INTERNATIONAL COMPUTER SCIENCE INSTITUTE

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

Dealing with negated knowledge and inconsistency in a neurally motivated model of memory and reflexive reasoning

Lokendra Shastri* and Dean Jeffrey Grannes

TR-95-041

August 1995

Abstract

Recently, SHRUTI has been proposed as a connectionist model of rapid reasoning. It demonstrates how a network of simple neuron-like elements can encode a large number of specific facts as well as systematic knowledge (rules) involving n-ary relations, quantification and concept hierarchies, and perform a class of reasoning with extreme efficiency. The model, however, does not deal with negated facts and rules involving negated antecedents and consequents. We describe an extension of SHRUTI that can encode positive as well as negated knowledge and use such knowledge during reflexive reasoning. The extended model explains how an agent can hold inconsistent knowledge in its long-term memory without being "aware" that its beliefs are inconsistent, but detect a contradiction whenever inconsistent beliefs that are within a certain inferential distance of each other become co-active during an episode of reasoning. Thus the model is not logically omniscient, but detects contradictions whenever it tries to *use* inconsistent knowledge. The extended model also explains how limited attentional focus or action under time pressure can lead an agent to produce an erroneous response. A biologically significant feature of the model is that it uses only *local* inhibition to encode negated knowledge. Like the basic model, the extended model encodes and propagates dynamic bindings using temporal synchrony.

*This work was partially funded by ONR grants N00014-93-1-1149 and N00014-95-C-0182. Thanks to David Bailey, Jerry Feldman, Jerry Hobbs, Dan Jurafsky, George Lakoff, D.R. Mani and Srini Narayanan for providing intellectual stimulation. *E-mail:* shastri@icsi.berkeley.edu

1 Introduction

Understanding language involves among other things, performing inferences to establish referential and causal coherence, generate expectations, and make predictions.[1] Nevertheless we can understand language at the rate of *several hundred words per minute*. This suggests that we can perform a wide range of inferences rapidly, automatically and without conscious effort — as though they were a *reflex* response of our cognitive apparatus. In view of this, we have described such reasoning as *reflexive* (Shastri 1991).[2] Our ability to perform reflexive reasoning poses a challenge for cognitive and neurosciences: How can a system of simple and slow neuron-like elements represent a large body of specific facts as well as context-sensitive rules and perform a broad class of inferences within hundreds of milliseconds?

Over the past few years we have been engaged in developing SHRUTI — a connectionist model of reflexive reasoning which attempts to address the above challenge (Ajjanagadde & Shastri 1991; Shastri & Ajjanagadde 1993; Mani & Shastri 1993). SHRUTI suggests a partial solution to the problem and demonstrates how a system of neuron-like elements could encode over a million facts and rules involving n-place relations, and concept hierarchies, and yet perform multi-step inferences within a few hundred milliseconds. For example, using biologically motivated values for system parameters, it takes SHRUTI 320 milliseconds to infer "Susan *owns* a car" after its internal state is initialized to represent "Susan *bought* a Rolls-Royce". Similarly, if SHRUTI's long-term memory contains the fact "John bought a Jaguar", it takes 420 milliseconds to answer "yes" to the query "Does John own a car?" [3]

A key feature of SHRUTI is that it uses synchronous firing of cells to establish dynamic bindings and represent dynamic relational structures. The view of information processing implied by SHRUTI is one where (i) reasoning is the transient but systematic propagation of a *rhythmic* pattern of activity over a *memory network* (ii) each active entity is a phase in this rhythmic activity, (iii) dynamic bindings are represented by the *synchronous* firing of appropriate role and filler nodes, (iv) rules are interconnection patterns that cause the propagation and transformation of rhythmic patterns of activity, and (v) long-term facts are subnetworks that act as temporal pattern matchers.

In essence, SHRUTI demonstrates how connectionist networks can represent relational structures and perform certain types of computations over such structures in an efficient manner. This involves the representation of both *static* and *dynamic*

[1] Empirical data suggests that inferences required to establish referential and causal coherence occur rapidly and automatically during text understanding (see e.g., McKoon & Ratcliff 1980; McKoon & Ratcliff 1981; Keenan, Baillet, and Brown 1984). The evidence for the automatic occurrence of *elaborative* or predictive inferences however, is mixed (see e.g., Kintsch 1988; Potts, Keenan, and Golding 1988).

[2] A formal characterization of reflexive reasoning appears in (Shastri 1993).

[3] This assumes that SHRUTI's long-term memory (LTM) includes the rule "if x buys y then x owns y" and the relations: "Rolls-Royce is a car" and "Jaguar is a car".

bindings, interactions between these two types of bindings, and the systematic but context sensitive propagation of dynamic bindings from one relational structure to another. Since schemas and frames are essentially relational structures, and since mappings between schemas and frames may be viewed as the propagation of bindings across these structures, the significance of the representational and inferential mechanisms developed in SHRUTI extends beyond reasoning to other cognitive tasks that involve computations over relational structures. Henderson (1994) has shown that the SHRUTI architecture is also appropriate for supporting real-time parsing of English.

SHRUTI however, only dealt with "positive knowledge". Thus while it could encode positive facts such as "John loves Mary", it could not encode negated facts such as "John does not love Susan". Similarly, while SHRUTI could encode rules involving positive predicates, it could not encode rules involving negated predicates. For example, it could not encode a rule such as "One cannot vote if one is not a citizen" (i.e., *x:person, y:country* $\neg$ *citizen(x,y)* $\Rightarrow$ $\neg$ *vote-in-elections(x,y)*). [4] Finally, since SHRUTI did not have any explicit representation of negation, it could only respond "yes" and "don't know" to queries. It had no firm basis for answering "no" to a question.

Due to the complexity it adds to the inference process, knowledge representation and reasoning systems often do not deal explicitly with negation. Some models deal partially with negation by adopting what is known as the *closed world assumption* in AI. The intuition behind this assumption is as follows: If an agent knows all the relevant facts about some domain, then it may assume that any fact it does not know is false! In view of this assumption, the agent may treat "don't know" answers as "no" answers.[5] As pointed out in (Shastri & Ajjanagadde 1993), we had also adopted the closed world assumption in SHRUTI. The use of the closed world assumption, however, has limited applicability and cannot be a substitute for the ability to deal explicitly with negated information.

A second way in which negation can be treated partially is by making use of mutual exclusivity of concepts. Since the membership of an entity in a certain category can rule out its membership in other non-overlapping categories, the knowledge of category membership can be used as a source of negated information. For example, given "Fido is a dog" one can conclude "Fido is not a cat" since dogs and cats are mutually exclusive classes of mammals. Although mutual exclusivity of categories can be a powerful means for capturing certain types of negated knowledge, it does not obviate the need for the explicit representation and use of negated predicates in facts and rules.

[4] We use the notation of first-order logic for notational convenience and its use does not mean that we view deduction to be the sole basis of reflexive reasoning.

[5] The closed world assumption states that any fact F that is neither in the knowledge base nor deducible from the knowledge base, may be assumed to be false.

1.1 The importance of explicit negated information

Certain types of explicit negated information play an obvious role in common sense reasoning. Clearly, we are capable of remembering negated facts and making use of such facts during reflexive reasoning. For example, if we are told "John has been to Canada" and "John has not been to Europe", we can readily answer "yes", "no", and "I don't know" to the questions (i) "Has John been to Canada?" (ii) "Has John been to France?" and (iii) "Has John been to Australia?," respectively. It also seems apparent that we can reason reflexively with rules involving certain types of negated conditions. So given "John is a bachelor", we can readily answer "no" to questions such as "Is John married? and "Is John married to Susan?". Note that answering these questions involves using knowledge that may be approximated as "A bachelor is not married to anyone" (i.e., *bachelor(x)* $\Rightarrow \neg$ *married(x,y)*).

1.2 Treatment of inconsistent knowledge

The encoding of negated knowledge raises the possibility of inconsistencies in an agent's long-term memory (LTM). We can, and often do, hold inconsistent beliefs in our LTM without being explicitly aware of such inconsistencies. At the same time, we are capable of detecting contradictions when we are faced with beliefs that are immediately contradictory — say, if we are presented with *P(a,b)* and $\neg$*P(a,b)*.

In view of the above, a cognitively plausible model of memory and reasoning should allow inconsistent facts and rules to co-exist in its LTM. But at the same time, the model should be capable of detecting contradictions whenever two contradictory beliefs that are within a certain inferential distance of each other become co-active during an episode of reasoning. As an example, consider the case wherein an agent's LTM contains rules that lead from $P(x, y)$ to $R(x, y)$ via a chain of inference, and from $Q(x, y)$ to $\neg R(x, y)$ via another chain of inference. It should be possible for the agent's LTM to contain the facts $P(a, b)$ and $Q(a, b)$ without the agent being aware that these facts render the LTM inconsistent. However, the agent should become aware of this inconsistency under suitable conditions. One situation in which this might happen is when the agent tries to answer the query *R(a,b)?* If the inferential distance between P and R and that between Q and $\neg R$ is not too great, the process of answering the query *R(a,b)?* would lead to the activation of both $P(a, b)$ and $Q(a, b)$, and hence, to the co-activation of *R(a,b)* and $\neg R(a, b)$. When this happens, the system should detect that it has contradictory beliefs.

The treatment of inconsistency pursued in this work has two desirable properties. First, the model does not strive for *logical omniscience*; it allows inconsistent knowledge to exist in the agent's LTM without requiring that all occurrences of inconsistencies be recognized by the agent. This is highly desirable given that an agent has only limited resources and must interact with a rich and complex environment with myriad sources of information. Consequently, the agent simply cannot afford to spend unbounded resources trying to detect *all possible* inconsistencies in its knowledge! Sec-

ond, the proposed treatment of inconsistency ensures that any inconsistencies in the agent's knowledge become apparent when the agent tries to bring such inconsistent knowledge to bear on a particular task.

Finally, any agent with limited resources must sometimes act with only limited attentional focus and often under time pressure. This means that an agent may sometimes overlook relevant information and act in an erroneous manner. More focused evaluation or an appropriate cue, however, might make the necessary information available and lead to a correct response. Several interesting aspects of such a situation are captured in the following scenario:

> **Post Office Example:** John runs into Mary on the street. "Where are you going?" asks John. "To the post office," replies Mary. "But isn't today Presidents' Day?" remarks John. "Oops! I didn't realize that today was a federal holiday," says Mary after a momentary pause and heads back.

Clearly, the knowledge in Mary's LTM was sufficient to infer that "today" was Presidents' Day — a federal holiday — and therefore, a postal holiday. But the fact that she was going to the post office means that she had supposed that the post office was open (so in a sense, Mary held inconsistent beliefs). John's rhetorical question served as a trigger that brought the relevant information to the surface and made Mary realize her mistake. A cognitively plausible model should be capable of modeling such situations.

In this report we describe an extension of SHRUTI that can deal with negated facts as well as negated antecedents and consequents in rules. The extended system deals with inconsistent knowledge in the manner characterized above and also explains situations such as the "Post Office Example". The related problem of dealing with negation in the type hierarchy and exploiting the mutual exclusivity of certain subtypes is not discussed here and will be described in a subsequent report.

Section 2 discusses several aspects of SHRUTI that are relevant for explaining the encoding of negated knowledge. Section 3 describes the extension of SHRUTI while Section 4 presents a few illustrative examples.

2 Overview of SHRUTI

2.1 Some representational problems associated with reflexive reasoning

Assume that an agent's LTM embodies the following systematic knowledge: 'If someone gives a recipient an object then the recipient comes to own that object'. Given the above knowledge an agent would be capable of inferring "Mary owns a book" on being told "John gave Mary a book". A network must solve several representational problems in order to incorporate the above behavior. Before discussing these problems let us introduce some notation. A specific event such as "John gave Mary a book"

can be viewed as an *instance* of the three place relation (or predicate) *give* with roles: *giver*, *recipient*, and *give-object* and expressed as the *fact give(John,Mary,a-Book)*. In what follows, we will use "fact" and "instance" interchangeably to refer to a relational instance. The systematic rule-like knowledge given above about giving and owning may be succinctly expressed as:

$give(x,y,z) \Rightarrow own(y,z)$ — (1)

wherein *own* is a two place relation with roles: *owner* and *own-object* and "⇒" informally means "leads to".

Dynamic representation of facts requires dynamic bindings: The reasoning system must be capable of rapidly representing facts such as *give(John,Mary,a-Book)* when they are "communicated" to it by perceptual or linguistic processes, or when they arise internally as a result of the reasoning process. Note that the fact *give(John,Mary,a-Book)* cannot be represented by simply activating the representations of the roles *giver*, *recipient*, and *give-object*, and the constituents 'John', 'Mary', and 'a-Book', since such a representation would be indistinguishable from that of *give(a-Book,Mary,John)*. This fact, like any other instantiation of an n-ary relation, is a composite structure wherein each constituent fills a distinct role in a relation. Consequently, the representation of such a fact requires the representation of *bindings* between the roles of the relation and its fillers. Thus the dynamic representation of *give(John,Mary,a-Book)* requires the creation of dynamic bindings *(giver=John, recipient=Mary, give-object=a-Book)*.

Reasoning involves systematic propagation of dynamic bindings: Starting with a dynamic representation of *give(John,Mary,a-Book)* the state of a network encoding rule (1) should evolve rapidly to include the dynamic representation of the inferred fact: *own(Mary,a-Book)*. Generating inferred facts involves the systematic *propagation* of dynamic bindings in accordance with various rules embodied in the system. The rule $give(x,y,z) \Rightarrow own(y,z)$ specifies that a *give* event leads to an *own* event wherein the *recipient* of the *give* event corresponds to the *owner* of the *own* event and the *give-object* of the *give* event corresponds to the *own-object* of the *own* event. Thus the application of this rule in conjunction with the instance *give(John,Mary,a-Book)* should create an instance of *own* with bindings *(owner=Mary, own-object=a-Book)*.

Long-term facts as temporal-pattern matchers: In addition to encoding domain rules, the reasoning system must also be capable of encoding facts in its LTM and using them during recall, recognition, query answering, and reasoning. For example, a reasoning system should be capable of encoding the fact "John bought a Rolls-Royce" in its LTM and using it to rapidly answer queries such as "Did John buy a Rolls-Royce?" and "Did John buy something?" Observe that the encoding of a long-term fact must store the bindings associated with a fact in the form of *static* bindings within a long-term structure. Once formed, this structure should be capable of detecting the occurrence of dynamic bindings that match its static bindings.

The multiple-instantiation problem: Reasoning often requires the simultaneous

activation of more than one fact pertaining to the *same* relation. For example, the system may have to encode *give(John,Mary,a-Book)* and *give(Mary,Tom,a-Car)* at the same time. A reasoning system must be capable of keeping multiple instantiations of the same relation active without cross-talk between instantiations.

2.2 Solutions incorporated in SHRUTI

2.2.1 General representation

Refer to the representation of some predicates and entities shown in Figure 1. Observe that predicates, their roles, and entities are represented using distinct nodes. In particular, there is a distinct cluster of nodes corresponding to each predicate. Rules are encoded by connections between the role nodes of appropriate predicates.

Nodes such as *John* and *Mary* correspond to *focal* nodes of more elaborate representations of the entities 'John' and 'Mary'. Information about the attribute values (features) of 'John' and his relationship to other concepts is encoded by linking the focal node *John* to appropriate nodes. Observe that information about the various perceptual and semantic features and aspects of 'John' may be distributed over various parts the memory network. This information, however, eventually converges to, and diverges from, the focal node for 'John'. Details of such an encoding may be found in (Shastri & Feldman 1986; Feldman 1989; Shastri 1991). As explained below, the cluster of nodes associated with each predicate also serves as a *focal area* to which all information pertaining to the relation converges, and from where all information pertaining to the relation may be accessed by fanning out along appropriate links.

For simplicity we assume that each node in the figure corresponds to an individual neuron-like (connectionist) node. This is an idealization. In actuality, each node corresponds to an *ensemble* of cells (Shastri & Ajjanagadde 1993).

2.2.2 Encoding of predicates: predicate clusters as convergence zones

The core component of the encoding of a predicate is illustrated in Figure 1. Consider the encoding of the ternary predicate *give* with three roles: *giver*, *recipient*, and *give-object*. This predicate is encoded by a bank of three role nodes (depicted as circular nodes) labeled *giver*, *recip*, and *g-obj*, an *enabler* node (depicted as a pentagon pointing upwards), and a *collector* node (depicted as a pentagon pointing downwards). In general, the cluster for an *n-ary* predicate contains n role nodes, one collector node, and one enabler node. For notational convenience we refer to the enabler and collector nodes of a predicate P as *e:P* and *c:P* respectively. The circular nodes are ρ-**btu** nodes while the pentagon shaped nodes are τ-**and** nodes.

We refer to the bank of $n + 2$ nodes associated with a predicate as the "core component" of a predicate's encoding since it provides a skeleton for attaching and accessing the complete encoding of a predicate. The latter includes all the interconnections that this structure makes with the rest of the memory network in order to

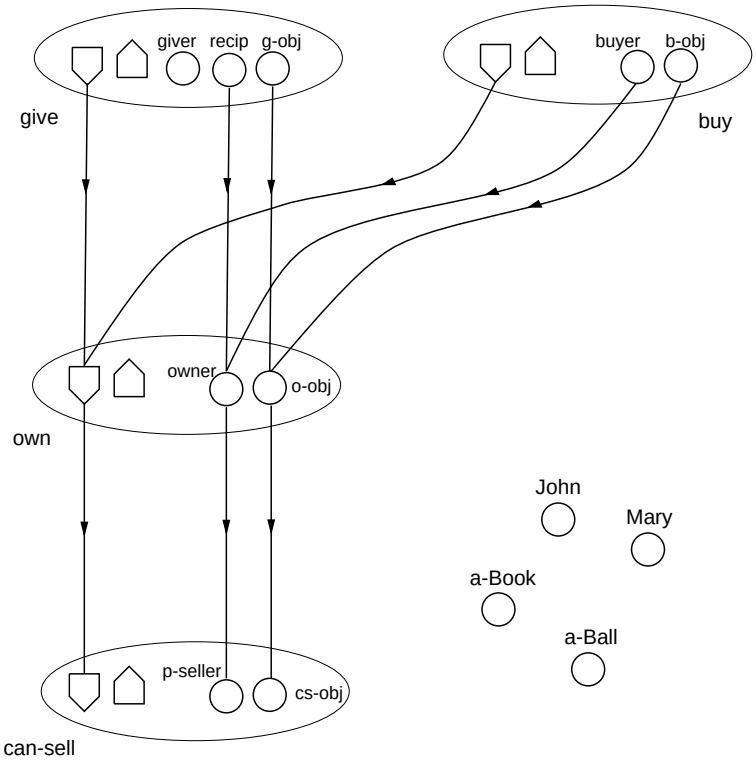

Figure 1: An example encoding of predicates, rules, entities. Links between roles reflect the correspondence between roles in the antecedents and consequents of rules.

encode rules and facts involving this predicate. Thus this cluster of nodes can be viewed as a *focal area* in that all information about the predicate converges to this area, and all information about the predicate can be accessed by fanning out from this area. Observe that the proposed representation of a predicate is consistent with the notion of "convergence zones" (Damasio 1989).

Before specifying the computational behavior of the nodes in a predicate cluster, let us examine their semantic import. The significance of role nodes is as expected — each role node represents a role of the predicate. The representational significance of the *enabler* and *collector* nodes is as follows. Assume that roles of a predicate P are dynamically bound to some fillers thereby representing a dynamic instance of P (we will see, shortly, how such dynamic bindings are represented). The activation of *e:P*, the *enabler* of P, means that the system is in a state wherein it is trying to explain whether the currently active dynamic instance of P is *supported* by the knowledge encoded in the memory. In other words, the system is "querying" itself about the currently active instance of P — Does this instance follow from what is known? Such a "query" might be generated internally by the reasoning model, or be communicated to it by some other system (e.g., the parsing or planning module) that might be interested in determining whether a particular instance of P follows from the knowledge encoded in the LTM. The semantic import of *c:P* is the complement of that of *e:P*. By activating *c:P* the system is "asserting" that the currently active instance of P — as composed of the dynamic bindings of the roles of P — is supported by the knowledge encoded in the system.

Let us consider an example to make this concrete. Assume that some process establishes dynamic bindings for the roles of *give*, say, *(giver=John, recipient=Mary, give-object=a-Book)*, and activates *e:give*. By doing so, the process is essentially "asking" whether the system believes that John gave Mary a book. Subsequently, if the node *c:give* becomes active as a result of spreading activation in the system, it means that the system is "asserting" that yes, it believes that John gave Mary a book.

The collector *c:P* can also be used by an external process to communicate information to the memory system. Thus some external process may establish some dynamic bindings over the roles of P and activate the node *c:P* in order to communicate an instance of P to the system. For example, the language understanding process might activate *c:give* and establish the bindings *(giver=John, recipient=Mary, give-object=a-Book)* upon hearing the utterance "John gave Mary a book".

Note that we are using intentional terms such as "query", "assert", "believe", and "know" to describe the significance of various nodes. This should not be construed to mean that individual nodes have such intensionality. Nodes simply serve a particular role in the overall behavior of the system by virtue of having certain computational properties and by having certain interconnections to other nodes.

2.2.3 Computational behavior of nodes

ρ-btu nodes have the following idealized behavior: When active, a ρ-btu node produces a train of spikes. If a ρ-btu node A is connected to another ρ-btu node B then the activity of B synchronizes with the activity of A. In particular, a periodic firing of A leads to a periodic and *in-phase* firing of B. It is assumed that ρ-btu nodes can respond in this manner as long as the period of firing, π, lies in the interval $[\pi_{min}, \pi_{max}]$. This interval can be interpreted as defining the frequency range over which ρ-btu nodes can sustain a synchronized response. A threshold, n, associated with a node indicates that the node will fire only if it receives n or more *synchronous* inputs. If unspecified, a node's threshold is assumed to be 1.

τ-and nodes have the following idealized behavior: A τ-and node becomes active on receiving a pulse of duration $\geq \pi_{max}$. We assume that this condition is satisfied by a periodic pulse train as long as the gap between successive pulses is less than some interval ϵ (where ϵ is comparable to a spike width). Thus a τ-and node behaves like a *temporal and* node. On becoming active, such a node produces an output pulse train similar to the input pulse. A threshold, n, associated with a τ-and node indicates that the node will fire only if it receives n or more synchronous pulses of duration $\geq \pi_{max}$. If unspecified, n is assumed to be 1.

The system also makes use of τ-or nodes which behave as temporal-OR nodes. The behavior of these nodes, however, is not relevant for our present purpose and is not discussed.

2.2.4 Encoding dynamic bindings

Dynamic bindings are represented in the system by the *synchronous* firing of appropriate role and filler nodes. With reference to the nodes in Figure 1, the *rhythmic* pattern of activity shown in Figure 2 represents the dynamic bindings *(giver=John, recipient=Mary, give-object=a-Book)*, and hence, the dynamic fact *give(John,Mary,a-Book)*. Observe that (i) the concepts *John, Mary*, and *a-Book* are firing in distinct phases but (ii) *John* and *giver* are firing in synchrony, *Mary* and *recipient* are firing in synchrony, and *a-Book* and *give-object* are firing in synchrony. The absolute phase of firing of filler and role nodes is not significant — what matters is the *coincidence* (or the lack thereof) in the firing of nodes.

2.2.5 Encoding of rules and the propagation of bindings

As discussed above, a step of inference may be viewed as taking an instance of the antecedent predicate and dynamically creating an instance of the consequent predicate, with the role bindings of the latter being determined by (i) the role bindings of the former and (ii) the role correspondence specified by the rule. Hence, a rule is encoded by linking the roles of the antecedent and consequent predicates so as to reflect the correspondence between roles specified by the rule. For example, the

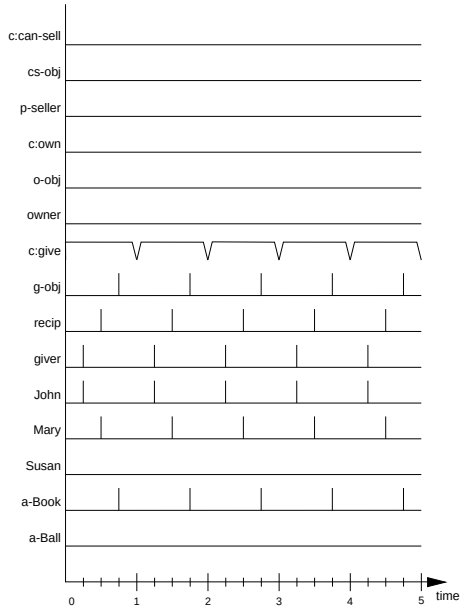

Figure 2: Rhythmic pattern of activation representing the dynamic bindings *give(John,Mary,a-Book)*. The binding between a role and a filler is represented by the in-phase firing of associated nodes.

rule $give(x, y, z) \Rightarrow own(y, z)$ can be encoded by establishing links between the roles *recipient* and *give-object* of *give* and the roles *owner* and *own-object* of *own*, respectively. If we also wish to encode the rule: $buy(x, y) \Rightarrow own(x, y)$, we can do so by connecting the roles *buyer* and *buy-object* of *buy* to the roles *owner* and *own-object* of *own*, respectively (see Figure 1).[6]

Given the above interconnection pattern and node behavior, the initial state of activation shown in Figure 2 representing *give(John,Mary,a-Book)* will lead to the state of activation shown in Figure 3, and subsequently, to the state of activation shown in Figure 4. The state of activity in Figure 3 represents the additional bindings *(owner=Mary, own-object=a-Book)* because *owner* is firing in synchrony with *Mary* and *own-object* is firing in synchrony with *a-Book*. Thus the state of activity in Figure 3 represents not only *give(John,Mary,a-Book)*, but also the inferred fact *own(Mary,a-Book)*. In addition to these two facts, the state of activity in Figure 4 also represents the fact *can-sell(Mary,a-Book)*.

The above example illustrates some significant aspects of SHRUTI:

- An episode of reasoning is a transient propagation of a rhythmic pattern of

[6]In the idealized model we are assuming that each role is represented as a single node and each role correspondence is encoded by a one to one connection between the appropriate role nodes. As discussed in (Shastri & Ajjanagadde 1993), each role is encoded as an ensemble of nodes and each role correspondence is encoded by many to many connections between the appropriate role ensembles.

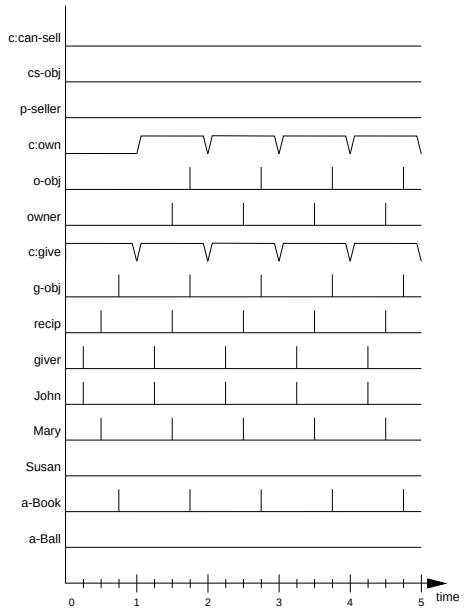

Figure 3: This state of activation represents two dynamic facts: *give(John,Mary,a-Book)* and *own(Mary,a-Book)*. The system has inferred *own(Mary,a-Book)*.

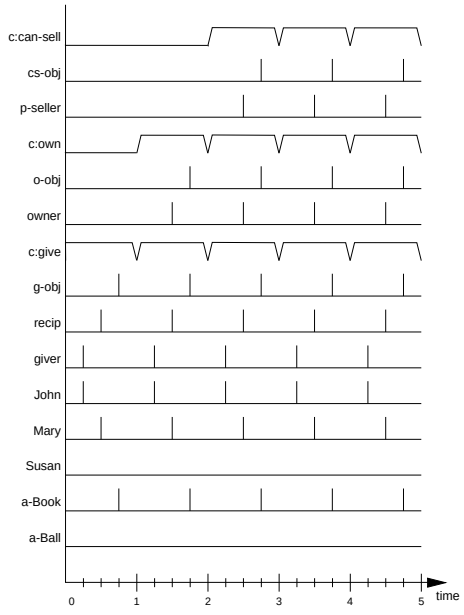

Figure 4: This state represents three dynamic facts: *give(John,Mary,a-Book)*, *own(Mary,a-Book)*, and *can-sell(Mary,a-Book)*. The system has inferred *own(Mary,a-Book)* and *can-sell(Mary,a-Book)*.

activity.

- The transient (short-term) representation of each entity filling a role in an episode is simply a distinct *phase* within a rhythmic pattern of activity
- The number of *distinct* phases within the rhythmic activation pattern only equals the number of *distinct* entities participating as role-fillers in dynamic bindings. This number does not depend on the *total* number of dynamic bindings being represented by the activation pattern.
- The number of distinct entities that can participate in dynamic bindings at the same time is limited by the ratio of (i) the period of the rhythmic activity and (ii) the width of individual spikes.[7]

Observe that reasoning is the spontaneous and natural outcome of the network's behavior. The network does not apply syntactic rules of inference such as *modus-ponens*. There is no separate interpreter that manipulates and rewrites symbols. The network encoding is best viewed as a vivid internal *model* of the agent's environment, where the interconnections between (internal) representations directly encode the dependencies between the associated (external) entities. When the nodes in this model are activated to reflect a given state of affairs in the environment, the model spontaneously simulates the behavior of the external world and in doing so makes predictions and draws inferences.

As discussed at length in (Shastri & Ajjanagadde 1993), there exists substantial neurophysiological evidence to suggest that the propagation of synchronous activity is neurally plausible. A detailed review of synchronous cortical activity appears in (Singer 1993). The idea that synchronous activity can encode feature bindings during visual processing had been suggested by von der Malsburg (1986) (also see Bienenstock & Geman 1995), but SHRUTI is the first detailed model that shows how synchronous activation can be harnessed to solve non-trivial problems in the representation of conceptual knowledge and reasoning.

2.2.6 Parallelism and the significance of structure

The encoding of rules by the explicit encoding of the inferential dependency between predicates and predicate roles, in conjunction with the use of temporal synchrony provides an efficient mechanism for propagating dynamic bindings and performing systematic reasoning. Conceptually, the proposed encoding of rules creates a directed *inferential dependency graph*: Each predicate role is represented by a node in this graph and each rule is represented by links between nodes denoting the roles of the

[7] For simplicity we have been assuming that nodes firing in synchrony fire precisely in-phase. This is an idealization. In general, we expect a coarser form of synchrony where nodes firing with a lag or lead of less than $\omega/2$ of one another are considered to be firing in synchrony. This corresponds to treating the width of the "window of synchrony" to be ω.

antecedent and consequent predicates. In terms of this conceptualization, it should be easy to see that the evolution of the system's state of activity corresponds to a *parallel* breadth-first traversal of the directed inferential dependency graph. This means that (i) a large number of rules can fire in parallel and (ii) the time taken to generate a chain of inference is independent of the total number of rules and just proportional to the *length* of the chain of inference.

The representational and inferential power of SHRUTI and its ability to draw inferences in parallel is directly attributable to its use of structured representations (Feldman et al. 1988; Shastri 1995). A system that uses *fully* distributed representations will be incapable of representing multiple dynamic facts and applying multiple rules simultaneously. Attempts to develop distributed systems to handle relations invariably end up positing several distinct banks — one for each role — thereby stepping away from a fully distributed mode, or fall back on serial processing. It is not surprising that distributed systems such as DCPS (Touretzky & Hinton 1988) have extremely limited capacity for encoding dynamic structures and are serial at the level of rule-application.

2.2.7 Multiple instantiation and type hierarchy

The complete SHRUTI encoding allows a bounded number of instantiations of the same predicates to be active at the same time (Mani & Shastri 93). This allows SHRUTI to represent multiple dynamic facts about the same predicate, and hence, compute inferences involving bounded recursion. Predicate representations are augmented so that each predicate consists of k banks instead of one, and thus can hold up to k dynamic instances (here k is a parameter). Each predicate also has an associated "switching" network which prevents cross-talk between different instantiations.

SHRUTI also supports the representation of a type (category) hierarchy and this allows types as well as instances to occur in rules, facts, and queries. Thus the reasoning system can combine rule-based reasoning with *inheritance* and *classification*. For example, SHRUTI can infer "Tweety is scared of Sylvester", based on the generic fact "Cats prey on birds", the rule "if x preys on y then y is scared of x" and the *is-a* relations "Sylvester is a cat" and "Tweety is a bird". SHRUTI can also use types/categories to encode semantic restrictions on rules. An example of a rule with semantic restrictions is: *x:animate, y:solid-obj walk-into(x,y)* $\Rightarrow$ *hurt(x)*. This rule specifies that the agent in a *walk-into* event gets hurt, if the agent and patient in that event are of the type 'animate' and 'solid-object', respectively.

2.2.8 Encoding long-term facts: Memory as a temporal pattern matcher

Long-term facts are essentially a permanent record of a set of bindings describing a particular situation. The representation of a long-term fact should encode the bindings pertaining to the fact in a manner that allows the system to rapidly recognize dynamic bindings that match the encoded fact. Given that dynamic bindings are

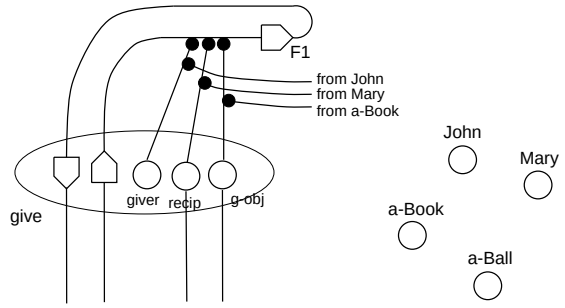

Figure 5: Encoding of a long-term fact. The interconnections shown here encode the *static* bindings corresponding to the fact *give(John,Mary,a-Book)*.

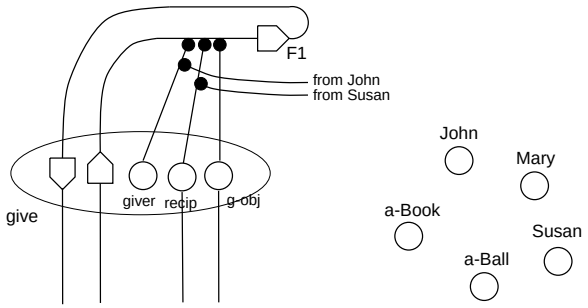

Figure 6: Encoding of the partially instantiated long-term fact *give(John,Mary,x)*, i.e., "John gave Mary something."

represented as temporal patterns, it follows that the encoding of a long-term fact should behave like a *temporal pattern matcher* that becomes active whenever the static bindings it encodes match the dynamic bindings represented in the system's state of activation.

The design of such a temporal pattern matcher is shown in Figures 5 and 6 which illustrate the encoding of the long-term facts *give(John,Mary,a-Book)* and *give(John,Susan,x)*, respectively (the latter means "John gave Susan something"). A long-term fact is encoded using a τ-and node which receives an input from the *enabler* node of the associated predicate. This input is modified by inhibitory links from role nodes of the associated predicate. If a role is bound to an entity, the modifier input from the role node is in turn modified by an inhibitory link from the appropriate entity node. The output of the τ-and node encoding a long-term fact is connected to the *collector* of the associated predicate. We refer to τ-and nodes associated with long-term facts as *fact* nodes. Note that there is only one *enabler* node, one *collector* node, and one set of role nodes for each predicate. These nodes are shared by all the long-term facts pertaining to that predicate.

The encoding of the long-term fact *give(John,Mary,a-Book)* will recognize states of activity that represent dynamic facts such as: *give(John,Mary,a-Book)*, *give(John,Mary,x)*,

give(x,Mary,y), or *give(x,y,z)*. However it will not recognize those states of activity that represent dynamic facts such as *give(Mary,John,a-Book)* or *give(John,Susan,x)*. Similarly, the encoding of the long-term fact *give(John,Susan,x)* will recognize states of activity that encode: *give(John,Susan,x)*, *give(x,Susan,y)*, or *give(x,y,z)*, but not *give(Susan,John,x)* or *give(John,Susan,a-Car)*.

2.3 Backward reasoning system and an example of inference

We briefly describe backward reasoning with SHRUTI which involves the use of systematic rules as well as long-term facts. The network in Figure 7 encodes the following domain knowledge.

give(x,y,z) $\Rightarrow$ *own(y,z)*,
buy(x,y) $\Rightarrow$ *own(x,y)*,
own(x,y) $\Rightarrow$ *can-sell(x,y)*,
give(John,Mary,a-Book),
buy(John,x), and
own(Mary,a-ball).

In the backward reasoning 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 roles of the consequent predicate to the roles of the antecedent predicate in accordance with the correspondence between these roles specified in the rule. Long-term facts are encoded as explained above.

A query is posed to the system by specifying the query predicate and its role bindings. The query predicate is specified by activating its *enabler* with a pulse train of width and periodicity π. Role bindings are specified by activating each entity, and the role nodes bound to that entity, in a distinct *phase*.

We illustrate the reasoning process with the help of an example (refer to Figure 8). Consider the query *can-sell(Mary,a-Book)?* (i.e., Can Mary sell a-Book?) The query is posed by (i) activating the *enabler e:can-sell*, (ii) activating *Mary* and *p-seller* in the same phase, say, ρ_1, and (iii) activating *a-Book* and *cs-obj* in the same phase, say, ρ_2 (ρ_1 and ρ_2 being distinct). As a result of these inputs, *Mary* and *p-seller* fire synchronously in phase ρ_1, while *a-Book* and *cs-obj* fire synchronously in phase ρ_2. The activation from the *can-sell* predicate propagates to the *own*, *give* and *buy* predicates via links connecting these predicates. As a result of this propagation the enablers of *give, own*, and *buy* become active. Furthermore, *Mary*, *p-seller*, *owner*, *buyer* and *recip* become active in phase ρ_1, while *a-Book*, *cs-obj*, *o-obj*, *g-obj* and *b-obj* become active in phase ρ_2. In effect, the system asks itself three more queries—*own(Mary,a-Book)?*, *give(x,Mary,a-Book)?* (i.e., Did *someone* give Mary a-Book?), and *buy(Mary,a-Book)?*. The τ-and node F1, associated with the fact

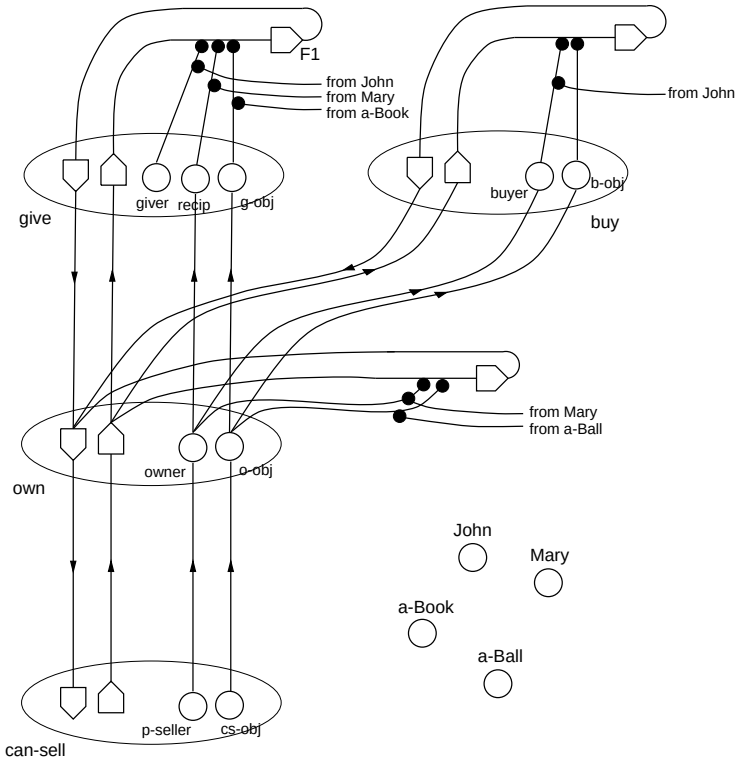

Figure 7: An example encoding of rules and facts for backward reasoning.

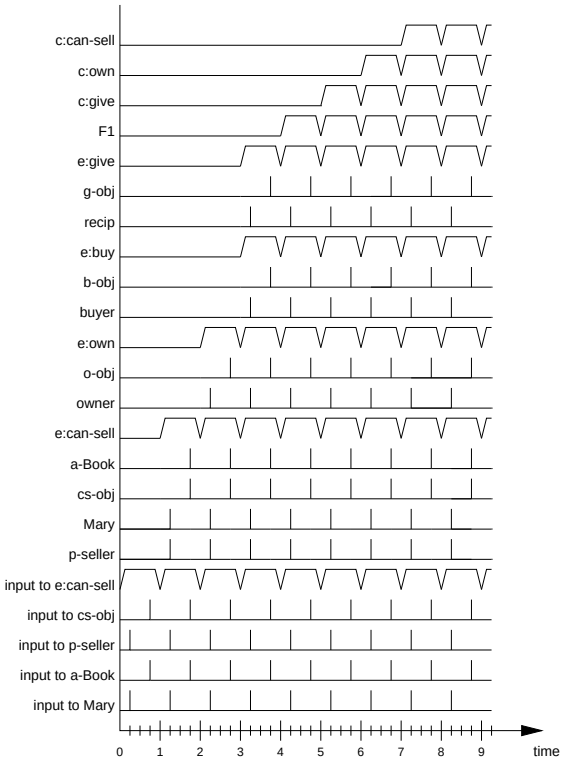

Figure 8: Activation trace for the query *can-sell(Mary,a-Book)?* The query is posed by activating the enabler and role nodes of *can-sell* and the nodes *Mary* and *a-Book* as shown.

give(John,Mary,a-Book) matches the query *give(x,Mary,a-Book)?* and becomes active. Observe that the role *giver* is not firing and the inhibitory inputs from the roles *recip* and *g-obj* are blocked by the synchronous inputs from the fillers *Mary* and *a-Book*, respectively. Therefore, F1 receives uninterrupted activation from *e:give* and becomes active. F1 activates *c:give* and this in turn activates *c:own* and *c:can-sell*. The activation of *c:can-sell* signals an affirmative answer to the initial query *can-sell(Mary,a-Book)?*.

2.4 Constraints and predictions

SHRUTI identifies a number of representational and processing constraints on reflexive processing. These relate to (i) the capacity of the "working memory" underlying reflexive processing, (ii) bounds on the depth of reasoning and differences in the time course of associative priming versus systematic reasoning and (iii) the form of rules that may participate in reflexive processing. These constraints are specific to SHRUTI and differentiate if from other connectionist models of memory and reasoning such as CONPOSIT (Barnden & Srinivas 1991), ROBIN (Lange & Dyer 1989), and CONSYDERR (Sun 1992).

Working memory underlying reflexive processing: Dynamic bindings, and hence, dynamic (active) facts are represented in SHRUTI as a rhythmic pattern of activity over nodes in the LTM network. In functional terms, this transient state of activation holds information temporarily during an episode of reflexive reasoning and corresponds to the *working memory underlying reflexive reasoning* (WMRR). Note that WMRR is just the state of activity of the LTM network and not a separate buffer. Also note that the dynamic facts represented in the WMRR during an episode of reflexive reasoning should not be confused with the small number of short-term facts an agent may *overtly* keep track of during *reflective* processing and problem solving. WMRR should not be confused with the short-term memory implicated in various memory span tasks (Baddeley 1986). In our view, in addition to the overt working memory, there exist as many "working memories" as there are major processes in the brain since a "working memory" is nothing but the state of activity of a network.

SHRUTI predicts that the capacity of WMRR is very large but at the same time it is constrained in critical ways (see below). Modulo these constraints, the number of dynamic facts that can be co-active in the working memory equals $k * R$, where k is the multiple instantiation constant and R is the number of relations known to the agent.

Most proposals characterizing the capacity of the working memory underlying cognitive processing have not paid adequate attention to the structure of items in the working memory and their role in processing. Even the recent proposal of Just & Carpenter (1992) characterizes working memory capacity in terms of "total activation". In contrast, the constraints on working memory capacity predicted by SHRUTI depend not on total activation but rather on the maximum number of *distinct* entities

that can participate in dynamic bindings simultaneously, and the maximum number of (multiple) instantiations of a predicate that can be active simultaneously.

Bound on the number of distinct entities referenced in WMRR During an episode of reflexive reasoning, each entity involved in dynamic bindings occupies a distinct phase in the rhythmic pattern of activity. Hence the number of distinct entities that can occur as role-fillers in the dynamic facts represented in the working memory cannot exceed π_{max}/ω where π_{max} is the maximum delay between two consecutive firings of cell-clusters involved in synchronous firing and ω equals the width of the window of synchrony — i.e., the maximum allowable lead/lag between the firing of synchronous cell-clusters. If we assume that a neurally plausible value of π_{max} is about 30 milliseconds and a conservative estimate of ω is around 6 milliseconds, we are led to the following prediction: As long as the number of distinct entities referenced by the dynamic facts in the working memory is five or less, there will essentially be no cross-talk among the dynamic facts. If more entities occur as role-fillers in dynamic facts, the window of synchrony ω would have to shrink appropriately in order to accommodate all the entities. As ω shrinks, the possibility of cross-talk between dynamic bindings would increase until eventually, the cross-talk would become excessive and disrupt the system's ability to perform systematic reasoning. The exact bound on the number of distinct entities that may fill roles in dynamic facts would depend on the largest and smallest feasible values of π_{max} and ω, respectively. However we can safely predict that the upper bound on the maximum number of entities participating in dynamic bindings can be no more than 10 (perhaps less).

Bound on the multiple instantiation of relations: The capacity of WMRR is also limited by the constraint that at most k dynamic facts pertaining to each relation may be active at any given time (recall that the total number of active dynamic facts can be very high). In general, the value of k need not be the same for all relations; some critical relations may have a higher value of k while some other relations may have a smaller value. The cost of maintaining multiple instantiations turns out to be significant in terms of space and time. For example, the number of nodes required to encode a rule for backward reasoning is proportional to the square of k. Thus a system that can represent three dynamic instantiations of each relation may have up to nine times as many nodes as a system that can only represent one instantiation per relation. Furthermore, the worst case time required for propagating multiple instantiations of a relation also increases by a factor of k. In view of the additional space and time costs associated with multiple instantiation, and given the necessity of keeping these resources within bounds in the context of reflexive processing, we predict that the value of k is quite small, perhaps no more than 3.

Bound on the depth of the chain of reasoning: Consider the propagation of synchronous activity along a chain of role ensembles during an episode of reflexive reasoning. Two things might happen as activity propagates along the chain of role ensembles. First, the lag in the firing times of successive ensembles may gradually build up due to the propagation delay introduced at each level in the chain. Second, the

dispersion within each ensemble may gradually increase due to the variations in the propagation delay of links and the noise inherent in synaptic and neuronal processes. While the increased lag along successive ensembles will lead to a "phase shift", and hence, binding confusions, the increased dispersion of activity within successive ensembles will lead to a gradual *loss of binding information.* Increased dispersion would mean less phase specificity, and hence, more *uncertainty* about the role's filler. Due to the increase in dispersion along the chain of reasoning, the propagation of activity will correspond less and less to a propagation of role bindings and more and more to an associative spread of activation. For example, the propagation of activity along a chain of rules such as: $P_1(x, y, z) \Rightarrow P_2(x, y, z) \Rightarrow ...P_n(x, y, z)$ due to a dynamic fact $P_1(a, b, c)$ may lead to a state of activation where all one can say about P_n is this: there is an instance of P_n which involves the entities a, b, and c, but it is not clear which entity fills which role of P_n. In view of the above, it follows that the depth to which an agent may reason during reflexive reasoning is bounded. In other words, an agent may be unable to make a prediction (or answer a query) — even when the prediction (or answer) logically follows from the knowledge encoded in the LTM — if the length of the derivation leading to the prediction (or the answer) exceeds this bound.

Form of rules that may participate in reflexive reasoning: Using complexity theory it can be shown that during backward reasoning (i.e., query answering) it is not possible to make use of rules containing equality constraints among antecedent roles unless (i) such roles map to a consequent role in the rule and (ii) the consequent role gets bound during the query answering process. A similar constraint applies to forward (predictive) reasoning. These constraints predict that certain queries cannot be answered in a reflexive manner even though the corresponding predictions can be made reflexively. For example, consider an agent whose LTM includes the rule "if x loves y and y loves z then x is jealous of z", and the long-term facts "John loves Mary" and "Mary loves Tom". We predict that if this agent is asked "Is John jealous of Tom?", she will be unable to answer the query in a *reflexive* manner. Note that the antecedent of the rule includes the equality condition: the second role of one instance of 'loves' should equal the first role of the other instance of 'loves'. Hence, answering this question will require deliberate and conscious processing unless the relevant long-term facts are active in the WMRR for some reason at the time the query is posed. However, an agent who has the above rule about love and jealousy in its LTM would be able to infer "John is jealous of Tom" in a reflexive manner, on being told "John loves Mary" and "Mary loves Tom".

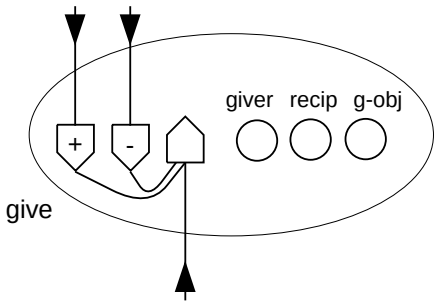

Figure 9: The structure of a predicate cluster augmented to deal with negated information

3 Extending SHRUTI to deal with negation

3.1 Modifications in the core predicate component

The treatment of negation involves a systematic but simple augmentation of the basic SHRUTI model (refer to Figure 9). The primary change is in the representation of the core component of a predicate. Instead of a single collector, the extended encoding of a predicate P now consists of two collectors: one associated with positive assertions about P and the other with negated assertions about P. We label these collectors $+c{:}P$ and $-c{:}P$ respectively. The enabler node and role nodes remain unchanged. A second change is that both the collectors $+c{:}P$ and $-c{:}P$ connect to the enabler $e{:}P$ via a weighted link.

As before: (i) each role node of P encodes one of the roles of P, and its synchronous activation with a filler encodes the dynamic binding of the role with that filler, (ii) the activation of the enabler $e{:}P$ indicates that the system (or an external process) is asking whether the currently active instance of P (composed of the dynamic bindings of the roles of P) is supported by the system's knowledge, and (iii) each collector node of P represents the system's response to the currently active instance of P — but with an important distinction. The system activates the positive collector $+c{:}P$ if the currently active instance of P is supported by the system's knowledge, but it activates the negative collector $-c{:}P$ if the negation of the currently active instance of P is supported by the system's knowledge. Thus if the system knows that John bought a car, it activates $+c{:}P$ when presented with the bindings *buyer=John,buy-obj=car*. But it activates $-c{:}P$ when presented with the same bindings if it knows that John did not buy a car. If the system does not know anything about John buying a car it activates neither $+c{:}P$ nor $-c{:}P$ thereby indicating "don't know".

3.1.1 Significance of collector to enabler connections

The links between the collectors and the enabler of a predicate serve an important role during reasoning. These links essentially convert any dynamic assertion into a

query about the assertion! This means that the system is always seeking support (or an explanation) for incoming facts. It is as though the system constantly "evaluates" incoming knowledge in the context of existing knowledge and seeks to affirm or reject it based on what it knows. The weights on links from collectors to enablers can be viewed as a measure of the system's propensity for seeking such evaluations. A system with a high weight on these links can be viewed as a highly critical and skeptical system, while one with very low weights can be viewed as a credulous system — one which accepts incoming information without actively seeking an explanation or determining how well it coheres with prior knowledge.

As we shall see, the ability of a system to evaluate incoming information gives it the ability to detect inconsistencies between incoming information and prior knowledge. The scope of inconsistency detection is however, *local*. In other words, inconsistencies are detected only between two facts that are within a limited inferential distance from each other. This bound on inferential distance is governed by the constraint on the depth of a chain of inference mentioned in Section 2.5. Observe that we are referring to a reflexive process of evaluation and not a deliberate search for explanations. Thus while this process would be very fast and automatic, it would be subject to the constraints on reflexive reasoning which include bounds on the depth of inference.

Finally, the links from the collectors of a predicate to its enabler serve to create positive feedback loops of spreading activation and thereby create stable coalitions of active nodes under appropriate circumstances. Assume that the system is seeking an explanation about the currently active instance of P, and therefore, the enabler of P is active. If the memory supports this instance of P it will activate the positive collector of P. This will create a feedback loop — or a stable coalition — consisting of *e:P*, the enablers of other predicates participating in the explanation of P, the appropriate collectors of these predicates, *+c:P*, and *e:P*.

3.1.2 Encoding of facts

The encoding of facts remains virtually unchanged. The only difference between positive facts and negated ones is that while a fact node associated with a positive fact about P feeds into *+c:P*, a fact node associated with a negated fact about P feeds into *–c:P*. Figure 10 shows the encoding of *loves(John,Mary)*, and ¬*loves(Susan,Tom)*. Observe that both the positive and negated facts make use of the same enabler and role nodes. The only difference lies in the collector that these fact nodes feed back into. Given queries such as *loves(John,Mary)*?, *loves(x,Mary)*?, or *loves(John,x)*?, the fact node F2 will become active and activate the collector *+c:love* indicating a "yes" answer. Similarly, given queries such as *loves(Susan,Tom)*?, *loves(x,Tom)*?, or *loves(Susan,x)*?, the fact node F3 will activate the *–c:P* collector indicating a "no" answer. Finally, given a query such as *loves(John,Susan)?*, neither *+c:love* nor *–c:love* would become active, indicating that the system has no knowledge to affirm or deny whether John loves Susan.

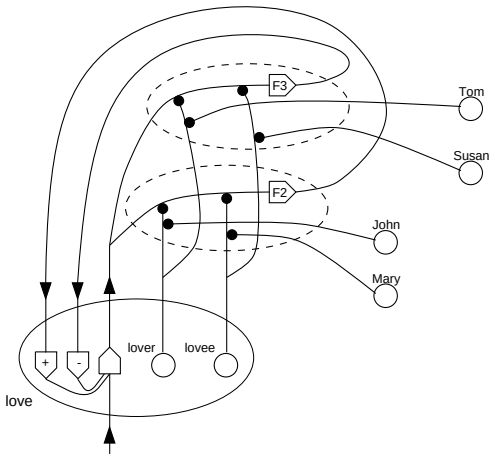

Figure 10: Example encoding of facts. The network encodes the fact *love(John,Mary)* and the negated fact $\neg$*love(Tom,Susan)*.

3.1.3 Encoding of rules

Rules in the extended system are also encoded as before except that the links from the collectors of antecedent predicates to the collector of the consequent predicate originate from $+c{:}P$ if P appears in its positive form in the antecedent and $-c{:}P$ if P appears in a negated form. Similarly, such links terminate at $+c{:}Q$ if Q appears in a positive form in the consequent of the rule, and at $-c{:}Q$ if Q appears in a negated form. Figure 11 shows the encoding of the rule $\neg citizen(x,y) \Rightarrow \neg vote(x,y)$ ("x cannot vote in y's election if x is not a citizen of y") while Figure 12 shows the encoding of the rule $bachelor(x) \Rightarrow \neg married(x,y)$ ("x is not married to any y if x is a bachelor").

Observe that the system does not encode the contrapositive of a rule by fiat. In our model, a rule and its contrapositive are two *distinct* rules. Given a rule in the LTM, its contrapositive form may, or may not, be present in the LTM.

The encoding of a rule in the augmented system supports the use of weighted links from the enabler of the consequent predicate to the enabler(s) of the antecedent predicate(s), and from the collector(s) of the antecedent predicate(s) to the collector of the consequent predicate. These weights lead to a gradual weakening of activation along a chain of inference during reflexive reasoning. Thus the activation level of enabler nodes get progressively lower as inference propagates along a chain of predicate banks until eventually it falls below threshold and terminates the chain of inference. The value of the weight and the threshold determine the bound on the depth of inference during reflexive reasoning. The weight on the collector to collector links have a similar effect. These weights can also be given an evidential or probabilistic interpretation and used to model the strength of a rule.

The encoding of rules with special conditions (e.g., the handling of constant role fillers, repeated variables, and existentially quantified variables), multiple antecedent

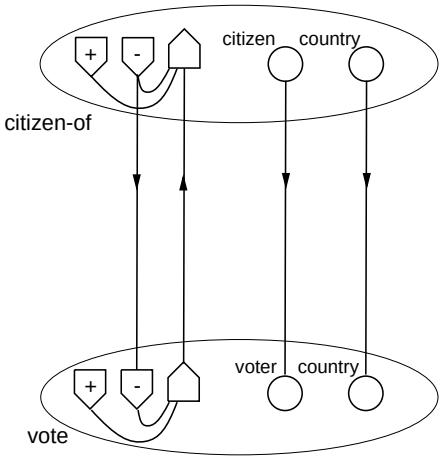

Figure 11: Example encoding of rules involving negated predicates. The network encodes the rule: $\neg citizen(x,y) \Rightarrow \neg vote(x,y)$

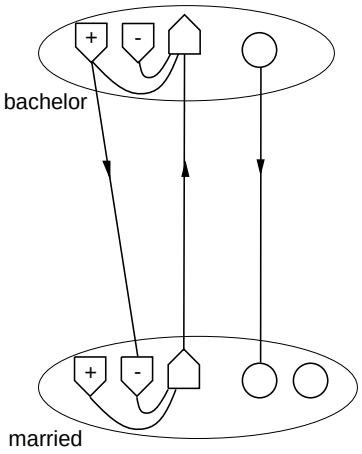

Figure 12: Encoding of the rule: $bachelor(x) \Rightarrow \neg married(x,y)$

rules, and multiple instantiation of predicates remains otherwise unchanged. In particular, the structure of the multiple instantiation banks and the multiple instantiation switch remains unchanged except that, analogous to the change in the predicate representation, single collector nodes are replaced by a pair of positive and negative collector nodes in the arbitrator of each switch (see (Mani & Shastri 1993) for details of the encoding of a switch).

The solution to the problem of negation and the treatment of inconsistency proposed above is considerably simpler than the one suggested in (Cottrell 1993). The latter had suggested that the entire predicate bank be duplicated for each predicate. In this scheme, each predicate P would have two separate banks of role, enabler and collector nodes: one for positive knowledge about P ($+P$), and another for negative knowledge about P ($-P$). In addition to requiring duplicate banks, this scheme would also require a mechanism for comparing bindings across the $+P$ and $-P$ banks in order to determine that a dynamic instance in the $+P$ bank contradicts a dynamic instance in the $-P$ bank. As explained below, the detection of inconsistency in our proposal is extremely simple since it only requires a comparison of the positive and negative collector activations within a single bank.

3.2 Detecting contradictions and inconsistencies

The system can evaluate answers and detect contradictions using an extremely simple (four node) circuit within each predicate cluster. Such a circuit receives three inputs: one from each of the two collectors and another from the enabler, and produces four outputs. The outputs are: "don't know", "yes", "no", and "contradiction". A "don't know" is produced if the enabler is active but neither of the collectors is active. A "yes" is produced if the positive collector is active but the negative collector is inactive. A "no" is produced if the *negative* collector is active and the positive collector is inactive. Finally, a "contradiction" is indicated if both the positive and the negative collectors are active.

The information available at the positive and negative collectors of a predicate, however, can be treated in a more flexible manner by viewing their activity not as a discrete on and off state, but rather as a graded level of activation. In particular, the activity of the positive and negative collectors of a predicate can be combined using a suitable evidence combination rule in order to evaluate the effective degree of support offered by the system to the currently active predicate instance. The result of this combination yields a graded belief ranging from "no" on the one extreme to "yes" on the other, with "don't know" in between. Such an evidential combination can be realized very simply via mutual inhibition between the positive and negative collectors of a predicate, and this is the approach we have adopted in the current implementation. Note that this only requires *local* inhibition *within* a predicate bank, and not across predicates.

If both the collectors receive comparable and strong activation then both collectors

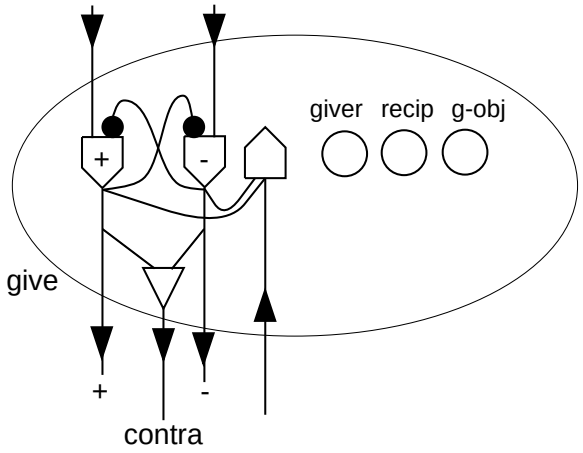

Figure 13: Encoding of a predicate structure augmented to combine evidence and detect contradictions.

can be in a high state of activity, in spite of the mutual inhibition between them. When this happens, a contradiction is detected. In the current implementation, a contradiction occurs whenever the combined activity level of the two collectors exceeds 1.5. This is realized by having one additional node within each predicate cluster. This node has a threshold of 1.5 and receives excitatory inputs from both the collectors. If this node becomes active, it means that a contradiction has occurred at the predicate.

To summarize, graded combination of positive and negative evidence and the detection of contradiction at each predicate can be realized by a simple modification of the core component of a predicate. This involves two additions: (i) the positive and negative collectors of a predicate mutually inhibit each other and (ii) the two collectors are connected to a "contradiction" node having a threshold of 1.5. Figure 13 shows the augmented predicate structure.

4 Two examples

In this section we present two examples that illustrate the treatment of negated information, detection of inconsistency, and the interaction of positive and negative evidence. The examples have been simplified as much as possible in order to focus on the key properties of the model.

4.1 Detecting inconsistency

Assume that the system has the following rule and fact in its LTM:

- "if you are not a citizen of a country, you cannot vote in the elections of that country"

- "John is not a citizen of USA"

i.e., $\neg citizen(x,y) \Rightarrow \neg vote(x,y)$, and $\neg citizen(John,USA)$.[8] Please refer to Figure 11.

The system is told "John voted in the elections in USA." This is communicated by activating the positive collector *+c:vote* and establishing the dynamic bindings *(voter=John, country=USA)* so as to assert *vote(John,USA)*. Activation now propagates from *+c:vote* to the enabler *e:vote* via the collector to enabler link. Because of the rule $\neg citizen(x,y) \Rightarrow \neg vote(x,y)$, the activation propagates from *e:vote* to *e:citizen* and at the same time from the role nodes of *vote* to the role nodes of *citizen*. This establishes the query *citizen(John,USA)?* At this time, the fact $\neg citizen(John,USA)$ matches the query and activates the negative collector *–c:citizen*. In turn, the activation from *–c:citizen* propagates down to *–c:vote-in-election* and activates it. Thus both the *positive* and *negative* collectors of *citizen* become active and signal a contradiction. This example illustrates how the system can not only represent negated knowledge, but also use such knowledge to detect contradictions in new and existing knowledge.

Inconsistencies in existing knowledge are also detected in an analogous manner when inconsistent knowledge is activated. This can happen during the processing of a query or during the assimilation of new information. For example, assume that the following (inconsistent) knowledge resides in the LTM:

1. $P(x,y) \Rightarrow R(x,y)$
2. $Q(x,y) \Rightarrow \neg R(x,y)$
3. $P(a,b)$
4. $Q(a,b)$

Now assume that the execution of some cognitive task results in the query *R(a,b)?* to the memory and reasoning system. As a result of rules (1) and (2), this query leads to the queries *P(a,b)?* and *Q(a,b)?* (see section 2.3). The facts (3) and (4) match the two queries, respectively, and activate *+c:P* and *+c:Q*. These collectors in turn activate *+c:R* and *–c:R* respectively. The activation of the positive and negative collectors of *R* leads to the detection of a contradiction. Thus the proposed encoding allows inconsistent knowledge to reside in the agent's memory, but detects an inconsistency whenever the agent tries to bring some inconsistent knowledge to bear on a particular task.

[8] We are simplifying matters in order to focus on the point at hand. A more realistic version of the rule would be $person(x) \wedge country(y) \wedge \neg citizen(x,y) \Rightarrow \neg vote(x,y)$ together with the *is-a* assertions: *is-a(John,Person)* and *is-a(USA,Country)*. The use of this more complex rule does not change the system behavior illustrated in the example.

4.2 The Post Office example

We describe a simulation of the Post Office Example introduced in Section 1.2. The simulation illustrates how an agent may sometimes overlook relevant information and act in an erroneous manner. More focused evaluation — or an appropriate cue, however, can make the relevant information accessible and lead to the correct response. The example also illustrates the interaction of default information with categorical information.

Let us model the agent's domain knowledge as follows (also refer to Figure 14):

1. *presidents-day(day)* $\Rightarrow$ *federal-holiday(day)*
2. *3rd-Mon-Feb(day)* $\Rightarrow$ *presidents-day(day)*
3. *3rd-Mon-Feb(20-Feb-95)*
4. $\neg$*3rd-Mon-Feb(21-Feb-95)*
5. *weekday(day)* $\wedge$ *post-office(x)* $\Rightarrow$ *open(x,day)* — [medium weight]
6. *weekend(day)* $\wedge$ *post-office(x)* $\Rightarrow$ $\neg$*open(x,day)*
7. *federal-holiday(day)* $\wedge$ *post-office(x)* $\Rightarrow$ $\neg$*open(x,day)*
8. *post-office(PO)*
9. *federal-holiday(4th-of-July)*

The first item encoded in the LTM states that Presidents' Days are federal holidays. The next item specifies that third Mondays in February are Presidents' Days. Ideally *3rd-Mon-Feb* would be realized as a mental process which can determine whether a given day is the third Monday in February. We indirectly simulate such a procedure by assuming that there is a mental process that can be accessed by the predicate *3rd-Mon-Feb* in order to determine whether the day bound to its role is a third Monday in February. For the purposes of this example, this mental "calendar" consists of two facts (items 3 and 4) which indicate that 20th February 1995 is a third Monday in February while 21st February 1995 is not. The next item states that post offices generally remain open on weekdays. Items 6 and 7 state that post offices are not open on weekends and federal holidays, respectively. Item 8 states that PO is a particular post office (supposedly the local post office visited by the agent). The last item captures the knowledge that the 4th of July is a federal holiday. Notice that *4th-of-July* has a special status since it is deemed to be a salient federal holiday. It has been encoded as an explicit type and individual fourths of July (such as 4th July 1995) can be viewed as its instances. Items 1, 2, 6, and 7 are categorical rules about

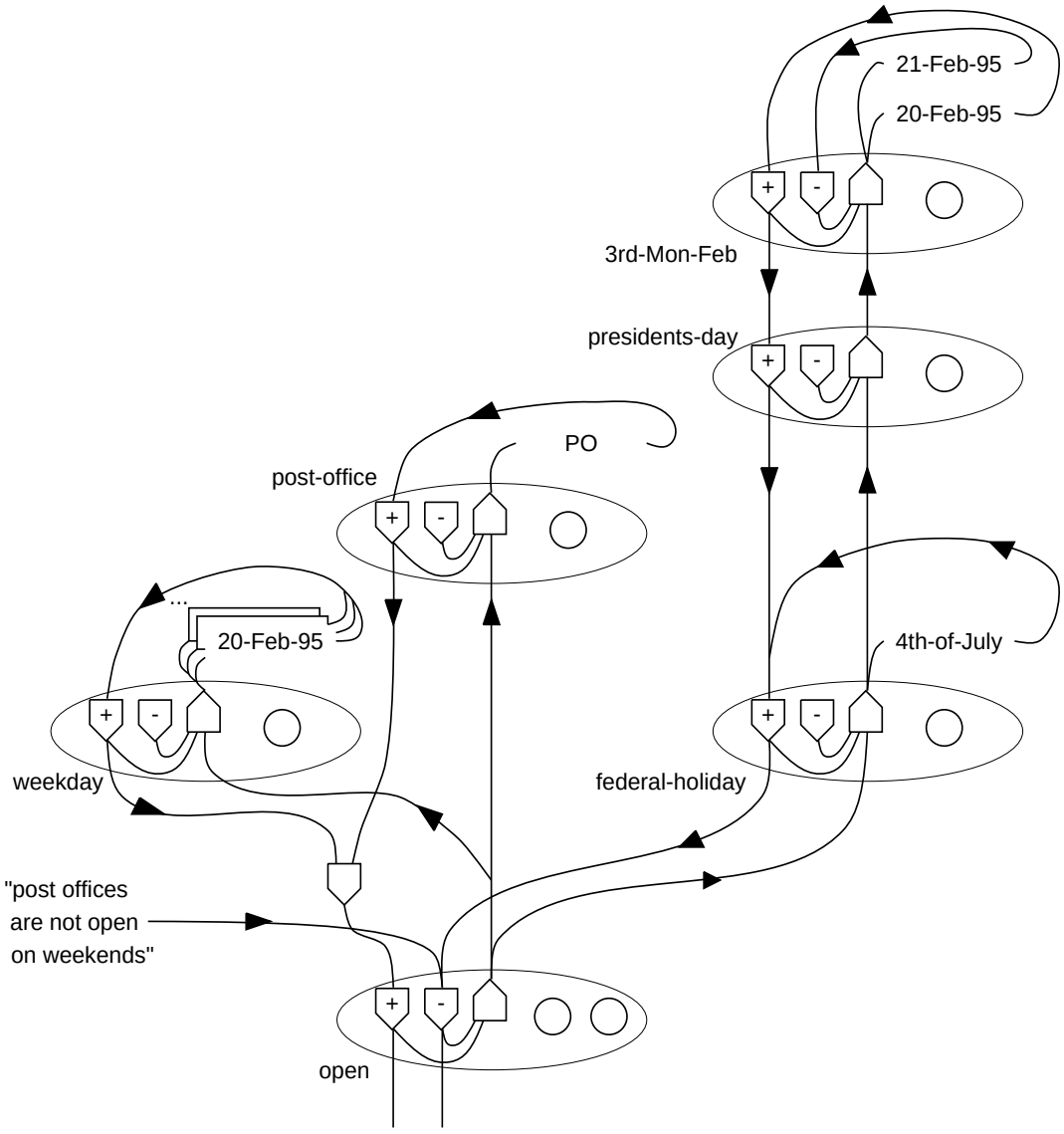

Figure 14: A graphical representation of some of the knowledge encoded in the Post Office Example. Links between role nodes are not shown to reduce clutter. For the same reason, the relation *weekend* and the encoding of the rule about post offices not being open on weekends has not been elaborated. The detailed encoding of facts is also not shown — the latter are shown simply as rectangles. The fact about *weekday* shown in the foreground is meant to indicate the binding between "Today" and the attribute "weekday". Facts such as *3rd-Mon-Feb(20-Feb-95)* and $\neg$*3rd-Mon-Feb(21-Feb-95)* are intended to simulate the effect of a mental process that can determine whether a given day is a third Monday of February or not. The fact *federal-holiday(4th-of-July)* is stored explicitly since 4th of July is a salient federal holiday.

the domain and have a high weight, but item 5 corresponds to default and defeasible information and hence, has a medium weight.[9]

We assume that "Today" is a concept which is bound each day to a different "date" and to "weekday" or "weekend" depending on the day. These bindings are assumed to be available as facts in the agent's memory.

Imagine that today is 20th February 1995 which is Presidents' Day. Our agent Mary works for an organization that does not observe Presidents' Day as a holiday and so she is not explicitly aware that today is Presidents' Day. Mary is out of postage stamps and is contemplating a quick trip to the nearby post office (PO).

Let us assume that her "go-to-post-office" schema has the precondition that the post office must be open. So before deciding to head to the post office, the schema poses the query *open(PO, Today)?* to the memory and reasoning system. Assume that after posing the query the schema monitors the activity of the collectors *+c:open* and *−c:open* and accepts an answer based on the criterion described below. Once it accepts an answer, the schema terminates the query and proceeds with its execution.

> The schema accepts a "yes" ("no") answer if the positive (negative) collector stays ahead of the competing collector and exceeds a threshold, θ_{accept}, for some minimum length of time, Δ_t.

Since today is 20th February 1995, "Today" is bound to *20-Feb-95* and the fact *weekday(20-Feb-95)* is present in Mary's memory. When the "go-to-post-office" schema asks the query *open(PO, Today)?*, the default rule about post offices remaining open on weekdays becomes active first and activates the positive collector *+c:open* (refer to Figures 14 and 15). If we assume θ_{accept} to be 0.5, the activation of *+c:open* exceeds the threshold after 12 cycles and stays above threshold for about 20 cycles (note that the vertical activation axis in Figure 15, and in subsequent figures, has been scaled by a factor of 1000). During this time, the negative collector does not receive any activation and stays at 0. If the schema uses a Δ_t of say, 10 cycles, it will accept *+c:open* as an answer and withdraw the query. So Mary will set off to the post office. The parameter values chosen in the simulation are only illustrative and meant to convey some of the qualitative aspects of the situation.

Had the query remained active, the inference process would have eventually inferred that the post office is not open today since it is the third Monday of February which is Presidents' Day, which is a federal holiday, and post offices are not open on federal holidays. The result of the inferential process, if the query *open(PO, Today)?* had not been terminated by the schema, is shown in Figure 16. The top panel shows the activation of the two collectors of *open* while the bottom panel shows the activations of the collectors of some other relevant predicates. The propagation of inference can be traced by noting the time delay in the activation of the various collectors.

[9] In the current implementation, default rules have a weight of 0.70 while categorical rules have a weight of 1.

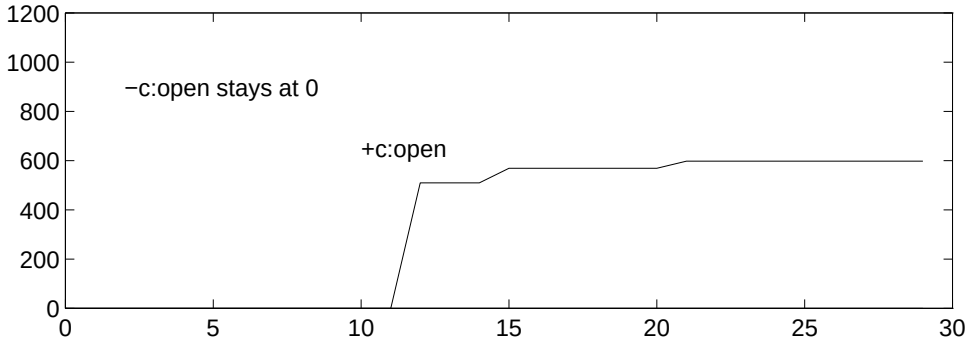

Figure 15: The activation trace for the query *open(PO,today)?*; today being 20th February 1995. The vertical axis denotes activation level and has a scale factor of 1000. The horizontal axis denotes number of simulation steps.

So first it is inferred that Today (20th February 1995) is a weekday. After a significant delay it is also inferred that today is the third Monday in February. This delay occurs because of the inferential distance between *open* and *3rd-Mon-Feb* (refer to Figure 14). Once it is inferred that today is the third Monday in February, the inference about today being Presidents' Day and a federal holiday follow and in turn lead to the inference that the post office, PO, is not open.

Imagine that subsequently, John asks Mary the question "Isn't today Presidents' Day?". This means that *e:Presidents-day* is activated directly and the role of Presidents' Day is bound to "Today" (i.e., *20-Feb-95*). This causes the immediate activation of *e:3rd-Mon-Feb* and subsequently of *+c:3rd-Mon-Feb* via the fact *3rd-Mon-Feb(20-Feb-95)*. The activation from *+c:3rd-Mon-Feb* works its way back and activates *–c:open*. Since this activation is due to categorical rules (rules 2, 1, and 7), it is stronger than that arriving at *+c:open* from the default rule (item 5). Thus, although the activation at *+c:open* arrives before it does at *–c:open*, the mutual inhibition between the more highly activated *–c:open* and the moderately activated *+c:open* results in the suppression of *+c:open*, making Mary realize that the post office is not open today (see Figure 17).

Now consider two other situations in which the query "Is the post office open today?" is posed. The first situation occurs on 4th July 1995 and the second on 21st February 1995. Both these days are also weekdays, and hence, the appropriate *weekday(Today)* fact would be asserted on these days.

The activation trace of the two collectors for *open* in the first situation (4th July 1995) is shown in Figure 18. In this situation, the information that 4th of July is a federal holiday is stored explicitly as a fact and becomes available earlier. Thus its effect reaches *–c:open* before the default rule about post offices being open on

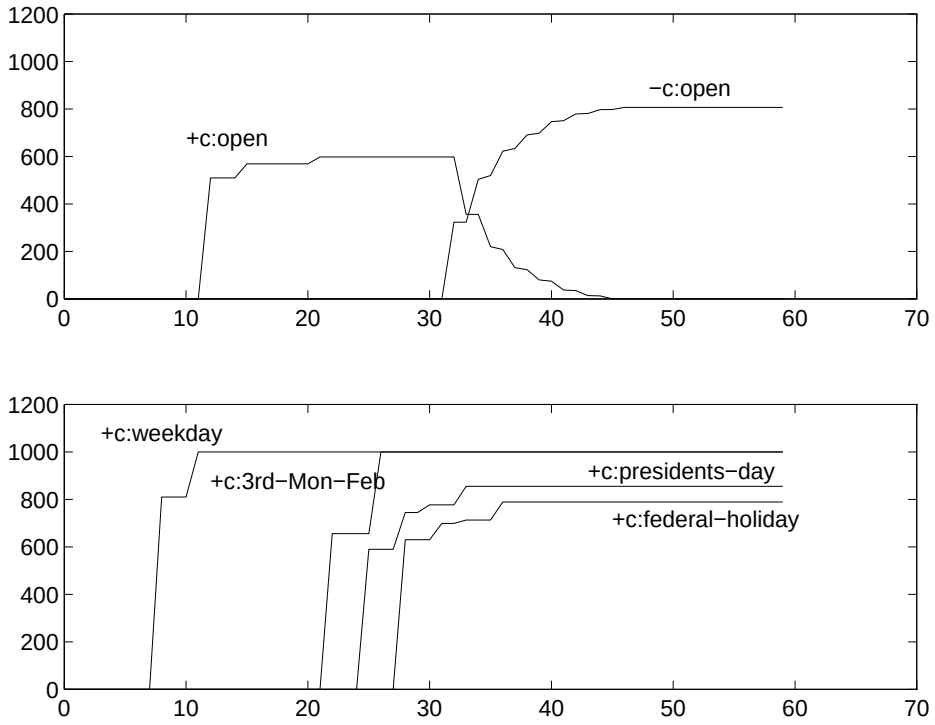

Figure 16: The activation trace for the query *open(PO,today)?* — today being 20th February 1995 — allowed to run its full course.

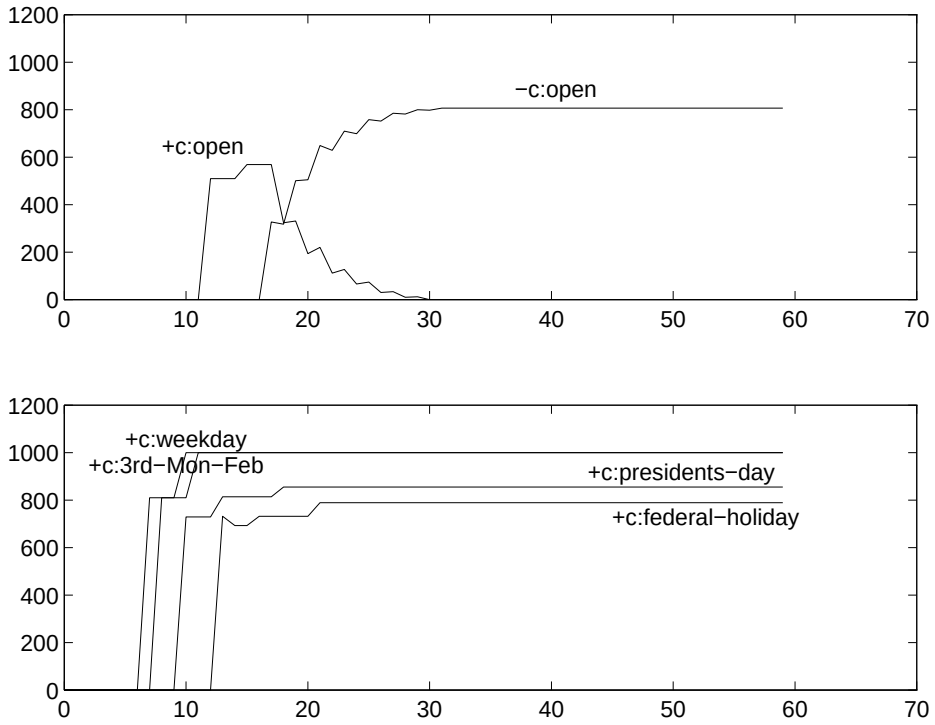

Figure 17: The activation trace for the query "Isn't today Presidents' Day?" posed to the agent going to the post office on 20th February 1995.)

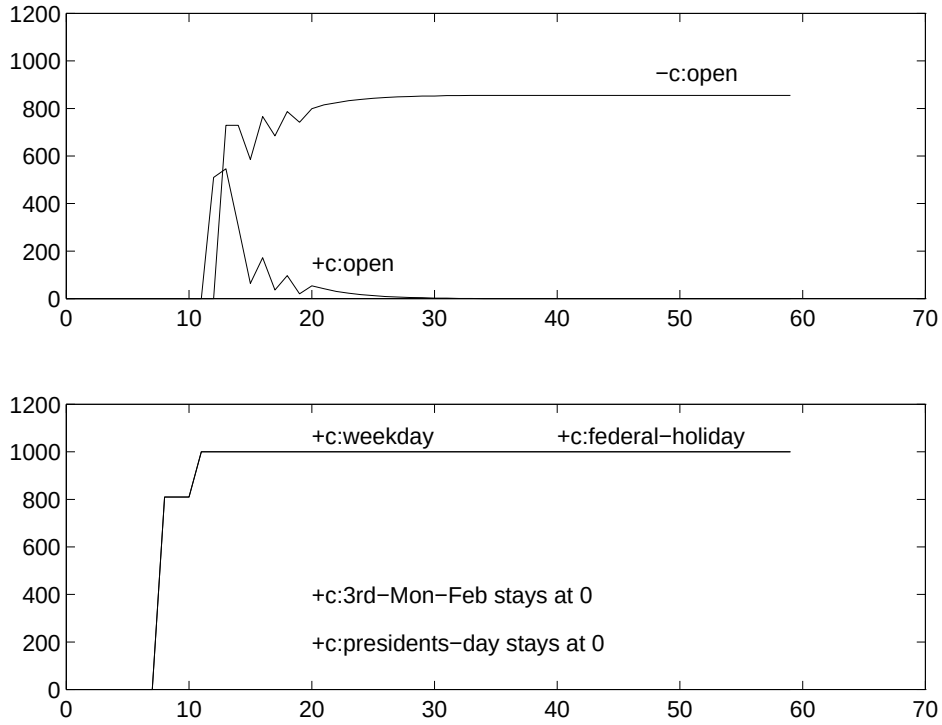

Figure 18: The activation trace for the query *open(PO,today)?*; today being 4th July 1995.

weekdays gets a chance to establish *+c:open* as the answer. Thus the activating schema would accept "no" as the answer to the question "Is the post office open today?".

The trace for the second situation (21st February 1995) is shown in Figure 19. Here the default rules causes *+c:open* to become active, and this activation is never countered by any activation arriving at *–c:open*. Thus the activating schema would accept "yes" as the answer to the question "Is the post office open today?".

5 Conclusion

This report describes an extension to SHRUTI that allows it to deal with positive knowledge as well as negated facts and systematic knowledge (rules) involving negated antecedents and consequents. The extension only requires *local* inhibitory connections. The extended model explains how an agent can hold inconsistent knowledge in its long-term memory without being "aware" that its beliefs are inconsistent, but detect a contradiction when two contradictory beliefs that are within a small infer-

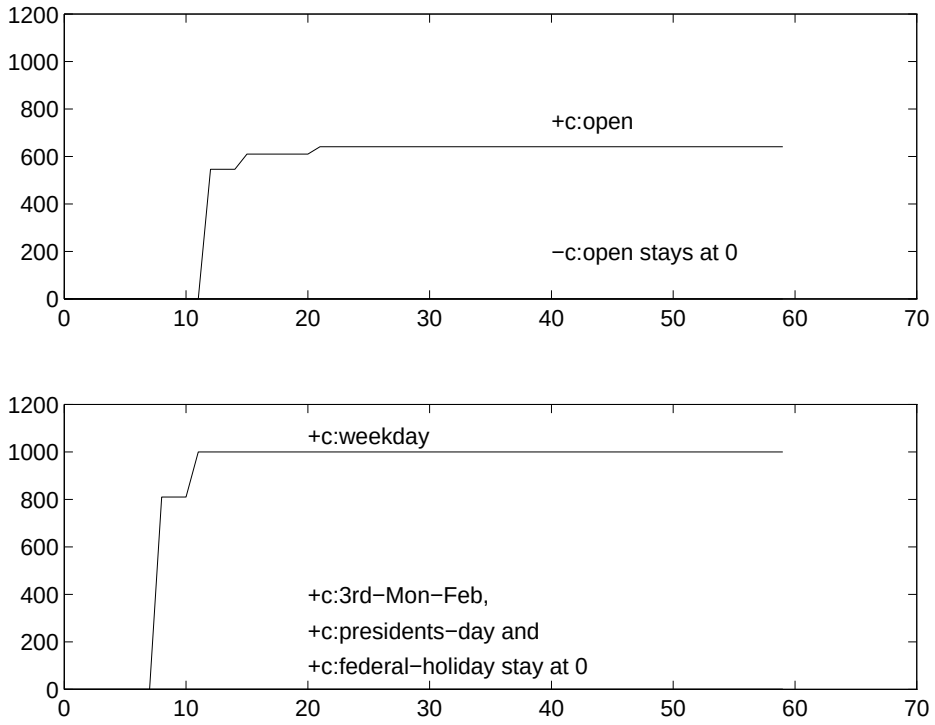

Figure 19: The activation trace for the query *open(PO,today)?*; today being 21st February 1995.

ential distance of each other become co-active during an episode of reasoning. Thus the model is not logically omniscient, but detects contradictions whenever it tries to make *use* of inconsistent knowledge in particular situations. The extended model also explains how limited attentional focus or action under time pressure can lead an agent to produce an erroneous response. The extended SHRUTI model is therefore capable of modeling a wider range of reflexive reasoning phenomena. A more detailed treatment of default information, especially the interaction between multiple default rules and categorical rules is under investigation.

References

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

Bienenstock, E. & Geman, S. (1995) Compositionality in Neural Systems. In *The Handbook of Brain Theory and Neural Networks* ed. M.A. Arbib. MIT Press.

Baddeley, A. (1986) *Working Memory*. Clarendon Press.

Barnden, J., & Srinivas, K. (1991) Encoding Techniques for Complex Information Structures in Connectionist Systems. *Connection Science*, Vol. 3, No. 3, 269-315.

Cottrell, C.W. (1993) From symbols to neurons: Are we yet there? *Behavioral and Brain Science*, 16:3 p.454.

Damasio, A. R. (1989) Time-locked multiregional retroactivation: A systems-level proposal for the neural substrates of recall and recognition. *Cognition*, 33, p. 25-62.

Feldman, J. A. (1989) Neural Representation of Conceptual Knowledge. In *Neural Connections, Mental Computation* ed. L. Nadel, L.A. Cooper, P. Culicover, & R.M. Harnish. MIT Press.

Feldman, J.A., Fanty, M.A., & Goddard, N.H. (1988) Computing with Structured Neural Networks. *IEEE Computer*, March 1988. pp 91-103.

Henderson, J. (1994) Connectionist Syntactic Parsing Using Temporal Variable Binding. *Journal of Psycholinguistic Research*, 23 (5) p. 353–379.

Just, M.A. & Carpenter, P.A. (1992) A Capacity Theory of Comprehension: Individual Differences in Working Memory. *Psychological Review*, Vol. 99, No. 1, 122-149.

Keenan, J. M., Baillet, S. D., & Brown, P. (1984) The Effects of Causal Cohesion on Comprehension and Memory. *Journal of Verbal Learning and Verbal Behavior*, 23, 115-126.

Kintsch, W. (1988) The Role of Knowledge Discourse Comprehension: A Construction-Integration Model. *Psychological Review*, Vol. 95, 163-182.

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. and 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, p. 205–242.

McKoon, G., & Ratcliff, R. (1980) The Comprehension Processes and Memory Structures Involved in Anaphoric Reference. *Journal of Verbal Learning and Verbal Behavior*, 19, 668-682.

McKoon, G., & Ratcliff, R. (1981) The Comprehension Processes and Memory Structures Involved in Instrumental Inference. *Journal of Verbal Learning and Verbal Behavior*, 20, 671-682.

Potts, G. R., Keenan, J. M., & Golding, J. M. (1988) Assessing the Occurrence of Elaborative Inferences: Lexical Decision versus Naming. *Journal of Memory and Language*, 27, 399-415.

Shastri, L. (1991) Relevance of Connectionism to AI: A representation and reasoning perspective. In *Advances in Connectionist and Neural Computation Theory*, vol. 1, ed. J. Barnden and J. Pollack. Ablex.

Shastri, L. (1993) A Computational Model of Tractable Reasoning – Taking Inspiration from Cognition. *Proceedings of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence.* France, pp. 202-207.

Shastri, L. (1995) Structured Connectionist Models. In *The Handbook of Brain Theory and Neural Networks* ed. M.A. Arbib. MIT Press.

Shastri, L. & Ajjanagadde V. (1993) From simple associations to systematic reasoning: A connectionist encoding of rules, variables and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:3 p. 417-494.

Shastri, L. & Feldman, J.A. (1986) Neural Nets, Routines and Semantic Networks. In *Advances in Cognitive Science*, N. Sharkey (Ed.), Ellis Horwood Publishers, Chichester, UK. pp. 158–203.

Singer, W. (1993) Synchronization of cortical activity and its putative role in information processing and learning. *Ann. Rev. Physiol.* 55:349-74

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

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

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