INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 666-2900 • FAX (510) 666-2956

A Connectionist Encoding of Parameterized Schemas and Reactive Plans[*]

Lokendra Shastri Dean Grannes Srini Narayanan
Jerome Feldman

TR-02-008

October 2002

Abstract

We present a connectionist realization of parameterized schemas that can model high-level sensory-motor processes and be a candidate representation for implementing reactive behaviors. The connectionist realization involves a number of ideas including the use of *focal-clusters* and feedback loops to control a distributed process without a central controller and the expression and propagation of dynamic bindings via temporal synchrony. We employ a uniform mechanism for interaction between schemas, low-level somatosensory and proprioceptive processes, and high-level reasoning and memory processes. Our representation relates to work in connectionist models of rapid — *reflexive* — reasoning and also suggests solutions to several problems in language acquisition and understanding.

[*]This Technical Report is the final manuscript of an article completed in June 1999 that is to appear as a chapter in the book *Hybrid Information Processing in Adaptive Autonomous vehicles,* G.K. Kraetzschmar and G. Palm (Eds.), Springer-Verlag, Berlin. The publication of the book, however, has been delayed, and hence, the article is being released as a technical report. *Contact:* `shastri@icsi.berkeley.edu`; `http://www.icsi.berkeley.edu/~shastri/shruti`

1 Introduction

Performing goal-directed behavior in an uncertain and dynamic environment requires an agent to use representations that tightly couple action and reaction and that are highly responsive to environmental, intentional, and motivational changes. Consider an agent performing simple actions such as *pull*, *push*, *crawl*, *grasp* or *walk*. Even these simple actions require that the agent be capable of controlling and coordinating multiple sub-actions (such as *preshaping* the hand while *reaching* for the object to be pushed), be highly reactive to perceptual and motor inputs (e.g., avoid obstacles while walking), be able to monitor execution and progress, and be able to recover from minor failures (e.g., recover from temporary instability while walking).

In this article we describe a connectionist architecture for encoding high-level sensorimotor processes, actions, and reactive behavior. Our long-term goal is to develop an architecture that is both neurally plausible and representationally adequate. Hence the architecture described below is both influenced by recent findings in biological motor control as well as informed by work on computational modeling of actions and reactive behavior.

1.1 Biological control theory

Ever since Bernstein's [Ber67] pioneering work, researchers in biological control theory have widely accepted the notion of *motor synergies*, which are sub-cortical parameterized continuous feedback controllers for stereotypical motions such as the vestibulo-ocular reflex.

The temporal details of motor actions are often mediated by afferent feedback. The Hering-Breuer reflex in the mammalian respiratory system is a good example of this phenomena. Here, feedback from the pulmonary stretch receptors activated during lung inflation terminates the respiratory phase [Fel86]. Researchers have also investigated the role of afferent signals in the walking system of the cat, where a sensory signal at the end of the stance phase switches the motor program from stance to swing. [Pea93] reviews the various results and hypotheses regarding the role of afferent feedback in vertebrate and invertebrate motor control. Not surprisingly, he concludes that in order to produce effective movements, motor output must be synchronized and coordinated with ongoing positions, forces and movements in peripheral structures. In general, feedback signals are important for establishing the execution trajectory of motor programs; they may reinforce ongoing motor activity or conversely help switch from one synergy to another.

In order to perform an action, we often need to execute multiple movements in a specific temporal order. Thus the control of complex actions requires the coordination of multiple synergies (a motor plan). After repetitive performance of a particular temporal arrangement of movements, we are able to memorize and execute the whole sequence without external guidance. Where and how in the brain do we store information necessary for the orderly performance of multiple movements?

Evidence that points to the existence of pre-compiled motor plans for complex movements comes from Sternberg and his colleagues [SMKW78] who focus on delays in starting or stopping typing sequences depending upon the length of the string to be typed. Work on grasping and picking up a ball [Jea97] demonstrates the need for concurrent activation of multiple synergies (arm movement and preshaping), serial execution, error-correction based on perceptual input [Arb94], and timed cerebellar modulation [Bul95]. The use of the word *schema* to explicitly refer to the neural representation of such parameterized action controllers appears to have been coined by Head [Hea20] to refer to posture maintenance and control. However, the first proposal that *schemas* included storage of movement related information, including predicted outcomes and error information to drive learning, was perhaps made by Schmidt [Sch75].

More recent work provides important clues about the possible site of motor schemas in the brain. Tanji and Shima [TS94] have found that the cerebral cortex of monkeys contains cells whose activity is exclusively related to a sequence of coordinated multiple movements. They found such activity in the supplementary motor cortex (1, 2) and have hypothesized that these cells contribute a signal about the control of anticipated movements for planning several movements ahead.

Recent evidence also suggests that some motor areas involved in planning motor sequences are active even when actions are only thought about, for example, during mental imagery and imitative imagination

[GFFR96, GAFR96, RGF96]. These and related findings suggest that uniform representational mechanisms might underlie high-level motor control and motor imagery.

1.2 AI and Robotics

In AI, formal approaches to model the ability to reason about changing environments have a long tradition. This research area was initiated by McCarthy [MH69], who observed that reasoning about actions plays a fundamental role in common sense. An advantage of deductive approaches to reasoning about actions is the universality inherent in any logical framework. A purely logical axiomatization which is suitable for temporal prediction, can just as well be employed to answer more general questions such as *postdiction* (i.e., what can be concluded form the current state as to states in the past) and *planning* (i.e., how to act in order that the system evolves into a desired state).

However, deductive approaches have suffered from several well known problems, the most famous of which, the *frame problem*, pertains to compactly specifying aspects of world that are unchanged when an action is executed. A number of associated problems and proposed solutions can be found in [Lif90]. Additionally, formal theories of action and change often make the assumption that environmental changes only occur as a result of some agent-initiated action [GL95]. This assumption is unduly restrictive and renders such systems incapable of dealing with any environment where changes may occur independent of agent-initiated action.

The above problems combined with the need to generate real time behavior in complex and dynamic environments renders deliberative reasoning about the effects of low-level action impractical. Recognition of this fact in AI and Robotics has resulted in various proposals for the representation and use of compiled plans and behaviors [Ros85, Bro86, Nil94, Ark90, AC87]. The basic idea is that rather than divide overall process of acting in the world into functional components such as planning, execution, and perception, one could instead divide the process into task specific pre-compiled plans for various *behaviors*. Examples of such behaviors may involve reaching, grasping, walking to a destination, stacking blocks, etc.

The basic insight here is that the necessity to act fast in an uncertain and dynamic world requires reactive planning agents (biological or robotic) to use representations that can tightly couple action, execution monitoring, error-correction and failure recovery. These are computationally quite similar to the schemas used in the biological motor control literature.

Taken together, the biological and AI results suggest that reactive plans or action schemas are *executable representations* that are capable of monitoring, controlling and coordinating multiple synergies. Moreover, such schemas cannot just be pre-compiled behaviors since they must be capable of flexible execution based on the run-time setting (or binding) of parameter values and continuously monitoring and responding to changes in in the world state. These findings impose strong requirements on the representational properties of action schemas.

1.2.1 X-Schemas.

In what follows, and elsewhere [FB98], we refer to parameterized action schemas as *x-schemas*. The prefix *"x"* stands for *executing* and is meant to emphasize that x-schemas are active representations that *execute* upon invocation. The latter attribute also distinguishes x-schemas from the more static or declarative senses of schemas prevalent in the AI and psychology literature.

1.3 Computational Requirements

In order to identify computational requirements for modeling x-schemas, let us consider the problem of realizing a controller for the *pushing* behavior. A high-level characterization of this behavior is depicted in Fig. 1. While this depiction involves several simplifications, it serves to illustrate the key computational requirements.

In the figure, there are three types of nodes. Circles (*places*) represent states including preconditions and resources. Rectangles (*transitions*) represent atomic actions and events. Hexagons depict actions that may be hierarchically decomposed into sub-actions. For binary conditions the presence of a *token* (a black dot) specifies that the condition holds in the current state. For measurable fluents and resources an integer number of tokens specifies the value or measure of the fluent (by default this value is 1). The current state of the system (called a *marking*) can be read off as the distribution of tokens in the graph. Execution corresponds to the firing of *transitions*. The firing of a transition changes the marking by removing tokens from the input places (resources) and adding tokens to the output places (effects) of the transition. Note that the actual execution trajectory is conditioned both by the structure of the graph as well as externally supplied parameters such as object and force-level.[1]

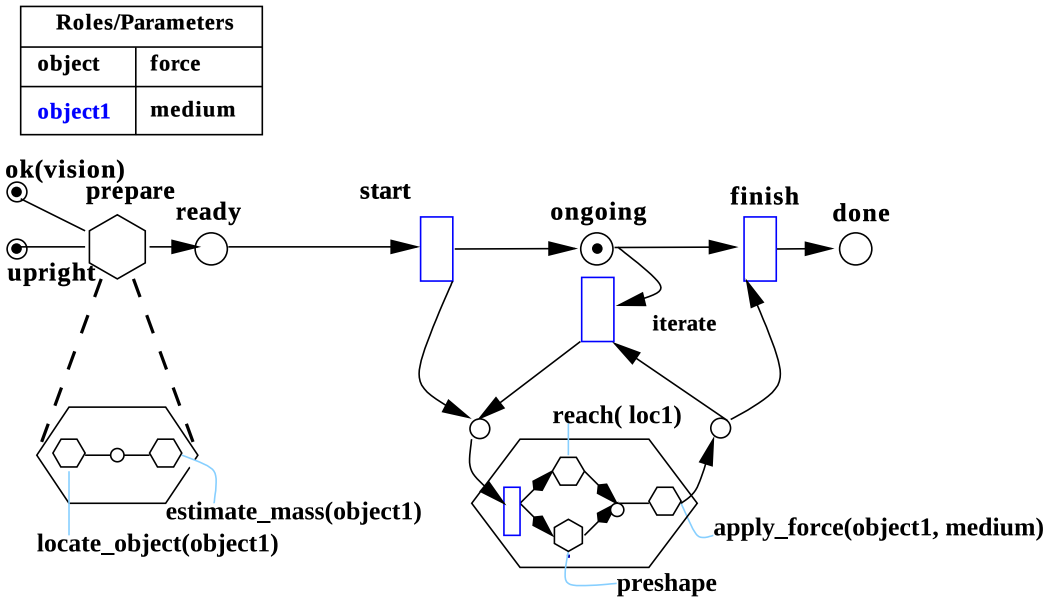


Figure 1: A simplified x-schema for the push action.

Once activated, the *push* x-schema controls the execution of the pushing behavior. The behavior is decomposable into a temporal arrangement of several sub-actions. Some of these sub-actions may be primitive (corresponding to motor synergies in biological control), others may be further decomposable. Only the first-level refinement of of the push behavior is depicted. The behavior can be further decomposed into constituent synergies that make up sub-actions such as *locate_object*, *estimate_mass*, *reach*, *preshape*, and *apply_force*.

The actual execution trajectory is conditional on the results of perceptual tests and/or world inputs. For example, the basic cycle of *locate_object* ⇒ *estimate_mass* ⇒ *ready* of the *push* x-schema may be modified if *locate_object* fails. On the other hand, if the object to be pushed is already in the visual field, then the agent may completely bypass the *locate_object* phase. The *apply_force* x-schema continues (at a force specified by the force input parameter) until the agent receives information (asynchronously from a proprioceptive or perceptual process) that the object to be pushed has been pushed as desired. At this point, the process of pushing is complete.

In general x-schemas must satisfy a number of computational requirements.

[1]Our informal description of the *push* x-schema can be made more precise in terms of a formal computational model based on extensions to Petri Nets [Mur89]. Narayanan [Nar99] has shown that x-schemas for modeling high-level motor plans are formally equivalent to a High Level Generalized Stochastic Petri Net (HLGSPN) and the reachability graph of a marked x-schema is isomorphic to a semi-markov process.

- **Run-time setting of parameters and role-bindings**: An x-schema should be applicable to a wide range of situations. For example, it should be possible to adapt the same x-schema to push either a book or a keyboard away from you (or perhaps, even to push open a door). Similarly, it should be possible to adapt the same x-schema to gently push away a book or to forcefully shove it away. This requires that x-schemas be able to *dynamically bind* to different entities at run time and be able to execute with different *parameter settings*. In Fig. 1, the push x-schema is shown to have one role *object* which is meant to specify the entity to be pushed, and one parameter *force* which provides a scalar measure of the force to be applied to the entity during the push action. Other relevant parameterizations include *direction* (push versus pull). As an additional example, consider the WALK schema, this schema may have a role for *destination* which may be bound to different landmarks at run-time (as in walk to the store). The need for encoding dynamic bindings and supporting run-time parameters distinguishes an x-schema from the simpler notion of a pre-compiled program.

- **Coordination and control**: Basic control behaviors should be modeled. This includes support for the following:

  - Partially ordered actions.
  - Conditional actions, i.e., the ability to choose one of several alternative actions based on the evaluation of specific.
  - Concurrent and iterative actions.
  - Compositional and hierarchical actions, i.e., x-schemas may be composed of other x-(sub)schemas.

- **Distributed control architecture**: X-schemas should function without a central controller. In particular, local control mechanisms should give rise to controlled behavior on a more global scale.

- **Responsiveness**: An x-schema must be *responsive* to both internal and external events. In particular,

  - An x-schema should be capable of event-based interruption and termination.
  - An x-schema must be responsive to the production and consumption of *resources*. This includes fluents such as energy and force.

- **A broad notion of action:** x-schemas should support a broad notion of action. This includes

  - Actions that affect the environment as well as actions that seek information from the environment.
  - Actions that update memory and actions that seek information from memory. The latter includes retrieval as well as inference.

The requirements above serve as a minimal set of properties to be satisfied by any representation of reactive plans and x-schemas.

In what follows we describe a neurally plausible (connectionist) implementation of x-schemas which satisfies several of the computational requirements enumerated above. This implementation draws upon our earlier work on "routines" [SF86] and more recent work on SHRUTI[2], a connectionist model of reflexive reasoning [SA93]. Section 2 provides an overview of SHRUTI. This is followed by a description of the connectionist encoding of x-schemas in Sect. 3. Section 4 briefly discusses the intimate relation between x-schemas, language acquisition, and certain aspects of language understanding.

[2]http:://www.icsi.berkeley.edu/~shastri/shruti

2 An Overview of SHRUTI

SHRUTI was initially developed to model our remarkable ability to draw common sense inferences with reference to a large body of semantic, episodic, and causal knowledge. Subsequently, work on language acquisition, especially early language acquisition in children [FLB$^+$96, Nar97a, BFNL97] lead us to the realization that bodily grounded representations of actions must play a crucial role in supporting a variety of common sense inference. This realization was based in part on the insight that the very representation used for *carrying out* an action must also subserve the *understanding* of that action. Furthermore, in many cases, understanding an utterance about an action may involve mentally simulating the action using the representational machinery required for its execution.

Our belief in a strong coupling between inference and simulated action motivated us to seek a connectionist realization of x-schemas that was compatible with SHRUTI. As we shall see, the x-schemas encoding described in Sect. 3 is not only compatible with SHRUTI, it also makes use of the representational machinery used in SHRUTI for the representation of conceptual knowledge.

2.1 Motivation behind the SHRUTI model

The task of understanding language involves, among other things, recognizing words, disambiguating word senses, incorporating grammatical constraints, and drawing inferences based on common sense knowledge to establish causal and referential coherence.[3] Yet we can understand language at the rate of several hundred words per minute. This suggests that we are capable of performing the requisite inferences rapidly, automatically, and without conscious effort — as though they were a *reflexive* response of our cognitive apparatus [Sha93].

This remarkable human ability poses a challenge for computational theories of intelligence and computational neuroscience: How can a system of slow neuron-like elements represent a large body of knowledge and perform a wide range of inferences with such speed?

We construe a bulk of what we read and hear in terms of events and situations. The latter are relational structures, and hence, their representation requires the coding of *bindings* between the *roles* of an appropriate relation and the *entities* that fill these roles in a given event or situation [SA93]. Consequently, any neurally plausible model of reflexive reasoning should also explain how a system of neuron-like elements can rapidly encode and propagate a large number of bindings.

SHRUTI has been proposed as a neurally motivated computational model of relational information processing and reflexive reasoning [AS91, SA93, MS83, SG96, Sha99a, SW99b, SW99a]. The model proposes that the encoding of relational information is mediated by neural circuits composed of focal-clusters and the dynamic representation and communication of relational instances involves the transient propagation of *rhythmic* activity across these clusters. A role-entity binding is represented within the rhythmic activity by the synchronous firing of appropriate cells within focal-clusters. Systematic rule-like knowledge is encoded by high-efficacy links that enable the propagation of rhythmic activity across focal-clusters. A fact encodes persistent bindings and fires when the bindings it encodes match the dynamic bindings encoded in the on going flux of rhythmic activity.

SHRUTI also identifies constraints on the capacity of the dynamic (working) memory underlying reflexive reasoning. First, on the basis of neurophysiological data pertaining to the occurrence of synchronous activity in the γ band, it predicts that a large number of relational instances can be active simultaneously and a large number of rules can fire in parallel. However, the number of distinct entities participating as role-fillers in these active relational instances and rules must remain very small (≈ 7) [SA93]. Second, since the quality of synchronization degrades as activity propagates along a chain of cell clusters, SHRUTI predicts that as the

[3]Causal coherence refers to the establishment of causal relationships among various events mentioned in a discourse. Referential coherence involves keeping track of entities referenced in a discourse and determining which entities are the same. It is well known that inferences required to establish causal and referential coherence occur rapidly and automatically during text understanding (see e.g., [Kin88, MR80, KBB84]). The evidence for the automatic occurrence of predictive inferences is mixed, but the occurrence of such inference cannot be ruled out [PKG88].

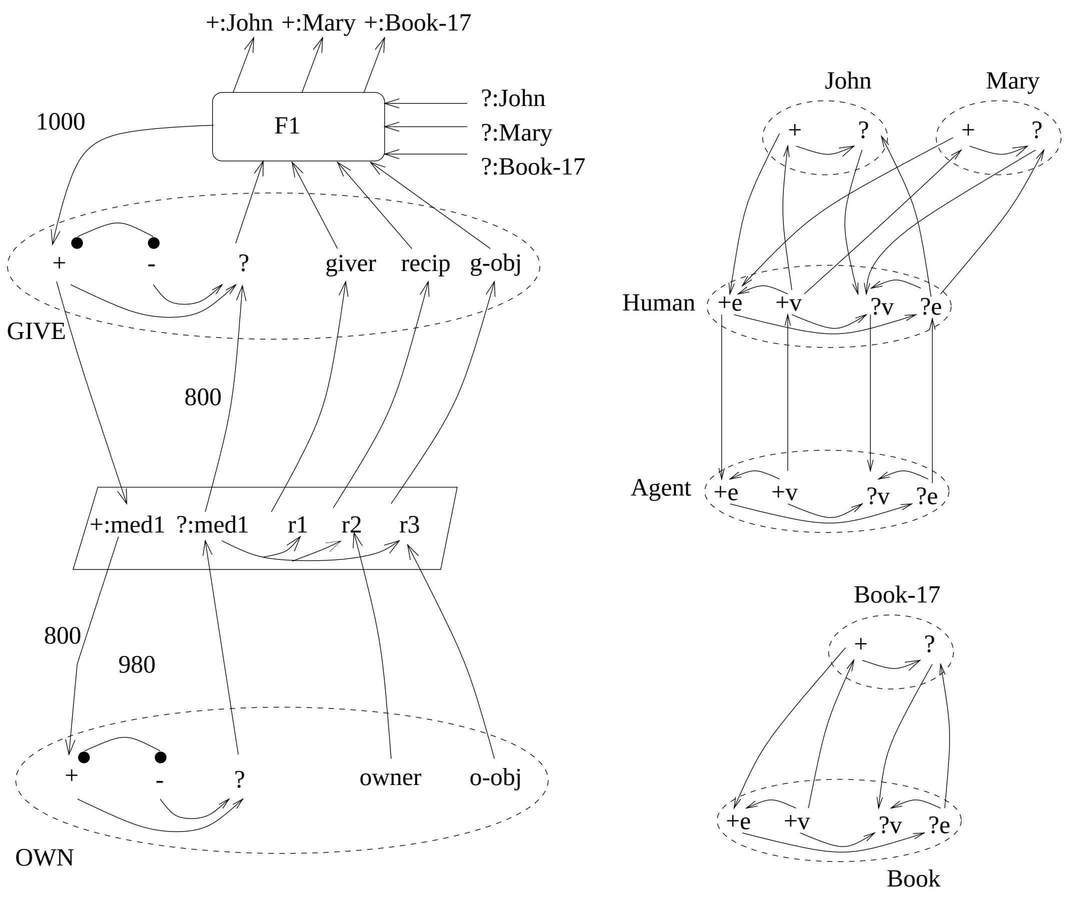


Figure 2: An example SHRUTI network. Links between the mediator and type structure have been omitted.

depth of inference increases, binding information is gradually lost and systematic inference reduces to a mere spreading of activation. Third, SHRUTI predicts that only a small number of instances of any given relation can be active simultaneously.

An implementation of SHRUTI on the CM-5 [Man95] can encode over 500,000 rules and facts, and yet respond to a range of queries requiring derivations of depth five in under 250 milliseconds.

2.2 Representational Machinery of SHRUTI

A description of SHRUTI requires the specification of its *structure* as well as a description of its *dynamic* behavior. The following simple example illustrates the basic representational machinery of SHRUTI. The example has been kept simple. In general, the model can encode positive as well as negated information; deal with inconsistent beliefs; represent rules with multiple antecedents and multiple consequents, exhibit priming effects, support context-sensitive unification of entities, and tune network weights and rule-strengths via supervised learning [Sha99a, SW99a]. Figure 2 illustrates a partial encoding of the following knowledge consisting of rules, facts, and type relationships:

1. $\forall$*(x:agent y:agent z:thing) give(x,y,z)* $\Rightarrow$ *own(y,z) [800,800];*
2. *give(John, Mary, Book-17) [1000];*
3. *is-a(John, Human);*
4. *is-a(Mary, Human);*
5. *is-a(Human, Agent);* and
6. *is-a(Book-17, Book);*

Rule (1) states that when an entity of type *agent* gives something to another entity of type *agent*, then the latter comes to own it. Fact (2) states that John gave Mary a particular book (Book-17). A discussion of how SHRUTI deals with evidence combination appears in [SW99a], for our purpose it suffices to note the following: The weight [a] associated with a fact indicates the strength of belief in the fact. The pair of weights [a,b] associated with a rule indicates that the degree of evidential support for the antecedent being the probable cause (or explanation) of the consequent is a, and the degree of evidential support for the consequent being a probable effect of the antecedent is b. Weights in SHRUTI lie in the interval [0,1000].

2.2.1 Encoding Relations Using Focal-Clusters.

Each generic relation is represented by a focal-cluster depicted by a dotted ellipse in Fig. 2. Consider the focal-cluster for the relation *give* (the dotted ellipse labeled GIVE). This cluster includes an enabler node labeled *?:give*, two collector nodes labeled *+:give* and *-:give*, and three role nodes labeled *giver, recipient* and *give-object* for its three roles. The positive and negative collectors are mutually inhibitory (inhibitory links are depicted by filled blobs). In general, the focal-cluster for an n-place generic relation contains n role nodes.

Nodes are computational abstractions and correspond to *small ensembles of cells*, and a connection from a node A to a node B corresponds to several connections from cells in the A ensemble to cells in the B ensemble. SHRUTI makes use of several node types. Two key node types are described below in Sect. 2.2.3.

Assume that the roles of a relation P have been dynamically bound to some fillers and thereby represent an active instance of P. The activation of the enabler *?:P* means that the system is seeking an explanation for the active instance of P. In contrast, the activation of the collector *+:P* means that the system is affirming the active instance of P. Similarly, the activation of the collector *-:P* means that the system is affirming the negation of the active instance of P.

The activation *level* of *?:P* signifies the strength with which information about P is being sought. Similarly, the activation *level* of *+:P* (*-:P*) signifies the (graded) degree of belief in the truth (falsity) of the active instance of *P*. This belief ranges from *no* on the one extreme (only *-:P* is active), to *yes* on the other extreme (only *+:P* is active), and *don't know* in between (neither collector is very active). A contradiction is indicated if both the collectors receive comparable and strong activation.

The link from the collector nodes to the enabler node of a generic relation converts a dynamic assertion of a relational instance into a query about the assertion. This means that the system continually seeks support (or an explanation) for active assertions.

2.2.2 Encoding of types and instances.

This encoding is illustrated at the right of Fig. 2. The focal-cluster of each entity, *A* consists of a *?:A* and a *+:A* node. In contrast, the focal-cluster of each type, *T* consists of a pair of *?* (*?e:T* and *?v:T*) and a pair of + nodes (*+e:T* and *+v:T*). While the nodes *+v:T* and *?v:T* participate in expression of facts involving the whole type *T*, the nodes *+e:T* and *?e:T* participate in the encoding facts involving particular instances of type *T*. The *levels* of activation of *?:A, ?v:T, and ?e:T* nodes signify the strength with which information about entity *A*, type *T*, and an instance of type *T*, respectively, is being sought. Similarly, the *levels* of activation of *+:A*, *+v:T*, and *+e:T* signify the degree of belief that the entity *A*, the type *T*, and an instance of type *T*, respectively, play appropriate roles in the current situation.

2.2.3 Node Types.

SHRUTI makes use of four node types: m-ρ-nodes, τ-and nodes, τ-or nodes of type 1, and τ-or nodes of type 2. This classification is based on the *computational* properties of nodes. Thus nodes serving different representational functions can be of the same type. The computational behavior of m-ρ-nodes and τ-and nodes is as follows:
m-ρ nodes: An m-ρ node with threshold θ becomes active and fires upon receiving θ units of synchronous activity. Here synchrony is defined relative to a window of temporal integration ω. An m-ρ node A receiving above-threshold periodic inputs from both nodes B and C (where B and C may be firing in different phases) will respond by firing in phase with both B and C.[4] Roles are encoded using m-ρ nodes.
τ-and nodes: A τ-and node becomes active on receiving an uninterrupted and above-threshold input over an interval $\geq \pi_{max}$, where π_{max} is a system parameter. Computationally, such input can be idealized as a pulse whose amplitude exceeds the threshold, and whose duration is greater than or equal to π_{max}. Physiologically, such an input may be identified with a high-frequency burst of spikes. Thus a τ-and node behaves like a *temporal and* node and becomes active upon receiving adequate and uninterrupted inputs over an interval π_{max}. Upon becoming active, such a node produces an output pulse of width $\geq \pi_{max}$.[5] Collectors and enablers are encoded using τ-and nodes.

2.2.4 Representation of Dynamic Bindings.

Dynamic bindings between roles and entities are represented by the *synchronous* firing of appropriate role and entity nodes. With reference to Fig. 2, the dynamic representation of the relational instance *(give: ⟨giver=John⟩, ⟨recipient=Mary⟩, ⟨give-object=a Book⟩)* (i.e., "John gave Mary a book"). involves the synchronous firing of *+:John* and *giver*, the synchronous firing of *+:Mary* and *recip*, and the synchronous firing of *+e:Book* and *g-obj*, with the entities *+:John*, *+:Mary* and *+e:Book* firing in distinct phases.

The possible role of synchronous activity in dynamic neural representations has been suggested by other researchers (e.g., [vdM86]) but SHRUTI is the first detailed computational model that shows how synchronous activation can be harnessed to solve problems in the representation and processing of high-level conceptual knowledge. There exists a rich body of neurophysiological evidence that suggests that synchronous activity might indeed play an important role in neural information processing (see [Sin93]).

[4] Each m-ρ node has an associated *activation combination function* (ECF). The strength of activity of an m-ρ node in any given phase is computed by the node's ECF by combining the weighted inputs arriving at the node in that phase [SW99a]. The strength of firing of a node in a given phase may be viewed as being related to the fraction of cells in the node's cluster firing in that phase.

[5] As in the case of m-ρ-nodes, the level of output activity of a τ-and node is determined by an ECF associated with the node.

2.2.5 Encoding of rules.

A rule is encoded via a mediator focal-cluster (shaded region in Fig. 2) that mediates the flow of activity between the antecedent and the consequent focal-clusters. The mediator consists of a collector and an enabler node and as many role-instantiation nodes as there are distinct variables in the rule. The encoding of a rule establishes links between nodes in the antecedent, consequent, and mediator focal-clusters as follows: (i) The roles of the consequent relations are linked to the roles of the antecedent relations via appropriate role-instantiation nodes in the mediator. This linking reflects the correspondence between antecedent and consequent roles specified by the rule. (ii) The enablers of the consequent relations are connected to the enablers of the antecedent relations via the enabler of the mediator. (iii) The appropriate (+/–) collectors of the antecedent relations are linked to the appropriate (+/–) collectors of the consequent relations via the collector of the mediator. A collector to collector link originates at the + (–) collector of an antecedent relation if the relation appears in its positive (negated) form in the antecedent. The link terminates at the + (–) collector of a consequent relation if the relation appears in a positive (negated) form in the consequent.

If a role-instantiation node receives activation from the mediator enabler and one or more consequent role nodes, it simply propagates the activity onward to the connected antecedent role nodes. If on the other hand, the role-instantiation node receives activity only from the mediator enabler, it sends activity to the *?:e* node of the type specified in the rule as the type restriction for this role. This causes the *?:e* node of this type to become active in an unoccupied phase. The *?:e* node of the type conveys activity in this phase to the role-instantiation node which in turn propagates this activity to connected antecedent role nodes. This interaction between the mediator and the type representation, in effect, creates activity corresponding to "Does there exist some role filler of the specified type?"

2.2.6 Encoding of facts.

SHRUTI encodes two types of facts in its long-term memory: episodic facts (E-Facts) and taxon facts (T-facts). These facts provide closure between the enabler node and the collector nodes. While an E-fact corresponds to a specific instance of a generic relation, a T-fact corresponds to a distillation or statistical summary of various instances of a generic relation and can be viewed as coding *prior probabilities*. In general, T-facts can be conditioned on the type of role-fillers (e.g., the T-fact *buy(a-person,a-Car) [50]* encodes the likelihood that an arbitrary person will buy a car).

2.3 An example of inference

Figure 3 depicts a schematized response of the SHRUTI network shown in Fig. 2 to the query "Does Mary own a book?" (*exists x:Book own(Mary, x)?*). This query is posed by activating *?:Mary* and *?e:book* nodes, the role nodes *owner* and *o-obj*, and the enabler *?:own*, as shown in Fig. 3. We will refer to the phases of activation of *?:Mary* and *?e:book* as ρ_1 and ρ_2, respectively. Activation from the focal-cluster for *own* reaches the mediator structure of rules (1) and (2). Consequently, nodes *r2* and *r3* in the mediator for rule (1) become active in phases ρ_1 and ρ_2, respectively. At the same time, the activation from *?:own* activates the enabler *?:med1* in the mediator of rules (1). Since *r1* does not receive any activation from any role in its consequent's focal-cluster (*own*), it activates the node *?e:agent* which becomes active in a free phase (say $\rho 3$) and, in turn, activates *r1* in this phase. The activation from nodes *r1*, *r2* and *r3* reach the roles *giver*, *recip* and *g-obj* in the *give* focal-cluster, respectively. In essence, the system has created new bindings for the *give* relation. These bindings together with the activation of the enabler node *?:give* encode the new query: "Did *some agent* give Mary a book?". At the same time, activation travels in the type hierarchy and maps the query to a large number of queries such as "Did a human give Mary a book?" and "Did John give Mary Book-17?". The fact *give(John, Mary, Book-17)* now becomes active as a result of matching the query *give(John, Mary, Book-17)?* and causes *+:give* to become active. This in turn causes *+:med1*, to become active and transmit activity to *+:own*. This results in an affirmative answer to the query "Does Mary own a book?" and creates a reverberant loop of activity involving the focal-clusters for *own*, *med1*, *give*, fact F1, and entities *John*, *Mary*, and *Book-17*.

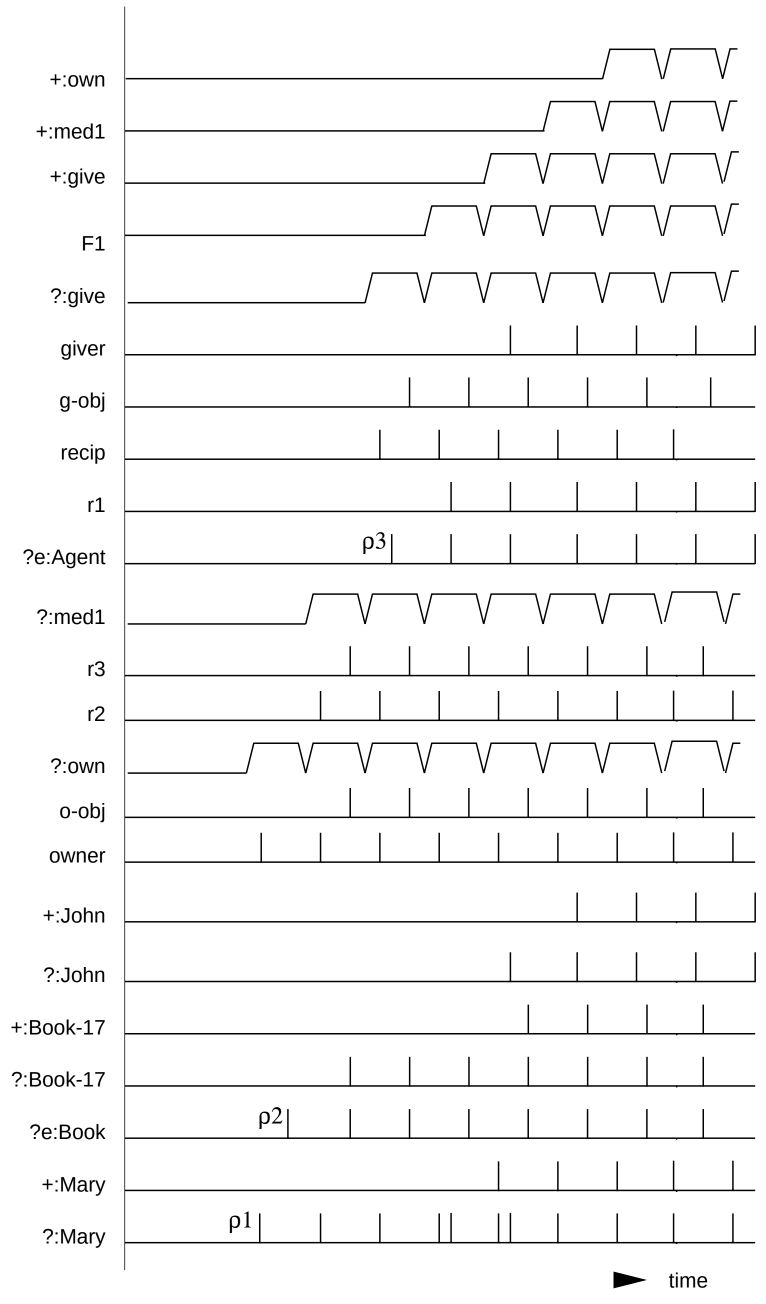


Figure 3: A schematized activation trace of selected nodes for the query *own(Mary,Book-17)?*.

The encoding of rules by the explicit encoding of the inferential dependency between relations, in conjunction with the use of temporal synchrony, provides an efficient mechanism for propagating bindings and performing reasoning. Conceptually, the proposed encoding of rules creates a directed *inferential dependency graph* and the evolution of the system's state of activity corresponds to a *parallel* breadth-first traversal of this inferential dependency graph. Consequently, the time taken to perform an inference is simply proportional to the depth of its derivation and is otherwise independent of the number of items in the knowledge base.

3 Connectionist Encoding of X-Schemas

The connectionist encoding of x-schemas involves a number of ideas including the use of focal-clusters and feedback loops to control a distributed process without a central controller, the expression and propagation of dynamic bindings via temporal synchrony, and a uniform mechanism for interaction between x-schemas, low-level somatosensory and proprioceptive processes, as well as high-level reasoning and memory processes. As we shall see, many of the connectionist representational mechanisms used in the encoding of x-schemas have been drawn from SHRUTI.

The network structure enclosed within the dotted hexagon in Fig. 4 shows a partial connectionist encoding of the PUSH x-schema shown in Fig. 1. The connectionist PUSH x-schema is an interconnected network of focal-clusters (depicted as ovals). The latter provide a locus of control and coordination, and a mechanism for exchanging role-bindings and parameter settings between components of the x-schema. The components of an x-schema can either be local to the x-schema, or they can be generic x-schemas shared by other x-schemas. For example, the PUSH x-schema has as its components generic x-schemas such as LOCATE-OBJECT. These are depicted as hexagons situated outside the PUSH x-schema.

Other x-schemas can invoke the PUSH x-schema by activating the ENABLE focal-cluster near the bottom of Fig. 4 and by communicating the appropriate parameter values and role bindings to this cluster. The first step in the PUSH x-schema execution involves locating the object to be pushed using the LOCATE-OBJECT x-schema (invoked by the LO focal-cluster). The successful completion of LOCATE-OBJECT is followed by the activation of the ESTIMATE-MASS x-schema (invoked by the EM focal-cluster) for estimating the mass of the object to be pushed. At the end of this step, the agent is ready to carry out the requisite motor action. This would be indicated by the activation of the READY focal-cluster. The next stage of execution consists of two concurrent steps, reaching for the object and preshaping the hand in a manner appropriate for pushing it. Reaching is realized by the REACH x-schema and preshaping is realized by an unspecified component which eventually invokes the SHAPE-HAND x-schema for generating the appropriate motor commands. Once the hand is properly shaped *and* correctly located to apply the force, the appropriate motor commands for applying force are executed.

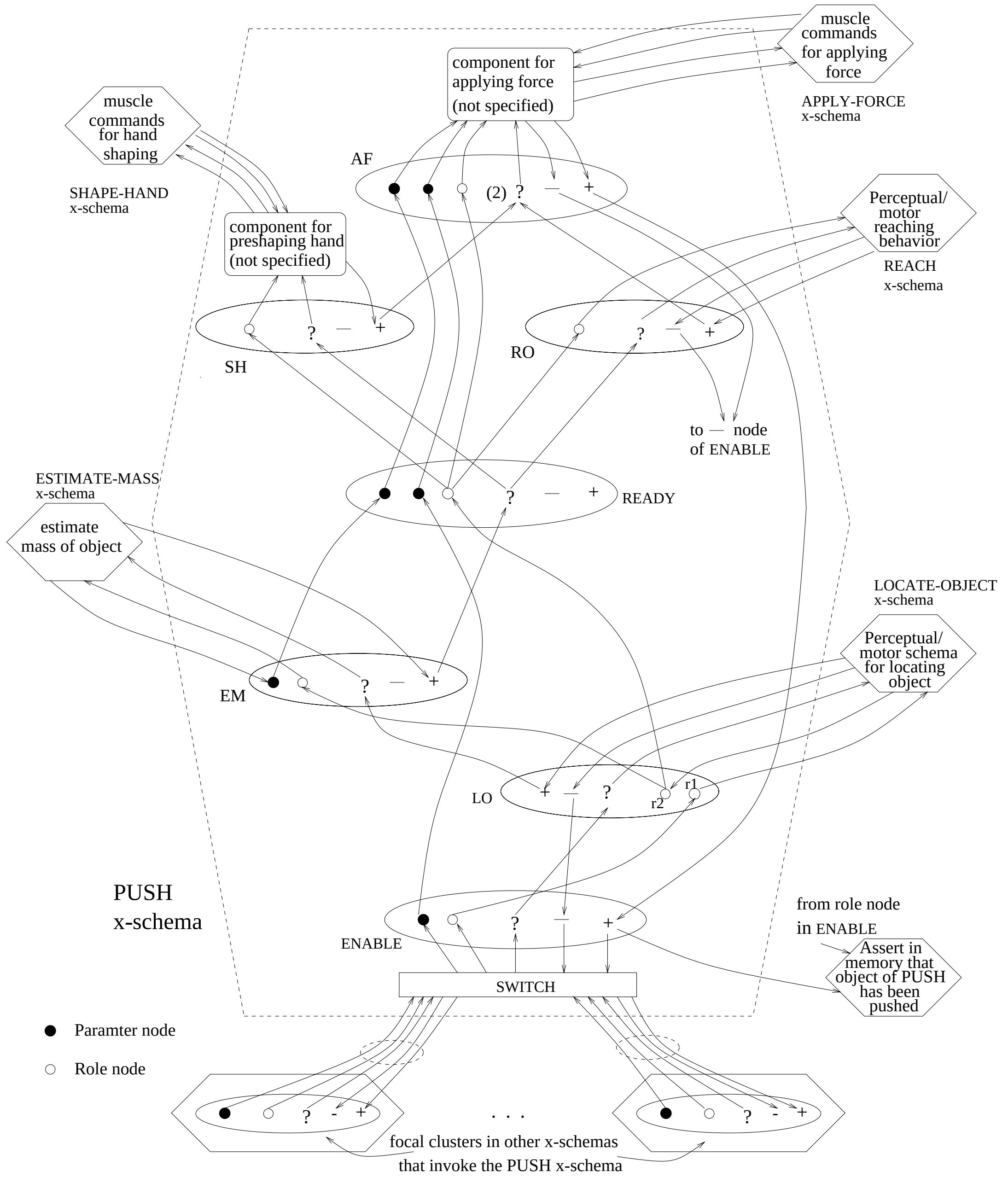
Observe that the x-schema brings together a number of distinct sensorimotor and cognitive functionalities that would typically be distributed across a large network. The interaction between these components is controlled and coordinated by the systematically wired network of focal-clusters.

Recurrent connections are a central aspect of x-schemas. In particular, a focal-cluster that initiates a x-schema also receives a signal when the x-schema is completed.

3.1 Focal-Clusters

An x-schema includes a focal-cluster for each "unit of activity" comprising the x-schema. Here a unit of activity may be:

- A component sub-schema (e.g., the x-schema LOCATE-OBJECT).
- Coordination between two or more components of the x-schema (e.g., the READY focal-clusters.
- A decision based on comparison ($<, >, =$, etc.) of two scalar quantities. The scalar quantities could be parameter values or attribute values of entities extracted from memory.



muscle commands for applying force
APPLY-FORCE x-schema
component for applying force (not specified)
AF
(2) ? — +
muscle commands for hand shaping
SHAPE-HAND x-schema
component for preshaping hand (not specified)
Perceptual/ motor reaching behavior
REACH x-schema
SH
? — +
RO
? — +
to — node of ENABLE
ESTIMATE-MASS x-schema
estimate mass of object
READY
? — +
LOCATE-OBJECT x-schema
Perceptual/ motor schema for locating object
EM
? — +
LO
+ — ?
r1
r2
PUSH x-schema
ENABLE
? — +
from role node in ENABLE
Assert in memory that object of PUSH has been pushed
SWITCH
● Paramter node
○ Role node
? - +
. . .
focal clusters in other x-schemas that invoke the PUSH x-schema

Figure 4: An x-schema for the push action.

- A decision based on equality or inequality of two role fillers.

- Posing of a relational query to other conceptual structures (e.g., to other SHRUTI networks encoding causal models, semantic knowledge, and episodic memory).

Each focal-cluster consists of a small number (possibly zero) of parameter nodes (depicted as filled circles), a small number (possibly zero) of role nodes (depicted as open circles), three control nodes depicted as ?, +, and – (see Fig. 4). The functional significance of nodes in a focal-cluster is as follows:

The activation level of a *parameter* node at the time of x-schema activation specifies the value of the associated parameter during the execution of the x-schema.[6] For example, in the PUSH x-schema, the activation level of the parameter node indicates the relative force to be applied during the push action. In general, parameter nodes can also encode discrete parameter values. In this case, the suprathreshold firing of such a parameter node would lead to the action being performed with the parameter set to the value encoded by the node.

Role nodes provide a mechanism for dynamically binding an x-schema role with a filler at the time of x-schema activation. The PUSH x-schema has one role for indicating the intended object of the push action. The dynamic binding between a role and its filler is expressed by the synchronous firing of the role node and the collector node of the entity filling the role. Note that the process of encoding and propagating dynamic bindings in an x-schema is the same as the one underlying reflexive reasoning.

The ?, +, and – nodes serve the control and coordination function. The ? node of a focal-cluster triggers the activity associated with the focal-cluster and may be viewed as an "initiate activity" node. The threshold of a ? node determines the number of active inputs it must receive in order to fire and trigger the associated unit of activity. Consequently, a ? node can enforce coordination constraints. As an example, consider the ? node of the AF focal-cluster. This node receives inputs from the + nodes of the SH and RO focal-clusters and has a threshold of 2. Hence this node will fire if and only if both the REACH x-schema and the preshaping component execute successfully (i.e., if the + nodes of both the SH and RO focal-clusters become active). Note that a simple sequential flow of control can be realized by simply concatenating focal-clusters in a chain and setting the threshold of ? nodes to 1.

The + and – nodes of a focal-cluster indicate the outcome of the activity associated with the focal-cluster. If the focal-cluster is associated with an x-schema, the activation of the + node by the x-schema indicates a successful *completion* of the x-schema. On the other hand, the activation of the – node by the x-schema indicates an unsuccessful completion. If the focal-cluster is associated with a comparison operation, the activation of + and – nodes encode the possible outcomes of the comparison operation, respectively. Similarly, for an equality operation.

Each x-schema has a head focal-cluster – ENABLE – that serves as the point of initiation. The ENABLE focal-cluster of a shared x-schema has a *switch* which controls the flow of signals into the x-schema. All x-schemas that invoke this x-schema reactively have their appropriate focal-clusters connected to this switch (see bottom of Fig. 4). It is assumed that these inter-schema connections are learned if two x-schemas are part of an automatic behavior or a reactive plan. The switch behaves like a bidirectional k to 1 switch, where k is the number of external x-schemas that are linked to this one. It ensures that at any given time, signals from at most one x-schema can propagate into the ENABLE focal-cluster of the (invoked) x-schema. This state is maintained until the invoked x-schema finishes execution (this is signaled by the activation of the + or – node in ENABLE). If several x-schemas try to activate a x-schema simultaneously, the switching mechanism selects signals from one of these x-schemas. The switch also channels the output of the + and – nodes in ENABLE to the appropriate links feeding back to the invoking x-schema. A connectionist realization of such a switch is described in [MS83].

[6]A parameter node is of the type *scalar*. Scalar nodes have a linear input-output function, that is, their output is simply proportional to their input. As with other types of nodes, a scalar node is an abstraction of a small ensemble of cells. The activation level of a scalar node corresponds to the (average) firing frequency of the cells within the node's ensemble.

3.2 X-Schema Execution

Let us walk through the PUSH x-schema (refer to Fig. 4). Imagine that our agent has the goal of vigorously pushing a red wagon. As a result, one of the x-schemas connected to the PUSH x-schema invokes it by:

- Activating the ? node in its ENABLE focal-cluster.
- Activating the parameter node in ENABLE with a high activation level (to indicate a high force value).
- Synchronizing the firing of the role node in ENABLE with the appropriate role node of the invoking x-schema. Since the object of the action is the "red wagon", the latter would be firing in synchrony with the focal nodes of the concepts "red" and "wagon".

As a result of the above activation, the PUSH x-schema is enabled with its force parameter set to high and its object role bound to the description "red wagon". The activity in ENABLE leads to the activation of the focal-cluster LO wherein the ? node becomes active and the role *r1* synchronizes with the role in ENABLE. Consequently, the role *r1* starts firing in synchrony with "red wagon" and hence, gets bound to that description. LO in turn invokes the LOCATE-OBJECT x-schema with the binding "red wagon". Upon successful execution, LOCATE-OBJECT activates the + node in LO and binds the role *r2* with the location where the object matching the description "red wagon" is located. This binding is expressed by the role node firing in synchrony with the appropriate "location" node in an egocentric spatial map. The activation of the + node in LO leads to the activation of the focal-cluster EM with its role bound to the location of the red wagon. EM activates the ESTIMATE-MASS x-schema which estimates the mass of the red wagon. Upon completion, ESTIMATE-MASS activates the + node of EM and activates the parameter node of EM at a level indicative of the mass of the red wagon.

The activation of the + and the parameter nodes in EM leads to the activation of the ? node in the READY focal-cluster and the setting of the parameter indicative of the estimated mass of the object to be pushed (the red wagon). By this time, the activity of the second parameter of READY is also set by the parameter node in ENABLE to indicate the desired relative force, and the role of READY is bound to the location of the red wagon by the role *r2* in the LO focal-cluster.

The activation of READY marks a significant state in the execution of the PUSH. At this point the agent has carried out the necessary preparatory steps but has not executed any actions that affect the environment. Thus READY provides a locus for high-level executive processes to exercise inhibitory control over the actual execution of the PUSH.

READY activates the focal-clusters RO and SH concurrently, and binds their roles to the location of the red wagon. This leads to the concurrent activation of (i) the REACH x-schema which brings the agent's hand to the location of the red wagon and (ii) the preshaping component which preshapes the hand in a manner appropriate for pushing the red wagon. The preshaping component is specific to the PUSH x-schema. Given a binding specifying the location of an object, it extracts the shape of the object and determines what should be the appropriate shape of the hand for pushing the object. It conveys this preshaping information to the generic x-schema SHAPE-HAND as a set of parameter values. SHAPE-HAND now executes and generates the necessary signals for driving the motor system.

REACH and the preshaping components execute independently and signal their completion (or failure) by activating the + (–) node in RO and SH respectively. The successful termination of these two components leads to the activation of the AF focal-cluster. Since the ? node of AF has a threshold of 2, it becomes active only upon receiving a completion signal from both SH and RO. By this time AF also has its parameters set with the appropriate values of estimated mass and desired relative force, and its role bound to the location of the red wagon. Now the final component for applying force is initiated which activates the low-level APPLY-FORCE x-schema which issues motor commands for applying the force required to push the red wagon. Once this component executes successfully it activates the + node of AF which activates the + node of ENABLE and signals the successful completion of PUSH. The activity of the + node in ENABLE is in turn propagated via the switch to the + node of the appropriate focal-cluster in the x-schema that invoked PUSH.

In addition to successful completion, failure is also communicated within the x-schema. For example, the failure to locate the object, reach the object, or apply force to the object can lead to the activation of the – nodes in LO, RO, and AF, respectively. This would lead to the activation of the – node in ENABLE and signal a failure of PUSH. The activity of the – node in ENABLE would be propagated via the switch to the – node of the appropriate focal-cluster in the x-schema that invoked PUSH. Alternately, the failure of a step within the PUSH x-schema can trigger alternate plans or remedial strategies. For example, the failure of REACH may trigger x-schemas for walking around an obstacle or using an implement to extend the agent's reach.

Upon completion, an x-schema can also trigger a memory update. Thus the completion of PUSH can lead to the agent's episodic memory recording the fact the agent has pushed the red wagon. It can also lead to the remembering of the position of the red wagon at the end of PUSH. Here memory retrieval is treated as being similar to a perceptual "look up". Similarly, the updating of episodic or working memory is treated as being similar to an external motor action.

Recently, Shastri has shown that a connectionist network architecture fully compatible with SHRUTI and the encoding of x-schemas presented here, can transform a transient pattern of activity into a persistent memory via long-term potentiation [Sha97, Sha99c, Sha99b].

3.3 Iteration and Interrupts

Figure 5 depicts a x-schema for clearing a specified object by removing things lying on its top. This example illustrates how x-schemas can encode iteration and use perceptual processes and the state of the external world to obviate the need for storing explicit state information.

The CLEAR x-schema is invoked by activating the ? node in the ENABLE focal-cluster and binding its role to the location of the object to be cleared. This in turn activates the VC focal-cluster which activates the IS-IT-CLEAR x-schema for verifying whether the specified object is clear. If IS-IT-CLEAR finds that the object is not clear, it activates the – node in VC. This activates the FT focal-cluster, and in turn, the FIND-TOP x-schema with its role *r1* bound to the object to be cleared. FIND-TOP finds the location of the topmost item above the object bound to *r1* and communicates this location to FT by binding its role *r2* to this location and activating its + node. FT now activates the focal-cluster PA and binds its role to the location of the topmost object. PA in turn activates the PUTAWAY x-schema with its role bound to the location of the topmost object. PUTAWAY puts away the topmost object and activates the + node in PA, which in turn activates the ? node in ENABLE (PA does not change the binding of the role in the ENABLE focal-cluster). This completes one iteration of a loop and starts a second iteration. In each iteration, FIND-TOP locates the topmost object and PUTAWAY puts it away. The cycle continues until IS-IT-CLEAR succeeds and activates the + node in VC which activates the + node in ENABLE thereby terminating the x-schema.

Notice that the x-schema does not require explicit control variables — the external world and the agent's perceptual mechanisms for locating objects in space provide the x-schema with the requisite information (cf. [BHPR97]). Now imagine that the execution of this x-schema was interrupted and resumed subsequently. At this time the only critical requirement is that the invoking x-schema bind the object role of CLEAR to the same object that was initially intended to be cleared. The state of the world itself captures the relevant state information.

3.4 Interaction Between X-Schemas and SHRUTI

The encoding of x-schemas is fully integrated with the SHRUTI system for reflexive reasoning. Thus a focal-cluster within an x-schema can pose a query about any relation encoded in SHRUTI. Upon receiving activation from the focal-cluster, circuits in SHRUTI rapidly carry out the necessary retrieval and inference and return the answer back to the focal-cluster that initiated the query. The answer takes the form of appropriate collector (+ or –) activation and the binding of roles via synchronous activation. Similarly, a focal-cluster within an x-schema can assert a dynamic fact pertaining to a relation in SHRUTI. Thus an x-schema can not only direct external actions, it can also access and update other forms of conceptual knowledge including the agent's causal model of the environment and semantic and episodic facts.

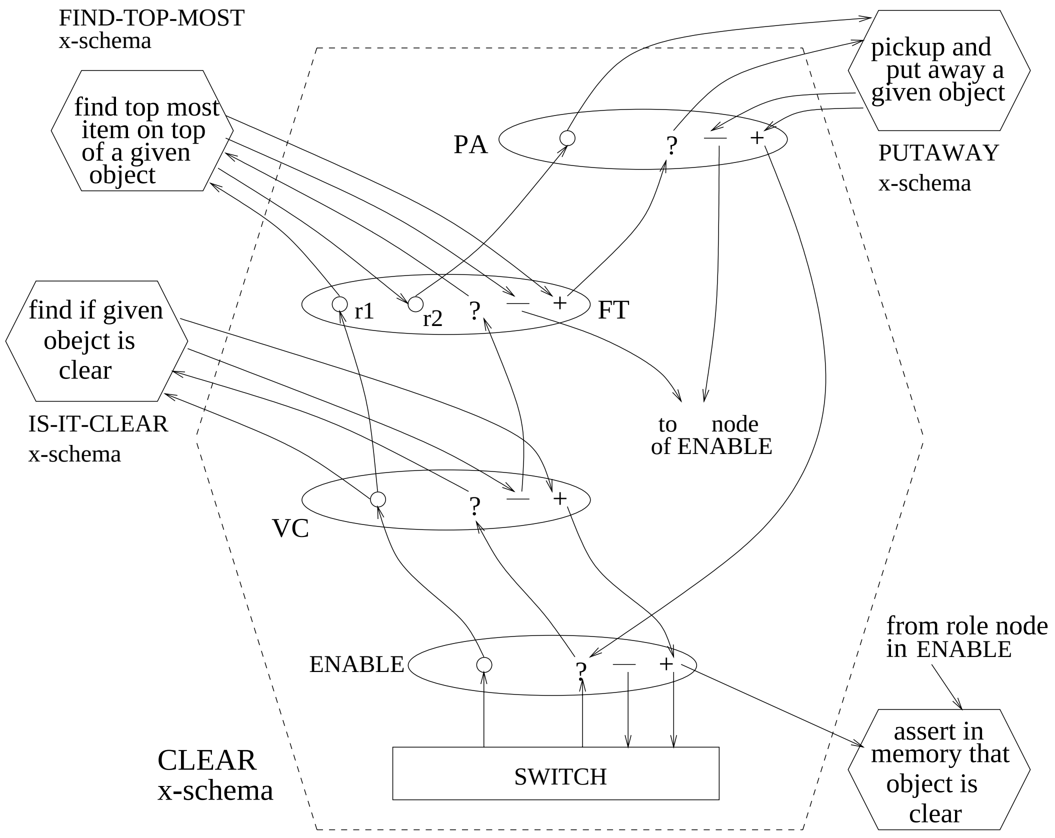

FIND-TOP-MOST
x-schema
find top most item on top of a given object
pickup and put away a given object
PUTAWAY
x-schema
PA
?
—
+
r1
r2
?
—
+
FT
find if given obejct is clear
IS-IT-CLEAR
x-schema
to node of ENABLE
VC
?
—
+
from role node in ENABLE
ENABLE
?
—
+
SWITCH
CLEAR
x-schema
assert in memory that object is clear

Figure 5: An x-schemas for *iteratively* removing things from the top of an object.

3.5 Implementation

The connectionist encoding of x-schemas as described above has been fully implemented within the SHRUTI simulation system, and the implementation has been tested on a number of examples including the PUSH and CLEAR x-schemas shown in Figs. 4 and 5. Figure 6 shows the timing of the internal focal-clusters of the PUSH x-schema resulting from the activation of the x-schema.

The following illustrates how the PUSH x-schema is specified to the simulation system using SHRUTI's input language.

```
*S:push(x;f) (y;m) <1> {
     /* x is a role, f is a parameter, ‘;’ is a separator.
        y is a local role, m is a local parameter */

     /* first specify focal-clusters, their arguments, and their
        associated x-schemas

        note: all x-schemas have a head focal-cluster ENABLED
        which is implicitly defined for all x-schemas. It is
        referred to by the x-schema name. */

  (lo:  locateObject(x,y;) ),     /* This defines a focal-cluster
                                     labeled ‘‘lo’’. The unit of
                                     activity associated with
                                     this cluster is the x-schema
                                     labeled locateObject. */
  (em:  estimateMass(y;m) ),
  (ready: (y;f,m)),
  (sh:  preshapeHand(y;) ),
  (ro:  reachObj(y;) ),
  (af:  applyForce(y;f,m) ),

    /* next describe the internal connectivity of focal-clusters
       ->| signifies an OR, the target node has a threshold of 1
       ->& signifies an AND, the target node has a threshold
           equal to the number of incident links. */

  ?:push         ->| ?:lo,
  -:lo           ->| -:push,
  +:lo           ->| +:em,
  +:em           ->| ?:ready,
  ?:ready        ->| ?:sh,
  ?:ready        ->| ?:ro,
  +:sh           ->& ?:af,
  +:ro           ->& ?:af,
  -:ro           ->| -:push,
  -:af           ->| -:push,
  +:af           ->| +:push
};

   /* now define the sub-schemas of the push x-schema */

*S:locateObject(x,y;) <1> {
    ...
};

    ...
```

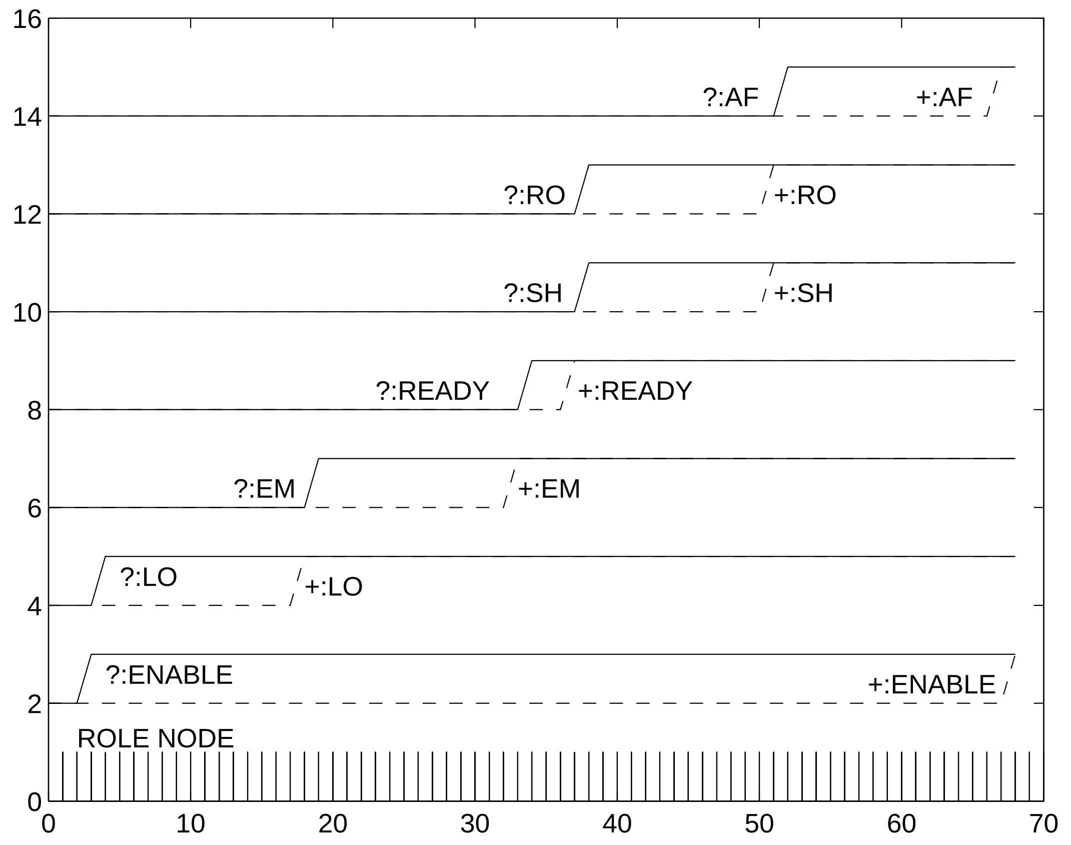


Figure 6: A schematized activity trace of selected nodes in the PUSH x-schema, generated from an actual simulation. The horizontal axis counts full cycles in the periodic activity, which consist of four distinct phases. The role node, shown at the bottom of the figure, fires in one phase in every cycle. Immediately above that are plotted the activity of the ?:ENABLE and the +:ENABLE nodes (these labels refer to the ? and the + nodes ofthe ENABLE focal-cluster, respectively). Above that, in order, are the ? and + nodes for the focal-clusters LO, EM, READY, SH, RO, and AF. For purposes of the example, all external x-schemas are encoded as "dummy" x-schemas that return positive results when activated.

4 X-Schemas and their Relation to Language

The embodiment of concepts and language is a central issue in cognitive science. How can a neural system represent and learn concepts, and organize them into a set of lexical items? The Neural Theory of Language (NTL) group[7] in Berkeley has sought computational insight into these questions by asking them of structured

[7]See `http:://www.icsi.berkeley.edu/NTL/`

connectionist systems, rather than the physical neural systems of the brain [FLB+96].

One major shortcoming of the standard view of lexical acquisition is that it provides no account of how a child learns to *make use* of the concepts she learns and the words that label them. This same weakness appears as a technical consequence of using back-propagation in PDP models: even when the network learns perfectly how to classify a domain, it has no mechanism for executing the action.

As pointed out in Sect. 2, a crucial insight resulting from the NTL project [BFNL97, Nar97a] is that the meaning of motion and manipulation words (such as *pull*, *push*, *walk*, *run*, *stumble* and *fall*) are grounded in x-schemas for carrying out the action. Different action verbs and associated grammatical devices (e.g., adverbs and prepositions) express parameters of the underlying action such as *rates*, *manner*, *forces*, and *posture and movement control parameters*, as well as associated intentional and motivational states such as *goals*, *resources*, *energy* and *effort*. This parameter specification interface between language and action is called the *linking structure*.

The tight link hypothesized between language and action allows us to employ language learning algorithms that produce usable representations of actions. [BFNL97] describes a project that resulted in a system that after training on examples of action-word pairs, is able to produce an appropriate label for a particular motor action based on features of both the action and the world state. In addition, however, the learned verb representation also functions as a command interface that allows the system to execute a given verb by filling in the linking structure. The system was able to learn hand action verbs in a variety of languages suggesting that x-schemas may provide a sufficiently strong inductive bias for verb learning.

The grounding of lexical semantics in action controllers enables the representation to flexible and adaptive in the way context affects interpretation. Two recent projects resulted in systems that exploit this context-sensitivity for language understanding. One project [Nar97b] used recurring monitoring and control schemas abstracted from the basic representation to provide a fine-grained simulation-based framework of processes and their interactions. The resulting model seems to offer a natural solution to the vexing linguistic problem of ”aspectual composition”. [Nar97a] describes a second project that uses an x-schema based language understanding system that models metaphoric reasoning about event descriptions in abstract domains such as international economics. A crucial aspect of the implemented model is its capacity to exploit domain knowledge of spatial motion and manipulation (implemented as x-schema simulations) for real-time simulative inference. Results of applying our model to discourse fragments from newspaper stories in international economics show that crucial facts about abstract plans, goals, resources and intent can be expressed by projections from embodied concepts. Our model can make these discourse inferences in real-time, consistent with the fact that such information is routinely available as automatic inferences in narrative understanding.

5 Conclusions

We have described a connectionist encoding of x-schemas and reactive plans that can model high-level sensory-motor processes and can be a candidate for representing reactive behaviors. It is interesting that the same mechanisms seem sufficient for reflexive inference as well as motor x-schemas. This is not surprising if one takes seriously the notion that language and conceptual structures are grounded in our body and shaped by our motor and perceptual systems.

A number of issues remain open. One key issue is that of learning. How are networks structures for x-schemas learned? Our group is investigating this question within the framework of model-merging [FB98] and recruitment learning [Fel82, Sha99b]. A second key issue concerns time. Different x-schemas operate over different time scales and even components of the same x-schema may have widely different completion times. This poses a potential problem for a connectionist encoding: the resulting state of a component that terminates early must be maintained while other components are executing. This can be done either by keeping the appropriate nodes active, or by converting the state information into a structural representation (i.e., memorizing it) and later retrieving it upon the completion of other components. The former solution is appropriate if the gap in completion times are small, while the latter is more suited for larger gaps. Both these solutions require further investigation. Other issues relating to time such as suspension, preemption/interruption, retraction, and time-out, also require additional work and are under investigation.

Acknowledgments

This work was partially funded by ONR grants N00014-93-1-1149 and N00014-95-C-0182, and NSF grants IRI-9619293 and SBR-9720398. Thanks to David Bailey, George Lakoff, Carter Wendelken, and other members of the NTL group.

References

[AC87] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87*, pages 268–272, Seattle, WA, 1987.

[Arb94] M.A. Arbib. *The Metaphorical Brain: 2*. Wiley Interscience, New York, 1994.

[Ark90] R.C. Arkin. Integrating behavioral, perceputal, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[AS91] V. Ajjanagadde and L. Shastri. Rules and variables in neural nets. *Neural Computation*, 3:121–134, 1991.

[Ber67] N.A. Bernstein. *The Co-ordination and Regulation of Movement*. Pergamon Press, New York, 1967.

[BFNL97] D.R. Bailey, J. A. Feldman, S. Narayanan, and G. Lakoff. Modeling embodied lexical development. In *Proceedings of the 19th Cognitive Science Society Conference*, pages 19–24, 1997.

[BHPR97] D.H. Ballard, M.M. Hayhoe, P.K. Pook, and R.P.N. Rao. Deictic codes for the embodiment of cognition. *Brain and Behavioral Sciences*, 20(4):723–767, 1997.

[Bro86] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23, March 1986.

[Bul95] D. Bullock. Motorneuron recruitment. *Handbook of Brain Theory*, pages 594–97, 1995.

[FB98] J.A. Feldman and D. Bailey. Layered hybrid connectionist models for cognitive science. *Workshop on Hybrid Systems, NIPS-98*, 1998.

[Fel82] J.A. Feldman. Dynamic connections in neural networks. *Biological Cybernetics*, 42:27–39, 1982.

[Fel86] J.A. Feldman. Neurophysiology of respiration in mammals. *Handbook of Neurophysiology: Section 1, The Nervous System*, 4:463–524, 1986.

[FLB$^+$96] J.A. Feldman, G. Lakoff, D.R. Bailey, Srini Narayanan, Terry Regier, and Andreas Stolcke. L_0—the first five years of an automated language acquisition project. *AI Review*, 10:103–129, 1996. Special issue on Integration of Natural Language and Vision Processing.

[GAFR96] S.T. Grafton, M.A. Arbib, L. Fadiga, and G. Rizzolati. Localization of grasp representations in humans by pet: 2. observation compared with imagination. *Experimental Brain Research*, 112:103–111, 1996.

[GFFR96] V. Gallese, L. Fadiga, L. Fogasi, and G. Rizzolati. Action recognition in the premotor cortex. *Brain*, 119(2):593–609, 1996.

[GL95] M. Gelfond and V. Lifschitz. Representing action and change by logic programs. *Handbook of Brain Theory*, pages 594–97, 1995.

[Hea20] H. Head. *Studies in Neurology*. Hodder and Stroughton, London, 1920.

[Jea97] M. Jeannerod. *The Cognitive Neuroscience of Action*. Blackwell, Oxford, UK, 1997.

[KBB84] J.M. Keenan, S.D. Baillet, and P. Brown. The effects of causal cohesion on comprehension and memory. *Journal of Verbal Learning and Verbal Behaavior*, 23:115–126, 1984.

[Kin88] W. Kintsch. The role of knowledge discourse comprehension: A construction-integration model. *Psychological Review*, 95:163–182, 1988.

[Lif90] V. Lifschitz. Representing frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[Man95] D.R. Mani. *The Design and Implementation of Massively Parallel Knowledge Representation and Reasoning Systems: A Connectionist Approach*. PhD thesis, Computer and Information Science Department, University of Pennsylvania, 1995.

[MH69] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artifical intelligence. *Machine Intelligence*, 4:463–502, 1969.

[MR80] G. McKoon and R. Ratcliff. The comprehension processes and memory structures involved in anaphoric reference. *Journal of Verbal Learning and Verbal Behaavior*, 19:668–682, 1980.

[MS83] D.R. Mani and L. Shastri. Reflexive reasoning with multiple-instantiation in a connectionist reasoning system with a typed hierarchy. *Connection Science*, 5(3 and 4):205–242, 1983.

[Mur89] T. Murata. Petri nets: Properties, analysis, and applications. In *Proceedings of IEEE-89*, volume 77, pages 541–576, April 1989.

[Nar97a] S. Narayanan. *Knowledge-based Action Representations for Metaphor and Aspect (KARMA)*. PhD thesis, Computer Science Division, EECS Department, University of California at Berkeley, 1997.

[Nar97b] S. Narayanan. Talking the talk is like walking the walk: A computational model of verbal aspect. In *Proceedings of the 19th Cognitive Science Society Conference*, pages 548–553, 1997.

[Nar99] S. Narayanan. Reasoning about actions in narrative understanding. In *Proceedings of IJCAI-99, the Sixteenth International Joint Conference on Artificial Intelligence (to appear)*. Morgan Kaufmann Press, 1999.

[Nil94] N. J. Nilsson. Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1:139–158, 1994.

[Pea93] K.G. Pearson. Common principles of motor control in vertebrates and invertebrates. *Annual Reviews in Neuroscience*, 16:265–297, 1993.

[PKG88] G.R. Potts, J.M. Keenan, and J.M. Golding. Assessing the occurrence of elaborative inferences: Lexical decision versus naming. *Journal of Memory and Language*, 27:399–415, 1988.

[RGF96] G. Rizzolatti, V. Gallese, and L. Fadiga. Premotor cortex and the recognition of motor actions. In *Cognitive Brain Research*, pages 131–141, 1996.

[Ros85] J.S. Rosenschein. Formal theories of knowledge in ai and robotics. *New Generation Computing*, 3(4):345–357, 1985.

[SA93] L. Shastri and V. Ajjanagadde. From simple association to systematic reasoning: a connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Brain and Behavioral Sciences*, 16(3):417–494, 1993.

[Sch75] R.A. Schmidt. A schema theory of discrete motor learning. *Physiological Review*, 82:225–260, 1975.

[SF86] L. Shastri and J.A. Feldman. Neural nets, routines, and semantic networks. In N.E. Sharkey, editor, *Advances in Cognitive Science 1*, pages 158–203. Ellis Horwood, Chichester, England, 1986.

[SG96] L. Shastri and D.J. Grannes. A connectionist treatment of negation and inconsistency. In *Proceedings of the Eighteenth Conference of the Cognitive Science Society*, pages 142–147, 1996.

[Sha93] L. Shastri. A computational model of tractable reasoning – taking inspiration from cognition. In *Proceedings of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence*, pages 202–207. Morgan Kaufmann Press, 1993.

[Sha97] L. Shastri. A model of rapid memory formation in the hippocampal system. In *Proceedings of the Nineteenth Conference of the Cognitive Science Society*, pages 680–685, 1997.

[Sha99a] L. Shastri. Advances in SHRUTI — a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence*, 11:79–108, 1999.

[Sha99b] L. Shastri. A biological grounding of recruitment learning and vicinal algorithms. Technical Report TR-009-99, International Computer Science Institute, Berkeley, CA, April 1999.

[Sha99c] L. Shastri. Recruitment of binding and binding-error detector circuits via long-term potentiation. *Neurocomputing*, 26-27:865–874, 1999.

[Sin93] W. Singer. Synchronization of cortical activity and its putative role in information processing and learning. *Annual Review of Physiology*, 55:349–374, 1993.

[SMKW78] S. Sternberg, S. Monsell, R.L. Knoll, and C.E. Wright. The latency and duration of rapid movement sequences: comparisons of speech and typewriting. In G. E. Stelmach, editor, *Information Processing in Motor Control and Learning*, pages 117–152. Academic, New York, 1978.

[SW99a] L. Shastri and C. Wendelken. Knowledge fusion in the large – taking a cue from the brain. In *Proceedings of FUSION'99, the 2nd International Conference on Information Fusion*, pages 1261–1269, 1999.

[SW99b] L. Shastri and C. Wendelken. Soft computing in SHRUTI: — a neurally plausible model of reflexive reasoning and relational information processing. In *Proceedings of IIA/SOCO'99, the Third International Symposium on Soft Computing*, pages 741–747, 1999.

[TS94] J. Tanji and S. Shima. The supplementary motor area in the cerebral cortex. *Nature*, 371(6496):413–416, 1994.

[vdM86] C. von der Malsburg. Am i thinking assemblies? In G. Palm and A. Aertsen, editors, *Brain Theory*, pages 161–176. Springer-Verlag, New York, NY, 1986.