INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Exploiting temporal binding to learn relational rules within a connectionist network

Lokendra Shastri*

TR-97-003

June 1997

Abstract

Rules encoded by traditional rule-based systems are brittle and inflexible because it is difficult to specify the precise conditions under which a rule should fire. If the conditions are made too specific a rule does not always fire when it should. If the conditions are made too general, the rule fires even when it should not. In contrast, connectionist networks are considered to be capable of learning soft and robust rules. Work in connectionist learning however, has focused primarily on classification and feature formation, and the problem of learning rules involving *relations* and *roles* (variables) has received relatively little attention. We present a simple demonstration of rule learning involving relations and variables within a connectionist network. The network learns the appropriate correspondence between roles of antecedent and consequent relations as well as the features that role fillers must possess for a rule to be applicable in a given situation. Each rule can be viewed as a mapping from the symbolic level to the symbolic level *mediated* by a *semantic filter* embedded within a subsymbolic level. The network uses synchronous firing of nodes to express dynamic bindings.

*This work was partially funded by ONR grants N00014-93-1-1149 and N00014-95-C-0182. Thanks to the L0 group for providing intellectual stimulation. *E-mail:* shastri@icsi.berkeley.edu

1 Introduction

A key limitation of the traditional rule-based approach is that rules are brittle and inflexible. When modeling complex domains, knowledge engineers find it difficult to specify the precise set of conditions under which a rule should fire. It they make the conditions too specific a rule does not always fire when it should. If they make the condition too general, the rule fires even when it should not. This is the well known qualification problem in AI.

With the introduction of powerful learning algorithms and high-performance machines, researchers have turned toward connectionist or neural network models that can learn appropriate "rules" from examples.[1] The ability to learn soft, fuzzy but robust rules is considered a strength of connectionist models and there exist several interesting demonstrations of this in the area of visual pattern recognition, speech recognition, and control systems.

Applications of connectionist learning, however, have focused primarily on classification and feature formation, and the learning of relational-rules, i.e., rules involving relations (predicates) and roles (variables), has received relatively little attention. An example of a relational rule is:

walk-into(x,y) = $(\alpha(x,y)) \Rightarrow$ *hurt(x)* ——- (1)

The above can be paraphrased as "If one walks into something, one gets hurt. Furthermore, the degree of hurt (i.e., the strength of the above rule-firing) in a given situation is determined by α which is a (potentially complex) function of the features of the actual role-fillers x and y in a given situation. For example, α may depend on, among other things, the hardness, the shape, and the relative size and weight of y. A brittle version of the above rule with α encoded as an additional condition in the antecedent might be:

walk-into(x,y) $\wedge$ *solid(y)* $\wedge$ *(mass(y)* = *heavy)* $\Rightarrow$ *hurt(x)*

Learning a relational rule such as (1) involves two subproblems. First, one must learn the correct correspondence between the roles of the relations *walk-into* and *hurt*. Thus one must learn that the first role of *walk-into* maps to the role of hurt. Second, one must learn the appropriate form of α.

1.1 Background

Over the past few years we have been engaged in developing SHRUTI, a connectionist model of rapid reasoning (see Ajjanagadde & Shastri, 1991; Shastri & Ajjanagadde, 1993; Mani & Shastri, 1993; Shastri & Grannes, 1996). This work attempts to answer the following question: How can a system of simple and slow neuron-like elements represent a large body of specific facts as well as general context-sensitive rules and perform a broad class of inferences within hundreds of msecs? SHRUTI presents a partial solution to this problem and demonstrates how it is possible to encode facts and rules involving n-ary relations (predicates), limited quantification, and concept hierarchies and perform a limited class of reasoning with requisite efficiency. As discussed in (Shastri & Ajjanagadde, 1993) reasoning involving n-ary predicates requires a solution to the dynamic binding problem (Feldman 1982; Malsburg 1986). SHRUTI incorporates a neurally plausible solution to this problem. It represents dynamic bindings between a role and a filler by the *synchronous* firing of

[1] At this point let we are ignoring the problem of extracting or identifying rules learned by a network in the course of solving a problem.

the appropriate role and filler nodes. There is growing neurophysiological evidence that synchronous activity may play an important role in neural information processing (e.g., Singer & Gray, 1995). The use of temporal synchrony for representing dynamic bindings points to a number of interesting constraints on the nature of rapid (reflexive) processing.[2]

The significance of the representational and inferential mechanisms developed in SHRUTI extends beyond reasoning. SHRUTI demonstrates how connectionist networks can represent relational structures in a *dynamic* manner and perform certain types of systematic computations over such structures in an efficient manner. In general, these relational structures can be viewed as schemas or frames and rules may be thought of as mappings between schemas or frames. Furthermore, these mappings (rules) can be deductive as well as evidential.

This report describes how a SHRUTI-like system can learn context sensitive rules involving n-ary predicates and variables from examples and observations. The solution allows a connectionist system to learn the mapping between roles of antecedent and consequent predicates and the combination of features that role fillers must possess for a rule to be applicable in a given situation. For example, the solution allows the system to learn rules such as "if the agent x of *walk-into* has features 'animate' and 'solid', and the patient y of *walk-into* has the feature 'solid' then: *walk-into(x,y)* $\Rightarrow$ *hurt(x)*".

Predicates, roles, and fillers in SHRUTI are part of the symbolic level of representation and rules are mapping from the symbolic level back to the symbolic level. These mappings, however, are *mediated* by *semantic filters* that are sensitive to the features and attributes of role-fillers in a given situation. These filters may be viewed as forming a subsymbolic level of representation mediating interaction between symbolic representations. The primary burden of learning is to acquire the relevant semantic filters via observations.

1.2 Overview of SHRUTI

We provide a brief overview of how predicates and rules are encoded in SHRUTI, how dynamic facts are represented using temporal synchrony, and how rule-firing occurs via propagation of synchronous activation. Only aspects of SHRUTI relevant to the current work are presented. Other details may be found in (Shastri & Ajjanagadde 1993). SHRUTI encodes an n-ary predicate as a cluster of nodes which includes n role nodes (these are the circular 'nodes' shown in Figure 1, the rectangular nodes are not relevant to our discussion). Nodes such as *John* and *Mary* correspond to *focal* nodes of a more complex representation of the individual concepts 'John' and 'Mary'. A rule is encoded by linking the roles of the antecedent and consequent predicates in accordance with the correspondence between roles specified in the rule. For example, the rule *give(x,y,z)* $\Rightarrow$ *own(y,z)* is encoded by connecting the roles *recip* and *g-obj* of *give* to the roles *owner* and *o-obj* of *own*, respectively. SHRUTI represents dynamic bindings using *synchronous* firing of the appropriate role and concept nodes. For example, the dynamic fact *give(John,Mary,Book1)* is represented by the *rhythmic* pattern of activity shown in Figure 2a. By virtue of the interconnections between role nodes of the predicates *give*, *own*, and *can-sell*, the state of activation described by the pattern shown in Figure 2a leads to the activation pattern shown in Figure 2b where the firing pattern of nodes corresponds to the dynamic facts *give(John,Mary,Book1)*, *own(Mary,Book1)*, and

[2]The model shares a number of features with ROBIN (Lange & Dyer 1989) which uses *signatures* instead of temporal synchrony to solve the dynamic binding problem.

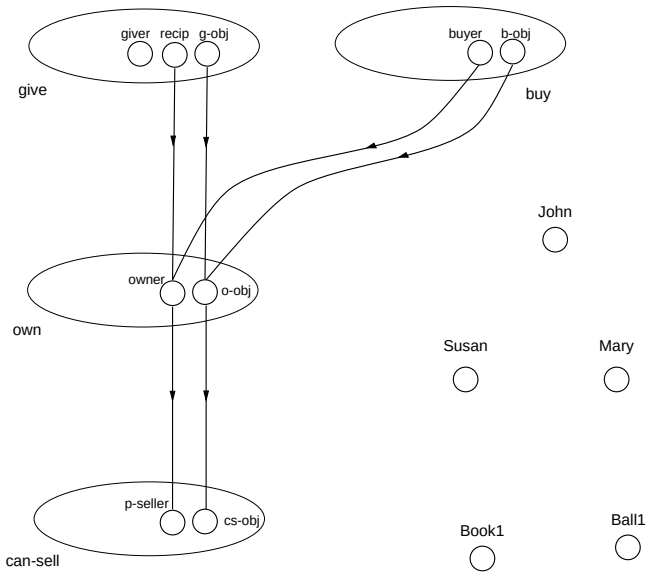

Figure 1: Encoding of predicates, individual concepts, and the rules: $\forall x, y, z\ [give(x, y, z) \Rightarrow own(y, z)]$, $\forall x, y\ [own(x, y) \Rightarrow can\text{-}sell(x, y)]$, and $\forall x, y\ [buy(x, y) \Rightarrow own(x, y)]$. Links between arguments reflect the correspondence between arguments in the antecedents and consequents of rules.

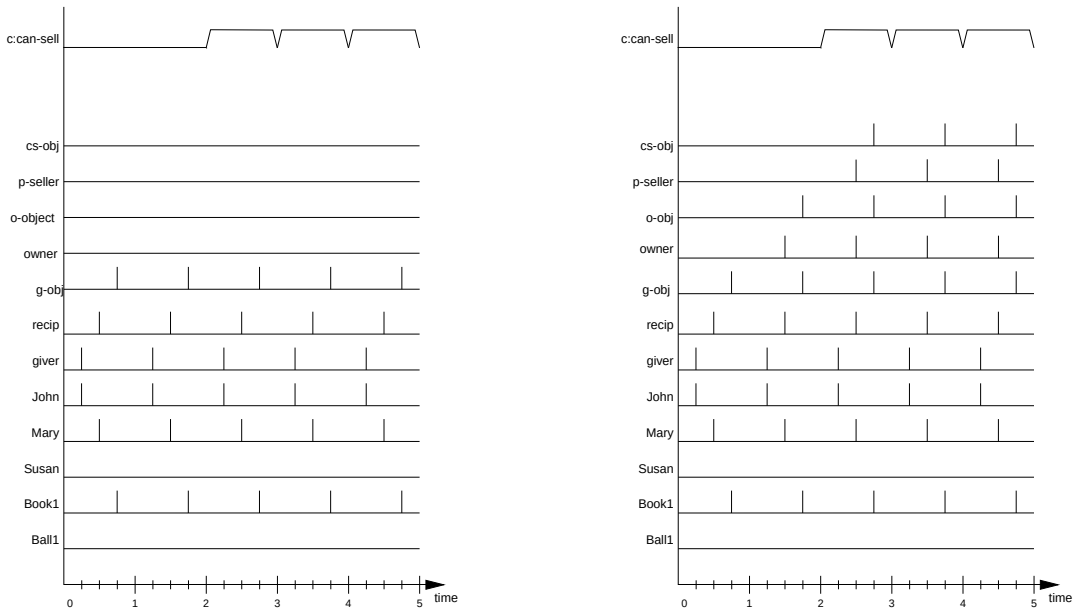

Figure 2: (a) Initial pattern of activation representing the bindings *(giver=John, recipient=Mary, give-object=Book1)*. (b) Pattern of activation after a few cycles. This pattern represents the additional dynamic bindings: *(owner=Mary, own-object=Book1, potential-seller=Mary, can-sell-object=Book1)*. The network has essentially inferred the facts *own(Mary, Book1)* and *can-sell(Mary, Book1)*.

can-sell(Mary,Book1). The key assumption here is that role nodes are ρ-btu nodes with the following idealized behavior: if there is a link from a ρ-btu node A to a ρ-btu node B, the firing of A leads to a synchronous firing of B.

SHRUTI can also represent long-term facts and a concept hierarchy. The latter allows categories as well as instances (individuals) to occur in rules, facts, and queries. SHRUTI can also encode multiple (but a small number of) dynamic instantiations of each predicate and concept. This enables it to deal with reasoning involving 'bounded recursion'. The time taken by SHRUTI to draw an inference is only proportional to the length of the chain of inference and is otherwise independent of the size of the knowledge base.

The representational choices made in SHRUTI together with the use of temporal synchrony for encoding dynamic bindings lead to predictions about the capacity of the *working memory* underlying rapid processing. For example, SHRUTI predicts that a large number of facts may be co-active and a large number of rules may fire simultaneously as long as: the maximum number of distinct entities that can occur as role-fillers in the dynamic facts is small (< 10). These constraints have implications for other rapid processing phenomena besides reasoning. For example, Henderson (1993) has shown that these constraints explain several properties of human parsing.

2 Learning rules from observations

To investigate rule-learning in SHRUTI we consider the forward reasoning version of SHRUTI and focus on role nodes within predicates clusters. As explained in Section 1.1, interconnections between role nodes play a crucial role in the encoding of rules and the propagation of bindings during rule-firing. Consider the simple network fragment depicted in Figure 3. This network includes two predicates: walk-into and hurt and a representation of types/feature in the domain. Encoding a context-sensitive rule such as: "if the agent x of *walk*-into has the features 'animate' and 'solid', and the patient of *walk-into* has the feature 'solid', then *walk-into(x,y)* $\Rightarrow$ *hurt(x)*" amounts to encoding a suitable "filter" between *walk-into* and *hurt* which would allow activation from the agent of *walk-into* to propagate to the patient of *hurt* whenever the dynamic role-fillers of *walk-into* have the appropriate features. In (Shastri & Ajjanagadde 1993) it was shown that such filters can be hard-wired using ρ-btu and τ-or nodes if the restrictions on the role fillers are known. In the present work we demonstrate that a SHRUTI-like system can automatically learn such filters from observations.

Encoding of observations

Given that dynamic bindings are encoded in SHRUTI using temporal synchrony, the encoding of an event is a time-varying pattern of activity. Figure 4 depicts the time varying activity corresponding to the event "John walked into the wall". Note that the salient feature nodes of John are active and firing in synchrony with the agent of *walk-into* and the salient feature nodes of wall are firing in synchrony with the patient of *walk-into*. The activity in Figure 5 depicts the joint activity for "John walked into a wall" and "John got hurt".

Rule learning is expected to occur as a result of observing a number of examples such as the one above. An observation consists of the network being shown the activity in Figure 4 as an *initial* pattern of activation, and the activity in Figure 5 as the *final* pattern of

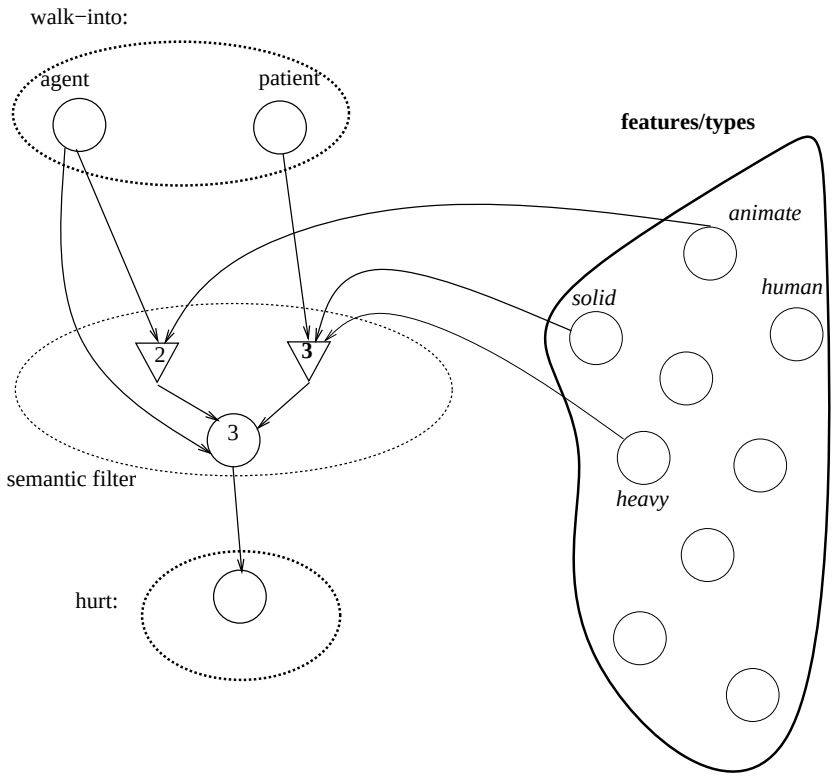

Figure 3: An example of a semantic filter for the rule: *x:animate* $\wedge$ *y:solid* $\wedge$ *y:heavy walk-into(x,y)* $\Rightarrow$ *hurt(x)*. The circular nodes are ρ-btu nodes and the triangular nodes are τ-or nodes. To make matters simple, let us assume that (i) a ρ-btu node with a threshold of k becomes active upon receiving k synchronous inputs and starts firing in synchrony with its inputs and (ii) a τ-or node with a threshold of k becomes active upon receiving k synchronous inputs and produces an uninterrupted burst of firing lasting π_{max}. The two τ-or together with the ρ-btu node in the semantic filter enforce the semantic constraint. The τ-or node on the left has a threshold of two and becomes active if the *agent* role of *walk-into* and the feature node *animate* fire in synchrony. In other words, it detects that the role-filler of *agent* is *animate*. Similarly, the τ-or node on the right detects that the role-filler of *patient* is *solid* and *heavy*. The ρ-btu node has a threshold of three and fires if both the τ-or nodes in the semantic filter become active along with the *agent* role of *walk-into*. Upon becoming active, it fires in synchrony with the *agent* role, since it receives three inputs only in the phase in which *agent* is firing.

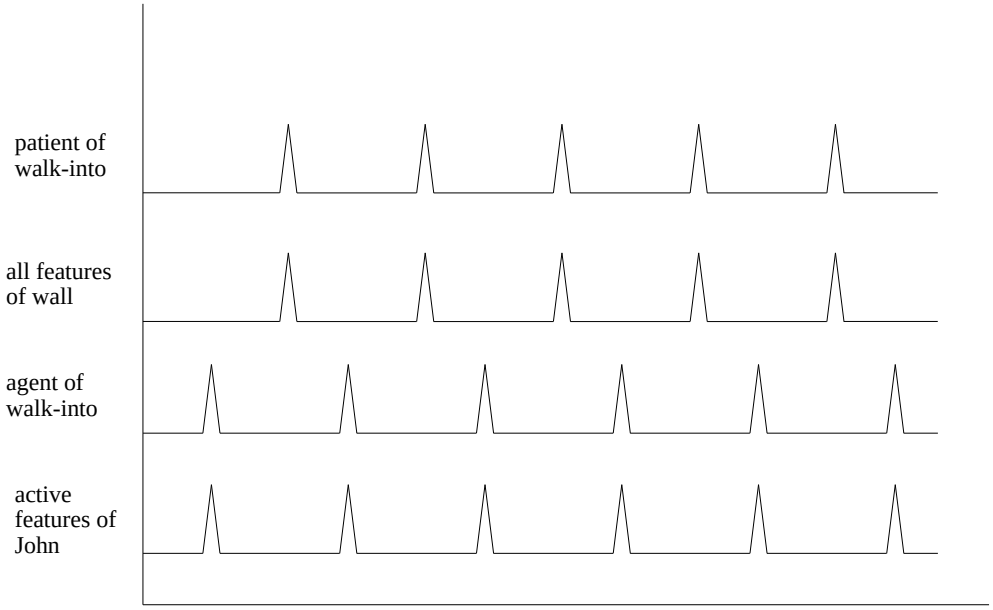

Figure 4: Pattern of activation depicting the "John walked into a wall".

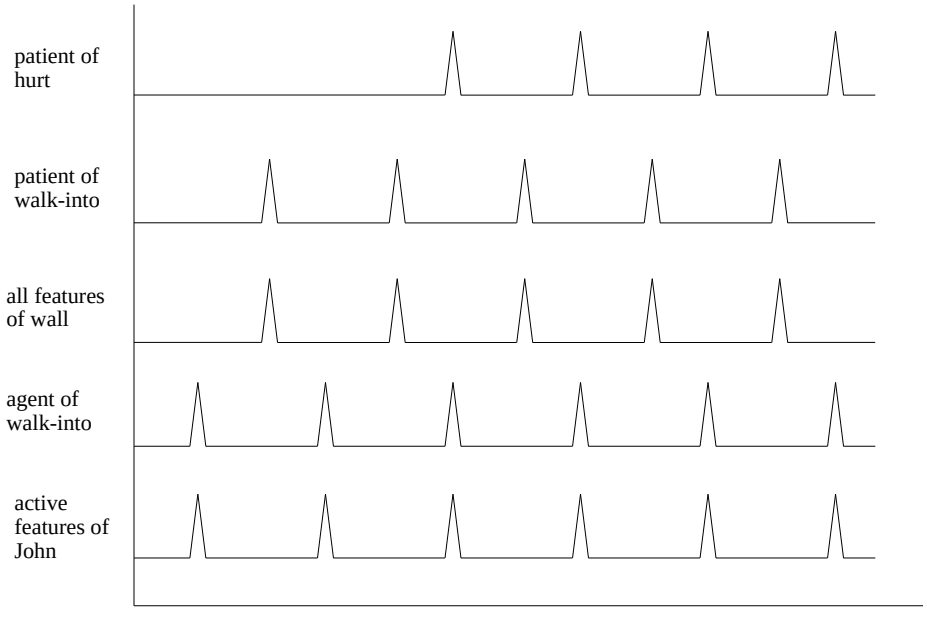

Figure 5: Pattern of activation depicting "John walked into a wall" and "John got hurt".

activation. This would convey to the network that when John walked into a wall, he got hurt. As a neutral example, the network may be shown the initial pattern for "Susan walked into the mist" followed by a final pattern which is the same as the initial pattern. This is intended to convey to the network that Susan's walking into the mist, did not have any consequences (at least in this limited domain). Based on such observations the network is expected to learn the systematicity in the domain and produce appropriate *final* patterns in response to given *initial* patterns.

Learning in SHRUTI requires spatio-temporal networks

In the usual case of supervised learning, each input and output pair corresponds to a fixed pattern of activity over the input and output nodes. In other words, the presentation of each input-output pair consists of a temporally *static* pattern of activity over input and output nodes. However, since SHRUTI uses temporal synchrony to encode dynamic bindings, the input and output patterns for a given event correspond to *time-varying* patterns of activity. We deal with this problem by mapping SHRUTI onto spatio-temporal networks that can explicitly represent temporal information. One such model is the Temporal Flow Model (TFM) (Watrous & Shastri, 1987) that has been applied to a number of problems in speech recognition Watrous (1990). TFM supports arbitrary link connectivity across layers, admits recurrent links, and allows variable propagation delays to be associated with links. The use of multiple links with variable delays allows the system to maintain context over a window of time and thereby carry out spatio-temporal feature detection and pattern matching.

3 Network structure

An important step in formulating the relational rule learning problem is the recognition that an inferential network such as the one in Figure 3 can be viewed as a network made up of three groups of nodes: one consisting of role nodes, another consisting of the various type/feature nodes, and a third consisting of a "hidden structure" within which the semantic filters for various rules are embedded and acquired as a result of learning. Such a conceptualization of a SHRUTI-like inference net is shown in Figure 6. Note that the network in Figure 6 may be thought of as a *fan-folded* version of an inferential network like the one in Figure 4, wherein all the role nodes fall at one end and all the semantic filters fall at the other end. The latter make up the hidden structure. As discussed in (Shastri & Ajjanagadde, 1993), the activation of an entity can be viewed as the activation of its handle node and other nodes representing its features and types. In order to concentrate on the problem of learning rules and semantic filters, we flatten the type hierarchy and treat is as a collection of feature nodes. This blurs the distinction between features that are directly associated with a concept and those that are associated indirectly via super-ordinate concepts. Though this distinction is important in general, it is not central to our present objective.

Observe that the nodes encoding predicates, roles, and fillers in SHRUTI can be viewed as being part of the symbolic level of representation. Each context sensitive rule can therefore be viewed as a mapping from the symbolic level to the symbolic level that is *mediated* by

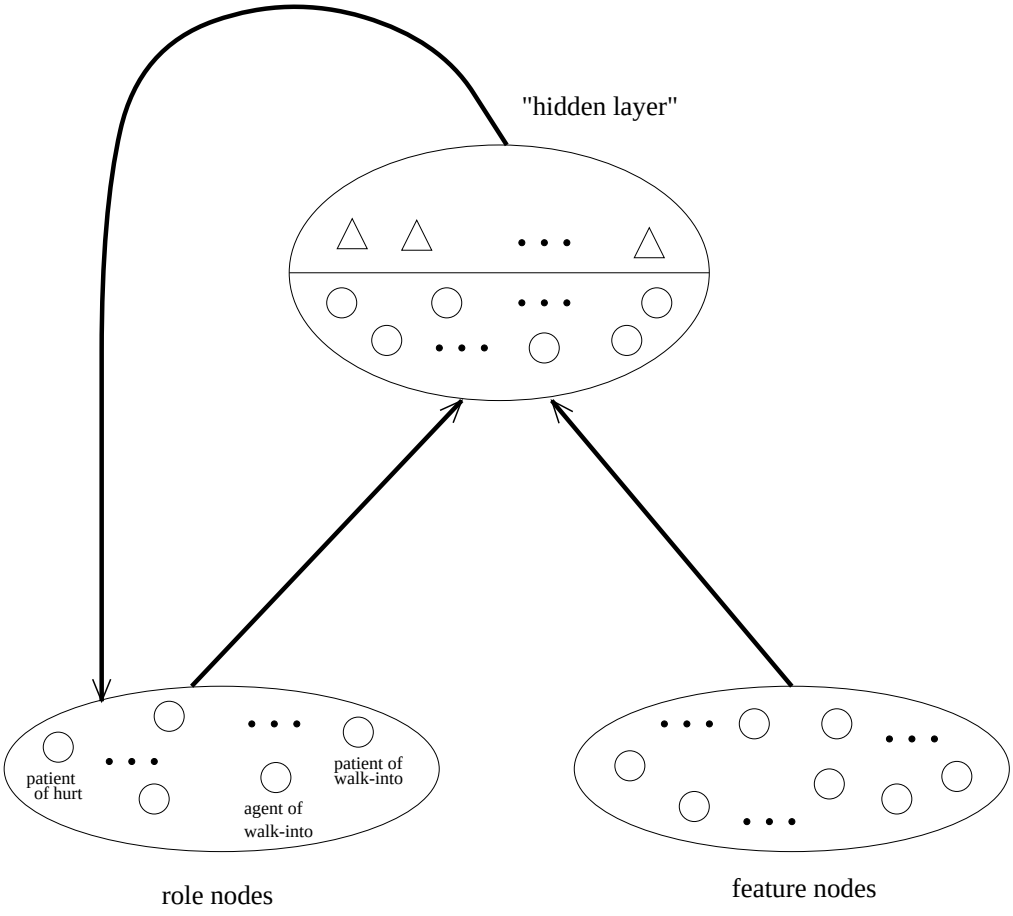

Figure 6: A schematic of the network structure for learning relational rules.

a *semantic filter* embedded within the subsymbolic level of representation (i.e., the hidden layer).

Thus the network architecture for investigating rule-learning consists of three subnetworks:

Role group: Each node in this group represents a distinct role. The role nodes are interconnected with links of delay 1.

Feature group: Each node in this group represents a distinct feature.

Hidden structure: The hidden structure consists of two layers. The computational behavior of nodes in these layers as well as the nodes in the role and feature groups is the same — each node is a sigmoid node which computes the weighted sum of its inputs and produces an output using a sigmoid transfer function. However, variable delay links, self-recurrence, and the interconnection pattern between the two hidden layers (see below), result in the nodes in the two hidden layers having different functional behavior. Based on this behavior, we refer to these layers as the ρ-btu layer and the τ-or layer respectively. The choice of names reflects the fact that nodes in the two layers perform the function of ρ-btu nodes and τ-or nodes, respectively.[3] These two node types were proposed in SHRUTI and have the following idealized behavior:

ρ-btu node: On receiving a spike train, a ρ-btu node produces a spike train that is *synchronous* with the driving input. The threshold of a node indicates the minimum number of *synchronous* inputs it must receive in order to fire. The window of synchrony, ω, specifies the amount of jitter (lead or lag) allowed between spikes deemed to be synchronous. Finally, ρ-btu nodes can respond in the desired manner as long as the delay, π, between two successive volleys of incident spikes lies in the interval $[\pi_{min}, \pi_{max}]$. In the simple case of periodic activity, π_{min} and π_{max} may be viewed as the shortest and the longest period of oscillation at which synchronous activity can propagate in the system.

τ-or node: A τ-or behaves like a *temporal* OR node, and becomes active on receiving patterns of coincident inputs. Unlike a ρ-btu node which fires if it receives a certain number of synchronous inputs in that phase, the firing criteria of a τ-or node may depend on activity spanning more than one phase. For example, a τ-or node may fire if it receives a certain number of inputs, within a period π, but spread over 2 or more phases. Furthermore, on becoming active, a τ-or node produces a train of wide pulses where each pulse is of width comparable to π.

Interconnection pattern across layers: Nodes in the role and feature groups are fully connected to the ρ-btu layer by links of delay 1. The nodes in the ρ-btu layer are in turn fully connected to nodes in the τ-or layer with delays of ranging from 1 to k. Thus each node in the ρ-btu layer is connected to each node in the τ-or layer with k links having delays of 1 through k. The value of k is monotonically related to the value of π_{max} and was set to 5. The existence of these multiple links into nodes in the τ-or layer and existence of self-recurrent connections *allows* nodes in this layer to behave like τ-or nodes if required. Nodes in the τ-or layer also have self recurrent links with a delay of 1. Finally, nodes in the τ-or layer are fully connected to role nodes with links of delay 1.

Faux input nodes: The backpropagation training regimen, requires that input nodes be distinct from output nodes. This presents a technical problem. Note that we want to

[3] The 'default' function of all nodes is that of ρ-btu nodes. Thus nodes in role feature groups are also ρ-btu nodes.

describe the input (i.e., initial) as well as the output (i.e., final) patterns of activity over the same set of role and feature nodes and certain role nodes that are inactive in the initial pattern may become active in the final pattern. This problem was overcome by adding a layer of *faux* input nodes. Each role/feature is encoded in this *faux* input layer as a pair of nodes (+,–). Each pair of input nodes (+,–) is connected to the associated role/feature nodes with links of (fixed) positive and negative weights respectively. The activation of the '+' node indicates activation of the associated role/feature, while the activation of the '–' node indicates the inactivity of the associated role/feature. The inactivity of both '+' and '–' nodes indicates that the activation level of role/feature is unspecified. It is important to encode such a unspecified state since we would only like to specify the activity of relevant roles and leave the state of *all other* roles unspecified.

4 Experimental Results

An artificial domain defined by four features: f1, animacy, solidity, and f2 was created. Some of the labels were chosen for obvious mnemonic reasons. Using these features, 14 distinct entities were created. Examples of entities are: John (1111), Casper (1100), wall (1011), and mist (1001). It was assumed that there are 5 roles in the domain. Two for walk-into (walk-into-agent and walk-into-patient), one for hurt (hurt-patient), and two others.

The above domain leads to 104 possible *input situations*. A possible input situation consists of any one of the 8 animate entities in the agent role of walk-into and any of the remaining 13 entities in the patient role of walk-into. Each input situation may be encoded as a time-varying *input pattern* in a number of ways. If we assume that the period π lies between 2 and 4, each input situation can be encoded using 20 different input patterns (π may range from 2 to 4, and the agent and patient may fire in any non overlapping phase). This gives a total of 2080 possible *input patterns*. The temporal extend of these patterns was chosen to be $6*\pi$. Thus each input pattern consisted of 2 to 4 phases replicated 6 times (i.e., input patterns extended from 12 to 24 time steps.) The target activation levels of nodes were set to 0.9 and 0.1 for firing and not-firing states, respectively.

The set of all possible input patterns was randomly split into a training set and a test set. After the split, some of the patterns were deliberately removed from the training set to ensure that the set did not include an input pattern for every input situation. This was done to make the learning problem more difficult. The size of the training set varied and included 30%-40% of the input patterns.

Two possible systematic worlds were considered. In World-1, only the agent of *walk-into* could get hurt whereas in World-2 both the agent and patient could get hurt. So in both World-1 and World-2, if an animate and solid entity walked into a solid entity, the former got hurt. In World-2, the latter also got hurt in case it was animate. Suitable target functions for describing the actual outcomes were created for various input patterns in each of the two worlds.

An actual network configuration: The net consisted of 18 faux input units, 5 role nodes, 4 feature nodes, 4 nodes in the ρ-btu layer, 2 nodes in the τ-or layer, and a threshold unit. The interconnection scheme was as described in Section 3.1 Learning was carried out using GRADSIM — a system for applying nonlinear gradient optimization techniques to train connectionist networks from examples consisting of time-varying input output patterns

(Watrous 1993).

Barring a few runs in which the network got stuck at an unsatisfactory local minimum, the optimization lead to a desirable network using the second-order gradient descent algorithm, BFGS. An example run consisted of a training set of 257 patterns. These were obtained by randomly selecting 453 patterns from the total set of 2080 patterns and then removing *all* patterns referring to John, Rasper and wall. Thus the training set did not include any situations involving these three entities. Optimization was stopped when the MSE error reached 0.002. This took 227 iterations. The resulting network responded appropriately to patterns in the training set as well as the test set. Thus the network generalized across individual entities as well as the relative phase of firing of entities. We also tested the generalization of the system along the π dimension. The training patterns had a maximum π of 4 and we tested the response of the system with patterns with higher values of π. The networks responded in an appropriate manner for values of π as high as 8.

5 Conclusion

The present work offers a simple demonstration that connectionist networks can learn relational rules from examples using spatio-temporal networks wherein dynamic bindings are expressed by the synchronous firing of appropriate nodes.

References

Ajjanagadde, V. & Shastri, L. (1990) Rules and variables in neural nets, *Neural Computation.* 3, 121–134.

Feldman, J. A. (1982) Dynamic connections in neural networks, *Bio-Cybernetics*, 46:27-39.

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

Lange, T. E., & Dyer, M. G. (1989) High-level Inferencing in a Connectionist Network. *Connection Science*, Vol. 1, No. 2, 181-217.

Mani, D.R. & Shastri, L. (1993) Reflexive Reasoning with Multiple-Instantiation in in a Connectionist Reasoning System with a Typed Hierarchy, *Connection Science*, Vol. 5, No. 3 & 4, 205–242.

Shastri, L. & Ajjanagadde, V. (1993) From simple associations to systematic reasoning, *Behavioral and Brain Sciences* Vol. 16, No. 3, 417–494.

Shastri, L. & Grannes, D.J. (1996) A connectionist treatment of negation and inconsistency. In *Proceedings of the Eighteenth Conference of the Cognitive Science Society*, San Diego, CA.

Singer, W. & Gray, C.M. (1995) Visual feature integration and the temporal correlation hypothesis. *Annual Review of Neuroscience* **18,** 555–586 (1995).

Watrous, R.L. & Shastri, L. (1987) Learning phonetic features using connectionist networks. R. L. Watrous and L. Shastri, *Proceedings of IJCAI-87, the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987. pp. 851–854.

Watrous, R.L. (1990) Phoneme discrimination using connectionist networks. *J. Accoust. Soc. Am.* 87 (4) 1753–1772.

Watrous, R.L. (1993) GRADSIM: A connectionist network simulator using gradient optimization techniques. Technical Report LS92-02, Siemens Research Laboratory, Princeton, NJ.