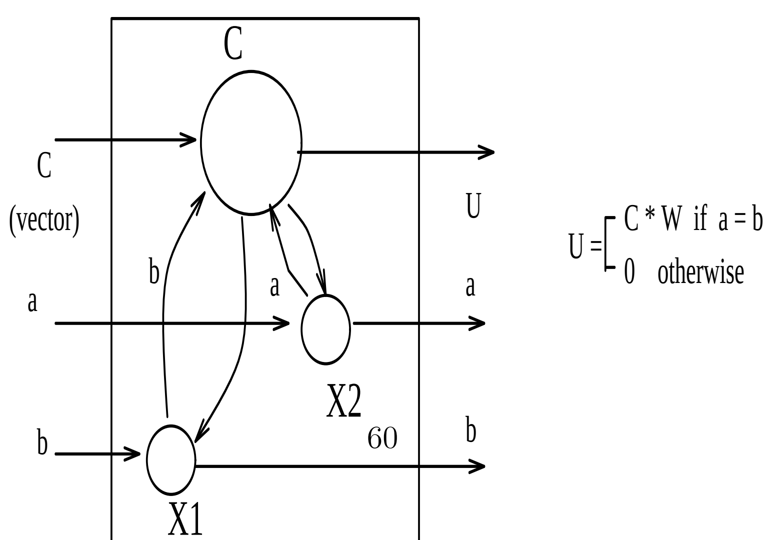

give
own
can-sell
x
y
y
y
z
z
z

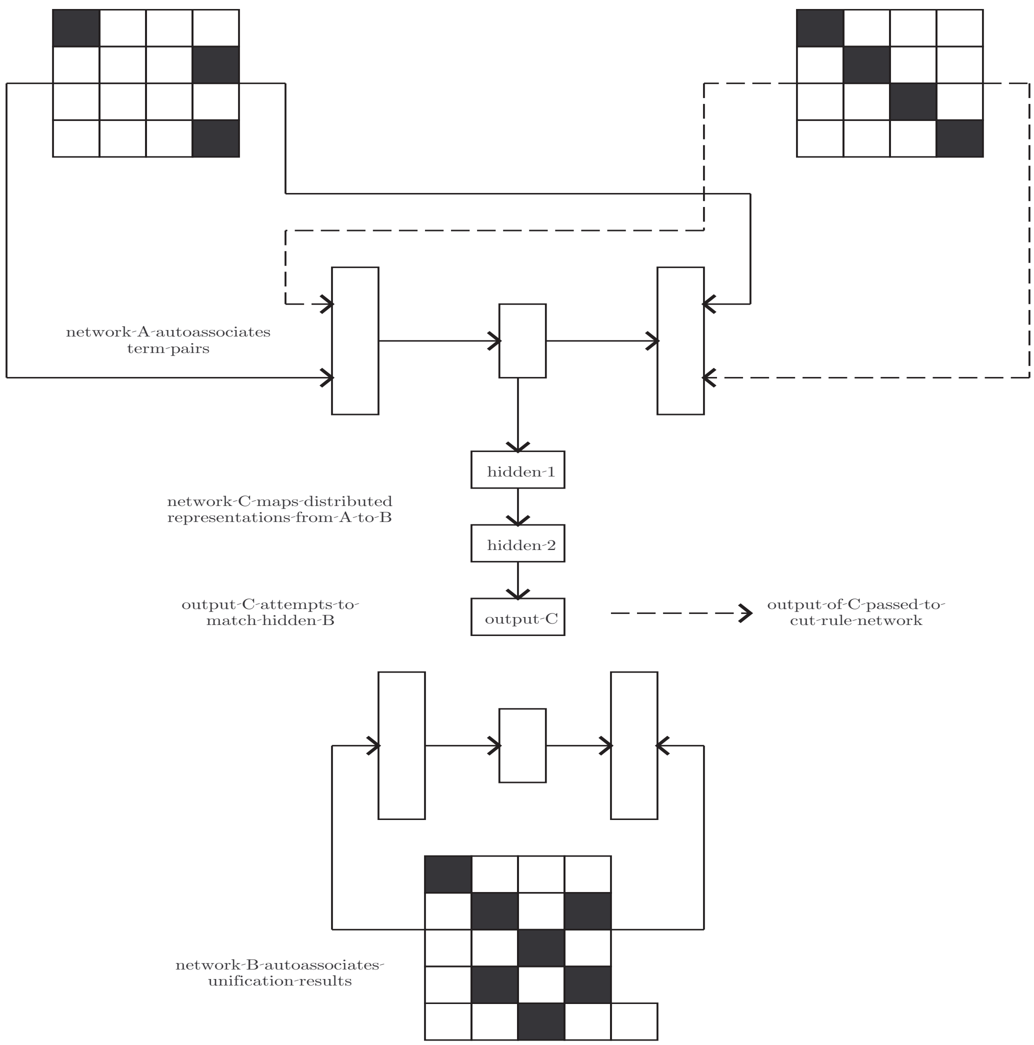

C
i p
i q
o= w p i p + w q i q
a
a
b
b
X2
X1

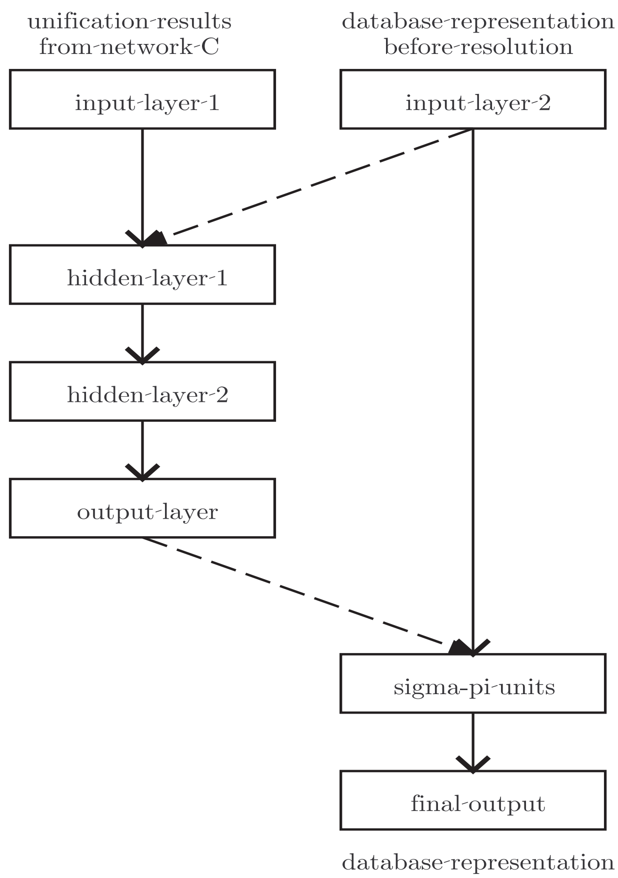
(1)


C
i p
i q
o = max(i p, i q)
a 1
a 2
a j
b 1
b 2
b j
X2
X1
j = 1 if i p > i q
2 otherwise

(2)


C
C
(vector)
U
U = C * W if a = b
0 otherwise
b
a
a
a
b
b
X2
X1


network-A-autoassociates
term-pairs
hidden-1
network-C-maps-distributed
representations-from-A-to-B
hidden-2
output-C-attempts-to-
match-hidden-B
output-C
output-of-C-passed-to-
cut-rule-network
network-B-autoassociates-
unification-results


unification-results
from-network-C
database-representation
before-resolution
input-layer-1
input-layer-2
hidden-layer-1
hidden-layer-2
output-layer
sigma-pi-units
final-output
database-representation