# A (Somewhat) New Solution to the Variable Binding Problem

Leon Barrett, Jerome Feldman, and Liam Mac Dermed

July 14, 2008

**Abstract**

To perform automatic, unconscious inference, the human brain must solve the "binding problem" by correctly grouping properties with objects. Temporal binding models like SHRUTI already suggest much of how this might be done in a connectionist and localist way by using temporal synchrony. We propose a set of alternatives to temporal synchrony mechanisms that instead use short signatures. This serves two functions: it allows us explore an additional biologically plausible alternative, and it allows us to extend and improve the capabilities of these models. These extensions model the human ability to both perform unification and handle multiple instantiations of logical terms. To verify our model's feasibility, we simulate it with a computer system modeling simple, neuron-like computations.

## 1 Introduction

There are several variants of the "binding problem," which asks how a massively parallel system can achieve coherence. The most striking examples involve subjective experience and therefore remain intractable to experimentation. For example, we know that visual processing involves dozens of separate brain areas, yet we perceive the world as a coherent whole. Even leaving subjective experience aside, there are still compelling technical problems in understanding how a neural network can perform crucial computational tasks, such as those that arise in reasoning about and acting in the world.

A basic problem, and the one that we will focus on, is the "variable binding" problem. As a first example, consider your ability to pick up objects. Depending on the object, its current position, and your goals, you have a very wide range of ways of grasping and manipulating the object, all realized by the network of neurons that is your brain. This is an instance of the variable binding problem because your choice of values for the three variables *object*, *position*, and *goal* has consequences throughout your brain on how the action is carried out. In conventional computing, we assume that different program modules all have access to the values of (global) variables and can modify their behavior appropriately. Any theory of neural computation needs some mechanism for achieving this kind of global effect.

In the domains of problem solving, language understanding, and other symbolic behavior, the variable binding task becomes even more complex and interesting. The linguist Ray Jackendoff has suggested that the variable binding problem is the key to any neural theory of language (Jackendoff, 2002). Both language and problem solving involve operations on variables before they have values assigned to them. For example, people routinely use general rules like "All humans are mortal," equivalently written *Human(X)* $\rightarrow$ *Mortal(X)*, to perform similar inferences about many different people. An obvious such use of variables in language understanding is reference resolution, which has several forms. The clearest case is the task of binding pronouns to their referred objects, but there are many other examples such as, "the winner of the 2012 U.S. Presidential election," which has meaning even if no one yet knows precisely who that person is. Humans can perform this inference effectively with a moderate number of variables without explicit effort, and we would like to model how this might work in a massively parallel neural system. At the same time, we see the relevance of localist representations (Page, 2000) and would like to use these concepts also.

A recent article (van der Velde and de Kamps, 2006) in the Behavioral and Brain Sciences and the accompanying commentary explores a wide range of connectionist approaches to the binding problem. Here, we discuss a few. Our objective is not to disparage other models, but rather to avoid existing pitfalls and expand the space of feasible binding mechanisms. The first such model is brute force enumeration of all possible variable bindings (Ballard, 1986; Lima, 1992), sometimes with coarse-coded conjunctive binding to mitigate its exponential complexity (O'Reilly et al., 2001). More recently, van der Velde and de Kamps (2006) employ such a crossbar network in their Neural Blackboard model. A second approach has been to use sign (signature) propagation. In sign propagation, each variable in an expression has its own node (a group of neurons working together). This node can represent and transmit a particular signature corresponding to a concept, so the signature is essentially treated as a name for the concept (Lange and Dyer, 1989; Sun, 1989, 1992; Browne and Sun, 2000; Lange, 1992). The main difficulty is that then there must be one signature for every representable object–so each signature must carry about 20 bits of information.

The most widespread approach is that of phase synchronization, also known as temporal synchrony. This approach breaks the cycle of neural firing into discrete time slices. When a variable node fires in-phase with a concept node, this represents a binding between them. For example, if the node representing "Fido" is firing at the same time as the node for "big," then this indicates that "Fido" has the "big" property. This idea has been around for a while (Milner, 1974), but the best-known model of this sort is SHRUTI (Shastri, 1999), and its mechanisms have been carefully examined. Despite its clean theoretical background, there is some experimental evidence (Shadlen and Movshon, 1999) and information theoretic argument (Averbeck et al., 2006; Golledge et al., 2003) that questions this sort of neural synchrony and its involvement in binding (Salinas and Sejnowski, 2001).

Another recently proposed solution to the binding problem is the Neural Blackboard architecture (van der Velde and de Kamps, 2006). In this design, rather than synchrony or passing around some sort of marker, there are connections between computational nodes that are ordinarily disabled, but may be enabled, and when enabled allow signals to travel between the two nodes for a period of time. Thus, it attempts to solve the binding problem by making temporary links between nodes. However, it is not designed to permit the complex inference that SHRUTI does. Some comparison of the neural blackboard architecture and SHRUTI has been written by van der Velde and de Kamps (2006) and Shastri (2006).

Another parallel inference method is marker passing, which uses simple binary markers. For instance, Fahlman (2006) suggests a model in which each concept is represented as a node with a collection of 20-40 "marker bits," with these concept nodes connected by links. A central coordinator regulates marker passing by broadcasting relatively complex commands to all nodes and links. Thus, Fahlman's scheme is based around a central structure that chooses and broadcasts the steps needed to perform inference, while local computations are used to actually execute the steps on all nodes and links. However, Fahlman specifically notes that this model is not intended to do complex reasoning, and he does not suggest using it for variable binding. Furthermore, while the system is capable of interesting inference (and could implement SHRUTI-like inference, thanks to its computationally flexible components), it is not designed to be neurally plausible. Lange (1992) discusses the advantages and disadvantages of marker passing models in general.

## 2 Our approach

Our model of neural binding in inference uses ideas from the signature, marker passing, and temporal synchrony approaches. It seems unbiological to assume that a network of neurons can reliably pass around enough signature information to uniquely identify any concept (∼20 bits). However, since we know that people can only deal with a limited number of variables (as many as about 8 (Miller, 1956), and possibly as few as 4 (Cowan, 2001)), a network only needs to reason with a few concepts at a time. Like SHRUTI, we suppose that the network only manipulates patterns that describe which of a conservative estimate of about 4-8 entities (∼2-3 bits) is involved in the current reasoning process. Then, we can use some of the same structures pioneered by the temporal synchrony approaches. Loosely following the usage in conventional logic, we will call these ∼8 short signatures **fluents**, and each one temporarily represents and is bound to a single object or concept.

The basic idea of replacing eight time slices with a 3 bit signature is straightforward and was suggested by Browne and Sun (2000). In fact, SHRUTI's computer simulation is implemented this way. However, while such networks are simple for computers to implement, it is nontrivial to develop a biologically plausible realization of this idea. Thus, one of our primary contributions is an explanation of how such a combination of SHRUTI and signatures could work.

Our other main contribution is the idea of a central structure that controls binding, thereby enabling some operations that SHRUTI cannot perform. First, it permits the network to keep track of specific bindings, where otherwise they would be lost as a time slice or signature spreads through the network. The central binding structure also allows for more complex abilities, such as the unification of signatures that have been determined to represent the same object. Furthermore, a central binder allows conflict-free fluent allocation, which cannot be performed without global information. This structure is discussed in § 2.2.

To show the feasibility of such a localist inference network, we have implemented a computer simulation intended to help solidify our model by exposing any weaknesses and causing us to consider every detail (Barrett et al., 2006b). It is designed to be biologically plausible, with simple nodes performing simple computations, although it contains some non-biological simplifications and does not implement every technique described in this paper, as we will explain.

Finally, it should be mentioned that, like SHRUTI, this model is difficult to describe in only a few pages. We describe it as well as we can given the constrained amount of space; please refer to the auxiliary tech report and interactive website (Barrett et al., 2006a,b).

## 2.1 Basic inference structures with fluents

The general structure of our model is based on SHRUTI (Shastri, 1999). In particular, the mechanisms for storing short-term relational knowledge, such as *Own(John, Book117)*, and computing implications, such as *Give(X,Y,M)* $\rightarrow$ *Own(Y,M)*, are similar to SHRUTI's—except for the treatment of dynamic binding, which is the main contribution of this paper. Here, we review SHRUTI's inference structures and the modifications we must make in order to use fluents instead of temporal synchrony for binding.

The most basic structures in our model, as in SHRUTI, are **nodes**, representing clusters of between several and hundreds of neurons that behave as simple computational units. There are two types of nodes: the **activation node** holds a numerical value, like SHRUTI's atemporal nodes, and the **fluent node** holds a fluent, just as SHRUTI's temporally-firing nodes would fire in a time slice. (A fluent node could be made of a few activation nodes, so that a pattern of activity of these represents a fluent. This might be done in any number of ways; the important thing is that it must hold a short signature representing 3 bits of information.) Weighted connections between nodes are used to perform calculations; for example, a node could be connected so as to represent the boolean "and" value of two other nodes, to pass values and fluents, compare fluents, and so on.

Our system uses combinations of nodes to represent short-term beliefs about the world. Each logical predicate is represented by a group of nodes called a **predicate cluster**. A predicate has a semantic meaning (e.g., the predicate in Figure 1 represents the relation *Give*), and it has one or more **roles**, which are implemented as fluent nodes and can be filled by fluents representing concepts. When a predicate (such as *Give*) has bindings (such as *giver=Mary*, *recipient=John*, *object=book117*), then the predicate describes a semantic relation between its roles (e.g. "Mary gave book117 to John."). These predicate clusters represent only short-term bindings; once the activities representing the role bindings have faded out, the information about the relation of these roles is lost. Importantly, these bindings can be fed from other short-term sources, so that information that is observed, heard, or inferred can be immediately represented and used for further inference. To store long-term information, we use fact structures, which are described below, to feed information to the predicates.

Each predicate cluster also has a pair of **collector nodes**, *+* and -, which indicate the degree of belief in the instantiated relation. They are mutually inhibitory activation nodes, so positive and negative evidence compete. If neither node wins, the predicate indicates uncertainty. In our model, as in SHRUTI, the predicate also has a **question node**, *?*, which is an activation node that is activated when the current set of bindings is being queried. This signal causes the system to actively seek out long-term knowledge and implications of this relation by passing fluents and questions to structures that might support this, and they then pass back bindings and *+* or - signals. It is the structure of the network that controls how these signals are passed and how different predicates imply each other. So in a properly-structured network, the question *?Give(Mary,John,book117)* (Did Mary give John book117?) will yield the answer *+Give(Mary,John,book117)* (Yes, Mary gave John book117.) if there is evidence to support this.

To store long-term knowledge, we use a **fact** structure. Each fact is associated with and connected to a particular predicate. It represents a mapping from a specific set of bindings of that predicate to a truth value. When a predicate's question node is activated, activation spreads to the question node of all related fact clusters. If the bindings match, then the fact sends a signal to the predicate. For example, if we have the fact "John owns book 117" (Figure 1), when we pose the question "Does John own book 117?" the corresponding fact will activate the query's plus collector, verifying that John does indeed own book 117.

Facts and predicates alone are not enough to perform inference, as they do not handle multiple antecedents, role consistency, and default role filling. These functions are performed by a **mediator cluster** (Figure 2), which conjunctively connects one or more antecedent predicates to a single consequent predicate. It has a role for every variable used by the antecedents and consequent, a single collector *+*, and a query node *?*. The collector is linked to the positive or negative collector of each antecedent, so that it will be active if all the antecedents are active, thus performing a conjunction over evidence. It also has a connection to the positive or negative collector of the consequent so that (positive or negative) conclusions will follow. When the consequent's *?* node is active, bindings are passed from that predicate to the mediator and then on to the antecedents, so that questions about objects are passed onwards. Because a mediator's collector can connect to both *+* and - collectors, it represents implications that can include negation in any antecedent or consequent.

Predicates and mediators perform different tasks. Predicates have semantic meaning and, by their collectors' connection weights and activation functions, perform "or" operations over mediators, with possible inversions caused by negative evidential weights. Mediators, on the other hand, represent implications and perform "and" operations over predicates, again

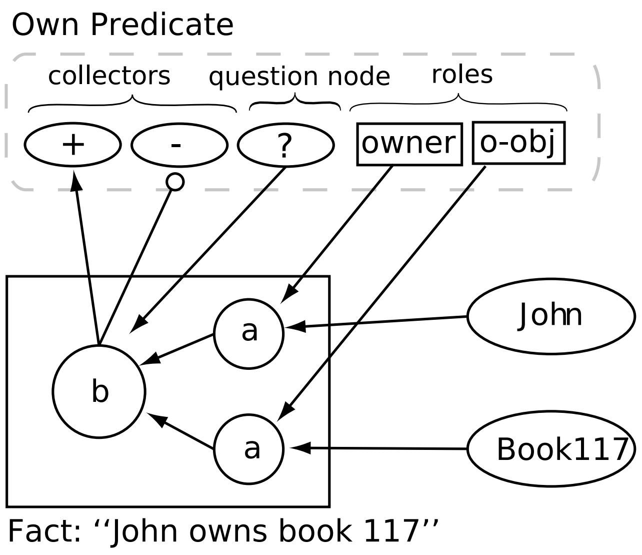

Figure 1: A predicate that represents working-memory "own" relationships, connected to a fact representing long-term knowledge of "John owns book 117"

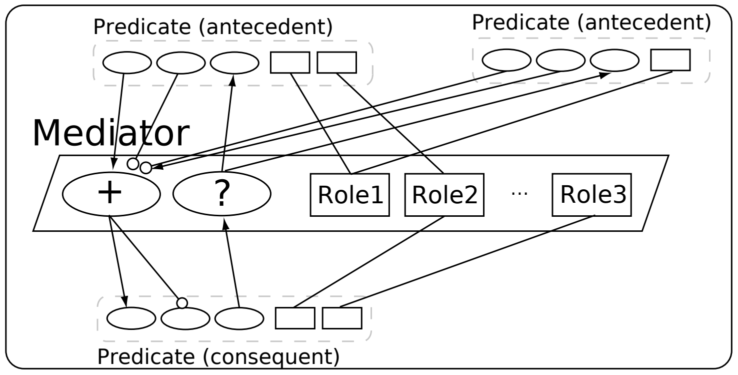

Figure 2: Detail of a mediator that implies its consequent when its antecedents match

with possible inversions. Thus, the basic logical operations (and, or, not) are supported. Also, predicates can be implied independently by many different circumstances, but each implication requires a conjunction of circumstances.

A very common type of relationship that an inference system must represent is the "is-a" relationship. To represent this simply, we include an **ontology** comprising a hierarchy of types. Each node in the ontology represents a single entity or category, and it sends activation to its super-categories and its subcategories via two separate paths. (By using two separate pathways, we can ensure that only sub- and super-types, and not other types, are activated.) No implication rules directly use nodes in the ontology as their antecedents; instead, facts bridge the gap between predicates and entities in the ontology. All of this is the same as SHRUTI.

However, there is one major complication that is introduced by our alternative to SHRUTI. SHRUTI's equivalent of a fluent node, called an m-$\rho$ node, can fire in more than one time slice, allowing it to simultaneously indicate more than one binding. This ability is not used in predicates, where only one binding is desired, but only in the ontology, where we would like to allow one entity or type to be related to more than one variable. However, our proposed fluent nodes can only indicate one binding. This adds complications for the ontology because SHRUTI's ontology relies on the near-simultaneous propagation of several bindings through the ontology. Instead, to do computation equivalent to SHRUTI's, our model must either have mechanisms for allowing multiple fluents to move through the ontology at a time (perhaps by having a copy of the ontology for each fluent) or restrict access to the ontology to a single fluent at a time, thus time-sharing the ontology over a number of phases (each long enough for activation to pass up and down the ontology). We propose to use the latter, restriction-based method, in which the fluents' access to the ontology is controlled by a central structure, described below.

This also allows another change that makes the ontology simpler; since the object nodes need not pass a whole fluent, but just an activation signal, they need not have fluent nodes, but only activation nodes. However, this comes at the expense of making the fact structures more complicated, since they must now compare fluents with activation signals. It is possible to do this if the central structure broadcasts what fluent currently has exclusive access to the ontology, but obviously incurs some complexity. Our current version does in fact use this activation-node strategy, but it is not a central feature of the model.

## 2.2 Central fluent binder

The above-described system could perform the basic operations of SHRUTI, although additional complexity might be required in the ontology. However, there are still needed abilities, such as ways to allocate new fluents or to unify existing fluents that refer to the same object. The SHRUTI model seems to perform these tasks by using local mechanisms (Wendelken and Shastri, 2004; Shastri, 2000), but in order to work correctly, global information is needed. In particular, the model must know which time slices are active and how their bound objects are related, which rather spoils the localness of the mechanisms. We propose to embrace the global nature of these tasks and suggest the use of a central structure to handle fluent allocation and unification for the whole system. The presence of a global structure also reduces the complexity needed in our ontology.

We introduce a central structure, called the **fluent binder**, that dynamically links each fluent to its bound entity. Recall that inference involves only passing and comparing a fluent (about 3 bits), and the effect should be similar to transmitting the full ($\sim$20 bit) address of the entity. This indirect addressing is achieved by using the central binder to link from the fluent to the entity. At the computational level, we can imagine a centralized bank of 8 registers, each of which is linked to, and can activate, the represented item. We suggest two distinct means for performing this linking: The first, which we call A1, exploits standard

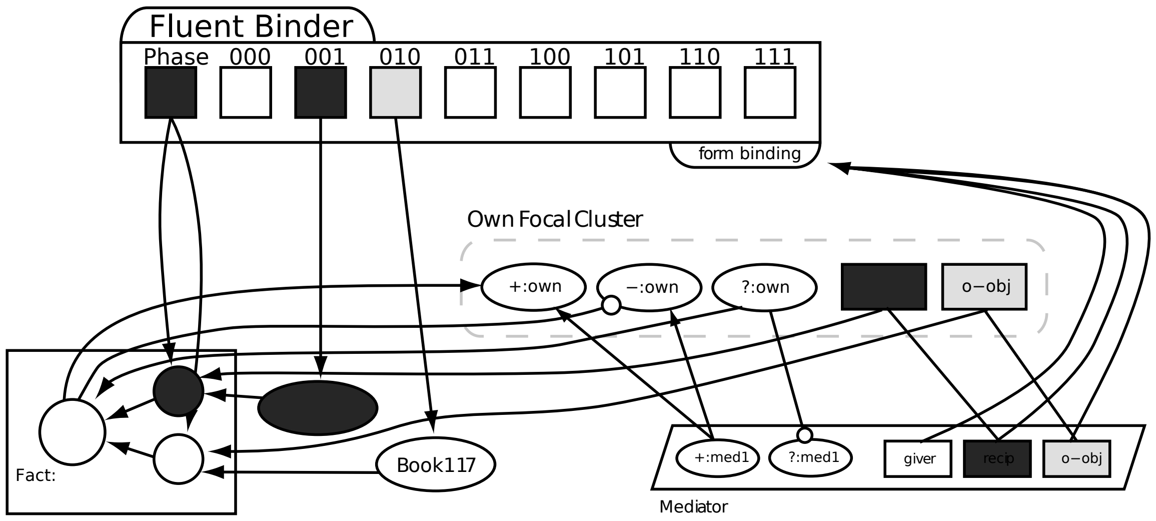

Figure 3: The fluent binder: the fluent registers are dynamically linked to objects in the ontology; the mediators' roles are linked to the fluent binder in order to request fluent allocation; and the fluent binder keeps track of the current phase of the ontology and broadcasts this to facts as needed

techniques for dynamic linking (Kappen and Nijman, 1995; Valiant, 2005). Essentially, the idea is that the object node and the desired fluent register, both embedded in a network of neurons, are strongly activated together, and then short-term potentiation creates a stable path through the network, linking the two. Then, to activate the object corresponding to a fluent, the fluent binder just activates that fluent register, and the activation spreads through the strenghtened connections in the network to the object node. The second solution, A2, uses tree addressing; in A2, the fluent binder's fluent register contains a bit string describing the path through a tree in which the leaves are entities in the ontology. To send activation to that entity, then, activation can be spread through a tree network that is switched by this address. Similarly, strong activation of both the register and the entity activates the relevant branches of the tree, allowing the bit string to be read off. Neither of these is implemented in the pilot system, but van der Velde and de Kamps (2006) use A1 in their neural blackboard architecture, and Tsotsos et al. (1995) has built essentially the same tree addressing method as A2.

One of the most important things that the fluent binder does is to allocate new fluents. With only local methods for assigning fluents, it is very easy for the same fluent to be assigned to several unrelated objects or variables at the same time, losing the valuable information the fluent is meant to carry. A central fluent binder could allocate fluents in a number of ways. One obvious method is simply to include a hierarchical winner-take-all network that selects a single variable at a time. Then, the binder can send back the next unbound fluent down the winner-take-all network in response. This is how our computational model implements this procedure. This hierarchical winner-take-all network should not take long to allocate a fluent, since with a branching factor of thousands, only a few levels are needed to select a

| Problem: | Approach: | Solutions: |
|---|---|---|
| A. initial setup | link input item to fluent | A1 ∼ X-bar; A2 ∼ address |
| B. propagate query | *?* node activation w/ fluents | B1 ∼ basic |
| C. get ± answer | return a single + or - | C1 ∼ basic |
| D. get item answer | bind each open fluent | D1 ∼ base, D2 ∼ unification |
| E. multiple instances | copies of logic terms | E1 ∼ extends B1, B2 |
| F. use ontology | taxonomic inferences | B2 ∼ with ontology |
| G. multiple answers | return multiple bindings | G1 |

Table 1: Subtasks and Possible Solutions

node even from the many that make up the ontology. A diagram showing the basic structure and connections of the fluent binder is shown in Figure 3.

The fluent binder is also useful for other tasks, including unification. These are discussed below, and unification in particular is discussed in § 3.8.

## 3 Subtasks

Even a simple inference case requires solving a number of problems in neural computation, and each problem might be approached in various ways. We have implemented a system, with particular choices of design, to test and illustrate the ideas, but we will describe a range of possible designs that might be of interest. Of course, what we really want is not just a means for doing computation, but also a model of how our brains actually work. So, we describe a range of computationally and biologically plausible solutions to these problems that can be sorted out by means of experiment. For convenience, we will name and label various tasks and proposed models of how they might work; they are enumerated in Table 1.

### 3.1 Subtask A. Linking a fluent with a fixed item

This is discussed in § 2.2 and is not repeated here.

### 3.2 Subtask B. Propagate a query though the structured rule and fact set

The SHRUTI group has worked extensively on connectionist variable binding and inference (Shastri, 1999), and our proposal derives largely from their effort. Like SHRUTI, we use a structure of predicates, mediators, and facts to represent and propagate role bindings, queries, and beliefs. The basic design is simple – each predicate or mediator cluster represents a logical predicate and has circuitry for representing both query (*?*) activation and bindings.

By connecting these nodes properly, queries and fluents pass from predicates to their causes as discussed in § 2.1; the detailed mechanics are described in Barrett et al. (2006a). This is process B1.

## 3.3 Subtask C. Return a + or - answer to a query with all fluents specified

As with subtask B, our solution builds on the SHRUTI model. Each predicate cluster has connections and local state for conveying a true (+) or false (-) answer. Mediator clusters need only carry a true (+) signal, because the converse could be represented by a second mediator with different weights on its connections. If a predicate or mediator cluster has an answer of any sort, it passes that answer back along connections going the opposite direction as the question signals. Similarly, facts check that their stored entities match their fluent inputs (which requires explicit comparison with the fluent binder's variable values, since the fact's stored entities cannot be represented as fluents), and this is how the system retrieves its stored knowledge. This is discussed some in § 2.1, and the details of these connections are described in Barrett et al. (2006a).

## 3.4 Subtask D. Returning unique answer items as well as + or - answers

We often want to know specific items that satisfy a query, such as *?Color(grass, X)*, which asks what color grass is, or *?Gives(Y, Bill, Z)*, which asks who gave what to Bill. Subtask D concerns the restricted case where there is only one answer returned for each open fluent, while subtask G (§ 3.7) addresses the more general case with multiple answers. For this restricted case, the new requirement is that a specific item gets associated with each open fluent ($X$) in a successful query answer. Our solution to this task involves all of the previous algorithms. The first operation is that of Task A—associate a particular item with a specific fluent. The trick is to simultaneously highly activate both an open fluent and its filler.

For simplicity, assume that all facts are unary or binary and that each fact is mutually linked to the items involved (this could be through triangle node linkage (Feldman, 1982)). Let us consider a simple case, the query *?color(grass, X)*. We suppose that grass is linked to fluent *f1*, and fluent *f2* is associated with the open variable $X$ (indicated by the fluent binder having that fluent register flagged as "open"). Now algorithm B1 can be run, yielding activation of all rules and facts that might bear on our query. Assuming that the only relevant fact is *color(grass, green)*, then the node for this fact would be activated and this yields high activation for the item *green*. In this very simple case, the simultaneous activation of *f2* and *green* will suffice to set up the required binding and thus fulfill task D. That is, *green* will be bound to *f2* and *f2* will propagate with a + signal back to the original query. Of course, it may not necessarily be true that the highest-activated entity is the correct binding for the fluent, so the system must always check it. Fortunately, it is easy to check—the system simply runs inference again with the fluent bound and verifies that the predicate is satisfied. If it is not, then it can inhibit that entity and run this procedure again. By applying this procedure once for each open fluent, an inference system could bind objects to as many fluents as needed.

## 3.5 Subtask E. Using multiple copies of clusters

The design so far described employs a cluster and its connections to encode a logical term and its relations to other terms. In a given query, each role of a predicate cluster becomes associated with a fluent. However, it is frequently the case that a query will involve two distinct assignments for the same logical term. For instance, the rule

$$\textit{Owns(X,M) and Owns(Y,N)} \rightarrow \textit{CanSwap(X,Y,M,N)}$$

requires that two separate instances of the Owns relation, linked to different fluents, be active in the same inference episode. The obvious way to represent this in a network like ours is to have multiple copies of each predicate and mediator. Unfortunately, these multiple instances must somehow be connected to each other in a useful way. There are several ways to make this work. Wendelken and Shastri (2004) propose that there be one mediator for each combination of predicate copies in an implication, though of course the number of such copies grows combinatorically.

We explore another option. The key idea is that all copies of a cluster are part of a coherent neural structure that we call a **family**. The family consists of the individual clusters plus a bit of control logic. At the start of an inference episode, all members of each family are available for binding. When one of these gets *?* activation, the accompanying fluent numbers are assigned to its first available copy and this is marked as busy.

If a family gets a *?* input with fluent numbers different from those stored in the first copy, it assigns them to its second copy. (This is done simply with inhibition and enabling of connections by the active first copy.) Whenever a family needs more copies than it has, the system simply ignores them, resulting in a failure of inference for that query. We can call this revised query propagating algorithm E1; it extends B1 and is also compatible with adding an ontology as described in the next section.

Some inter-family communications are easier than others. As a trivial case, each mediator family implies exactly one predicate family and, in fact, each mediator corresponds to exactly one predicate. So, it is simple to send responses from the mediator family to the predicate family; one simply connects the response directly from the specific mediator to the corresponding predicate, as we have previously described. The mapping is 1-1, so the messages are easy to route, and the existence of families adds no complications. Similarly, sending questions from predicates to mediators is easy for exactly the same reason. This structure can be seen in Figure 4.

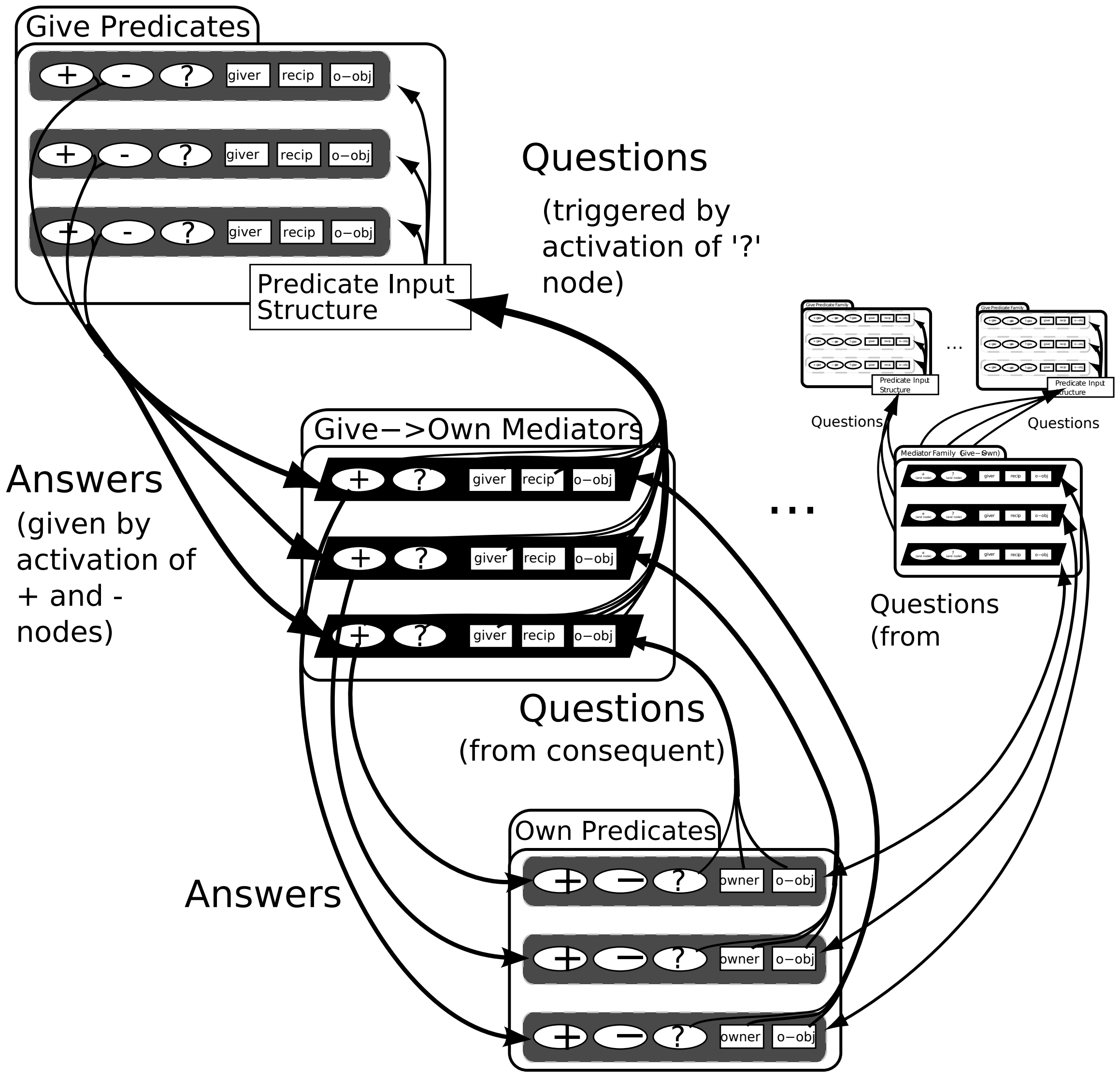

Figure 4: Overview of inference with multiple clusters

The difficult communications come when mediators ask questions of predicates and when predicates send responses. Each predicate may be queried by many different mediators in many families, and its responses are not necessarily useful to all mediators. These properties make interference likely, so we must be sure that the messages are routed appropriately. This is done with winner-take-all message receivers, called "question input structures." When a mediator needs to send a question to a predicate, it sends a message to the predicate family's question input structure. The question input structure is a selection network that chooses only a single question input at a time, as shown in Figure 5. It then routes the question to an unoccupied predicate (such routing can be done easily by a simple network of inhibitory nodes) and continues on to the next question. Thus, it prevents interference between questions to the same predicate family.

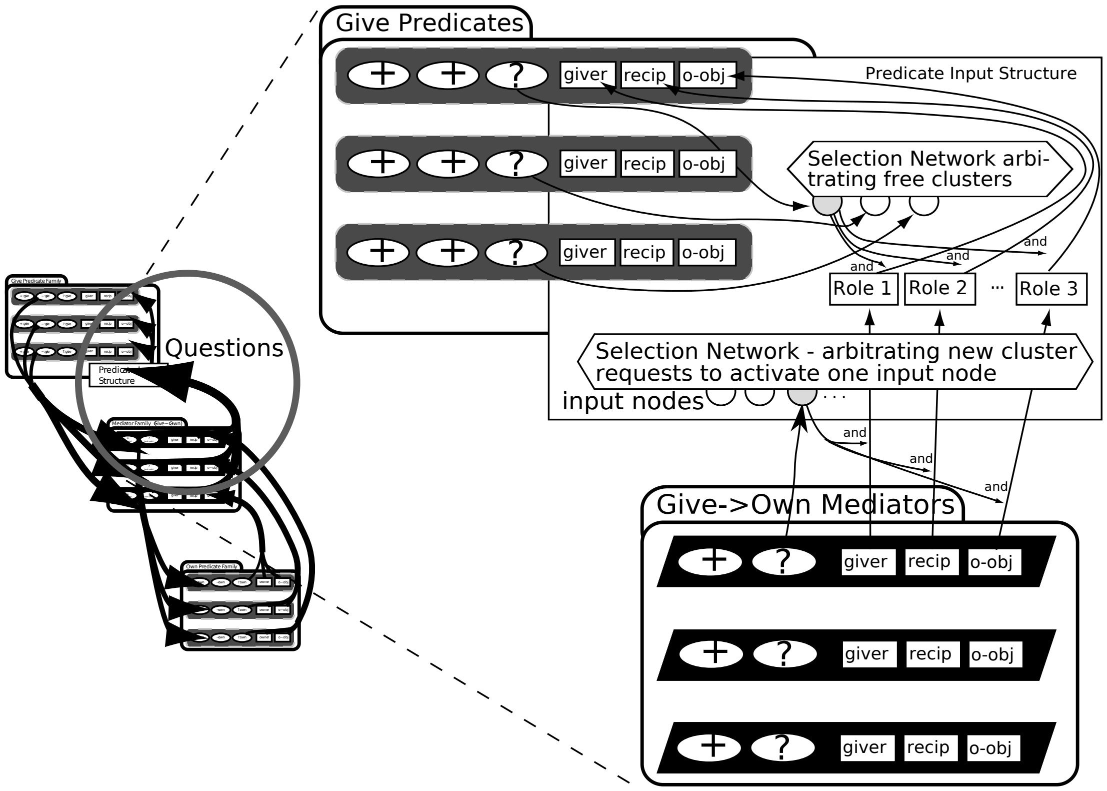
Give Predicates
Predicate Input Structure
giver
recip
o-obj
Selection Network arbitrating free clusters
and
Role 1
Role 2
Role 3
Questions
Selection Network - arbitrating new cluster requests to activate one input node
input nodes
Give->Own Mediators

Figure 5: Details of the Predicate Input Structure

Later, when a predicate has received an answer, the predicate sends a response to every mediator in every family that might possibly be able to use the result. All the mediators test for a match, which can be done easily by comparing fluents and inhibiting this input if any of the fluents of the mediator and predicate do not match. In this way, responses are routed appropriately.

## 3.6 Subtask F. Adding ontology structure to the set of items

This is discussed in § 2.1, and so is not discussed here.

## 3.7 Subtask G. Treating multiple answer sets

All of the discussion above assumes that at most one set of fluent bindings will be returned for a query. One might argue that this is a reasonable model of automatic, reflexive reasoning, but multiple answers are sometimes needed internally for an inference system to find one consistent set of bindings. For instance, one might want to query all of John's children (*Child(John,X)*) to determine if one of them owns *book117*. One possible solution to this issue is to simply extract one binding at a time, using technique D1 (§ 3.4). The first binding is extracted by following D1 to get a set of bindings. Then, the query is posed again with the previously-bound objects suppressed, so that re-running D1 finds new ones. In this way, an inference network can extract multiple bindings. This is algorithm G1.

## 3.8 Unification

In a computational realization of logical reasoning or natural language understanding, unification is an important operation. Again, there are various versions of unification, but they all involve identifying two entities that were distinct and then treating them as the same. A simple case might involve recognizing that the Governor of California and the Terminator were the same person. Most of the previous connectionist models of variable binding lacked a way of doing unification, but it is easy to model with fluents. SHRUTI (Shastri, 1999) is proposed to perform unification by somehow merging time slices that have a large degree of correspondence, but it is not clear how this can be done without some sort of global system (not present in SHRUTI) to detect the correspondence and then actually change the firing of all those nodes.

We treat unification by assuming that each fluent "register" in the fluent binder has a way of indicating that it has been unified with some other fluent. Performing unification is then just setting this tag for one fluent and using the other one as usual. That is, the active fluents will get bound to specific entities as part of the answer generation of algorithm D1. Unified fluents will refer indirectly to the entities of their partners. This leads to a minor modification in the algorithm for checking and returning fluent bindings.

In general, when a cluster receives a +/- answer signal, it always needs to compare the accompanying fluent numbers for each position with the ones it has stored from the query. When the fluent binder is signaling information about unification, this check also is satisfied if the incoming fluent number has been unified with the original one. This modification for task D is called D2. This is another instance where the passing and storing of explicit fluent bindings supports functionality not available with time slices. This was not implemented in the demo system.

We do not propose a system for detecting when unification should be performed, just as we do not propose a system for entering the initial questions. Such detection (and question-posing) would have to be complex, making use of available reasoning, context, and linguistic cues, so it is beyond the scope of this paper.

## 4 Discussion

### 4.1 Simulation

We performed a computer simulation of this model, largely using simple computational units so as to verify the model's feasibility. There is not room to discuss it here, but see Barrett et al. (2006b,a).

### 4.2 Learning

Any serious neural modeling effort should include considerations of the how the proposed structures might arise in the brain. Conceptual knowledge is obviously not innate and thus requires some kind of learning story. We can assume that a basic architecture of neural clusters like our fluent binder, predicates, and mediators could be part of our genetic endowment. The question is then how these might get wired up correctly as new concepts and relations are learned. The standard structured connectionist approach to this kind of problem is *recruitment learning* (Feldman, 1982; Browne and Sun, 1988). The basic idea is that there is a pool of uncommitted linking units that can be recruited to build conceptual structure.

The SHRUTI project has done extensive investigation of learning for their theory. This includes not only computational models and simulations but very detailed analysis of how recruitment learning might map on to the hippocampal complex (Shastri, 2001a). Virtually all of that work can be carried over to our model. However, there are still a large number of details that would need to be worked out. At this point, one can only say that there is no apparent barrier to building a fluent-based conceptual learning model, but it would be at least a doctoral thesis worth of work.

### 4.3 Biological plausibility

More generally, the validity of any connectionist proposal depends on its biological plausibility. As a computational level model, our fluent binding system implements all of the required functionality while honoring the known biological and behavioral constraints: It reasons about its knowledge base using parallel mechanisms, so it takes time proportional to the depth of the inference chain rather than the size of the base. It performs this reasoning with sparse, structured coding, like the best connectionist models to date (van der Velde and de Kamps, 2006), and uses simple computational units that could be built from neurons or collections of neurons. Much of this has already been discussed relative to SHRUTI, and our model is similar enough to SHRUTI that much of that argument holds (Shastri, 2001b). There seems to be no way to eliminate some common attentional mechanism like the central binder of § 2.2 (Simons and Chabris, 1995).

The structures and units in our model (and all others), though computationally simple, can seem quite complex. Complexity is often considered not to be biologically plausible since it can be hard to imagine how such complexity would arise. However, we should not be surprised to find that the reasoning abilities unique to humans require some complex structure, especially given the many cell types and varieties of connections found in the brain. And surely complexity should not seem less plausible in the brain than in other body parts; even more apparently simple systems, such as skin or blood, are wildly complex, with huge numbers of interacting proteins, genes, and cell types. Furthermore, the complexity we propose is a few structures that are repeated many times, and these structures would only need to arise once and be replicated, rather than requiring that each copy be learned independently.

The biological plausibility of a central binder seems to be the most controversial aspect of this model. As all neuroscience students have been reminded many times, the brain is a parallel, distributed system, and there is no particular component that is critical for all systems. However, there are many centralized or sequential processes and tasks, such as attention and generating elements of arbitrary categories (e.g. "Name a book where a female character has a masculine name."), so it should not be surprising to hypothesize at least one logically (if not physically) centralized element. Furthermore, the various components of the fluent binder have simple, plausible implementations such as crossbar networks and winner-take-all networks.

As we noted, the model and simulations in this paper are unrealistically binary and deterministic. We see no inherent problem in redesigning it with more realistic abstract neuron models. van der Velde and de Kamps (2006) do this kind of recasting of their similar style model into physics-based units, but we believe that a statistical treatment along the lines of (Averbeck et al., 2006) would scale better.

## 5 Conclusions

It has been clear for decades that variable binding is a difficult task for a massively parallel inference system. Human brains need this ability for routine thinking and language use, and computational considerations dictate that the mechanism be quite direct. The structures and algorithms in this paper provide a range of suggestions on how our brains might get the effect of variable binding while meeting the biological and computational constraints.

Of course, the model and implemented simulation is only a coarse digital approximation of a complex analog/chemical reality. No one believes that the brain has registers or communicates with binary signal pathways, but the idea of storing and communicating enough information to resolve approximately 4-8 alternatives is quite plausible. We have suggested a variety of alternate representations and algorithms that are consistent with known results. It would be great to get experimental data that help choose among alternatives, but even before that more computational work can help provide further insight into the brain.

## References

B. B. Averbeck, P. E. Latham, and A. Pouget. Neural correlations, population coding and computation. *Nature Reviews Neuroscience*, pages 358–366, May 2006.

D. H. Ballard. Parallel logical inference and energy minimisation. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 203–208, 1986.

L. Barrett, J. Feldman, and L. Mac Dermed. A (somewhat) new solution to the binding problem. Technical Report tr-06-001, 2006a.

L. Barrett, J. Feldman, and L. Mac Dermed. Fluent inference model demonstration. url(http://fluents.barrettnexus.com/), 2006b.

A. Browne and R. Sun. *Connectionist variable binding.* Springer Verlag, Heidelberg, 2000.

A. Browne and R. Sun. Connectionist recruitment learning. *European Conference on Artificial Intelligence*, pages 351–356, 1988.

N. Cowan. The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 1(24), 2001.

Scott E. Fahlman. Marker-passing inference in the scone knowledge-base system. In *First International Conference on Knowledge Science, Engineering and Management (KSEM'06)*, Guilin, China, August 2006. Springer-Verlag.

J. Feldman. Dynamic connections in neural networks. *Biological Cybernetics*, 46:27–39, 1982.

P. W. Glimcher. *Decisions, Uncertainty, and the Brain.* The MIT Press, Cambridge, Massachusetts, 2003.

H. D. R. Golledge, S. Panzeri, F. Zheng, G. Pola, J. W. Scannell, D. V. Giannikopoulos, R. J. Mason, M. J. Tovee, and M. P. Young. Correlations, feature-binding and population coding in primary visual cortex. *NeuroReport*, 14(7):1045–1050, May 2003.

R. Jackendoff. *Foundations of Language.* Oxford University Press, Oxford, 2002.

H. Kappen and M. Nijman. Dynamic linking in stochastic networks. In Moreno-Diaz R., editor, *Proceedings W.S. McCullock: 25 years in memoriam*, pages 294–299, Las Palmas de Gran Canaria, Spain, 1995. MIT Press.

T. Lange and M. Dyer. Frame selection in a connectionist model. In *Proc. 11th Cognitive Science Conf*, page 706 713, Hillsdale, NJ, 1989. Lawrence Erlbaum Associates.

Trent E. Lange. Hybrid connectionist models: Temporary bridges over the gap between the symbolic and the subsymbolic. In John Dinsmore, editor, *The Symbolic and Connectionist Paradigms: Closing the Gap*, pages 237–289, 1992.

P.M.V. Lima. *Logical abduction and prediction of unit clauses in symmetric hopfield networks.* Elsevier, Amsterdam, The Netherlands, 1992.

G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63:81–97, 1956.

P. M. Milner. A model for visual shape recognition. *The Psychological Review*, (81):521–535, 1974.

R. O'Reilly, R. Busby, and R. Soto. Three forms of binding and their neural substrates: Alternatives to temporal synchrony. Technical report, Department of Psychology University of Colorado at Boulder, 2001.

M. Page. Connectionist modelling in psychology: A localist manifesto. *Behavioral and Brain Sciences*, 23(4):443–512, 2000.

E. Salinas and T. J. Sejnowski. Correlated neuronal activity and the flow of neural information. *Nat Rev Neurosci*, 2(8):539–550, August 2001. ISSN 1471-003X. doi: 10.1038/35086012. URL `http://dx.doi.org/10.1038/35086012`.

M. N. Shadlen and J. A. Movshon. Synchrony unbound: a critical evaluation of the temporal binding hypothesis. *Neuron*, 24:67–77, 1999.

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.

L. Shastri. A computational model of episodic memory formation in the hippocampal system. *Neurocomputing*, 38:889–897, 2001a.

L. Shastri. Comparing the neural blackboard and the temporal synchrony-based shruti architectures. *Behavioral and Brain Sciences*, 29:37–70, 2006.

L. Shastri. Types and quantifiers in shruti — a connectionist model of rapid reasoning and relational processing. In S. Wermter and R. Sun, editors, *Hybrid Neural Symbolic Integration*, Lecture Notes in Computer Science, Lecture Notes in Artificial Intelligence, Berlin, 2000. Springer-Verlag.

Lokendra Shastri. *Biological grounding of recruitment learning and vicinal algorithms in long-term potentiation.* Springer-Verlag, Berlin, 2001b.

D. Simons and C. Chabris. Gorillas in our midst: Sustained inattentional blindness for dynamic events. *British Journal of Developmental Psychology*, 13:113–142, 1995.

R. Sun. A discrete neural network model for conceptual representation and reasoning. In *Proceedings of the 11th Conference of the Cognitive Science Society*, pages 916–923, Hillsdale, NJ, 1989. Lawrence Erlbaum.

R. Sun. On variable binding in connectionist networks. *Connection Science*, 4:93–124, 1992.

J. Tsotsos, S. Culhane, W. Wai, Y. Lai, N. Davis, and F. Nuflo. Modeling visual attention via selective tuning. *Artificial Intelligence*, 78:507–547, 1995.

Leslie G. Valiant. Memorization and association on a realistic neural model. *Neural Computation*, (17):527–555, 2005.

F. van der Velde and M. de Kamps. Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, 29:84–86, 2006.

C. Wendelken and L. Shastri. Multiple instantiation and rule mediation in SHRUTI. *Connection Science*, 16:211–217, 2004.