A Connectionist Solution to the Multiple Instantiation Problem using Temporal Synchrony*

D. R. Mani and Lokendra Shastri
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
mani@linc.cis.upenn.edu
shastri@central.cis.upenn.edu

Abstract

Shastri and Ajjanagadde have described a neurally plausible system for knowledge representation and reasoning that can represent systematic knowledge involving n-ary predicates and variables, and perform a broad class of reasoning with extreme efficiency. The system maintains and propagates variable bindings using temporally synchronous—i.e., in-phase — firing of appropriate nodes. This paper extends the reasoning system to incorporate *multiple instantiation of predicates*, so that any predicate can be instantiated up to k times, k being a system parameter. The ability to accommodate multiple instantiations of a predicate allows the system to handle a much broader class of rules, including bounded transitivity and recursion. The time and space requirements increase only by a constant factor, and the extended system can still answer queries in time proportional to the length of the shortest derivation of the query.

Introduction

In (Shastri & Ajjanagadde, 1990a, 1990b and Ajjanagadde & Shastri, 1991), Shastri and Ajjanagadde have described a solution to the variable binding problem (Feldman, 1982, Malsburg, 1986) and shown that the solution leads to the design of a connectionist reasoning system that can represent systematic knowledge involving n-ary predicates (relations) and *variables*, and perform a broad class of reasoning with extreme efficiency. The time taken by the reasoning system to draw an inference is only proportional to the *length* of the chain of inference and is independent of the number of rules and facts encoded by the system. The reasoning system maintains and propagates variable bindings using temporally synchronous—i.e., in-phase—firing of appropriate nodes. The solution to the variable binding problem allows the system to maintain and propagate a large number of bindings *simultaneously* as long as the number of *distinct* entities participating in any given episode of reasoning remains bounded. Reasoning in the proposed system is the transient but systematic flow of *rhythmic* patterns of activation, where each *phase* in the rhythmic pattern corresponds to a distinct *entity* involved in the reasoning process and where variable bindings are represented as the synchronous firing of appropriate argument and entity nodes. A fact behaves as a temporal pattern matcher that becomes 'active' when it detects that the bindings corresponding to it are present in the system's pattern of activity. Finally, rules are interconnection patterns that propagate and transform rhythmic patterns of activity.

Several other researchers have proposed connectionist solutions to the dynamic binding problem using a variety of techniques. These include the use of dynamic connections (Feldman, 1982), parallel constraint satisfaction (Touretzky & Hinton, 1988), position specific encoding (Barnden & Srinivas, 1991), tensor product representations (Dolan & Smolensky, 1989) and signatures (Lange & Dyer, 1989). (Shastri & Ajjanagadde, 1992) compares and contrasts these other approaches with the temporal synchrony approach used in this paper.

The system described in (Shastri & Ajjanagadde, 1990b) has the limitation that any predicate in the reasoner can be instantiated *at most once*.[1] It is not difficult to find examples which suggest that *reflexive* (effortless) reasoning involves dealing with multiple instances of predicates. For example, if we know that Mary is John's spouse, we would not have any difficulty in realizing that John is Mary's spouse. In other words, given *spouse-of(Mary,John)* we can reflexively answer 'yes' to the query *spouse-of(John,Mary)?* Such behavior would require the *spouse-of* predicate to be instantiated *twice*: once with *spouse-of(John,Mary)* and again with *spouse-of(Mary,John)*. As another example, consider the situation in which we know that Mary is older than John's father. If we now hear that John married Mary, we can instantly sense the unusualness of the situation, since Mary is obviously much older than John. But the fact that Mary is older than John has not been explicitly stated. This would suggest that we may have inferred *older-than(Mary,John)* using the facts *older-than(Mary,John's-father)* and *older-than(John's-father,John)*,[2] and the transitive nature of the *older-than* predicate. To model this scenario

*This work was supported by NSF grant IRI 88-05465 and ARO grants DAA29-84-9-0027 and DAAL03-89-C-0031.

[1] An extension for dealing with multiple instantiation using two levels of temporal synchrony was outlined in (Shastri & Ajjanagadde, 1990b). The solution proposed here is distinct from that in (Shastri & Ajjanagadde, 1990b).

[2] *older-than(John's-father,John)* follows from the knowledge that fathers are older than their children.

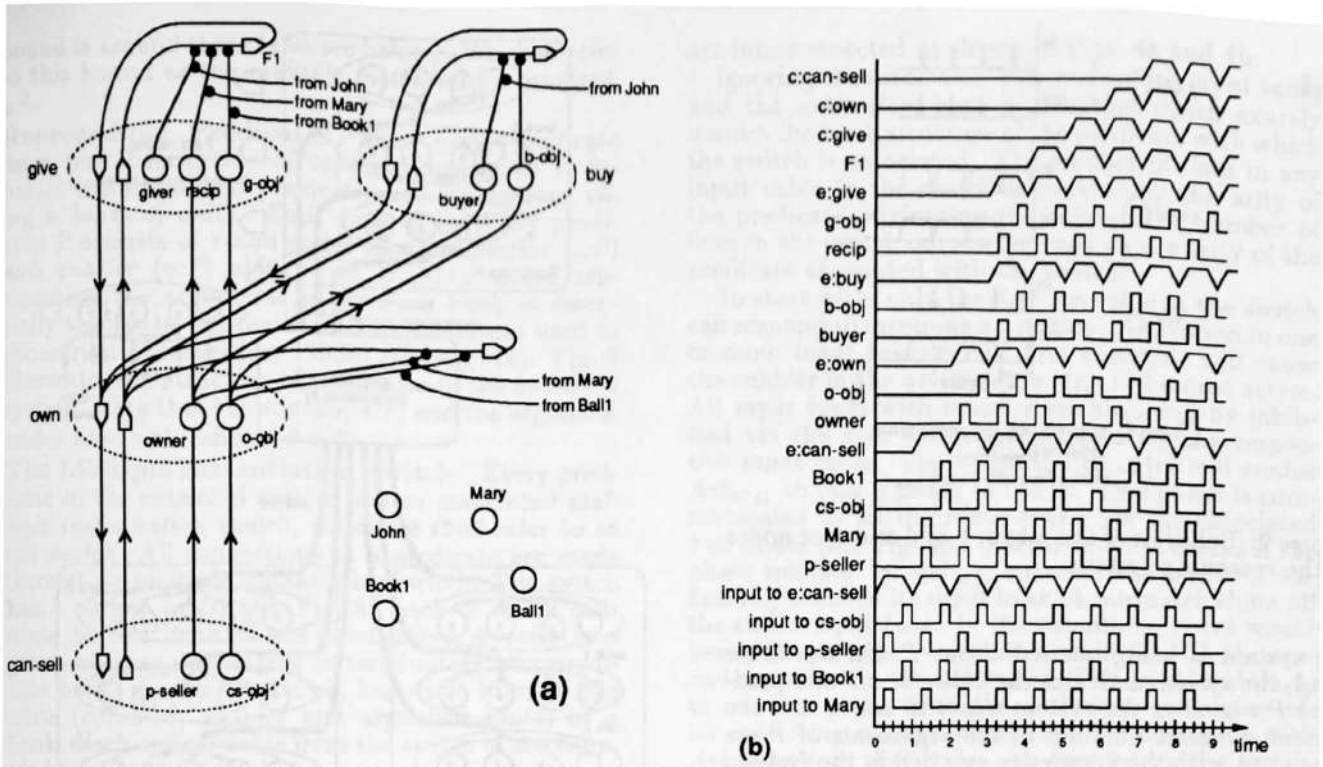

Figure 1: (a) An example encoding of rules and facts. (b) Activation trace for the query *can-sell(Mary,Book1)?*.

in the reasoning system, we would need to simultaneously handle three instantiations of *older-than*. Similarly, we can, without conscious deliberation, infer that John may be jealous of Tom if we know that John loves Mary and Mary loves Tom. Here again, we would need the ability to represent multiple instantiations of the *loves* predicate to capture the situation. Thus, any system which purports to model common-sense, reflexive reasoning should be capable of representing multiple instantiations of predicates.

This paper describes how the basic reasoning system of (Shastri & Ajjanagadde, 1990b) may be extended to incorporate multiple instantiation of predicates. We begin with a brief overview of the rule-based reasoning system, followed by an exposition of multiple instantiation in the reasoning system and its implementation. We will primarily concern ourselves with backward reasoning. Forward reasoning will be considered only briefly towards the end.

The rule-based reasoning system

Fig. 1a illustrates how long-term knowledge is encoded in the rule-based reasoning system. The network encodes the following *rules* and *facts*: i) $\forall x,y,z$ [*give(x,y,z)* $\Rightarrow$ *own(y,z)*], ii) $\forall x,y$ [*buy(x,y)* $\Rightarrow$ *own(x,y)*], iii) $\forall x,y$ [*own(x,y)* $\Rightarrow$ *can-sell(x,y)*], iv) *give(John,Mary,Book1)*, v) *buy(John,x)*, and vi) *own(Mary,Ball1)*.

The encoding makes use of two types of nodes (see Fig. 2): **ρ-btu** nodes (depicted as circles) and **τ-and** nodes (depicted as pentagons). These nodes have the following idealized behavior: On receiving a periodic spike train, a ρ-btu node produces a periodic spike train that is *in-phase* with the driving input. We assume that ρ-btu nodes can respond in this manner as long as the period of oscillation, π, lies in the interval $[\pi_{min}, \pi_{max}]$, where π_{min} and π_{max} are the minimum and maximum periods at which nodes can oscillate. For a discussion of biologically motivated values for these parameters, see (Shastri & Ajjanagadde, 1992).

A τ-and node behaves like a *temporal* AND node, and becomes active on receiving an oscillatory input consisting of a train of pulses *of width comparable to the period of oscillation*. On becoming active, a τ-and node produces an oscillatory pulse train whose period of oscillation and pulse width matches that of the input. A third type of node we use later is the τ-or node which becomes active on receiving *any* activation; its output is a pulse whose width and period equal π_{max}. Fig. 2 summarizes the behavior of the ρ-btu, τ-and and τ-or nodes.

The maximum number of distinct entities that may participate in an episode of reasoning equals $\lfloor \pi/\omega \rfloor$ where π is the period of oscillation and ω is the width of the window of synchronization—nodes firing with a lag or lead of less than $\omega/2$ would be considered to be in synchrony. The encoding also makes use of *inhibitory modifiers*—links that impinge upon and inhibit other links. A pulse propagating along an inhibitory modifier will block a pulse propagating along the link it impinges upon. In Fig. 1a, inhibitory modifiers are shown as links ending in dark blobs.

Each entity in the domain is encoded by a ρ-btu node. An n-ary predicate P is encoded by a pair of τ-and nodes and n ρ-btu nodes, one for each of the n arguments. One of the τ-and nodes is referred to as the *enabler*, *e:P*, and the other as the *collector*, *c:P*. In Fig. 1a, *enablers* point upward while *collectors* point downward. The *enabler e:P* becomes active whenever

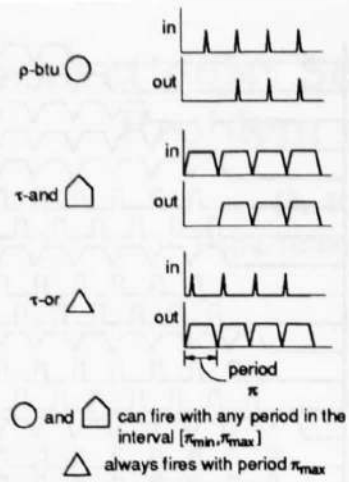

Figure 2: Behavior of the ρ-btu, τ-and and τ-or nodes in the reasoning system.

the system is being queried about P. On the other hand, the system activates the *collector* $c{:}P$ of a predicate P whenever the system wants to assert that the current dynamic bindings of the arguments of P are consistent with the knowledge encoded in the system. A rule is encoded by connecting the *collector* of the antecedent predicate to the *collector* of the consequent predicate, the *enabler* of the consequent predicate to the *enabler* of the antecedent predicate, and by connecting the arguments of the consequent predicate to the arguments of the antecedent predicate in accordance with the correspondence between these arguments specified in the rule. A fact is encoded using a τ-and node that receives an input from the enabler of the associated predicate. This input is modified by inhibitory modifiers from the argument nodes of the associated predicate. If an argument is bound to an entity in the fact then the modifier from such an argument node is in turn modified by an inhibitory modifier from the appropriate entity node. The output of the τ-and node is connected to the *collector* of the associated predicate (refer to the encoding of the fact *give(John,Mary,Book1)* and *buy(John,x)* in Fig. 1a.)

Inference Process Posing a query to the system involves specifying the query predicate and the argument bindings specified in the query. The query predicate is specified by activating its *enabler* with a pulse train of width and periodicity π. Argument bindings are specified by activating each entity and the argument nodes bound to that entity in a distinct *phase*. Phases are just non-overlapping time intervals within a period of oscillation.

We illustrate the reasoning process with the help of an example. Consider the query *can-sell(Mary, Book1)?* (i.e., Can Mary sell Book1?) The query is posed by i) activating the *enabler e:can-sell* ii) activating *Mary* and *p-seller* in the same phase (say, phase-1), and iii) activating *Book1* and *cs-obj* in some other phase (say, phase-2). As a result of these inputs, *Mary* and *p-seller* fire synchronously in phase-1 of every period of oscillation, while *Book1* and *cs-obj* fire synchronously in phase-2. See Fig. 1b. The activation from the *can-sell* predicate propagates to the *own*, *give* and *buy* predicates via the links encoding the rules. Eventually, as shown in Fig. 1b, *Mary*, *p-seller*, *owner*, *buyer* and *recip* will all be active in phase-1, while *Book1*, *cs-obj*, *o-obj*, *g-obj* and *b-obj* would be active in phase-2. The activation of *e:can-sell* causes the enablers of all other predicates to go active. In effect, the system is asking itself three more queries—*own(Mary,Book1)?*, *give(x,Mary,Book1)?* (i.e., Did *someone* give Mary Book1?), and *buy(Mary,Book1)?*. The τ-and node F1, associated with the fact *give(John,Mary,Book1)* becomes active as a result of the uninterrupted activation it receives from *e:give*, thereby answering *give(x,Mary,Book1)?* affirmatively. The activation from F1 spreads downward to *c:give*, *c:own* and *c:can-sell*. Activation of *c:can-sell* indicates an affirmative answer to the original query *can-sell(Mary,Book1)?*.

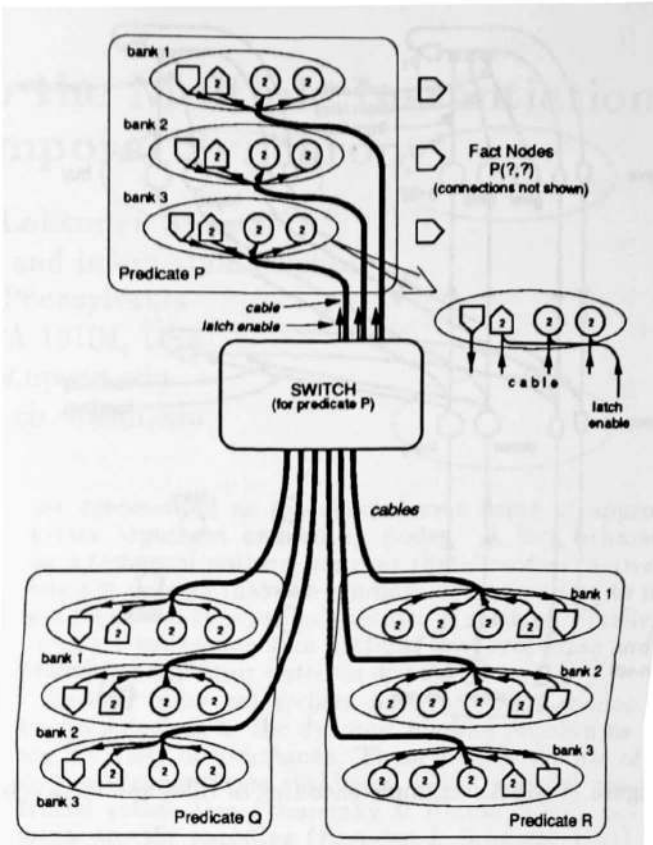

Figure 3: An overview of the multiple instantiation system. P and Q are binary predicates while R is a ternary predicate. The multiple instantiation constant $k = 3$.

Multiple Instantiation in the Reasoning System

Introducing multiple instantiation relies on the assumption that, during an episode of reflexive reasoning, any given predicate need only be instantiated a *bounded* number of times. In (Shastri & Ajjanagadde, 1992), it is argued that a reasonable value for this

bound is around three (also see below). We shall refer to this bound as the *multiple instantiation constant*, k.[3]

Representing Predicates Since every predicate must now be capable of representing up to k dynamic instantiations, predicates are represented using k *banks* of units. Each *bank* of an n-ary predicate P consists of τ-and nodes for the *collector* (*c:P*) and *enabler* (*e:P*) along with n ρ-btu nodes representing the arguments of P. *Each* bank is essentially similar to the predicate representation used in (Shastri & Ajjanagadde, 1990b) (see Fig. 1a). Fig. 3 illustrates the structure of predicates in the extended system. Note that the *enabler*, *e:P*, and the argument nodes have a threshold[4] $\theta = 2$.

The Multiple Instantiation Switch Every predicate in the extended system has an associated *multiple instantiation switch*, which we shall refer to as the *switch*. All connections to a predicate are made through its multiple instantiation switch. The switch has k output *cables* (see Fig. 3), each of which connects to one bank of the predicate. A *cable* is a group of wires originating or terminating at a predicate bank; a cable, therefore, has wires from all the units (collector, enabler and argument units) in a bank. Each output cable from the switch is accompanied by a *latch enable* link.

The switch arbitrates input instantiations to its associated predicate and brings about efficient and automatic dynamic allocation of predicate banks by ensuring the following: (1) Fresh predicate instantiations are channeled to the predicate banks only if the predicate can accommodate more instantiations. (2) All inputs that transform to the *same instantiation* are mapped into the *same predicate bank*. Thus, new instantiations selected for representation in the predicate are always unique.

Structure and Operation of the Multiple Instantiation Switch Figures 4a and 4b illustrate the construction of the the multiple instantiation switch. The switch consists of k groups or *ensembles* of units. Each ensemble consists of an *arbitrator bank*, and several *input banks*. The *arbitrator* consists of n ρ-btu nodes representing the arguments of the associated n-ary predicate, $(n-1)$ τ-or nodes and two τ-and nodes for the collector and enabler. Each ρ-btu node except the *first* is associated with a τ-or node, as shown in Fig. 4b. The i-th *arbitrator bank* directly connects with the i-th bank of the predicate. *Input banks* (Fig. 4b) consists of n ρ-btu units representing the arguments of the predicate, and two τ-and nodes representing the collector and enabler of the bank. Each *input bank* also has a τ-or node associated with it. The cable terminating at the *input bank* is an input to the switch; the outputs of the *input bank* connect to the *arbitrator* of the respective ensemble. Corresponding *input banks* across ensembles are interconnected as shown in Figs. 4a and 4b.

Ignoring the associated τ-or nodes, the *input banks* and the *arbitrators* have a structure which exactly mimics the bank structure of the predicate with which the switch is associated. The number of lines in any input cable to the switch is decided by the arity of the predicate originating the cable. The number of lines in the switch output depends on the arity of the predicate associated with the switch.

To start with, only the first ensemble in the switch can respond to incoming activation. Activation in one or more *input banks* of the first ensemble will cause the enabler in the *arbitrator*, *e:Arb*, to become active. All *input banks* with inactive enablers will be inhibited via the τ-or nodes associated with the respective *input banks*. The activation of *e:Arb* will enable Arb_{arg_1} to pick a phase to fire in. This phase is communicated to all the *input banks*, via the associated τ-or nodes (see Fig. 4b). Each τ-or node checks if the phase selected by Arb_{arg_1} matches the phase of the first argument of its *input bank*. A mismatch shuts off the entire *input bank*. In the meantime, *e:Arb* would have activated the τ-or node associated with the second argument in the *arbitrator*. This enables Arb_{arg_2} to select a phase from the activation remaining after inhibiting instantiations that did not agree with Arb_{arg_1}. Note that Arb_{arg_2} is enabled by the associated τ-or node *independent* of Arb_{arg_1} and will select a phase to fire in *even if* Arb_{arg_1} *is inactive* (which would be the case if all incoming instantiations have an unbound first argument). The process continues, allowing Arb_{arg_3}, ..., Arb_{arg_n} to select phases to fire in. After, Arb_{arg_n} has made its choice, the *latch enable* becomes active and the selected instantiation is transferred to the first predicate bank. A link from the last τ-or node to *e:Arb* in the second ensemble enables the second ensemble to select a fresh instantiation. Once the second ensemble makes its choice, it enables the third, and so on. The process continues until k instantiations have been channeled to the predicate, after which, any fresh input instantiations are ignored.

Note that if the i-th ensemble ($1 < i \leq k$) is making its choice, it will always select an instantiation which is *different* from those picked by the first $i-1$ ensembles, ensuring that all instantiations channeled to the predicate are unique. A more detailed description of the structure and operation of the switch can be found in (Mani & Shastri, 1992).

Encoding Rules and Facts

Every predicate in the system has k banks of units for representing k instantiations. Further every predicate has an associated multiple instantiation switch which arbitrates the instantiations which will be represented in the predicate. Given this modified underlying framework, we encode rules and long-term facts using an appropriate extension of the scheme detailed in (Shastri & Ajjanagadde, 1990b).

Dynamic instantiation which matches a long-term fact for predicate P could be present in *any one* of its k banks, necessitating a fact-pattern-matcher for *each* of the predicate banks. Thus, any fact of the form $P(C_1, \ldots, C_n)$ will be encoded using k τ-and

[3] For simplicity, we assume that k is the same for all predicates. This need not be the case.

[4] This applies to a predicate in the backward reasoning system. In a forward reasoner, the *collector*, *c:P*, and the argument nodes have a threshold $\theta = 2$.

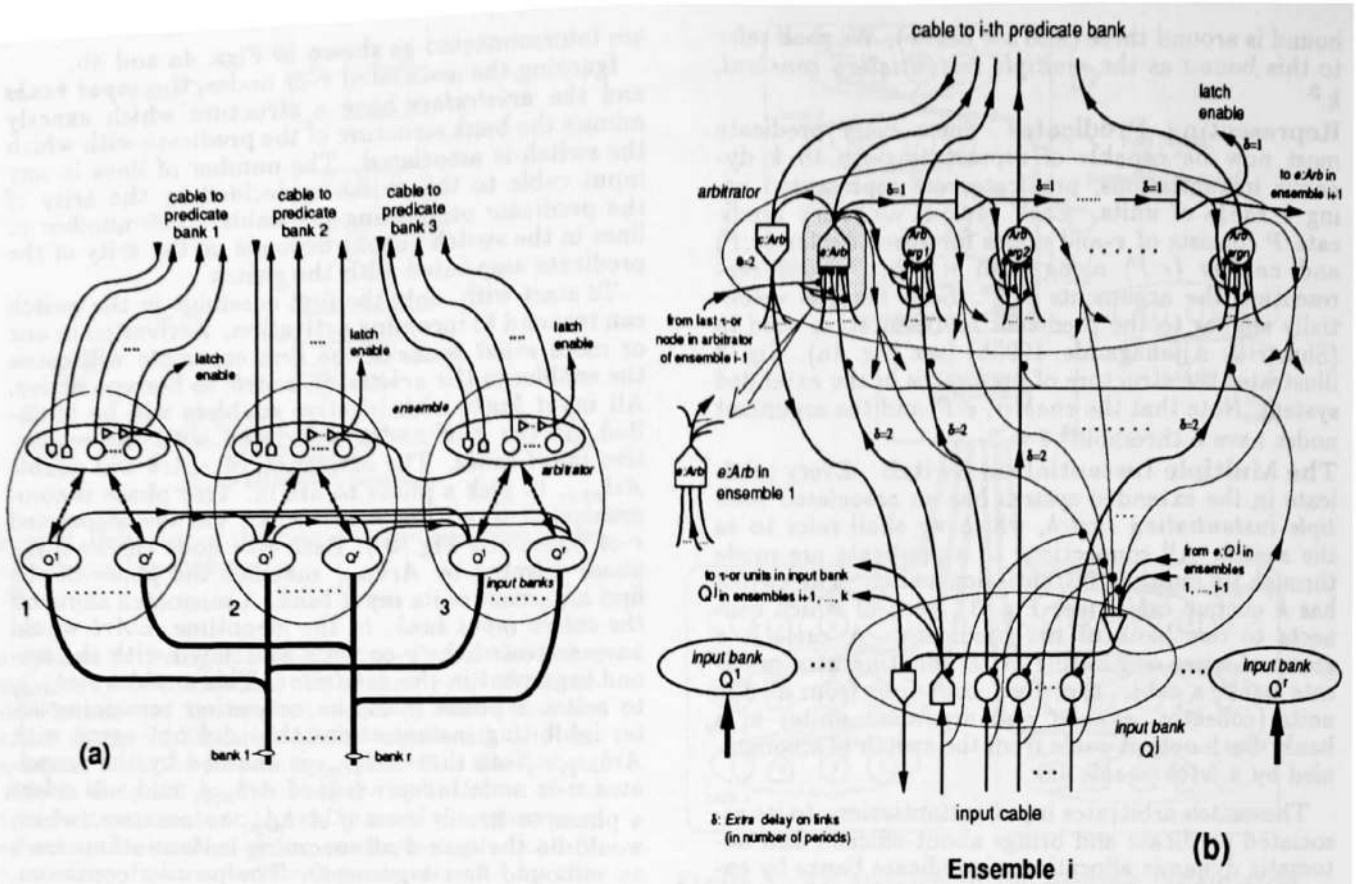

Figure 4: (a) Overview of the multiple instantiation switch. The multiple instantiation constant $k = 3$. (b) Structure of the i-th ensemble in the switch. Only connections between *input bank* Q^j and the *arbitrator* are shown. Connections between other *input banks* and the *arbitrator* are implied. As indicated, connections to *e:Arb* in the first ensemble are different.

nodes—one for each bank of P (Fig. 3). *Each* τ- and node encodes $P(C_1, \ldots, C_n)$ as described in (Shastri & Ajjanagadde, 1990b) (also see Fig. 1a).

Fig. 3 illustrates rule encoding at a very gross level. Suppose the rule relating P and Q in Fig. 3 is $\forall x,y$ [$P(x,y) \Rightarrow Q(y,x)$]. To encode this rule in a backward reasoning system, each bank of predicate Q is connected to an *input bank* in every ensemble of the switch for P. Thus, the k banks of predicate Q require a total of k^2 *input banks*—k *input banks* in each of the k ensembles of the switch. The *input banks* in the switch for P have a structure identical to the bank structure of predicate P. The cable from a bank of Q connects to the corresponding *input bank* as though the *input bank* itself represented the predicate P. The connection pattern between the bank of Q and the *input bank* is therefore identical to the connection pattern between the actual predicates in the system of (Shastri & Ajjanagadde, 1990b) (see Fig. 1a). Further, activation of the collector in any bank of P is transmitted (via the switch) to the corresponding bank in Q which *originated the instantiation* represented in that bank of P. A detailed description of rule and fact encoding in given in (Mani & Shastri, 1992).

Network Complexity The extended reasoning system requires $O(\mathcal{C} + \mathcal{F} + \mathcal{P})$ nodes and links. $\mathcal{C}$ and $\mathcal{F}$ represent the total number of entities and long-term facts in the system, respectively. $\mathcal{P}$ is the sum of the arities of all predicates in the rule base. The constant of proportionality for the network complexity is proportional to k^2, where k is the multiple instantiation constant. Thus, as in (Shastri & Ajjanagadde, 1990b, 1992), the network complexity is *linear* in the size of the knowledge base although the constant of proportionality is now larger. A similar comment holds for the time complexity: the system can still answer queries in time proportional to the length of the shortest derivation (Shastri & Ajjanagadde, 1990b, 1992); but now, the constant of proportionality is slightly larger, since we also need to consider the time required for the activation to propagate through the switches. Given a predicate P, the worst case propagation time for activation passing through its switch is proportional to k.

Multiple Instantiation in a Forward Reasoning System

To introduce multiple instantiation of predicates in the *forward reasoner*, we structure predicates and

their associated switches in a manner similar to the backward reasoning system.[5] Rules with a single predicate in the antecedent can be encoded directly: each bank of the antecedent predicate is connected to *input banks* in every ensemble of the switch for the consequent predicate. For rules like $\forall x,y,z$ [$P(x,y) \wedge Q(y,z) \Rightarrow R(x,y,z)$], which have multiple predicates in the antecedent, we would need to pair each bank of P with all the banks of Q and check if the second argument of P is the same as the first argument of Q. The obvious way to do this requires $O(k^m)$ nodes and links to encode each rule with m predicates in the antecedent.

Typically, we expect m to be around 2, since most predicates in a multiple-predicate antecedent serve to specify constraints on the arguments of a few key predicates. These type-enforcing predicates can be replaced by typed variables, significantly reducing the number of predicates in the antecedent. For example, the rule $\forall x,y$ [*collide(x,y)* $\wedge$ *animate(x)* $\wedge$ *solidobj(y)* $\Rightarrow$ *hurt(x)*] with three predicates in the antecedent is equivalent to $\forall$ *x:animate, y:solidobj* [*collide(x,y)* $\Rightarrow$ *hurt(x)*], which can be directly encoded as rule with a *single* predicate in the antecedent (Mani & Shastri, 1991). Even if this "compression" of the antecedent were not possible, we could introduce dummy predicates and split a rule with several predicates in the antecedent into several rules with just a few predicates in the antecedent, so as to maintain $m \approx 2$.

Maintaining, propagating and using multiple instantiations of a predicate entails a significant cost in space and time. In the context of reflexive reasoning, it is essential that these resources be bounded. This leads us to predict that the value of k is quite small, perhaps on more than 3 (Shastri & Ajjanagadde, 1992). Thus, with $k \approx 3$ and $m \approx 2$, the extra cost of encoding rules in the forward reasoner with multiple instantiation is a factor of about 10 ($\approx 3^2$)—which is about the same as the cost increase for a backward reasoner with multiple instantiation.

Conclusions

Extending the connectionist rule-based reasoning system to accommodate multiple instantiation of predicates enables the system to handle a wider and more powerful set of rules, facts and queries. The extended system can encode and reason with rules that capture symmetry, transitivity and recursion, provided the number of multiple instantiations of a predicate required to draw a conclusion remains bounded. The multiple instantiation reasoning system has been combined with a connectionist type hierarchy (Mani & Shastri, 1991) to provide a more flexible and powerful system. All these features have been successfully included in the simulation system reported in (Mani, 1992).

[5] Except for a few minor functional differences, the multiple instantiation switch associated with a predicate in the forward reasoning system is structurally identical to the switch used in the backward reasoner. See (Mani & Shastri, 1992) for details.

References

(Ajjanagadde & Shastri, 1991) Ajjanagadde, V., and Shastri, L. Rules and Variables in Neural Nets. *Neural Computation* **3**:121–134.

(Barnden & Srinivas, 1991) Barnden, J., and Srinivas, K. Encoding Techniques for Complex Information Structures in Connectionist Systems. *Connection Science* **3**(3):269–315.

(Dolan & Smolensky, 1989) Dolan, C. P., and Smolensky, P. Tensor Product Production System: A Modular Architecture and Representation. *Connection Science* **1**(1):53–68.

(Feldman, 1982) Feldman, J. A. Dynamic Connections in Neural Networks. *Bio-Cybernetics* **46**:27–39.

(Lange & Dyer, 1989) Lange, T. E., and Dyer, M. G. High-level Inferencing in a Connectionist Network. *Connection Science* **1**(2):181–217.

(Malsburg, 1986) von der Malsburg, C. Am I Thinking Assemblies? In G. Palm and A. Aertsen (Eds.). *Brain Theory*. Berlin: Springer-Verlag.

(Mani, 1992) Mani, D. R. Using the Connectionist Rule-Based Reasoning System Simulator. Technical Report, Department of Computer and Information Science, Univ. of Pennsylvania. Forthcoming.

(Mani & Shastri, 1991) Mani, D. R., and Shastri, L. 1991. Combining a Connectionist Type Hierarchy with a Connectionist Rule-Based Reasoner. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 418–423. Hillsdale NJ: Lawrence Erlbaum Associates.

(Mani & Shastri, 1992) Mani, D. R., and Shastri, L. 1992. Multiple Instantiation of Predicates in a Connectionist Rule-Based Reasoner. Technical Report MS-CIS-92-05, Department of Computer and Information Science, Univ. of Pennsylvania.

(Shastri & Ajjanagadde, 1990a) Shastri, L., and Ajjanagadde, V. An optimally efficient limited inference system. In *Proceedings of AAAI-90, the Twelfth National Conference of the American Association of Artificial Intelligence*, 563–570. Cambridge MA: American Association for Artificial Intelligence.

(Shastri & Ajjanagadde, 1990b) Shastri, L., and Ajjanagadde, V. From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables and Dynamic Bindings. Technical Report MS-CIS-90-05, Department of Computer and Information Science, Univ. of Pennsylvania.

(Shastri & Ajjanagadde, 1992) Shastri, L., and Ajjanagadde, V. From Simple Associations to Systematic Reasoning: A Connectionist Representation of Rules, Variables and Dynamic Bindings using Temporal Synchrony. *Behavior and Brain Sciences*. Forthcoming.

(Touretzky & Hinton, 1988) Touretzky, D. S, and Hinton, G. E. A Distributed Connectionist Production System. *Cognitive Science* **12**(3):423–466.