Holyoak, K. J., & Hummel, J. E. (2000). The proper treatment of symbols in a connectionist architecture. In E. Deitrich & A. Markman (Eds.), *Cognitive dynamics: Conceptual change in humans and machines*. Cambridge, MA: MIT Press.

The Proper Treatment of Symbols in a Connectionist Architecture

Keith J. Holyoak John E. Hummel

University of California, Los Angeles

Running head: Symbolic Connectionism

Address for proofs: Keith J. Holyoak
Dept. of Psychology
Franz Hall
UCLA
Los Angeles, CA 90095-1563

Email: holyoak@lifesci.ucla.edu

Acknowledgements

Preparation of this chapter was supported by NSF Grant SBR-9511504. Eric Dietrich, Art Markman and Jay McClelland provided valuable comments on an earlier draft.

Physical Symbol Systems

A foundational principle of modern cognitive science is the Physical Symbol System hypothesis, which states simply that human cognition is the product of a physical symbol system (PSS). A symbol is a pattern that denotes something else; a symbol system is a set of symbols that can be composed into more complex structures by a set of relations. The term "physical" conveys that a symbol system can and must be realized in some physical way in order to create intelligence. The physical basis may be the circuits of an electronic computer, the neural substrate of a thinking biological organism, or in principle anything else that could implement a Turing machine-like computing device. Classical presentations of the PSS hypothesis include Newell and Simon (1976) and Newell (1980); more recent discussions include Newell (1990) and Vera and Simon (1993, 1994).

The PSS hypothesis, which implies that structured mental representations are central to human intelligence, was for some time uncontroversial, accepted by most cognitive scientists as an axiom of the field scarcely in need of either theoretical analysis or direct empirical support. In the mid-1980s, however, the hypothesis came under sharp attack from some proponents of connectionist models of cognition, particularly the advocates of models in the style of "parallel distributed processing", or PDP (Rumelhart, McClelland & the PDP Research Group, 1986; more recently, Churchland, 1990, 1995; Elman, 1990; Elman et al., 1996; Seidenberg, 1994, 1997; and many others; see Marcus, 1997, for a review). The representations used in such models are often described as "subsymbolic" because the elementary units correspond to (relatively) low-level features, over which meaningful concepts are represented in a distributed fashion. In so far as models based on "subsymbolic" representations are actually non-symbolic, yet adequate as accounts of human intelligence, the need for symbol systems would be eliminated; hence models of this general class constitute "eliminative" connectionism (Pinker & Prince, 1988). Eliminative connectionism offers a direct challenge to the PSS hypothesis, thereby transforming the latter from an axiom of cognitive science into a controversial theoretical position, which has been vigorously

defended by Fodor and Pylyshyn (1988), Pinker and Prince (1988), and Marcus (1997), among others.

Regardless of whether models based on distributed representations provide genuine alternatives to physical symbol systems, it is apparent that they have attractive properties as possible algorithmic accounts of cognition. Discrete symbols represent entities in an "all-or-none" fashion, thereby violating the principle of least commitment (e.g., using the presence or absence of the symbol "dog" to represent the presence or absence of a dog affords no direct basis for expressing inconclusive evidence that there may be a dog). Discrete symbols also fail to express the semantic content of the represented entities (e.g., the symbols "dog" and "cat" do not signify what dogs and cats have in common and how they differ). Distributed representations overcome both these limitations, capturing some basic properties of human perception and thinking more effectively than do classical symbolic representations. By allowing similar inputs to elicit similar outputs, distributed representations capture broad regularities in human inductive inference, and also endow the system with error tolerance. They also support a variety of learning algorithms that can capture regularities in environmental inputs, and which provide simple types of automatic generalization.

Another desirable property of connectionist architectures is that they are at least roughly consistent with neural architectures: Both consist of discrete computing elements that communicate in densely-connected networks. In contrast to symbols in a traditional symbolic system, which can move around freely (e.g., from one function or role to another), nodes occupy fixed locations in connectionist networks, much as neurons occupy relatively fixed locations in the brain. As we will see, this difference between symbolic systems on the one hand and connectionist or neural systems on the other is important because it implies that nodes or neurons in a network need some special properties to bind fillers to roles or values to variables—the “binding problem” poses difficulties for the architecture of connectionist and neural networks. More generally, connectionist models provide a convenient language for linking cognitive phenomena to their possible neural substrates.[1]

But is it possible, or even desirable, for connectionist models to eliminate physical symbol systems? This question really has two parts. The first is, can distributed connectionist models

eliminate *symbols*? The answer to this question hinges on a terminological issue regarding what it takes to be a "symbol". If a symbol is narrowly defined as an atomic unit corresponding to a concept, then feature-based models may indeed be subsymbolic. But if a symbol is defined more broadly as a representation that designates something, then distributed representations are as symbolic as the localist variety (see Touretzky & Pomerleau, 1994, and Vera & Simon, 1994, for a debate that focuses on this definitional issue). We find the less restrictive definition to be more useful, but will not consider this part of the question further. The second part of the question is more substantive: Can distributed representations eliminate symbol *systems*? That is, is it possible to model the full scope of human cognition—including reasoning, relational generalization, language use, and complex object and scene recognition—with representations that do not allow the systematic composition of complex structures from simpler elements?

We will argue that the answer is "no". If this answer is accepted, then it follows that the PSS hypothesis is correct, and the ultimate aim of eliminative connectionism is unattainable. However, the PSS hypothesis itself is an abstract description of the requirements for a cognitive architecture, rather than a prescription for any particular architecture. The core difference between the PSS hypothesis and the eliminative connectionist hypothesis is that the former postulates systematic, compositional mental representations, whereas the latter rejects them; hence the resolution of the debate hinges solely on the compositionality of human mental representations. But the failure of eliminative connectionism (which founders on the compositionality of human mental representations) does not obviate the potential virtues of more realistic connectionist instantiations of the human cognitive architecture. What is required, then, is not eliminative connectionism, but rather a proper treatment of symbols within a connectionist architecture—an architecture that simultaneously retains the strengths of distributed representations and instantiates the PSS hypothesis—and hence constitutes *symbolic* connectionism (Holyoak, 1991; Hummel & Holyoak, 1998).

In the remainder of this chapter we will develop the case for symbolic connectionism. We first review evidence that central aspects of cognition depend on compositional symbol systems.

We will then suggest certain requirements for a proper treatment of symbols in a connectionist network. Finally, we will sketch an example of a connectionist architecture for reasoning and learning that meets these requirements.

Roles and Fillers: The Necessity for Variable Binding

The best-known argument for the necessity of symbolic representations—the argument from systematicity—was made by Fodor and Pylyshyn (1988). They observed that knowledge is systematic in that the ability to think certain thoughts seems to imply the ability to think certain related thoughts. For example, a person who understands the meanings of the concepts "John", "Mary" and "loves", and can understand the proposition "John loves Mary," must surely be able to understand the proposition "Mary loves John." Eliminative connectionist models do not ensure such systematicity. (In fact, as elaborated shortly, they ensure the *absence* of truly general systematicity.) A network of the PDP type can learn to respond in an appropriate fashion to an input representing any particular proposition; however, there is no assurance that learning one proposition will enable a sensible response to a systematically-related proposition (see Marcus, 1997).

Systematicity is the hallmark of a system in which complex symbols are composed in a regular fashion from simpler ones (see Halford, Wilson & Phillips, in press). More primitive varieties of cognition can safely rely on specialized representational systems that do not require composition of complex symbols; instead, every significant stimulus configuration can be linked to appropriate responses, either innately or by associative learning. Within this range, eliminative connectionist models may well be adequate. Strong evidence for systematicity has only been found for higher primates, most notably humans. Newell (1990) characterized the development of compositional symbol systems as the "Great Move" of evolution, triggered by the pressure to represent and manipulate increasingly diverse information about the physical and social environment. For example, humans can recognize scenes in which known or novel objects enter into varied spatial relationships. Thus the relation *above (object1, object2)* can be instantiated by a triangle above a square, *above (triangle, square)*, or the reverse, *above (square, triangle).* Human

scene recognition is systematic with respect to a limited set of spatial relationships, and for this reason requires models based on composed symbols (i.e., structural descriptions; see Hummel, this volume). Thinking and language require systematicity on a grander scale, as the pool of potential relationships over which complex symbols can be composed is indefinitely large (e.g., *loves (lover, beloved), sells (seller, buyer, object), pretends (person (is (object1, object2))*, and so on). There is reason to think that the human ability to represent and manipulate domain-general relations is linked to evolutionary advances in prefrontal cortex (Robin & Holyoak, 1994).

As all the above examples suggest, composability of symbols requires representations that distinguish variables from their values, or equivalently, roles from their fillers. "John loves Mary" is similar to "Mary loves John" in that both propositions involve the same relation and objects, but the two differ in that the assignments of objects as fillers of roles are reversed. It is this combination of similarity and difference between systematically related symbol structures that eliminative connectionist models fail to capture. Lacking any capacity to explicitly bind roles to their fillers, eliminative connectionist models must resort to various forms of conjunctive coding to bind fillers to roles (as elaborated shortly). For example, one node (or collection of nodes) might represent John in the agent role of the love relation (the conjunction *John + lover*), with a completely separate node (or pattern) representing John in the patient role (*John + beloved*). As a consequence, such models do not preserve object identities across relational contexts. This problem, already apparent with simple relational structures, becomes even more pernicious as the complexity of composed symbol structures increases. Eliminative connectionist models have only one basic resource for representing propositions: a fixed-length vector of units. This fixed vector thus becomes the procrustean bed into which all symbols must be force-fit. Because symbol structures can be of varying size and complexity, there is no way to guarantee that a given symbol will be represented on the same (or even overlapping) set of units in two different structures. Thus, the units that code "Mary" in "John loves Mary" may not overlap with those that code "Mary" in "Mary loves John", far less with those that code "Mary" in "John believes that Peter's anger toward Mary caused him to write her a strongly-worded letter."[2]

The inadequacies of eliminative connectionist models are especially apparent in reasoning tasks that require placing roles and fillers into correspondence (Barnden, 1994). Consider a simple inference rule, "If person1 loves person2 and person2 loves person3, then person1 is jealous of person3." We can readily recognize a match between the antecedent ("if") portion of the rule and the propositions, "John loves Mary" and "Mary loves George." The resulting inference, "John is jealous of George", requires carrying over the correspondences established for the "if" portion (John --> person1, George --> person3) to the "then" portion, and using them to create the structurally appropriate inference (and not, for example, "George is jealous of John"). Such structural inferences require more than detecting some global similarity between the specific propositions and the "if" portion of the rule. The global similarity between the propositions and the antecedent of the rule is (at best) enough to suggest that someone is likely to be jealous of someone else; it is not adequate to indicate who will be jealous of whom. Drawing this specific inference requires establishing, maintaining and using a set of specific correspondences between roles and fillers (i.e., a set of variable bindings). No model that lacks the capacity to preserve object identities across roles could make systematic inferences of this type.

These problems are not limited to reasoning based on established general rules with explicit abstract variables, such as "person1". Fundamentally the same issues arise in reasoning by analogy to specific cases. Suppose the reasoner lacked the "jealousy rule", but had encountered a specific situation, "Alice loved Sam, and Sam loved Betty, so Alice was jealous of Betty." Now the reasoner learns that John loves Mary and Mary loves George. Analogical mapping (e.g., Falkenhainer, Forbus & Gentner, 1989; Holyoak & Thagard, 1989) can readily establish the correspondences John --> Alice, Mary --> Sam, and George --> Betty. When these correspondences are passed to an inference engine capable of "copying with substitution" (Falkenhainer et al., 1989; Holyoak, Novick & Melz, 1994) from the source to the target analog, the conjecture "John is jealous of George" can be inferred. Moreover, once the target is extended by this inference, the full set of correspondences between the two analogs provides the basic ingredients for forming a new relational generalization. If the reasoner can take the structured intersection between the two

analogs, keeping the commonalities while dropping the differences (i.e., generalizing over John the man and Alice the woman to construct a "person" variable), then the result will be the "jealousy" rule. As has often been argued (Gick & Holyoak, 1983; Ross & Kennedy, 1990), analogical mapping sets the stage for relational generalization, which can yield abstract rules and schemas. But none of this is possible for models that lack the capacity to represent roles, fillers, and the bindings between them.

As the above examples illustrate, both rule-based and analogical inferences depend on the capacity to detect and exploit *linking* relationships between role assignments (or mappings). In the rule-based example, the binding John --> person1 links the "if" portion of the rule to the "then" portion; in the analogical example, the mapping John --> Alice links the initial mapping to the eventual inference. Marcus (1997, this volume) has shown that eliminative connectionist models, lacking the capacity for variable binding, are incapable of learning generalizations based on such linking relationships. Instead, such models are inherently limited to learning the specific instantiations of linkages that hold for the set of examples on which they are trained. Although an eliminative connectionist model can then make "inferences" on which it has been directly trained (i.e., the model will remember particular associations that have been strengthened by learning), the acquired knowledge may not generalize *at all* to novel instantiations of the linking relationships based on cases that lie outside of the training set (also see Phillips & Halford, 1997).

These limitations can be illustrated by the performance of a particularly sophisticated example of an eliminative model, the Story Gestalt model of story comprehension developed by St. John (1992; St. John & McClelland, 1990). In one computational experiment (St. John, 1992, Simulation 1), the Story Gestalt model was first trained with 1,000,000 short texts consisting of propositions based on 136 different constituent concepts. Each story instantiated a script such as "<person> decided to go to <destination>; <person> drove <vehicle> to <destination>" (e.g., "John decided to go to a restaurant; John drove a jeep to the restaurant"; "Harry decided to go to the beach; Harry drove a Mercedes to the beach"). After learning a network of associative connections based on the 1,000,000 examples, the generalization ability of the model was tested by presenting it

with a text containing a new proposition, such as "George decided to go to the airport," and having the model attempt to complete the "driving" script. St. John reports that when given a new proposition about deciding to go to the airport, the model would typically activate the restaurant or the beach (i.e., the destinations in specific prior examples) as the destination, rather than making the contextually appropriate inference that the person would drive to the airport. This type of error (which would appear quite unnatural in human text comprehension) results from the model's lack of a capacity to learn generalized linking relationships (e.g., that if a person wants to go somewhere, that place will be the person's destination). As St. John noted, "Developing a representation to handle role binding proved to be difficult for the model" (1992, p. 294).

A particularly simple example of a linking relationship that reveals such generalization failures is the identity relation. Holyoak and Thagard (1995) have argued that recognition of identity or sameness of one object to another is the most basic form of systematic analogical reasoning. The concept of identity appears to be within the cognitive capacity of both humans (including young children) and other primates. Both monkeys and chimpanzees are able to first learn to solve match-to-sample problems (e.g., picking a target object that is identical to a sample object), and then to transfer successfully to problems based on novel objects (e.g., D'Amato, Salmon, Lukas & Tomie, 1986; Oden, Thompson & Premack, 1988). The ability to transfer to new objects suggests that these primates can recognize and respond to the identity relation in a way that goes beyond the training examples.

Marcus (1977) has analysed the limitations of eliminative connectionist models in acquiring a function based on the identity relation. Suppose, for example, that a human reasoner was trained to respond with "1" to "1", "2" to "2", and "3" to "3". Even with just these three examples, the human is almost certain to respond to "4" with "4", without any direct feedback that this is the correct output for the new case. In contrast, an eliminative connectionist model (e.g., a feed-forward or recurrent network trained by back-propagation[3]) will be unable to make this obvious generalization. Such a model will have learned the specific input-output relations on which it was trained; but lacking the capacity to represent variables, generalization outside the training set is

impossible. In other words, the model will simply have learned to associate "1" with ""1", "2" with "2" and "3" with "3". A human, by contrast, will have learned to associate *input (number)* with *output (number)*, for any number; doing so requires the capacity to bind any new number (whether it was in the training space or not) to the variable *number*. Indeed, most people are willing to generalize even beyond the world of numbers. We leave it to the reader to give the appropriate outputs in response to the following inputs: "A"; "B"; "flower"; "My ability to generate these responses indicates that I am binding values to variables."

The power of human reasoning and learning, then, is dependent on the capacity to represent roles and bind them to fillers. This is precisely the same capacity that permits composition of complex symbols from simpler ones. The human mind is the product of a physical symbol system; hence any model that succeeds in eliminating symbol systems will *ipso facto* have succeeded in eliminating itself from contention as a model of the human cognitive architecture.

Three Requirements for a Symbolic-Connectionist Architecture

As we noted earlier, establishing the validity of the PSS hypothesis places broad constraints on the nature of the human cognitive architecture, but does not suffice to identify any specific architecture as psychologically real. Ultimately, the empirically correct model of the human cognitive architecture, as a *physical* symbol system, will need to specify the neural code for thought. A long road remains ahead before this goal is attained, as little is yet known about the detailed neural substrate for propositional representation. Indeed, it appears in retrospect that the attraction of eliminative connectionism was in part due to premature and overly-restrictive presumptions about "neural plausibility", according to which symbol systems (narrowly identified with specific "symbolic" architectures in the cognitive-science literature) were viewed as inherently neurally implausible. The unknown often seems implausible. But as Sherlock Holmes observed, once we have eliminated the impossible, what remains, however implausible, must be the truth. The human brain supports symbol systems; rather than pretending otherwise, we need to investigate how it does so.

There is nothing in the general notion of neural networks that precludes variable binding and composition of symbol structures. Indeed, many researchers in the connectionist tradition have seriously considered the question of how symbol systems could be embodied in a neural network (e.g., Feldman & Ballard, 1982; Hinton, 1990; Hummel & Holyoak, 1997; Plate, 1991; Pollack, 1990; Shastri & Ajjanagadde, 1993; Smolensky, 1990; Touretzky & Hinton, 1988). Given that the PSS hypothesis is accepted, and that the brain is apparently a neural network (of some sort), the search for the human cognitive architecture leads in the direction of *symbolic* connectionism (Holyoak, 1991).

It is not our purpose here to describe and evaluate in detail the many proposed symbolic connectionist models. Some models perform rule-based inferences (e.g., Shastri & Ajjanagadde, 1993) and a few perform analogical mapping (e.g., Halford et al., 1994), but only our own model (Hummel & Holyoak, 1997) performs a wide range of the types of structured comparisons typical of human symbol processing. Here we will state three apparent requirements for an adequate model of the human cognitive architecture that have motivated our own theoretical tack (Hummel & Biederman, 1990, 1992; Hummel & Holyoak, 1993, 1996, 1997, 1998; in press; Hummel & Stankiewicz, 1996), and that highlight limitations of alternative approaches (see also Hummel & Holyoak, 1998). Each of these requirements is motivated by a mix of computational considerations and empirical evidence about human cognition.

1) Independent, dynamic variable binding. The cognitive architecture must be a symbol system: It must enable structured comparisons between complex symbol structures, allowing the computation of systematic role-filler bindings, analogical mappings, and mapping of universal functions (see Holyoak & Thagard, 1995). This implies that it must provide mechanisms for the composition of symbol structures, and therefore, variable binding. A variable binding espresses a role-filler or variable-value conjunction, and has two essential properties.

1.1) Dynamic binding. A variable binding is *dynamic* in the sense that it can be created and destroyed on the fly: *John* can be bound to the agent role of *love (x y)* on one occasion and to some other role on another occasion.

1.2) Independent binding. The binding must be *independent* of the entities it binds. Binding is something a symbol system *does to* elemental units such as roles and fillers; it is not an intrinsic property of the units themselves, and it does not change the identities of those elements. For example, a propositional representation uses list position to express role-filler bindings: *John* is bound to the agent role of *loves (x y)* by placing it in the first slot of that predicate. This is a "true" variable binding because list position is external to (i.e., independent of) the elements themselves, so neither *John* nor *loves (x y)* changes as a result of the binding. This independence is important because it allows the representation of John in the context of "John loves Mary" to overlap in a perspicuous manner with the representation of John in "Mary believes that Susan's anger toward John caused her to write him a strongly-worded letter." The independence of binding and unit identity in human cognition is supported by the fact that people can effectively use constituents as retrieval cues to access larger structures stored in memory (e.g., Lesgold, 1972; Wanner, 1968). As discussed later, it is also supported by our ability to generalize rules universally.

It is important to distinguish independent binding from conjunctive coding, the dominant approach to binding in the connectionist literature. Conjunctive coding uses separate units (or patterns of activation) to represent separate bindings. For example, to represent *loves (John Mary)*, a conjunctive code would designate one unit or pattern, A, to represent the binding of *John* to the *lover* role, and a separate unit or pattern, B, to bind *Mary* to the *beloved* role; *loves (Mary John)* would be represented by two more patterns, C binding *Mary* to *lover*, and D binding *John* to *beloved*. Critically, A, B, C and D must differ from one another in order to bind objects to their roles unambiguously. As a result, John bound to lover (unit A) differs from John bound to beloved (unit D). Conjunctive coding is similar to variable binding in that it represents role-filler (or variable-value) conjunctions. It is also similar to variable binding in that it can be dynamic: It is possible to create and destroy conjunctive codes on the fly, as in the case of tensor product representations of binding (see Halford et al., 1994). But it differs from true variable binding because it carries binding information in the units themselves, rather than representing it independently of those units (i.e., conjunctive coding fails Requirement 1.2): A unit that represents

the conjunction *John+lover* is simply a symbol for that conjunction; it does not explicitly bind the symbol *John* to the role *lover.* As a result, conjunctive codes do not have the expressive power of symbolic representations based on independent dynamic variable binding (see also Hummel & Biederman, 1992).

2) Static binding in long-term memory. Although independent dynamic variable binding is a necessary prerequisite for symbolic representation, a cognitive architecture must also be able to establish static bindings—for example, by conjunctive coding—in order to code facts and rules in long-term memory (Hummel & Holyoak, 1993, 1997; Shastri, 1997). A code for independent dynamic binding based on temporal patterns (e.g., binding by synchrony of firing, as discussed shortly) is necessarily transient (hence naturally associated with working memory), and therefore must be supplemented by a static representation that stores bindings over extended periods of time. The static form of the binding must be capable of responding to the corresponding dynamic form (or a similar structure) when the latter enters working memory (recognition), and it must be able to reinstate the independent dynamic form when the structure (e.g., proposition) is called back into working memory (recall) (Hummel & Holyoak, 1997).

It is interesting to consider whether something analogous to the distinction between dynamic and static binding arises in a traditional symbolic representation. For example, does the symbolic representation of a proposition on the hard drive of a computer differ—in a way that is analogous to the dynamic/static distinction—from the representation of that proposition in the computer's random-access memory? Although these representations certainly differ in some respects (for example, the latter is represented as a set of electronic currents in the registers that comprise the computer's memory, whereas the former is a pattern of magnetic states on the computer's disk), it is unclear whether such differences map onto the dynamic/static distinction that arises for connectionist representations of symbolic structures. If not, then this would imply that Requirement 2 is unique to symbolic connectionist systems.

3) Distributed representations of propositional content. Finally, these representations and operations must be sufficiently robust to tolerate partial matches and imperfect correspondences.

This capability is essential to rule-based and analogical inference, as well as relational generalization. Therefore, concepts must have distributed representations of their meanings in order to provide simple mechanisms for error tolerance and similarity-based retrieval. In other words, symbols must be coded by distributed patterns, rather than atomic elements. Requiring that symbols have distributed representations implies acceptance of the broader definition of "symbol" advocated by Vera and Simon (1994).

Numerous connectionist models have been proposed that satisfy Requirements 2 (static binding for long-term storage) and 3 (distributed representations). However, localist connectionist models (e.g., Feldman & Ballard, 1982; Shastri & Ajjanagadde, 1993) lack the benefits of distributed representations (see Hummel & Holyoak, 1993), as do traditional symbolic models (e.g., Anderson's, 1993, ACT-R and its precursors; Rosenbloom et al.'s, 1991, SOAR). Most distributed models do not satisfy Requirement 1.2 (independent binding), in that the representation of a symbol in isolation (or as a constituent in one symbol structure) may have no overlap with the representation of the same symbol as a constituent in some other symbol structure. The models that exhibit this limitation include all eliminative models, as well as models based on tensor products (Smolensky, 1990) and their relatives, such as holographic reduced representations (HRR; Plate, 1991) and recursive autoassociative memories (the RAAM model of Pollack, 1990). This failure to satisfy Requirement 1.2—that is, to represent roles and fillers independently of their bindings—is the direct consequence of relying solely on conjunctive bindings.

The Problem with Tensor Products for Variable Binding

The fact that models based solely on static bindings fail to represent roles and fillers independently of their bindings has generally been overlooked, so we will sketch the reason for the problem (see Hummel & Holyoak, 1998, for a more complete mathematical proof). To a first approximation, tensor products and their relatives seem adequate as a solution to the variable binding problem (Requirement 1). However, inasmuch as satisfying Requirement 1 entails satisfying Requirement 1.2 (independent binding), tensor-based approaches are inadequate as a general solution to the binding problem. The limitations of tensor-based approaches—and the

importance of Requirement 1.2—are important but relatively subtle, and so warrant detailed consideration.

A tensor product is an outer product of two or more vectors. For example, in the case of a tensor, **ab**, formed from vectors **a** and **b**, the *ij*th element of **ab** is simply the product of the *i*th element of **a** with the *j*th element of **b** (see Figure 1):

$$\mathbf{ab}_{ij} = \mathbf{a}_i\mathbf{b}_j. \quad (1)$$

Tensors can be formed from any number of vectors in this way. For instance, a tensor can be formed from three vectors by setting the *ijk*th element of the tensor to the product of the *i*th element of the first vector, the *j*th element of the second, and the *k*th element of the third (see Figure 2). Smolensky (1990), Halford et al. (1994) and others have shown that tensor products can be used to bind variables to values, or fillers to roles. For example, as illustrated in Figure 2, it is possible to represent the proposition *loves (John Mary)* with a three-dimensional tensor, **abc**, in which one vector (**a**) codes the predicate (loves), the second vector (**b**) codes the filler of the agent role (John), and the third vector (**c**) codes the filler of the patient role (Mary). Switching the roles to represent *loves (Mary John)* changes the assignment of John and Mary to role slots, and thereby changes the tensor product. If *loves (John Mary)* is represented by **abc**, then *loves (Mary John)* would be represented by **acb** (compare Figures 2a and 2b).

Figures 1 and 2 about here

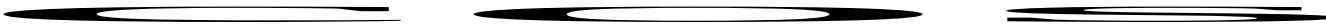
A tensor product is analogous to a weight matrix between the simple vectors from which it is generated (the product rule for generating the tensor is precisely a Hebbian learning rule; Smolensky, 1990). As a result, it can be used to answer "questions" about the bindings of roles to fillers. For example, consider the tensor representation of "John runs" in Figure 1a. Imagine activating the tensor and the vector for "run", leaving the vector where John would be represented inactive. If the vector for "run" is treated as an input and the tensor is treated as a weight matrix, then the network will activate John on the argument vector, effectively answering "John" to the

question "who is running?". In this sense, the tensor binds the argument John to the slot of the predicate run (see Halford et al., 1994; Smolensky, 1990).

However, tensor products do not adequately model role binding in human mental representation. Although a tensor can be used to *generate* one element of a binding given another element as a cue (as in the above example), the tensor itself does not explicitly represent those elements and their bindings. As noted previously, symbols in a symbol system are free to change bindings without changing their identities. That is, the identity of a symbol is invariant with whatever role bindings it happens to be participating in at any given time.

The tensor representation of a variable binding is not invariant in this way. Rather, the representation of a filler (or role) in a tensor changes as a function of the role (or filler) to which it happens to be bound. For example, consider the hypothetical tensor representations of *run (John*) in Figure 1a, and *walk (John)* in Figure 1b. The representation of *run (x)* is similar but not identical to the representation of *walk (x)*, so the tensor for *run (John)* is similar but not identical to the tensor for *walk (John)*. Predicates that do not overlap at all produce tensors that do not overlap. For example, the representation of *eat (John)* in Figure 1c does not overlap at all with the representation of *run (John)* in Figure 1a. The tensor thus captures the binding of John to these various roles, but it fails to capture the fact that John remains the same entity in each role. This point is somewhat subtle because *we* (the modeler or the reader of a modeling paper) know that John is the same in both cases; and looking at the graphical representation of the tensor, we can "see" John in both cases—the fact that John is the argument in both cases is the reason why the first, third and fifth units (but not the second, fourth and sixth) are active within active columns of the tensor.

But although *we* know John is "in there", the tensor itself does not. To demonstrate this limitation more formally, let us define the similarity of two vectors, **a** and **b**, in terms of the cosine of the angle between them:

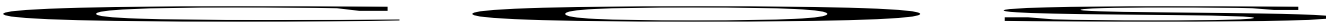


(2)

where ||**x**|| is the length of vector **x**, and **a**·**b** is the inner product (or "dot product"):

(3)

The cosine is a measure of the similarity between two vectors. It is at a maximum (1.0) when the vectors are identical (i.e., when they point in the same direction, regardless of their lengths), zero when the vectors are unrelated (i.e., orthogonal), and at a minimum (-1.0) when the vectors are opposites (i.e., with positive values in one corresponding to negative values in the other) (see Jordan, 1986). The cosine of the angle between any two tensors, **ab** and **a'b'** (i.e., their similarity), scales with the product of the similarities of the vectors from which they were created (Hummel & Holyoak, 1998). For a tensor created from two vectors, **a** and **b**:



. (4)

The cosine of the angle between two tensors goes to zero when either of the more basic similarities (cos(**a**,**a**') or cos(**b**,**b**')) is zero, and goes negative if either of the more basic similarities is negative. In a tensor representation, binding the same object to non-overlapping roles results in non-overlapping tensors.

It is tempting to reply that this is not a problem because the tensor really only needs to express binding information: The responsibility for expressing similarity lies, not with the tensor, but with the simple vectors from which the tensor is generated. According to this reply, the preceding analysis actually reveals a strength of tensor-based representations because it shows that the tensor can, in principle, express binding information unambiguously.

But this reply fails for two reasons. The first is that tensors describing bindings of similar roles to similar fillers will in fact be similar to one another (Equation 4). Thus, even if we wished to grant that it is not the tensor's responsibility to carry similarity information, the mathematics ensures that it is inevitably the tensor's burden. This problem is most extreme when the individual vectors (**a**, **a**', **b**, and **b**') are maximally dissimilar. If cos(**a**,**a**') = -1 (i.e., **a** and **a**' are opposites), and cos(**b**,**b**') = -1, then cos(**ab**,**a**'**b**') will be *positive* 1.0 (see Eq. 4): In this case, the dot products are maximimally *similar* precisely because their constituent roles and fillers are maximally *dissimilar*! A second and more serious problem is that even the simple vectors from which the tensor is constructed are not invariant across bindings, so it does not help to assign the "responsibility" for similarity to them. Consider a predicate, such as *loves (x y)*, that takes more than one argument (Figure 2). Bound to the agent role of such a relation, an object is represented in one vector space (i.e., collection of units); but bound to the patient role, the same object is represented in a completely different vector space. For example, as the agent of *loves (x y)*, John is represented on the "vertical" units (Figure 2a); but as the patient, John is represented on the "horizontal" units (Figure 2b). The representation of John in one role does not overlap at all with the representation of John in the other, even on the simple vectors.

The problems with tensor-based binding are compounded in schemes based on "compressed" tensors (e.g., Plate, 1991; Pollack, 1990). For example, in a holographic reduced representation (Plate, 1991), a tensor is compressed (by summing over reverse diagonals) into a vector whose dimensionality is given by the diagonal of the original tensor. Because the HRR is derived from a tensor, it inherits the binding-identity tradeoff of the tensor; and because the dimensionality of the HRR is lower than the dimensionality of the tensor, the HRR encounters the additional problem that it underconstrains the tensor. That is, for any given HRR, there are multiple tensors that could in principle have produced it. The recovery of the tensor—that is, the recovery of the binding—is ill-posed in an HRR. The same problems arise in other schemes for compressing tensors, including circular convolutions and Recursive Autoassociative Memories (RAAMs; Pollack, 1990).

Distributing a Symbol System Over Space and Time

The problems with the tensor approach to binding stem from the fact that tensors are a brand of conjunctive coding: Each unit in a tensor represents a role-filler *conjunction* (see Hummel & Biederman, 1992). As a result, the representation of a role or filler is fundamentally in conflict with the representation of role-filler bindings (Hummel & Holyoak, 1993): To the extent that the tensor preserves one, it must sacrifice the other (Equation 4).

To satisfy Requirement 1.2 (independent binding), a representational system needs a second degree of freedom—independent of the units' identities and their activations—to represent binding information: Units need a "tag" to express binding (i.e., such that units in the same group have the same value on their "tags"). The tag must be dynamic, so that units representing roles can be rapidly but temporarily bound to units representing the fillers of those roles. Recall that units in a connectionist network (like neurons) are not free to "move around", so list position (the binding tag used in propositional representations) is not available. But in principle, many possible tagging systems are conceivable. For example, units that are bound together could be spray-painted with a shared color; Mozer et al. (1992) describe a network that uses imaginary numbers as a binding tag. At present, however, the only proposed basis for tagging that has any apparent neural plausibility is based on the use of *time.* In particular, it has been proposed that units fire in synchrony with one another when they are bound together, and out of synchrony when they are not (Milner, 1974; von der Malsburg, 1981, 1985; see Gray, 1994, for a review). For example, to represent *loves (John Mary)*, units representing John would fire in synchrony with units for "lover", while units for Mary fire in synchrony with units for "beloved" (the John+lover set must fire out of synchrony with the Mary+beloved set); *loves (Mary John)* would be represented by the very same units, but the units for Mary would fire in synchrony with the units for lover while the units for John fire in synchrony with the units for beloved (Hummel & Holyoak, 1992).

There is some neurophysiological evidence for binding by synchrony in visual perception (e.g., in striate cortex; Eckhorn et al., 1988; Gray & Singer, 1989; König & Engel, 1995) and in higher-level processing dependent on frontal cortex (Desmed & Tomberg, 1994; Vaadia et al.,

1995). Numerous connectionist models use synchrony for binding. This mechanism has been applied in models of perceptual grouping (e.g., Eckhorn, Reitboeck, Arndt, & Dicke, 1990; von der Malsburg & Buhmann, 1992), object recognition (Hummel & Biederman, 1992; Hummel & Saiki, 1993; Hummel & Stankiewicz, 1996, in press), rule-based reasoning (Love, 1997; Shastri & Ajjanaggade, 1993), episodic storage in the hippocampal memory system (Shastri, 1997), and analogical reasoning (Hummel & Holyoak, 1992, 1996, 1997, in press).

Similarity in Dynamic Binding

Equation 4 characterizes how the similarity of different tensor products scales with the similarity of the simple vectors from which they are composed. It is possible to perform the same analysis on synchrony-based representations of binding, as illustrated in Figure 3. In synchrony-based models, predicate roles and fillers occupy different regions of the same vector space, and—more importantly—a given role or filler always occupies the *same* part of the space (i.e., activates the same units) regardless of whatever else is bound to it. (Geometrically, this is what it means for role and filler identity to be invariant with binding.) Binding by synchrony corresponds to activating two or more vectors at the same time (one for the role and one for the filler); mathematically, binding by synchrony is vector addition (see Figure 3). (By contrast, recall that tensor-based binding is vector multiplication; Equation 1.) As a consequence, the similarity of different bindings in a synchrony-based representation scales additively (rather than multiplicatively) with the similarity of the simple vectors (Hummel & Holyoak, 1998):

$$\cos(\mathbf{a}+\mathbf{b},\mathbf{a}'+\mathbf{b}') = (\mathbf{a}\cdot\mathbf{a}' + \mathbf{a}\cdot\mathbf{b}' + \mathbf{b}\cdot\mathbf{a}' + \mathbf{b}\cdot\mathbf{b}')/||\mathbf{a} + \mathbf{b}||\ ||\mathbf{a}' + \mathbf{b}'||, \quad (5)$$

where $\mathbf{a}+\mathbf{b}$ is the vector generated by synchronizing $\mathbf{a}$ with $\mathbf{b}$, and $\mathbf{a}'+\mathbf{b}'$ is the vector generated by synchronizing $\mathbf{a}'$ with $\mathbf{b}'$. If roles ($\mathbf{a}$ and $\mathbf{a}'$) and fillers ($\mathbf{b}$ and $\mathbf{b}'$) are assumed to occupy non-overlapping regions of vector space (i.e., assumed to share no units; see Hummel & Holyoak, 1997), then $\mathbf{a}\cdot\mathbf{b}'$ and $\mathbf{b}\cdot\mathbf{a}'$ go to zero, and Equation 5 simplifies to:

$$\cos(\mathbf{a}+\mathbf{b},\mathbf{a}'+\mathbf{b}') = (\mathbf{a}\cdot\mathbf{a}' + \mathbf{b}\cdot\mathbf{b}')/||\mathbf{a} + \mathbf{b}||\ ||\mathbf{a}' + \mathbf{b}'||. \quad (6)$$

Multiplication (as in the tensor scheme) corresponds to logical AND, whereas addition (as in the synchrony-based scheme) corresponds to logical OR. Tensor bindings are similar to the

extent that their roles *and* fillers are similar (Equation 4), whereas synchrony-based bindings are similar to the extent that their roles *or* fillers (or both) are similar (Equation 6). The practical consequence of this property is that, in a synchrony-based scheme, *walk (Bill)* is guaranteed to be identical to *eat (Bill)* on the units representing Bill, even if *walk (x)* and *eat (x)* have nothing whatsoever in common. Moreover, because the numerator of Eq. 6 is based on addition rather than multiplication, negative simple dot products (i.e., where $\mathbf{a}\cdot\mathbf{a}' < 0$ and $\mathbf{b}\cdot\mathbf{b}' < 0$) will produce a negative value for $\cos(\mathbf{a}+\mathbf{b},\mathbf{a}'+\mathbf{b}')$ (rather than a positive value, as in tensor-based schemes). That is, synchrony-based representations are similar precisely to the degree that they express similar concepts.

Figure 3 about here

Using Synchrony to Form Symbolic Representations

It is one thing to show that synchrony-based bindings preserve the similarity structure of the entities they bind; it is another to show that the resulting bindings constitute useful symbolic representations. To count as symbolic, a knowledge representation must function as part of a system that can perform symbolic computations. We have recently developed a model that uses synchrony-based bindings to form representations that are meaningfully symbolic in this sense. This model, LISA (*Learning and Inference with Schemas and Analogies*), is a model of the major stages of analogical inference and relational generalization, namely, retrieval from long-term memory, mapping of structures in working memory, analogical inference, and schema induction (Hummel & Holyoak, 1996, 1997, in press; for earlier versions of the model see Hummel, Burns & Holyoak, 1994; Hummel & Holyoak, 1992; Hummel, Melz, Thompson, & Holyoak, 1994). We will describe LISA in only very general terms here. The details of LISA's operation as an analogical retrieval and mapping engine can be found in Hummel and Holyoak (1997), and the details of its operation as an inference and schema induction engine can be found in Hummel and Holyoak (1996, 1998).

LISA represents role-filler bindings in working memory as synchronized patterns of activation distributed over a collection of *semantic* units. For example, "John loves Mary" would be represented by units for John firing in synchrony with units for the agent role of loves, while units for Mary fire in synchrony with units for the patent role. Propositions are represented in LISA's long-term memory by a hierarchy of *structure units* (see Figure 4). *Predicate units* (triangles in Figure 4) bind semantic features into predicate roles, *object units* (circles) bind semantic features into objects, *sub-proposition* (*SP*) *units* (rectangles) bind roles to their fillers, and *proposition* (*P*) *units* (ovals) bind role-filler conjunctions into complete propositions. Note that all the bindings in LISA's long-term memory are static in that they are coded conjunctively (as dictated by Requirement 3). As such, these units do not directly represent the semantic content of a proposition; rather, they serve only to store that content in long-term memory and respond to it when it enters working memory (i.e., as patterns of activation on the semantic units). An analog in LISA is represented as a collection of structure units coding the propositions in that analog. Separate analogs consist of non-overlapping sets of structure units, but share the semantic units. Note that structure units are created as needed, rather than pre-stored; as we will illustrate below, they can be learned by an algorithm for unsupervised learning.

Figure 4 about here

Based on these representations, LISA performs analog retrieval and analogical mapping as a form of guided pattern recognition. When a proposition becomes active in one analog (a *driver* analog), it generates synchronized patterns of activation on the semantic units (one pattern for each role-filler binding). In turn, these patterns activate structure units in other *recipient* analogs. This process is analog retrieval: Patterns of activation generated by the driver activate (i.e., retrieve from LTM) units in other analogs. Mapping differs from retrieval solely by the addition of modifiable *mapping connections* between units of the same type in the driver and recipient analogs. During mapping, weights on the mapping connections grow larger when the units they link are active simultaneously and more negative when one unit is active but the other is not. These connections

permit LISA to learn the correspondences generated during retrieval. They also serve to constrain subsequent memory access, and thus constrain subsequent mappings. By the end of a simulation run, corresponding structure units will have large positive weights on their mapping connections, and non-corresponding units will have strong negative weights. Using these operations, LISA simulates a large body of findings in human analog retrieval and mapping, and accounts for some complex asymmetries between retrieval and mapping (Hummel & Holyoak, 1997). These same operations also form the basis of LISA's capacity for schema induction, analogy-based inference (Hummel & Holyoak, 1996) and explicit rule-based inference. Let us consider analogy- and rule-based inference first.

Analogical Inference and Rule Use

Imagine that we give LISA an analog (henceforth *Analog1*) containing the following two propositions (Figure 5a):

P1 = input (X)

P2 = output (X),

where P1 and P2 are the names of the propositions, *input (x)* and *output (x)* are simple one-argument predicates (e.g., let *input* be connected to the semantic units *role* and *input*, and let *output* be connected to *role* and *output*), and *X* is a simple semantically-empty object (e.g., let *X* connect either to the semantic unit *variable*, or to no semantics at all; as we shall see, it does not matter which). Analog1 is a typical analog in LISA notation (Hummel & Holyoak, 1997), and it can also be interpreted as a rule stating "X is input" and "X is output". That is, Analog1 is LISA-ese for the identity function. Next let us give LISA Analog2 (Figure 5b):

P1 = input (1),

where the predicate unit *input* is connected to the very same semantics as *input* in Analog1 (but note that the predicate *input (x)* is represented by separate units in Analog1 and Analog2; Figures 5a and 5b), and the object unit, *1*, is connected to semantics indicating that it is a number, and that its value is one. Critically, *1* is connected to none of the same semantics as *X* in Analog1, so it bears no

similarity whatsoever to that object (i.e., the variable X in the rule). As we shall see, this property distinguishes LISA from all eliminative connectionist approaches to modeling the identity function.

Figure 5 about here

Now let us map Analog2 onto Analog1. P1 (in Analog2) is a single-place proposition, and therefore has only one SP (namely, *input+1*; Figure 5b). When this SP fires, it will activate the predicate unit *input* and the object unit *1*, which will activate (in synchrony) their respective semantic units (*input, role, number*, and *1*). Although the semantics *number* and *1* excite nothing in Analog1, *role* excites both *input* and *output* (in Analog1), and *input* (the semantic unit) excites *input* (in Analog1). Because the predicate *input* (in Analog1) is receiving more bottom-up excitation than the predicate *output, input* will "win" the inhibitory competition, becoming fully active and inhibiting *output* to inactivity. The predicate unit *input* (in Analog1) will in turn excite the SP *input+X*, which will excite the object *X*. As a consequence, *X* in Analog1 is now active at the same time as *1* in Analog2, so the mapping connection between them will grow: LISA has bound the value 1 to the variable X and stored this binding as a connection between them. Similarly, *input* (Analog1) will have learned an excitatory mapping connection to *input* (Analog2) (since they were active at the same time). But *output* (Analog1) was inactive while *input* (Analog2) was active, so *output* (Analog1) will have learned a *negative* (inhibitory) mapping connection to *input* (Analog1).

Next let us make Analog1 the driver and Analog2 the recipient. When P1 fires in Analog1, it will simply reinforce (i.e., strengthen) the mapping connections from *input* (in Analog1) to *input* (in Analog2) and from *X* to *1*. It will also strengthen—i.e., make more negative—the inhibitory connection between *output* [Analog1] and *input* [Analog2].

However, when P2 fires, something more interesting will happen. P2 will activate the SP *output+X*, activating *output* and *X*, which will activate (in synchrony) the semantics *role* and *output.* At the same time, *X* will activate *1* (in Analog2) directly by way of the mapping connection between them. Meanwhile, the semantic unit *role* will excite the predicate unit *input* (in Analog2), but *output* (in Analog1) will inhibit *input* (in Analog2) due to the negative mapping connection between them.

In fact, although the object *1* is receiving excitatory input over its mapping connection, all the predicates in Analog2 (all one of "them") are receiving inhibitory input over their mapping connections, a situation that indicates that no predicates in Analog2 correspond to the currently active predicate in Analog1. This situation serves as a cue that it is necessary to "invent" a new predicate corresponding to whatever predicate is currently active in Analog1 (a variety of "copy with substitution and generation"; Holyoak et al., 1994). LISA will invent a new predicate (call it **output*, where the "*" indicates that LISA invented it, and "output" indicates that it corresponds to *output* in Analog1), and connect it to whatever semantic units (corresponding to predicates) are currently active (in this case, *output* and *role*).

The predicate **output* is now coactive (in synchrony) with the object unit *1* in Analog2. But for the same reason *input* was inhibited by *output*, the SP *input+1* (in Analog2) will be inhibited by the SP *output+X* (Analog2), and *P1* will be inhibited by *P2*. As a consequence, LISA will invent the SP **output+1* and the P unit **P2* (in Analog2), connect them to one another, and connect **output+1* to the predicate **output*, and the object *1*. (LISA knows what to connect to what on the basis of the units' co-activity. It simply connects together all the active units.) Together, these operations—mapping from Analog2 to Analog1, mapping back from Analog1 to Analog2, and filling in the gaps in Analog2—cause LISA to infer the proposition P2 = output (1). That is, given the identity function (Analog1) and the question *input (1)*, LISA answers *output (1)* (i.e., "the output of the identity function run on the input 1 is 1").

We ran LISA on exactly this problem, and on several others like it (Hummel & Holyoak, 1998). In each case, it gave the right answer as output: *input (1) --> output (1), input (2) --> output (2), input (3) --> output (3).* We also ran it on the non-numerical problems *input (flower)* and *input (Mary).* Not surprisingly, it also gave the correct responses to these problems. It is important to emphasize that LISA was able to solve the identity problem in spite of the fact that there was no semantic overlap whatsoever between the object (X) in the rule (i.e., Analog1) and the objects in the problems on which it was tested (1, 2, 3, flower, and Mary). Like a human reasoner, but unlike any eliminative connectionist model (see Marcus, 1997, this volume), LISA generalized the identity

function universally. Its ability to do so stems directly from its ability to bind values (such as 1, flower, and Mary) to variables (such as X).

Note also that LISA would have been far less successful in solving this problem had it represented variable-value (or role-filler) bindings with tensors (or their variants) rather than synchrony. The ease with which LISA maps the identity function hinges on the fact that the predicate *input (x)* is represented in exactly the same way regardless of what its argument happens to be (Requirement 1.2 and Equation 6): It is the mapping of *input* (in Analog2) to *input* (in Analog1) that binds the value (1, 2, etc.) to the variable (X) and bootstraps the solution to the problem. Had LISA bound *input (x)* to its argument (the object X) using tensor products, then the resulting tensor would depend on both the predicate (*input (x)*) and its argument (X), so there would be no guarantee that the tensor representing *input (X)* (in Analog1) would overlap at all with the tensor representing *input* (*y*) (where *y* denotes any arbitrary object) in Analog2 (recall Equation 4). This is not to say that tensor-based models might not solve the identity function in a different way, for example, by treating the tensor as a weight matrix, as discussed previously. But mapping the identity function in this way is formally equivalent to the approach of the eliminative connectionist models (inputs are represented by one vector, outputs by another, and the function is mapped by the weights in between). As such, this version of the tensor-based approach would be subject to all the same limitations as traditional eliminative connectionist models (discussed below; see also Marcus, 1997).

Analogy-Based Rule Induction

It might be objected that our demonstration of identity mapping in LISA—and especially our comparison of LISA with traditional eliminative models—is misleading. After all, the eliminative models *learn* to solve the identity function by example; we simply *gave* LISA the rule. Granted the rule, it is no surprise that LISA solved the problem.

This objection fails for two reasons. The first is that the eliminative model could not solve the identity function even if we gave it the rule (or more accurately, *tried* to give it the rule). "Giving" an eliminative model the identity rule would be a matter of giving it *N* input units and *N*

output units, and connecting the *i*th input unit to the *i*th output unit (for all $i = 1...N$). In such an arrangement, the network will simply copy to the output units whatever it is given on the input units. Voilá—we have given the eliminative model the identity function. Or have we? Note that even this model will not generalize universally because there is a finite number of inputs it can even represent in the first place (given by the dimensionality, *N*, of the input and output vectors). That is, this model expects—indeed, demands—its inputs and outputs to be representable in a *particular* feature space, as given by *N* (see also Marcus, 1997, this volume). LISA, by contrast, does not care at all how its inputs are represented. (Recall that it maps the function even though there is no semantic overlap between X and any value bound to X.) LISA's solution to the identity function hinges instead on the predicate *input (x)*, so once Analog1 is established, LISA can then map the identity function on any argument bound to *input (x)*. Another way to put it is that we have indeed given LISA a *rule*, in the true sense of a function that binds values to variables. The trouble is not that we gave LISA the rule, but that there is *no way* to give the eliminative model such a rule, even if we wanted to.

The second answer to the above objection is that, although the previous demonstration assumes the pre-existence of the rule, LISA is quite capable of learning the rule by example. Moreover, as we shall show, LISA can learn to generalize universally from only one or two examples. Whereas the eliminative model requires a number of examples that scales with the number of problems it will eventually be asked to solve, LISA can solve *any* problem (i.e., an infinity of them) after just one or two examples. LISA induces the identity rule in the same way as it induces any schema—by unsupervised learning (of the kind that allows it to "invent" structure units, as in the previous example) plus *intersection discovery* (see Hummel & Holyoak, 1996, 1998).

Imagine that we give LISA the following example:

Analog1	Analog2
P1 = input (1)	P1 = input (2)
P2 = output (1)	P2 = output (2),

and have it map Analog1 onto Analog2 as in the previous example. The predicate unit *input* in Analog1 will map to *input* in Analog2, *output* will map to *output, 1* will map to *2*, and the corresponding SPs and P units will likewise map to one another. Since every structure unit in Analog1 has a corresponding unit in Analog2, this mapping will not require LISA to invent (i.e., learn or infer) any new structures in Analog2.

Now let us create a third analog, Analog3, which initially contains no structure units at all. Once P1 in Analog1 maps to P1 in Analog2, these units will excite one another directly via their mapping connection. But Analog3 contains no units, so there is no unit to develop positive mapping connections to P1 in Analog1 (or P1 in Analog2). That is, nothing in Analog3 maps to P1 in Analog1. Recall that this lack-of-mapping is LISA's cue to invent new structure units. In Analog3, LISA will invent the P unit **P1*, the SP **input+1*, the predicate **input*, and the object **1* (let us assume that Analog1 is the driver, and that the names of invented units are taken from the driver; hence, the new object is **1* rather than **2* or **number*). The principles underlying learning in Analog3 are so far just the same as those underlying the "copy with substitution and generation" (i.e., inference) in the previous example.

However, analogs that are learning to be schemas, such as Analog3, are subject to one additional constraint: The object and predicate units in these analogs have a connection-level threshold that prevents them from learning connections to any semantic units with activations below a certain value, Θ (Hummel & Holyoak, 1996). Otherwise, these units update their connections in the "usual" fashion (i.e., via a modified Hebbian rule; see Hummel & Holyoak, 1996). In addition, predicate and object units in the recipient (Analog2, if Analog1 is the driver) send activation back to the semantic units. As a result, semantic units that are connected to active units in both the driver and the recipient will tend to have about twice as much input as semantic units that are connected to one but not the other. For example, the semantic unit *number* is connected both to the object *1* (in

Analog1) and to the object *2* (in Analog2). When P1 in Analog1 maps to P1 in Analog2, *number* will therefore have two sources of input. By contrast, the semantic unit *1* is connected to the object *1* in Analog1, but is not connected to anything in Analog2; and the semantic unit *2* is connected to the object *2* in Analog2, but to nothing in Analog1. The semantic unit *number* thus receives about twice as much excitatory input as either *1* or *2*, and therefore becomes more active.

In combination with the threshold, Θ, on the predicate and object units in the schema (Analog3), this feedback from recipient analogs to semantics causes Analog 3 to perform a kind of intersection discovery. Units in Analog3 only learn connections to highly active semantic units—that is, semantics that are common to the driver and recipient. In the case of Analog3, this means that the object unit **1* in Analog3 will only learn a connection to the semantic unit *number*. **1* represents numbers *generally*, not just the numbers 1 and 2, from which it was induced by example. The same process operates on the predicate unit **input* in Analog3. But in this case, *input* in Analog1 has all the same semantics as *input* in Analog2, so all their semantic units receive two sources of input. Therefore, **input* in Analog3 learns connections to all those semantic units, and ends up connected to exactly the same units as both examples from which it was induced. Once these operations have run on both P1 and P2 (in Analogs 1 and 2), Analog3 is the equivalent of:

P1 = input (number)

P2 = output (number).

Based on Analog3, LISA can now map the identity function for any number. And in fact, Analog3 is prepared to generalize much more universally than that. Let Analog4 be:

P1 input (flower),

where *flower* is assumed to have no semantic overlap whatsoever with *number* in Analog3. Our first example showed that LISA can map the identify function even when the object (1, 2, flower, etc.) has no semantic overlap with the variable in the function (X in the previous example). Since *flower* has no semantic overlap with *number*, mapping Analog4 onto Analog3 is just a repeat of that first example: LISA will infer *P2 = *output (flower) in Analog4. (Hummel & Holyoak, 1998, ran these simulations and this is exactly what it does.) Hence, after just one training example (mapping

Analog1 onto Analog2 and inducing Analog3), LISA can generalize universally. If, while Analog3 is being mapped onto Analog4, a new empty analog, Analog5, is allowed to learn a schema (rule) from their intersection, then Analog 5 will end up being the equivalent of:

P1 = input (X)

P2 = output (X),

where X is semantically empty (it will connect to the intersection of *flower* and *number*, which is the empty set). LISA has now induced the rule we gave it in the very first example.

Not only is LISA's rule-learning blindingly fast compared with back-propagation learning (as used in many eliminative connectionist models), the results are also much more general. After just one example, LISA knows how to "play the identity function game," and can play it with any new input. The learning trials required by people—and their subsequent ability to generalize universally—are much more on the scale of LISA than of an eliminative connectionist model, or a model based on tensor binding. The difference between LISA and both these alternative approaches is that LISA can bind values to variables and arguments to roles while preserving the similarity relations among the constituent concepts. As a result, LISA is a connectionist implementation of a symbol system that can map and learn symbolic functions (such as the identity function). Tensor product models attempt to bind values to variables, but fall short of the mark; as a result, their ability to generalize also falls short of the mark. Eliminative connectionist models do not even attempt to bind values to variables, and as a result, their performance falls far from the mark. After hundreds or thousands (or even millions) of iterations through its training set, a back-propagation model is still just as guaranteed to fail to generalize universally as it was before training started. Universal generalization is *in principle* out of reach for any model that cannot bind values to variables, so it would not matter how long one trained the back-propagation model—it would never truly learn the identity function (Marcus, 1997).

A more important criticism of LISA's ability to learn the identity function is that we gave it the predicates input (X) and output (X) in the examples from which the rule was induced. The question of how a human reasoner discovers these predicates in the first place is an important one

for which we cannot yet offer a complete answer (but see Hummel & Holyoak, in press, for progress in this direction). However, it is safe to assume that at least for trivial problems such as the identity function, adult reasoners come armed with predicates corresponding to input (X) (e.g., "This is the example on which I am being tested") and output (X) ("This is the response I am supposed to give"). Even if we assume the existence of these predicates, it is not a trivial matter to specify how the reasoner can use them to solve the problem. Eliminative connectionists would also be willing to postulate concepts such as "input" and "output", but their models are nonetheless incapable of using those concepts to perform useful work. It is this capacity that requires symbol processing, and that the preceeding simulations are intended to demonstrate.

Why Symbolic Connectionism is Not "Mere" Implementation

Fodor and Pylyshyn (1988) observed that a connectionist model might, in principle, capture the systematicity (compositionality) of human cognition, but that in so doing, the resulting model would simply implement a (traditional-style) symbolic model. The "invited inference" was that nothing is to be gained from the exercise of implementing symbol processing in a connectionist framework. Is symbolic connectionism just a roundabout way of getting "back where we started"?

The answer is a resounding "no". The issue of whether the mind is a physical symbol system is a question at the level of computational theory (Marr, 1982): What function is the mind computing? In the most abstract terms, the answer is that the mind is performing symbol manipulation. This question and its answer are very important, as the failings of eliminative connectionist models attest. But the answer does not tell us *how* the mind is doing symbol manipulation, which is a question at Marr's level of representation and algorithm. It is here that symbolic connectionism represents a striking advance over traditional symbolic architectures of cognition (e.g., Anderson, 1993; Rosenbloom et al., 1991).

One advantage of symbolic connectionism derives from an apparent weakness: It is *hard* to do symbol manipulation in a connectionist architecture. This is because symbol manipulation requires dynamic binding, and dynamic binding is difficult to perform in a connectionist architecture (see Hummel & Stankiewicz, 1996, in press). In the case of dynamic binding by

synchrony of firing, some mechanism has to get the right units into synchrony with one another and (what is even more difficult) keep them *out* of synchrony with all the other units. It takes *work* to establish synchrony and (especially) asynchrony, and some process must perform this work. By contrast, dynamic binding in a symbolic model is trivially easy: The correct bindings are simply *given.* By definition, placing the symbol "John" into the first slot of the predicate *loves (x y)* binds John to the agent role of that predicate. End of story. There is nothing else to say, and no other work to do. If you then want to bind "John" to some other role, you can just do it, as many times as you want, with as many predicates as you want, and as many other objects as you want.

In a traditional symbol architecture, bindings are free, so you can have as many as you need. Of course, a theorist may opt to impose some limit on binding, in deference to the glaring fact that people have limited capacity to make and break role bindings; but this will simply be an ad hoc "add on" rather than a deep implication of the proposed symbolic architecture. It is here that the computational weakness of symbolic connectionism becomes a psychological virtue. A model that represents bindings with synchrony (such as LISA and related models, such as JIM; Hummel & Biederman, 1992; Hummel & Stankiewicz, 1996), is *inherently* limited in the number of things it may simultaneously have active and mutually *out* of synchrony with one another (although there is no theoretical limit on the number of entities in any one synchronized group). That is, there is a limit on the number of distinct bindings such a model may have in working memory at any one time (Hummel & Holyoak, 1997; Shastri & Ajjanaggade, 1993). Humans, too, have limited working memory, and limited attention. Thus Hummel and Stankiewicz (1996, in press) argue that a primary function of visual attention is to keep the separate elements of a visual display out of synchrony with one another. Symbolic connectionism—as an *algorithmic* theory of symbol systems—provides a natural account of the fact that humans have a limited working memory capacity. Similar symbolic-connectionist considerations predict various other limitations of human symbolic reasoning as well (see Hummel & Holyoak, 1997). One thing to be gained by asking *how* the human cognitive architecture implements symbols (rather than simply assuming *that* it

does, as in the traditional symbolic approach) is an understanding of some of the limitations of that architecture.

Symbolic connectionism also explains some strengths of the human cognitivearchitecture that are equally mysterious from the traditional symbolic perspective. One is the capacity to map semantically related predicates that take different numbers of arguments, for example, mapping *taller (A B)* and *taller (B C)* onto *tallest-to-shortest (D E F).* LISA can solve this mapping (Hummel & Holyoak, 1997). Traditional symbolic models, by contrast, must enforce an inviolable "*N*-ary restriction" (whereby a predicate with *N* arguments may only map to another predicate with *N* arguments), which precludes such mappings (see Hummel & Holyoak, 1997). Other strengths of symbolic connectionism derive from the value of distributed representations of semantic content (see Hummel & Holyoak, 1997). The early connectionists were right about the value of distributed representations, and symbolic connectionism is just as able to exploit those strengths as "traditional" (eliminative) connectionism. In fact, it is *better* able to do so, because symbolic connectionism embeds these representations into systematic structures. Armed with dynamic binding, LISA can implement fast inductive learning of universal generalizations using a simple variant of the Hebbian algorithm for unsupervised learning. As an aside, it is interesting to note that LISA's learning by analogy is a variety of learning by example—a property that it shares with back-propagation. It is thus more constrained than traditional algorithms for unsupervised learning (e.g., Kohonen, 1982; Marshall, 1995; von der Malsburg, 1973). But at the same time it is less "heavy-handed"—and much more psychologically plausible—than the explicit error-correction algorithm of back-propagation. In LISA, the "teacher" is just a familiar example (i.e., a source analog), not an all-knowing external device.

A further advantage of symbolic connectionism over either traditional symbolic modeling or eliminative connectionist modeling is that it provides a vocabulary for talking about the relationship between truly associative, non-symbolic processes and more complex symbolic processes. In symbolic connectionism, these are all part of the same system: Take symbolic connectionism, strip away dynamic variable binding, and the result is simple (connectionist-style) associationism.

Finally, symbolic connectionism maintains the basic architecture of earlier connectionist models (densely connected networks of local computing elements) while adding a more fine-grained use of the informational capacity of time. As compared to the elements of traditional symbolic models (lists of localist symbols, which can be constructed and modified by explicit list operations), the elements of symbolic connectionism provide more direct links to neural architecture, and hence set the stage for addressing questions at Marr's (1982) implementation level. A *physical* symbol system, as embodied in a human or other biological organism, is realized in the brain. There is a neural code for thought, and symbolic connectionism—the proper treatment of symbols—may guide us in cracking the code.

Footnotes

1. At the same time, some apparent similarities between connectionist networks and neural networks—that nodes operate like neurons and connections operate like synapses—must be interpreted with caution. Neural processes are complex and not yet understood. The similarity of current connectionist models to actual neural networks lies more in their gross architectures than in the operation of their basic elements.

2. Arguments (or roles) may suggest different shades of meaning as a function of the roles (or fillers) to which they are bound. For example, "loves" suggests a slightly different interpretation in *loves (John Mary)* than it does in *loves (John chocolate).* However, such contextual variation does not imply in any general sense that the filler (or role) itself necessarily changes its identity as a function of the binding. For example, our ability to appreciate that the "John" in *loves (John Mary)* is the same person as the "John" in *bite (Rover John)* demands explanation in terms of John's invariance across the different bindings. If we assume invariance of identity with binding as the general phenomenon, then it is possible to explain contextual shadings in meaning when they occur (Hummel & Holyoak, 1997). However, if we assume lack of invariance of identity as the general rule, then it becomes impossible to explain how knowledge acquired about an individual in one context can be connected to knowledge about the same individual in other contexts.

3. Although eliminative models often are based on back-propagation learning, their most basic limitations arise not from the learning algorithm *per se*, but rather from their lack of explicit role-filler representations. As we will discuss below, models of this sort are unable to represent the knowledge necessary for true universal generalization, and hence cannot succeed in modeling human relational generalization even if the modeler is allowed to hand-code the network.

References

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ: Erlbaum.

Barnden, J. A. (1994). On the connectionist implementation of analogy and working memory matching. In J. A. Barnden & K. J. Holyoak (Eds.*), Advances in connectionist and neural computation theory, Vol. 3: Analogy, metaphor, and reminding* (pp. 327-374). Norwood, NJ: Ablex.

Churchland, P. (1990). Cognitive activity in neural networks. In D. H. Osherson & E. E. Smith (Eds.), *An Invitation to Cognitive Science: Thinking (Volume 3).* Cambridge MA: MIT Press.

Churchland, P. (1995). *The Engine of Reason, the Seat of the Soul: A Philosophical Journey into the Brain.* Cambridge MA: MIT Press.

D'Amato, M. R., Salmon, D. P., Loukas, E., & Tomie, A. (1985). Symmetry and transitivity of conditional relations in monkeys (*Cebus apella*) and pigeons (*Columba livia*). *Journal of the Experimental Analysis of Behavior, 44,* 365-373.

Desmedt, J., & Tomberg, C. (1994). Transient phase-locking of 40 Hz electrical oscillations in prefrontal and parietal human cortex reflects the process of conscious somatic perception. *Neuroscience Letters, 168,* 126-129.

Eckhorn, R., Bauer, R., Jordan, W., Brish, M., Kruse, W. Munk, M. & Reitboeck, H.J. (1988). Coherent oscillations: A mechanism of feature linking in the visual cortex? Multiple electrode and correlation analysis in the cat. *Biological Cybernetics, 60,* 121-130.

Eckhorn, R., Reitboeck, H., Arndt, M., & Dicke, P. (1990). Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. *Neural Computation, 2,* 293-307.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14,* 179-212.

Elman, J. L., Bates, E. A., Johnson, M. K., Karmiloff-Smith, A., Parisi, D., & Plunkett, K. (1996). *Rethinking innateness: A connectionist perspective on development*. Cambridge, MA: MIT Press.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41,* 1-63.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science, 6,* 205-254.

Fodor, J. A., & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition, 28,* 3-71.

Gick, M. L., & Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology, 15,* 1-38.

Gray, C. M. (1994). Synchronous oscillations in neuronal systems: Mechanisms and functions. *Journal of Computational Neuroscience, 1,* 11-38.

Gray, C. M., & Singer, W. (1989). Stimulus specific neuronal oscillations in orientation columns of cat visual cortex. *Proceedings of the National Academy of Sciences, USA, 86,* 1698-1702.

Halford, G. S., Wilson, W. H., Guo, J., Gayler, R. W., Wiles, J., & Stewart, J. E. M. (1994). Connectionist implications for processing capacity limitations in analogies. In K. J. Holyoak & J. A. Barnden (Eds.), *Advances in connectionist and neural computation theory, Vol. 2: Analogical connections* (pp. 363-415). Norwood, NJ: Ablex.

Halford, G. S., Wilson, W. H., & Phillips, S. (in press). Processing capacity defined by relational complexity: Implications for comparative, developmental, and cognitive psychology. *Brain and Behavioral Sciences.*

Hinton, G. E. (Ed.) (1990). *Connectionist symbol processing*. Cambridge, MA: MIT Press.

Holyoak, K. J. (1991). Symbolic connectionism: Toward third-generation theories of expertise. In K. A. Ericsson & J. Smith (Eds.), *Toward a general theory of expertise: Prospects and limits* (pp. 301-335). Cambridge: Cambridge University Press.

Holyoak, K. J., Novick, L. R., & Melz, E. R. (1994). Component processes in analogical transfer: Mapping, pattern completion, and adaptation. In K. J. Holyoak & J. A. Barnden (Eds.), *Advances in connectionist and neural computation theory, Vol. 2: Analogical connections* (pp. 130-180). Norwood, NJ: Ablex.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13,* 295-355.

Holyoak, K. J., & Thagard, P. (1995). *Mental leaps: Analogy in creative thought.* Cambridge, MA: MIT Press.

Hummel, J. E. (this volume). View-based theories of human object recognition: Let's get serious.

Hummel, J. E., & Biederman, I. (1992). Dynamic binding in a neural network for shape recognition. *Psychological Review, 99,* 480-517.

Hummel, J. E., Burns, B., & Holyoak, K. J. (1994). Analogical mapping by dynamic binding: Preliminary investigations. In K. J. Holyoak & J. A. Barnden (Eds.), *Advances in connectionist and neural computation theory, Vol. 2: Analogical connections* (pp. 416-445). Norwood, NJ: Ablex.

Hummel, J. E., & Holyoak, K. J. (1992). Indirect analogical mapping. *In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (pp. 516-521). Hillsdale, NJ: Erlbaum.

Hummel, J. E., & Holyoak, K. J. (1993). Distributing structure over time. *Behavioral and Brain Sciences, 16,* 464.

Hummel, J. E., & Holyoak, K. J. (1996). LISA: A computational model of analogical inference and schema induction. In G. W. Cottrell (Ed.), *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society* (pp. 352-357). Hillsdale, NJ: Erlbaum.

Hummel, J. E., & Holyoak, K. J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review, 104,* 427-466.

Hummel, J. E., & Holyoak, K. J. (1998). Symbolic connectionism. Manuscript in preparation, Department of Psychology, UCLA.

Hummel, J. E., & Holyoak, K. J. (in press). From analogy to schema induction in a structure-sensitive connectionist model. In T. Dartnall & D. Peterson (Eds.), *Creativity and computation.* Cambridge, MA: MIT Press.

Hummel, J. E., Melz, E. R., Thompson, J., & Holyoak, K. J. (1994). Mapping hierarchical structures with synchrony for binding: Preliminary investigations. In A. Ram & K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 433-438). Hillsdale, NJ: Erlbaum.

Hummel, J. E., & Saiki, J. (1993). Rapid unsupervised learning of object structural descriptions. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society* (pp. 569-574). Hillsdale, NJ: Erlbaum.

Hummel, J. E., & Stankiewicz, B. J. (1996). An architecture for rapid, hierarchical structural description. In T. Inui & J. McClelland (Eds.), *Attention and performance XVI: Information integration in perception and communication* (pp. 93-121). Cambridge, MA: MIT Press.

Hummel, J. E., & Stankiewicz, B. J. (in press). Two roles for attention in shape perception: A structural description model of visual scrutiny. *Visual Cognition*.

Jordan, M. I. (1986). An introduction to linear algebra in parallel distributed processing. In D. E. Rumelhart, J. L. McClelland & the PDP Research Group, *Parallel distribute processing: Explorations in the microstructures of cognition, Vol. 1* (pp. 365-422). Cambridge, MA: MIT Press.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, 43, 59-69.

König, P., & Engel, A. K. (1995). Correlated firing in sensory-motor systems. *Current Opinion in Neurobiology, 5*, 511-519.

Lesgold, A. M. (1972). Pronominalization: A device for unifying sentences in memory. *Journal of Verbal Learning and Verbal Behavior, 11*, 316-323.

Love, B.C. (1997). Asynchronous connectionist binding. Manuscript in preparation, Northwestern University.

Marcus, G. F. (1997). Rethinking eliminative connectionism. Manuscript in preparation, New York University.

Marcus, G. F. (this volume).

Marr, D. (1982). *Vision.* Freeman: San Francisco.

Marshall, J. A. (1995). Adaptive pattern recognition by self-organizing neural networks: Context, uncertainty, multiplicity, and scale. *Neural Networks, 8,* (3), 335-362.

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

Mozer, M. C., Zemel, R. S., Behrmann, M., & Williams, C. K. (1992). Learning to segment images using dynamic feature binding. *Neural Computation, 4,* 650-665.

Newell, A. (1980). Physical symbol systems. *Cognitive Science, 4,* 135-183.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM, 19,* 113-126.

Oden, D. L., Thompson, R. K. R., & Premack, D. (1988). Spontaneous transfer of matching by infant chimpanzees (*Pan troglodytes*). *Animal Behavior Processes, 14,* 140-145.

Phillips, S., & Halford, G. S. (1997). Systematicity: Psychological evidence with connectionist implications. In M. G. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Conference of the Cognitive Science Society* (pp.614-619). Hillsdale, NJ: Erlbaum.

Pinker, S., & Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model. *Cognition, 28,* 73-193.

Plate, T. (1991). Holographic reduced representations: Convolution algebra for compositional distributed representations. In J. Mylopoulos and R. Reiter (Eds.), *Proceedings of the 12th International Joint Conference on Artificial Intelligence* (pp. 30-35). San Mateo: Morgan Kaufmann.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence, 46,* 77-106.

Robin, N., & Holyoak, K. J. (1994). Relational complexity and the functions of prefrontal cortex. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 987-997). Cambridge, MA: MIT Press.

Rosenbloom, P. S., Laird, J. E., Newell, A., & McCarl, R. (1991). A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artificial Intelligence, 47,* 289-325.

Ross, B. H., & Kennedy, P. T. (1990). Generalizing from the use of earlier examples in problem solving. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 16,* 42-55.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. *Parallel distributed processing, Vol. 1: Foundations*. Canbridge, MA: MIT Press.

Seidenberg, M. S. (1994). Language and connectionism: The developing interface. *Cognition, 50,* 385-401.

Seidenberg, M. S. (1997). Language acquisition and use: Learning and applying probabilistic constraints. *Science, 275,* 1599-1603.

Shastri, L. (1997). A model of rapid memory formation in the hippocampal system. In M. G. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Conference of the Cognitive Science Society* (pp. 680-685). Hillsdale, NJ: Erlbaum.

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

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence, 46,* 159-216.

St. John, M. F. (1992). The Story Gestalt: A model of knowledge-intensive processes in text comprehension. *Cognitive Science, 16,* 271-302.

St. John, M. F., & McClelland, J. L. (1990). Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence, 46,* 217-257.

Touretzky, D., & Hinton, G. (1988). A distributed production system. *Cognitive Science, 12,* 423-466.

Touretzky, D., & Pomerleau, D. A. (1994). Reconstructing physical symbol systems. *Cognitive Science, 18,* 345-353.

Vaadia, E., Haalman, I., Abeles, M., Bergman, H., Prut, Y., Slovin, H., & Aertsen, A. (1995). Dynamics of neuronal interactions in monkey cortex in relation to behavioural events. *Nature, 373,* 515-518.

Vera, A. H., & Simon, H. A. (1993). Situated action: A symbolic interpretation. *CognitiveScience, 17*, 7-48.

Vera, A. H., & Simon, H. A. (1994). Reply to Touretzky and Pomerleau: Reconstructing physical symbol systems. *Cognitive Science, 18*, 355-360.

von der Malsburg, C. (1973) Self-organization of orientation selective cells in the striate cortex. *Kybernetik, 14*: 85-100.

von der Malsburg, C. (1981). The correlation theory of brain function. Internal Report 81-2, Department of Neurobiology, Max-Planck-Institute for Biophysical Chemistry.

von der Malsburg, C. (1985). Nervous structures with dynamical links. *Ber. Bunsenges. Phys. Chem., 89*, 703-710.

von der Malsburg, C., & Buhmann, J. (1992). Sensory segmentation with coupled neural oscillators. *Biological Cybernetics, 67*, 233-242.

Wanner, H. E. (1968). *On remembering, forgetting, and understanding sentences: A study of the deep structure hypothesis.* Ph.D. dissertation, Harvard University.

Figure Captions

Figure 1. A tensor product is an outer product of two or more vectors. (a) A tensor product representing a binding of the object *John* to the single-argument predicate *run (x)*. Black circles indicate values of 1 (active units) and circles indicate values of 0 (inactive units). *Run (x)* is represented by the vector [1,1,0,0,0,0]. *John* is represented by the vector [0,1,0,1,0,1]. The *ij*th element of the tensor *run (John)* is the product of the *i*th element of *run (x)* with the *j*th element of *John*. (b) A tensor product representing a binding of *John* to the predicate *walk (x)*. The vector for *walk (x)* shares active units with (but is not identical to) the vector for *run (x)*, so the tensor for *walk (John)* shares active units with (but is not identical to) the vector for *run (John)*. (c) A tensor product representing a binding of *John* to the predicate *eat (x)*. The vector for *eat (x)* shares no active units with the vector for *run (x)*, so the tensor for *eat (John)* shares no active units with the vector for *run (John)*.

Figure 2. (a) A tensor representation of the proposition *love (John Mary)*. Black circles indicate values of 1 (active units) and while circles indicate values of 0 (inactive units). (b) A tensor representation of *love (Mary John)*. Note that *John*, which is bound to different roles in the two propositions, is represented by different vectors in the two propositions: *John* is represented by the "vertical" (agent) vector in the first proposition, and by the "horizontal" (patient) vector in the second.

Figure 3. (a) Hypothetical vector representations of *John*, *Mary*, the agent and patient roles of *love (x y)* (*lover* [**L1**] and *beloved* [**L2**], respectively), and the agent and patient roles of *fear (x y)* (*fearer* [**F1**] and *feared* [**F2**], respectively). Black circles indicate values of 1 (active units) and circles indicate values of 0 (inactive units). (b) Matrix of dot-products (similarities) of the vectors in (a). For example, the entry in row **J**, column **M** is the dot product of the vector for *John* with the vector for *Mary*. Empty cells indicate values of zero. (c) Vectors formed by synchronizing (i.e., adding) each object vector in (a) with each role vector. For example, vector **J+L1** is the vector produced by synchroniziong (adding) the vector for *John* with the vector for *lover*. Pairs of vectors represent propositions. For example, **J+L1** and **M+L2** represent the role-filler bindings in *love (John Mary)*

and jointly represent that proposition. (d) Matrix of dot-products (similarities) of the (synchronized) vectors in (c). Note that the dot product for any pair of synchronized vectors (from c) is the sum of the dot products of the corresponding simple vectors (from a). For example, the dot product of **J+L1** (*John+lover*) with **M+F1** (*Mary+fearer*) (4) equals the dot product of *John* with *Mary* (2) plus the dot product of *lover* with *fearer* (2). The similarity of synchronized vectors scales with the sum of the similarities of the simple vectors from which they are composed.

Figure 4. Illustration of the representation of the proposition *love (John Mary)* in LISA's long-term memory. See text for details.

Figure 5. (a) LISA representation of the rule, "X is input (proposition P1) and X is output (proposition P2)". (b) LISA representation of the question, "1 is input. What is output?" LISA "answers" the question in (b) by mapping the analog in (b) onto the analog in (a), (which binds the value 1 to the variable X) and then mapping back, creating the proposition P2 = "1 is output" in the analog in (b). See text for details.

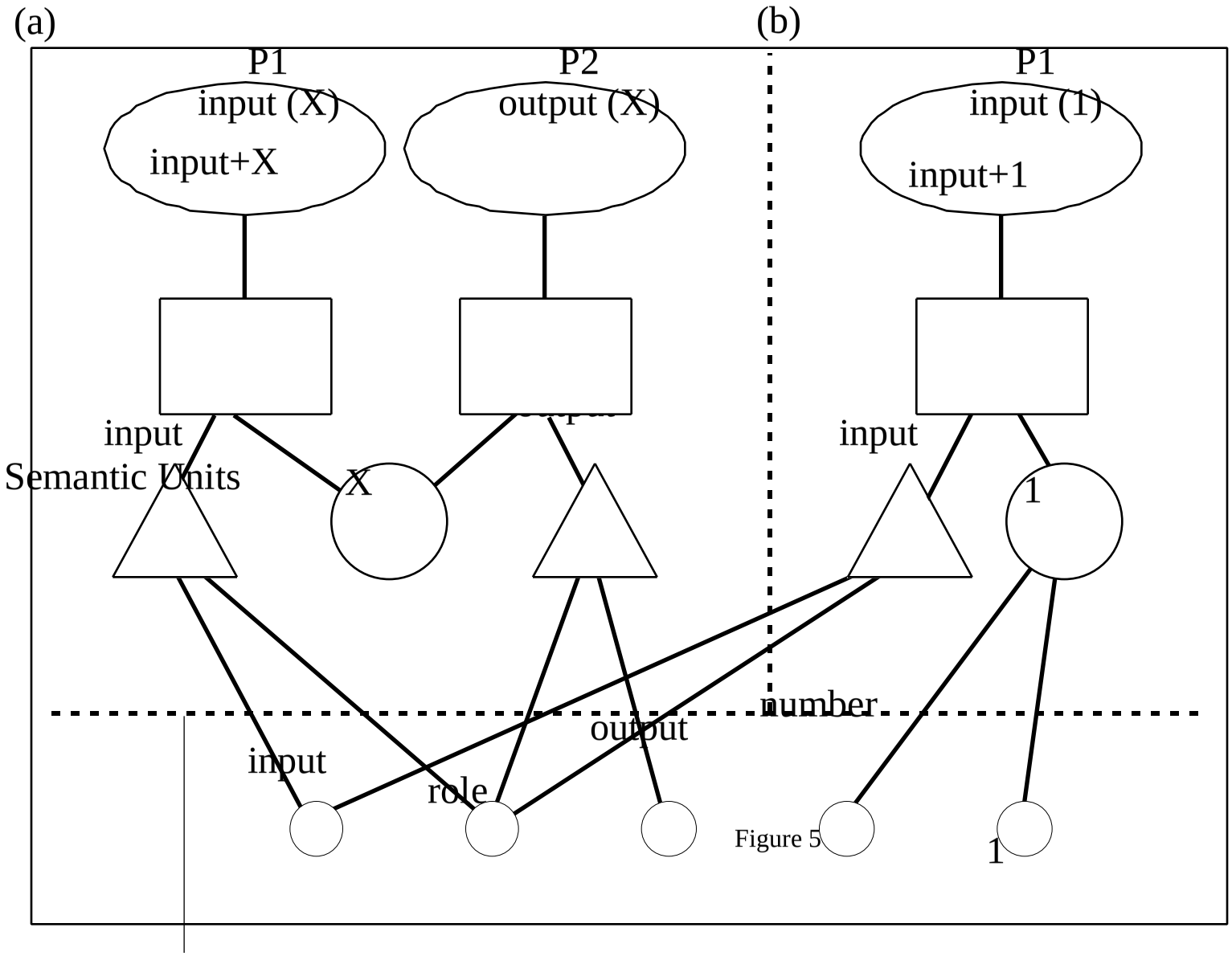

(a)
(b)
P1
input (X)
input+X
P2
output (X)
P1
input (1)
input+1
input
Semantic Units
X
input
1
number
input
role
output
1

Figure 5

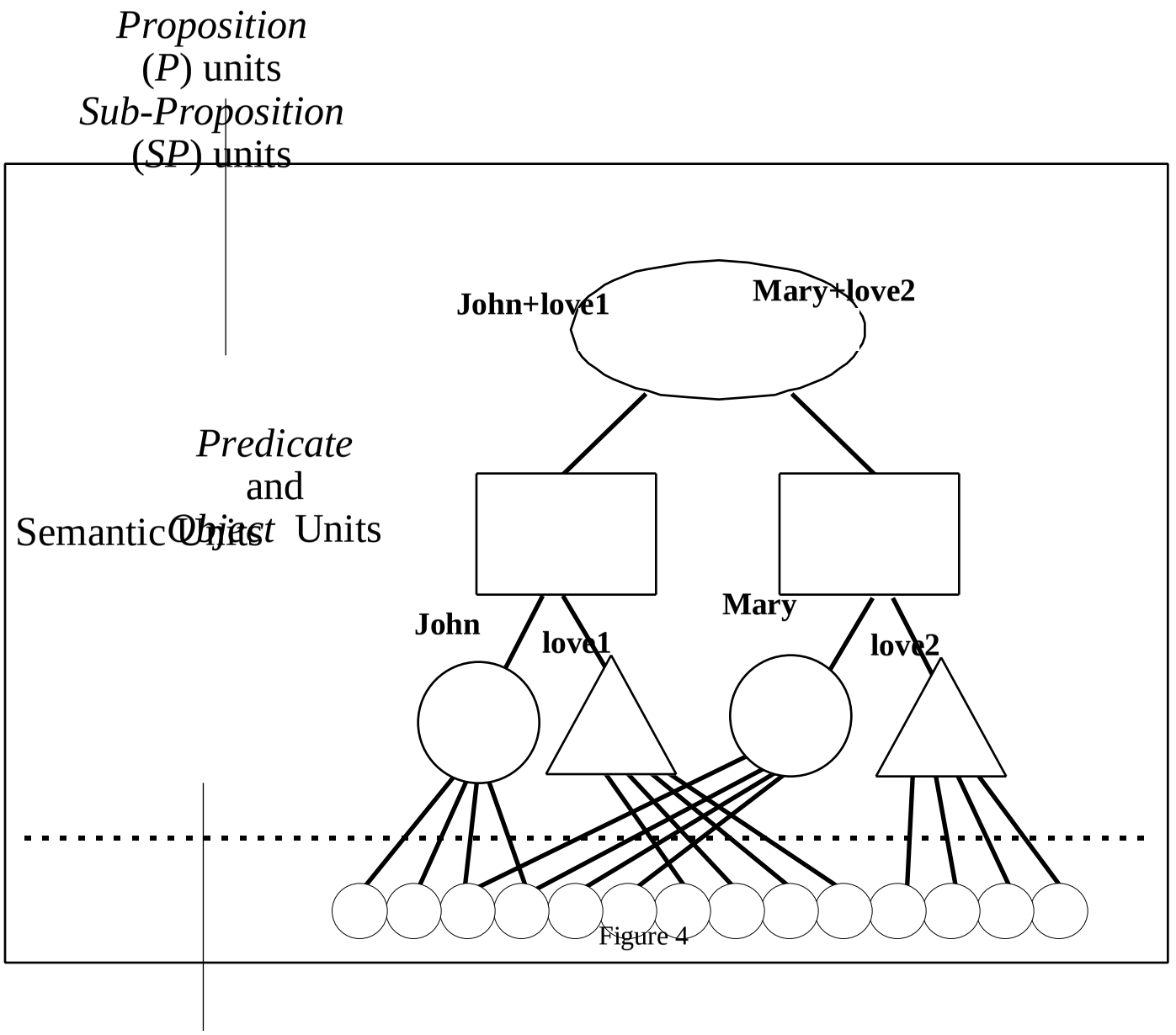

Proposition
(*P*) units
Sub-Proposition
(*SP*) units
Predicate
and
Object Units
Semantic Units
John+love1
Mary+love2
John
love1
Mary
love2

Figure 4

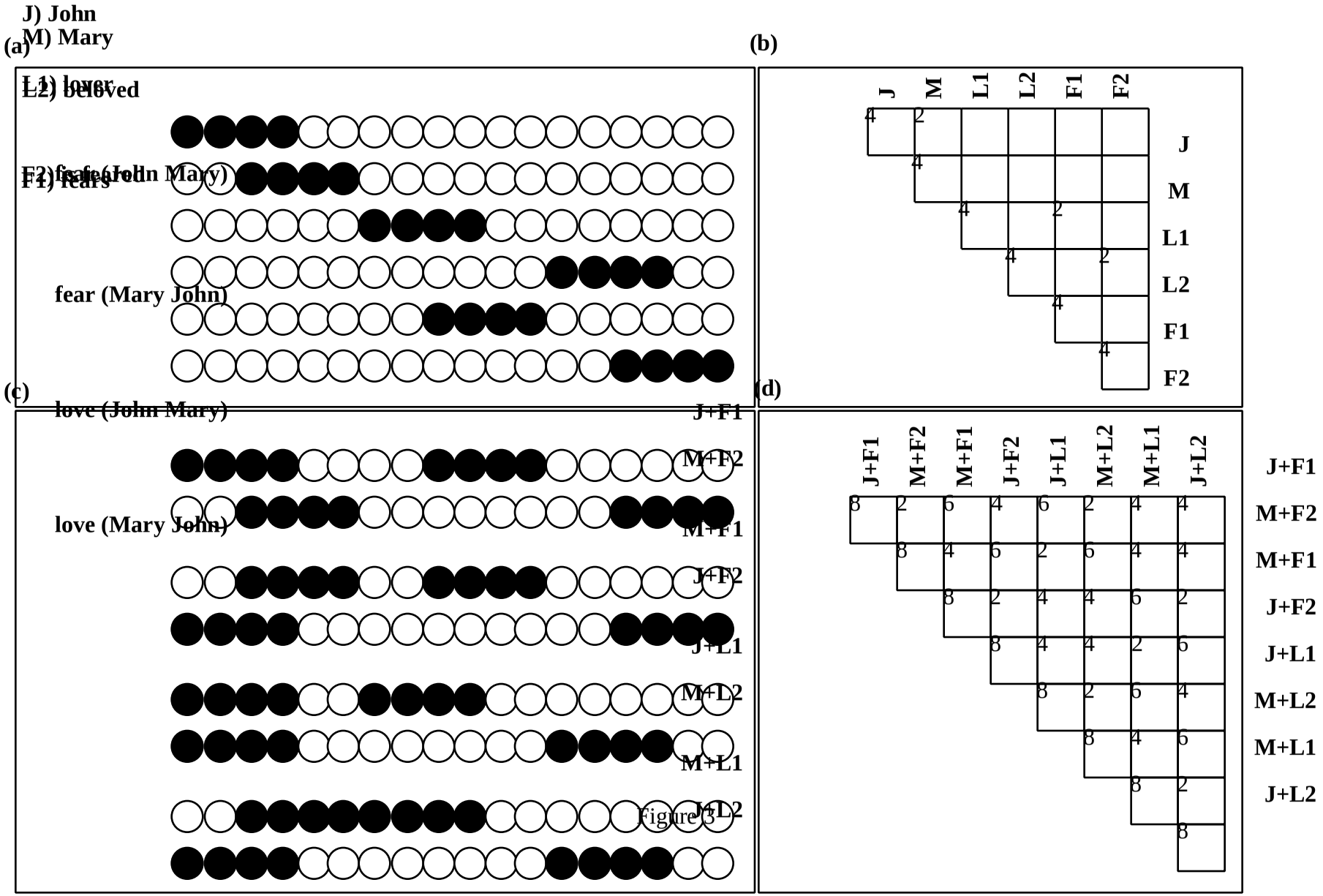

J) John
M) Mary
(a)
L1) lover
L2) beloved
F2) fear (John Mary)
F1) fears
fear (Mary John)
(b)
J
M
L1
L2
F1
F2
(c)
love (John Mary)
love (Mary John)
J+F1
M+F2
M+F1
J+F2
J+L1
M+L2
M+L1
J+L2
(d)
J+F1
M+F2
M+F1
J+F2
J+L1
M+L2
M+L1
J+L2

Figure 3

Predicate
love

Predicate
love

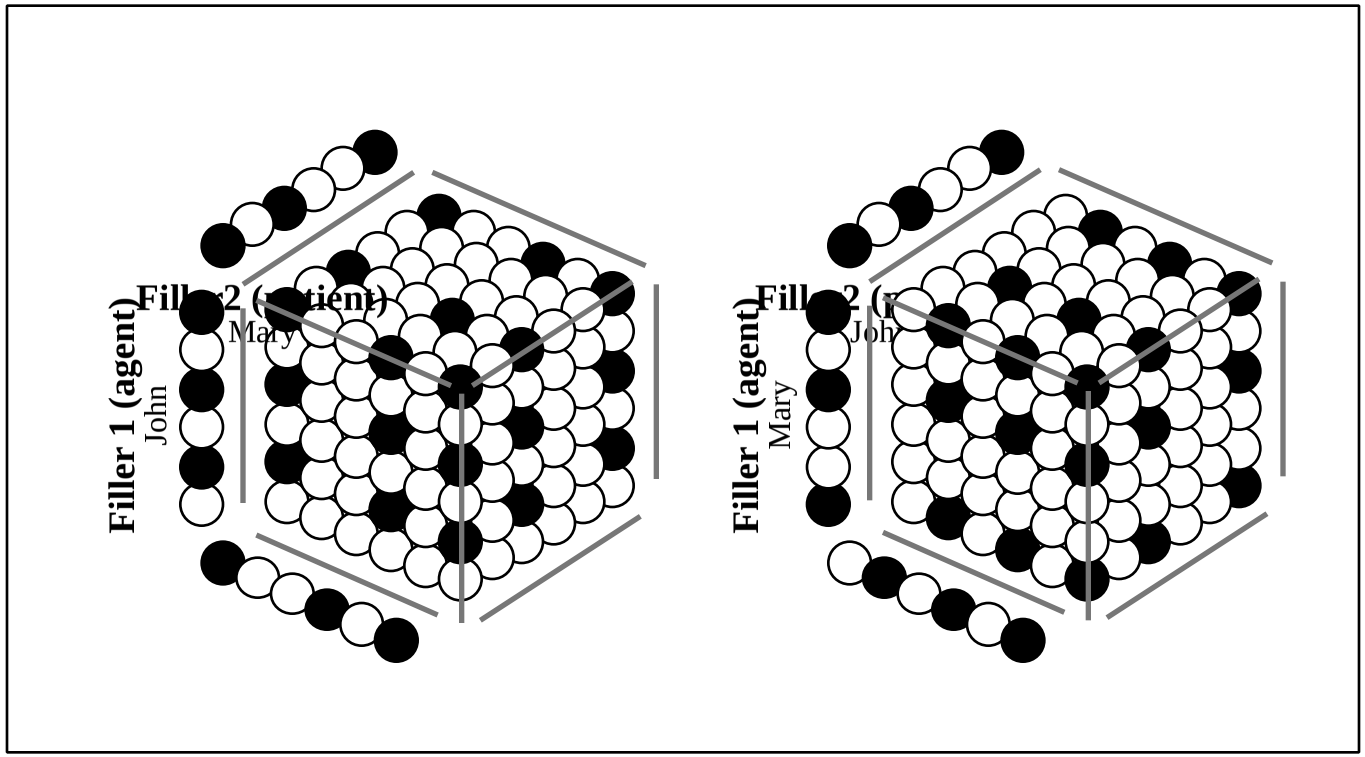


Figure 2

run (John)

walk (John)

eat (John)

Predicate **Predicate** **Predicate**

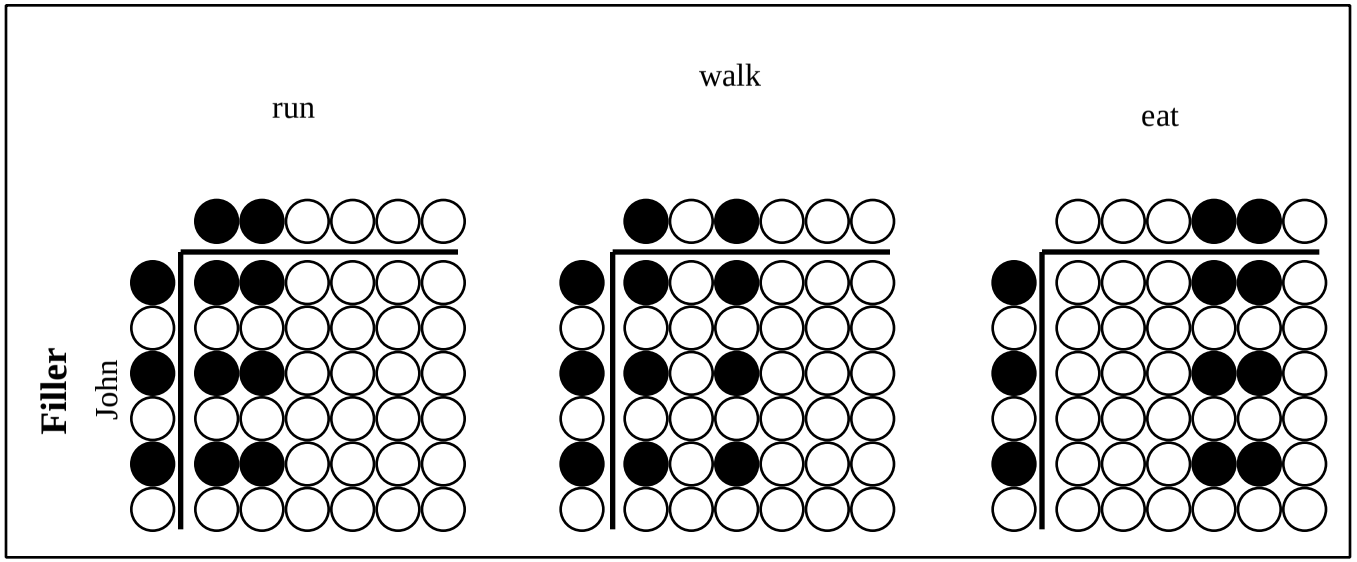


Figure 1