Reasoning with Limited Unification in A Connectionist Rule-Based System*

Nam Seog Park†, Dave Robertson†, Keith Stenning‡

†Dept. of Artificial Intelligence, University of Edinburgh, Scotland
namseog@aisb.ed.ac.uk, dr@aisb.ed.ac.uk

‡HCRC, University of Edinburgh, Scotland
keith@cogsci.ed.ac.uk

Abstract

At the intersection between symbolic inference and connectionism, there is interest in producing systems constructed from connectionist components which perform types of inference comparable to symbolic systems. The attraction of such systems is that, under restricted conditions, they may be capable of very fast, "reactive" responses to external stimuli. A major open question in this field is the extent to which a connectionist architecture can accommodate basic concepts of symbolic inference, such as unification of terms. One of the current leaders in this area is the rule-based system proposed by Shastri & Ajjanagadde (1993). We demonstrate that the mechanism for variable binding which they advocate is fundamentally limited and show how a reinterpretation of the primitive components of their system can extend the range of inference which can be supported.

1 Introduction

Two standard operations in theorem proving systems are matching of terms and substitution of the results of matching through the rest of a formula. Suppose we have the following rule encoded into a knowledge base:

$$\forall X, Y \;\; p(X, Y) \rightarrow q(X, Y)$$

To infer the proposition $q(a, b)$ from the given rule and the fact $p(a, b)$ firstly involves a match computation to verify that the given fact instantiates the antecedent of the rule. The output is a set of variable bindings: $match(p(X, Y), p(a, b)) =$

*This paper was presented at the postconference workshop on "Logic and Reasoning with Neural Networks" at the International Conference on Logic Programming ICLP'94, Santa Margherita Ligure, Italy June 13–18.

$\{X/a, Y/b\}$. After the match procedure, the bindings are used to substitute constants for variables in the consequent: $substitute(\{X/a, Y/b\}, q(X,Y)) = q(a,b)$. These match and substitute procedures constitute a mechanism for identifying and propagating variable bindings. In a conventional symbolic system, we have a free choice of algorithm for the match procedure. In a connectionist system, however, we are more constrainted because of the uniformity of its structure and the simplicity of its processing elements. Feldman (1982), Barnden(1984), Malsburg (1986), and Fodor & Pylyshyn (1988) describe the variable binding issue in connectionist systems in greater detail. As for the substitution procedure, unlike symbolic systems, most of connectionist systems obtain propagation of initial bindings from one group of nodes to the other through explicit links between them.

Although various connectionist systems have been proposed to achieve symbolic inference, for example: Touretzky & Hinton (1988), Dolan & Smolensky (1989), Lange & Dyer (1989), Smolensky (1990), Barnden & Srinivas (1991), Sun (1992), Shastri & Ajjanagadde (1993), only a few have dealt closely with knowledge representation issues such as pseudo bindings, binding generation, consistency checking, and unification. Specifically, the example system we consider here is the connectionist reasoning system developed by Shastri & Ajjanagadde [Shastri & Ajjanagadde (1991), Shastri& Ajjanagadde (1993)]. Their system encodes rules and facts involving n-ary predicates and performs an interesting class of inferences very quickly. A key component of their mechanism is the use of synchronous activity of "neurons" to represent dynamic bindings of variables. This is a powerful, but limited, technique we aim to extend.

In this paper, we only focus on the unification issue in Shastri & Ajjanagadde's (Henceforth S & A) system, which is important in enabling the system to achieve a broader range of symbolic processing capabilities. We first discuss the unification problem observed in their system. Then we propose how their mechanism can be modified and extended to get over the problem. Finally we will show how unification can be dealt with in the extended system. As a result, the modified system deals more comprehensively with unification, while retaining the merits of the original model such as an efficiency in inference and neural plausibility.

2 Unification and The Temporal Synchrony Approach

The form of rules allowed in S & A's system is Horn clause in the following form:

$$p_1(\ldots) \wedge p_2(\ldots) \wedge p_n(\ldots) \rightarrow q(\ldots)$$

S & A's system uses phase sensitive nodes of four types to encode such rules. Figure 1 depicts the temporal behaviours of these nodes. The number of phases in one period of oscillation is assumed to be 6 here. A τ-and node in the figure produces an oscillatory pulse whose period of oscillation and pulse width matches a period of

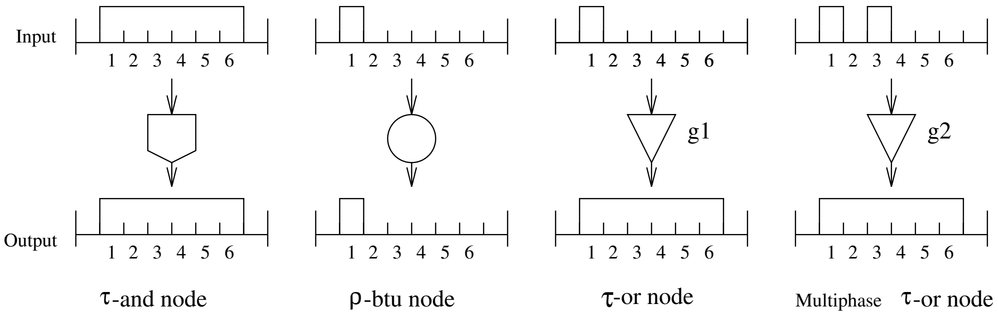

Figure 1: The temporal behaviour of phase-sensitive nodes

oscillation after it receives the same oscillatory input. However, a ρ-btu node, when receives a periodic spike train, produces a periodic spike train that is in phase with the driving input. The behaviour of the τ-or and *multiphase* τ-or node look alike in that they produce an oscillatory pulse train whose pulse width is comparable to the period of oscillation on receiving the periodic spike train, with the following small difference. A *multiphase* τ-or node becomes active only if it receives more than one spike trains in different phases within a period of oscillation as shown in the figure. On the other hand, τ-or node becomes active not only in the above situation but also when it receives one spike train from one of its input nodes within a period of oscillation. The behaviours of these nodes are differentiated based on the window of time over which they sample their inputs and the width of their output pulse. The width of phase, ω, and the period of oscillation cycle, π, are limited to those indicated by physiological measurements.

Suppose we have encoded the following rule in S & A's system:

$$p(X, Y, a) \rightarrow q(X, Y)$$

Note that this rule has the constant argument a in the antecedent. When the fact p(b,c,a) is presented to the rule to infer q(b,c), firstly consistency checking is required to see that the filler corresponding to the third argument of p (a in this case) is the same as a or not during the match procedure. On the other hand, when the fact p(b,c,U) is presented to the rule, unification between a and U is required.

Figure 2 depicts the network which encodes the rule above to show how S & A's system deals with each situation in more detail. In the figure, a group of nodes, {c1, e1, a1, a2, a3} represent the antecedent p, in which c1 and e1 nodes represent the *collector* and *enabler* and each a_i represents the ith argument. The consequent q is also represented in the same way. These two predicate assemblies with corresponding links represent the given rule.

When the fact p(b,c,a) is presented to the antecedent predicate of the network, S & A's mechanism presents dynamic bindings in the following way using activation synchrony: it activate and oscillate the constant node b and the *a1* node in the

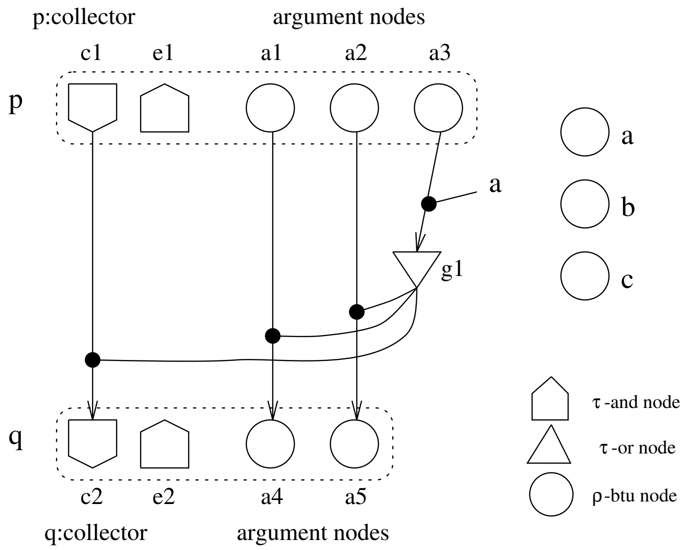

Figure 2: Consistency checking and unification on a constant argument

first phase, c and the *a2* in the second, and a and the *a3* node in the third phase. Thus in phase oscillation between argument nodes and filler nodes represent dynamic bindings: $\{a1/b, a2/c, a3/a\}$. To infer the right result, consistency checking and unification should be dealt with during inference. The *g1* node, which is of τ-or type, is introduced for this purpose. When the *a3* node gets bound to a constant (a real binding) other than a, the *g1* node will be active and project a inhibitory signal which prevents the rule from firing. So consistency of the rule is maintained. When *a3* gets bound to a variable (a pseudo binding), the *a3* node remains inactive during instantiation of inference and the *g1* node also remains inactive during inference. Thus activation of the nodes in the predicate p propagates to the corresponding nodes in the predicate q to produce the desired answer.

If we apply the same mechanism to the following rule, we have problems.

$$p(X,Y) \wedge q(Y,Z) \rightarrow r(X,Z)$$

Note that the rule has a repeated variable in the antecedent, which also appears in the consequent. When the system performs inference with this rule, the mechanism needs to force every occurrence of the repeated arguments in the antecedent to get bound to the same constant (consistency checking). In the same way, when one of repeated arguments gets bound to another variable, we need to keep track of the bindings between variables.

Figure 3 shows how the above rule is encoded in S & A' system. Again the *g1* node whose threshold is 2 is used to force consistency checking and unification during inference. One may work through the network to confirm that this produces

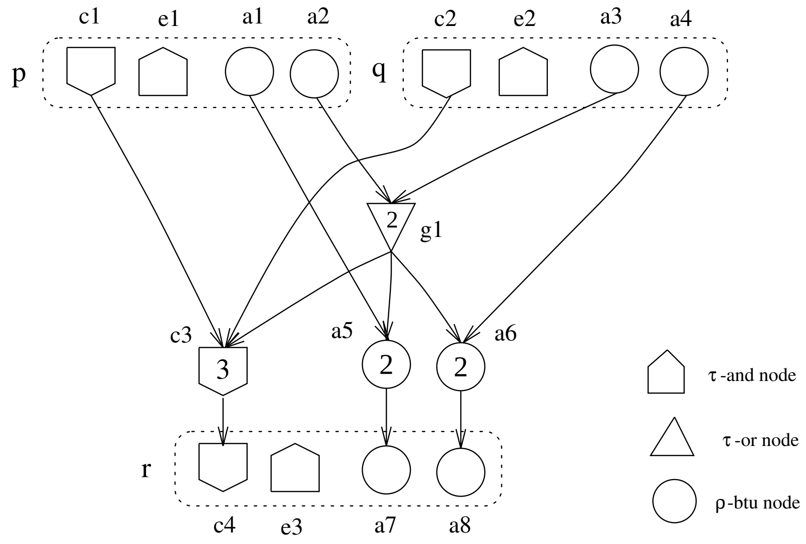

Figure 3: Consistency checking and unification over repeated arguments

the desired answer when the fact $p(a, b) \wedge q(b, c)$ is presented to the antecedent. However, when $p(a, b) \wedge q(U, c)$, S & A's unification algorithm breaks down. S & A's mechanism would allocate the *a2* node and the constant node b to the same phase but would leave the *a3* node inactive. This leads the *g1* node inactive and consequently the rule is not fired. See Figure 4 how propagation of phases is stopped after the *g1* node. Note that simply activating the *g1* node in the distinct phase is not a solution, since this makes it impossible to distinguish pseudo bindings from constant bindings.

Hence, although S & A's system provides a powerful mechanism for dynamic bindings it also has a fundamental limitation in dealing with unification, which prevents it from handling transitive inference with variables. In the next section, we show how S & A's mechanism can be modified and extended to cope with this problem using a more general, and simple, solution.

3 Modification of The Mechanism

From the symbolic point of view, a variable in a rule is considered something which can unify with a constant, a variable, or even in some languages a function. Implementing unification in a connectionist system is constrainted because of the uniformity of its structure and the simplicity of its processing elements. As discussed in the previous section, the limitation of S & A's system in dealing with unification is mainly due to their way of representing pseudo bindings. As long as a pseudo binding is represented simply by not providing any temporal phases, identifying bound variables is a problem. Therefore, we need to find a way of representing a pseudo variable binding and

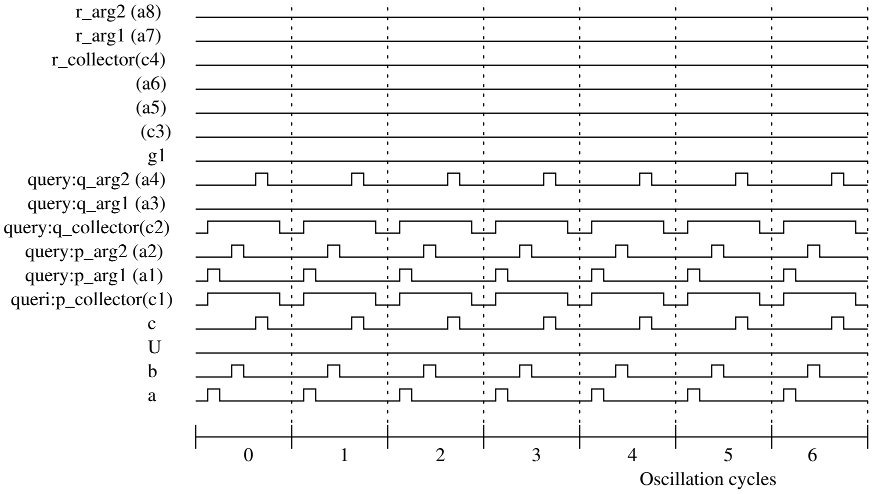

Figure 4: Phases propagation when consistency checking and unification is performed over repeated arguments

a mechanism which supports the desired inference based on synchronous activity of neurons.

One simple way of representing a pseudo binding may be allocating an unused phase for it. This, however, makes it impossible to distinguish a real binding from a pseudo binding because both of them will be represented by phases of the same type. What we are looking for is a representation by which a pseudo binding and a real binding can be differentiated and on which a basic mechanism can be built to deal with unification. An alternative way of representing it is representing a pseudo binding using special temporal label which is activated throughout an oscillation cycle. The additional advantage of this representation is that a real binding can be viewed as a subset of a pseudo binding when they are represented by phases.

Having adopted the second solution for representing a pseudo binding, we need to define another node which will use the special temporal label to represent pseudo bindings. This new node which we will call an ω-btu node will replace an existing ρ-btu node. Figure 5 depicts the behaviour of an ω-btu node. When the ω-btu node gets bound to a constant, its behaviour is the same as that of the ρ-btu node. But, as can be seen, the way of representing a pseudo binding is different. A pseudo binding is now represented by an oscillatory pulse whose period of oscillation and pulse width matches the sum of all phases in a cycle.

4 Dealing with Unification Using an ω-btu Node

Let us reconsider our problem example:

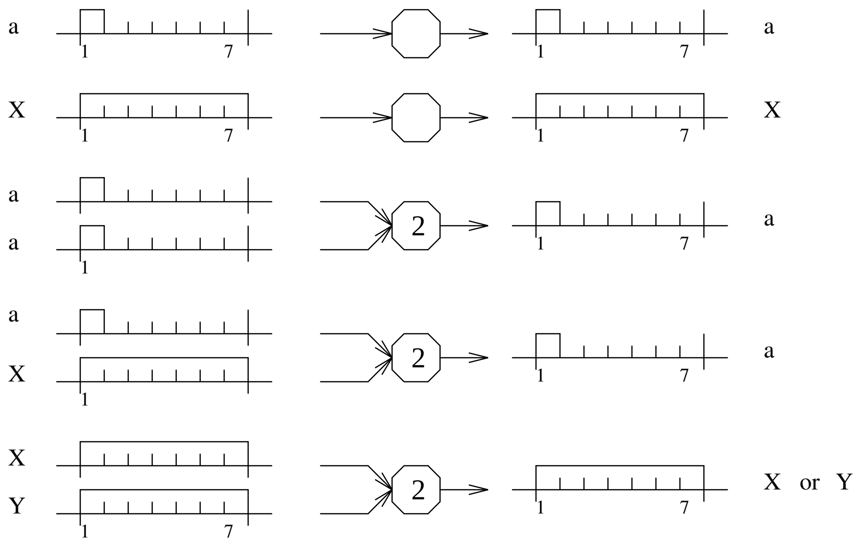

Figure 5: Temporal behaviour of an ω-btu node

$$p(X,Y) \wedge q(Y,Z) \rightarrow r(X,Z)$$

Encoding these rules using ω-btu node simply could be done by replacing ρ-btu node with ω-btu node from the network in Figure 2. In practice, as shown in Figure 6, two layers of nodes have been added between the antecedent and the consequent of the rule because an ω-btu node affects binding propagation with different temporal behaviour than a ρ-btu node. These layers are used to deal with not only the unification problem but also other knowledge representation issues such as binding generation. More detailed descriptions of these can be found in Park & et. al. (1993).

In the modified mechanism, unification is naturally carried out by the behaviour of the corresponding ω-btu node in the acknowledgement layer. As shown in the figure, the outputs of repeated arguments are connected to the same ω-btu node, whose threshold is 2. This node becomes active on receiving two the same constant inputs, or two variable inputs, or one variable and the other constant input. For example, the rule fires when $p(a,b) \wedge q(b,c)$ is presented since the argument nodes corresponding to the variable Y get bound to the same constant b. The rule also fires when $p(a,b) \wedge q(U,c)$ or $p(a,U) \wedge q(V,c)$ are presented to the network. However, the rule does not fire when $p(a,b) \wedge q(e,c)$ is presented. Therefore, unification and consistency checking is maintained at the same time during inference.

Figure 7 depicts how this unification procedure is recognised by the modified mechanism based on synchronous activity of nodes over the network.

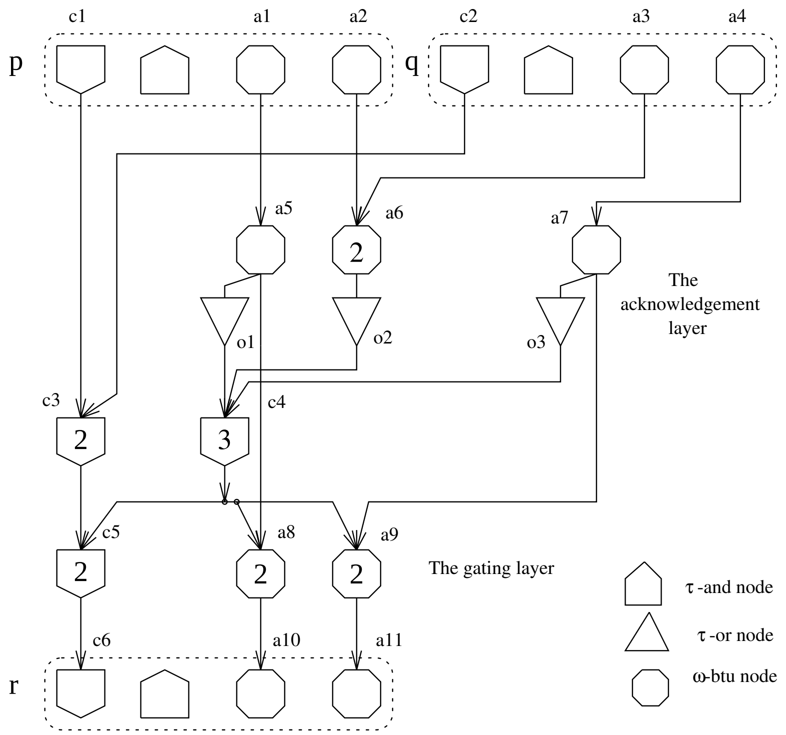

Figure 6: Consistency checking and unification using ω-btu nodes

5 Discussion

This paper has shown that how S & A's temporal synchrony approach to the dynamic variable binding can be modified to deal with unification. As we have seen, the modified system now can deal with unification within each repeated argument based on synchronous activation of nodes while retaining simple network components similar to those of the original system.

There have been some other connectionist systems which have offered solutions to this issue. Touretzky & Hinton (1985) can deal with unification but only with very restricted forms of rules. Sun (1992) has also suggested a possible solution to the unification problem. The major difference between Sun's approach and our extension of S & A's approach lies in the functional requirements of nodes in a network. S & A's system uses only one type of assembly nodes which consist of a τ-and node and ρ-btu node which was replaced by ω-btu node. The behaviour of the assembly is quite transparent such that each node in the assembly performs the same type of simple processing, weighted sums of inputs and thresholding. Sun's approach, however, uses three different types of assemblies and unification is performed by what Sun calls a Complex assembly. Although Sun's system does not use temporal phases, it imposes

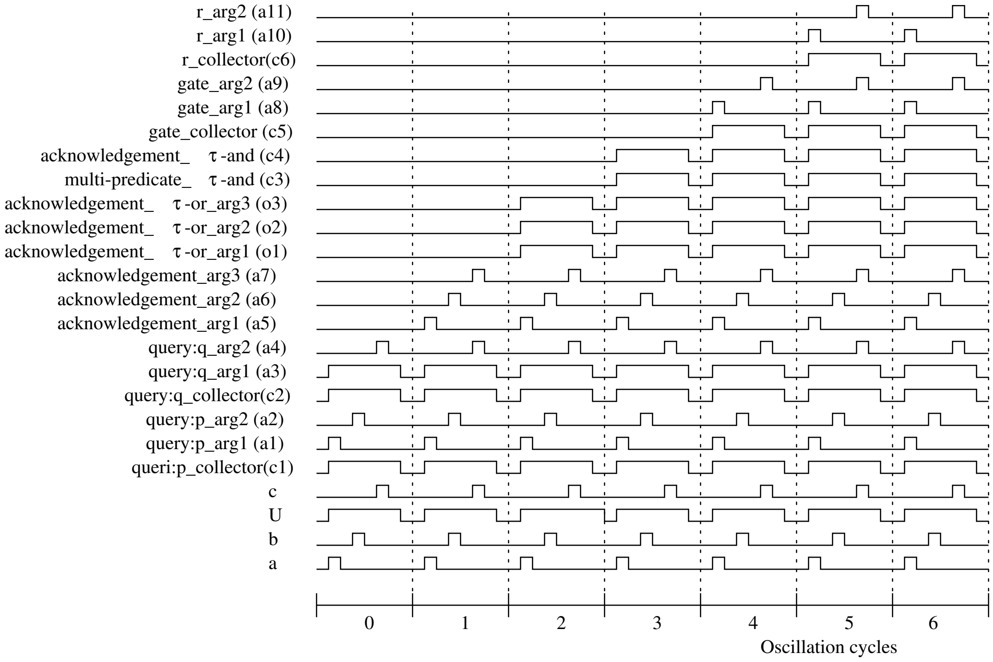

Figure 7: Propagation of phases on the network when the fact $p(a, b) \wedge q(U, c)$ is presented

a higher computational requirement on each node in the assembly, beyond simple summation with thresholding. For example, when Sun's system performs inference on the following rule:

$$p(X, Y) \wedge q(Y, Z) \rightarrow r(X)$$

the predicate node of the Complex assembly of the predicate r must perform the following computation in order to carry out unification properly.

> Suppose variables are represented by values greater than a certain threshold θ; if the presented value for argument Y in the predicate p equals to the presented value for argument Y in the predicate q (when both values are constant), or if the presented value for Y in the predicate p is greater than θ (when the first Y gets bound to a variable), or if the presented value for Y in the predicate q is greater than θ (when the second Y gets bound to a variable), then they are unifiable.

Although Sun's mechanism can deal with the unification during inference, the need for high computational ability in the assemblies and the use of abstraction which hides network details seems neurally implausible and may impose a high computational overhead.

In our extension of S & A's system, a small change has been made in order to maintain transparent behaviour of the original system. This change has provided a way of representing a group of unifying variables, even when it is not known to which individual those variables are bound. We represent such group of arguments by tagging them all with the same specific temporal phase which is *active throughout a cycle*. Sometimes, however, it may be necessary to represent many groups of such arguments in a rule, i.e. unification across different argument groups. For example, we may want to obtain $q(a,a)$ by presenting the fact p(a,U,U) to the following rule:

$$p(X,X,Y) \rightarrow q(X,Y)$$

Here the variable filler U in the fact appears across different argument positions, X and Y, in the antecedent. So the result of unification, {U/a}, obtained from the repeated argument X should be used during the match procedure to produce the bindings, {X/a,Y/a}. These bindings will then propagate to the consequent of the rule to produce the desired result. However, even the extended mechanism described in this paper will produce $q(a,U)$ in this situation. This is because all pseudo bindings in the extended system are now represented using the same temporal phase which is active throughout an oscillation cycle and this will not differentiate the bindings {X/U} and {Y/V}. To get over this, we need to tag different pseudo bindings with different phases. This further extension can be made by changing the structure of an argument node: instead of using a single ω-btu node, a pair of ω-btu nodes can be used to represent an argument. For example, to represent argument bindings {X/U,Y/V}, we need two pairs of ω-btu nodes, one for X and on for Y. And the bindings will be represented by activating the left argument nodes of X and Y throughout an oscillation cycle (to represent pseudo bindings) and the right argument node of X in the first phase and the right argument node of Y in the second (to tag each pseudo binding). Additional nodes and links will be required for the propagation of these bindings to the consequent of the rule.

As can be seen, S & A's system has the fundamental problem which can not identify bindings between variables. And this eventually limits their system in dealing with the important knowledge representation issue, unification. The extension of S & A's system and further extension proposed in this paper provides a mechanism which translate a significant subset of Horn-clause FOPL into a connectionist representation which may be executed very quickly. However, in order to have the full expression power of Horn clause FOPL, we need to add the ability to represent recursion in rules and negation of predicates. Currently, no connectionist system has provided a convincing solution to implementing recursion in rules.

References

[Barnden 1984] J. Barnden. On short-term information processing in connectionist theories. *Cognition and Brain Theory*, 7(1):25–59.

[**Barnden & Srinivas 1991**] J. Barnden and K. Srinivas. Encoding techniques for complex information structures in connectionist systems. *Connection Science*, 3(2):269–315.

[**Dolan & Smolensky 1989**] C. P. Dolan and P. Smolensky. Tensor product production system: A modular architecture and representation. *Connection Science*, 1.

[**Feldman 1982**] J. A. Feldman. Dynamic connections in neural networks. *Bio-Cybernetics*, 46:27–39.

[**Fodor & Pylyshyn 1988**] J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: a critical analysis. In S. Pinkerr & J. Mehler (Eds), *Connection and Symbols*, Cambridge, Mass., MIT Press.

[**Lange & Dyer 1989**] T. E. Lange and M. G. Dyer. High–level inferencing in a connectionist network. *Connection Science*, 1(2).

[**Malsburg 1986**] C. von der Malsburg. Am I thinking assemblies? In G. Palm & A. Aertsen (Eds), *Brain Theory*, Springer-Verlag.

[**Park & et. al. 1993**] N. S. Park, D. Robertson, and K. Stenning. An extension of the temporal synchrony solution to dynamic variable bindings in a connectionist system. RP93-666, Department of Artificial Intelligence, The University of Edinburgh.

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

[**Shastri & Ajjanagadde 1993**] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16(4).

[**Smolensky 1990**] P. Smolensky. Tensor product variable binding and the representation of symbolic structure in connectionist systems. *Artificial Intelligence*, 46(1):159–216.

[**Sun 1992**] R. Sun 1992. On variable binding in connectionist networks. *Connection Science*, 4(2):93–124.

[**Touretzky & Hinton 1988**] D. S. Touretzky and G. E. Hinton. A distributed connectionist production system. *Cognitive Science*, 12(3).