Learning Relations in an Interactive Architecture

Randall Stark
School of Cognitive and Computing Sciences, University of Sussex
Falmer, Brighton, East Sussex BN1 9QH UK
email: randalls@uk.ac.sussex.cogs

Abstract

This paper presents a connectionist architecture for deriving unknown role fillers in relational expressions. First, a restricted solution to the binding problem is presented which ensures systematicity in principle, and allows for sufficient compositionality so as to enable instantiation of shared variables in conjunctive expressions where the same object may fill a variety of roles in a variety of relations. Next, a more detailed architecture is explicated (an extension of McClelland's 1981 "Interactive Activation Competition" architecture) which allows for systematicity in practice while providing a training procedure for relations. Finally, results of the learning procedure for the Family Tree data set (Hinton, 1990) are used to demonstrate robust generalization in this domain.

1. Introduction

This paper outlines an architecture for connectionist symbol processing. The task driving this architecture is that of **variable instantiation**. This task involves contexts with any number of objects playing any number of roles in any number of relations. Given a subset of the objects, the goal is to simultaneously derive all of the unknown role fillers (Stark, 1992). The focus is on learning relations in a manner compatible with a principled binding strategy (one that allows bindings to be propagated so as to allow role fillers to be derived).

2. An Interactive Binding Strategy

While connectionist networks are good at representing single distinct (or schematic) objects, they do not perform as well when simultaneously representing multiple objects, making it difficult to distinguish which features belong to which objects, or which roles objects are playing in a relation. This is known as the *binding problem* (Hinton, McClelland, & Rumelhart, 1986; Smolensky, 1990).

The variable instantiation task serves to constrain the nature of the binding problem. Rather than being concerned with developing a universal binding scheme to encode arbitrary structures (e.g. Smolensky, 1990; Pollack, 1990), the primary concern here is with providing a connectionist architecture that exhibits *systematicity* (the ability to allow in principle any object to appear in any role of any relation) without having to *a priori* dedicate hardware to allow for all possibilities (cf. Fodor & Pylyshyn, 1988).

Consider an *object-representing network* (Figure 1), viewed as a vector of feature units and a connectivity matrix. Here, the auto-associative network functions as a content-addressable memory that will settle on an appropriate object representation. Such a network can realize a distributed representation of a single object, or a single object schema; the features define a vector space in which points correspond to specific objects or possible schemas.

The problem of simultaneously representing multiple distinct objects can be addressed by utilizing multiple copies of the basic object-representing network (Hinton, McClelland, & Rumelhart, 1986), and arranging them in an interactive architecture. A *context* involving two objects will be computed with a *context network* consisting of two object-representing subnetworks, with additional connections between them governing their interaction. The additional connections are derived from the roles the objects are to play in a given relation in the context network. Functioning as content-addressable memories, the networks can cooperate in simultaneously forming representations of distinct

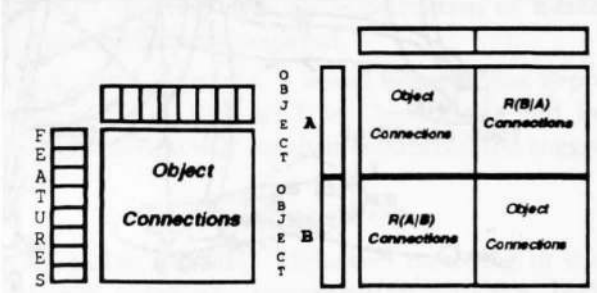

Figure 1: **Object Representing Net**

Figure 2: **R(A,B) Context Network**

The work reported in this paper was supported by a grant from the Joint Council Initiative for Cognitive Science (MRC G8920680). The author wishes to thank Dr. Chris Thornton for helpful discussions at various stages of this work.

objects in specific roles, constraining each other's attempts to settle into minima.

A two-object context network operates on a vector made by concatenating two copies of the feature vector associated with an object-representing network, and a connection matrix divided into four equal submatrices. The upper-left and lower-right submatrices (on the main diagonal) are each copies of the object-representing network's connection matrix. The other two submatrices, called *binding matrices*, each contain constraints on one object given the other object.

Figure 2 shows a network for computing a context with two objects (**A** and **B**) that are in role one and role two of the relation **R** respectively. In this network, the "within layer" connections for each object are just exact copies of the basic connection matrix shown in Figure 1. The other two submatrices are the binding matrices. The lower-left binding matrix contains connections representing constraints on object **A**'s feature vector given that it is in the first role of relation **R** with object **B** in the second role (denoted **R(A|B)**), and the upper-right binding matrix contains connections representing constraints on object **B**'s feature vector given that it is in the second role of relation **R** with object **A** in the first role (denoted **R(B|A)**).

Contexts of arbitrary complexity can be created using such binding matrices, for example as in Figure 3, where three distinct objects play roles in two relations, with one object (**B**) playing different roles in different relations. A generative, compositional syntactic description can be used to describe each context (see Figure titles); furthermore, any context describable with this conjunctive, predicate-based syntax has a corresponding network representation.

Systematicity is realized because each object-representing network is in principle capable of representing any object or schema. Since each variable has its own subspace (containing points corresponding to possible bindings), crosstalk problems are brought under control. When crosstalk is desired (as in the case of a variable being bound to a schema), superpositional representation still occurs *within* a variable subspace. Object-representing connection matrices (along the main diagonal of a context) provide mappings *within* these subspaces, while the binding matrices provide mappings *between* variable subspaces.

Most of the work reported in this paper is devoted to describing an advantageous vector representation of objects and explicating procedures for determining the contents of the binding matrices. That such procedures exist can be seen by considering random object vectors and a simple learning procedure (such as that in a Hopfield net). The object-representing network can be trained by applying the Hopfield learning procedure with a training set consisting of all the object vectors. The binding matrices for each relation can be learned by applying the same learning procedure on a network twice the size (as in Figure 2), where the training set for each relation consists of vectors obtained by concatenating the object vectors of each pair of objects observed in the relation. When learning the binding matrices, the connections in the object-representing networks are "frozen", that is, only connections in the upper-right and lower-left quadrants are learned.

Binding is then a *constructive* process in which a context network is generated by creating a unit vector which consists of n concatenated copies of the basic object vector, where n is the number of objects (or *variables*) in the context. The overall connection matrix can then be constructed dynamically, using the object-representing network's matrix along the main diagonal of the context matrix, and filling in the binding matrices learned for each of the relations, as in Figure 3. Thus contexts involving any configuration of objects in any conjunctive configuration of relations can be modelled. It is this property of the architecture that I refer to as *compositional*.

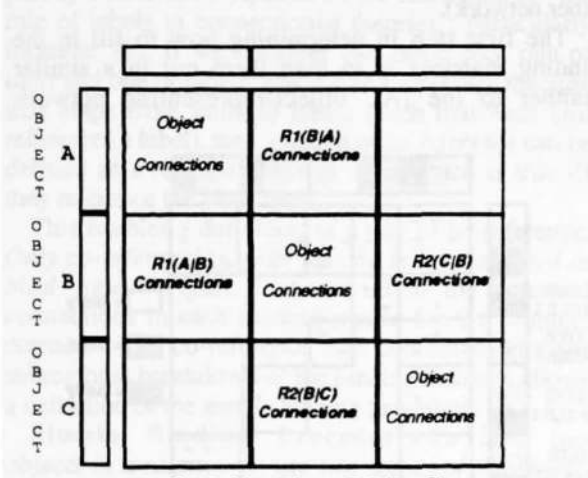

Figure 3: **R1(A,B) ∧ R2(B,C) Context Network**

3. Interactive Representation of Objects

This section looks at one aspect of the more detailed architecture by considering structure *within* the object-representing networks (Figure 1). The architecture used to represent objects is based on McClelland's (1981) "Interactive Activation Competition" (IAC) architecture, best known as the one underlying the "Jets and Sharks" model. The IAC architecture provides both a localist representation with an *instance* subnetwork containing a unit for each object, and a form of distributed representation whereby each object is represented by a pattern of activation over the remaining feature units. These units are further divided into *attribute* subnetworks, each with unique units for each value an attribute may take. Networks

using this architecture are able to represent objects and exhibit a number of interesting properties, including the ability to form schemas and function as a content-addressable memory (the network is able to "fill in" an object's attribute values given a subset of them).

When considering the problem of learning relations, however, the *right* features and attributes need be present. The binding scheme described in Section 2 is dependent on features being explicitly represented if they are important in deriving the nature of a relationship. There is nothing inherent in the architecture to guarantee that this condition is met. While not claiming to have found a general solution, it is suggested that certain specific attributes are useful in representing and learning relations.

In particular, attributes associated with the relations themselves can be seen to be of use. If *John loves Mary*, then John has the attribute of loving someone, namely Mary. Likewise, Mary has the attribute of being loved (by John). The IAC architecture offers a simple way of modeling such attributes in the same manner as any other attribute; for a given two-place relation, two additional attribute subnetworks (one for each role of the relation) may be incorporated in the model, each with value units for each object that has been observed in the given role of the given relation. This is a form of *conjunctive encoding* (Hinton, McClelland, & Rumelhart, 1986), since each unit represents the conjunct of a relation, a role, and an object. Thus part of an object's distributed representation will involve features which indicate that object's relationship to other objects in the domain. The remainder of this paper will focus solely on these relational attributes.

Consider an example domain with three individuals (*John*, *Mary*, and *Sally*), with the five observed facts: `loves(John,Mary)`, `loves(Sally,John)`, `hates(Mary,John)`, `hates(John,Sally)`, `hates(Sally,Mary)`. The corresponding IAC network (in both schematic and vector/matrix form) is shown in Figure 4 (the connection matrix is a detail of the object-representing matrix [Figure 1] duplicated along the main diagonal in Figures 2 and 3).

While this approach allows the realization of simple relational attributes, the limitations of conjunctive coding are well known (Hinton, McClelland, & Rumelhart, 1986; Fodor & Pylyshyn, 1988). As the goal is to be able to compute with contexts of arbitrary complexity, allowing for any configuration of objects in any configuration of relations, a degree of *compositionality* is required that cannot be attained through conjunctive coding alone, if we are to be able to represent not just, e.g., the person that John loves, but also the person who hates the person who John loves, or the mother of the person who hates the person who John loves, etc.

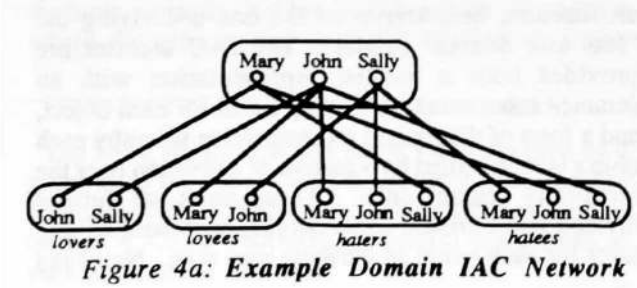

Figure 4a: Example Domain IAC Network

4. Simple Binding

It is to overcome these limitations that the interactive binding strategy was developed. We will first consider a very simple binding procedure (i.e., a method of deriving the connection weights in the binding matrices) to demonstrate the basic principle using the IAC architecture. This procedure does not require any training procedure or learning of weights, as does the full binding procedure presented in the next section.

This simple binding procedure can be seen in terms of Hinton's (1990) discussion of "expanding part/whole hierarchies". An IAC network (the *whole*) consists of a number of subnetworks, some of which represent a specific attribute (a *part*). While the whole is able to represent objects using a distributed representation (which includes all of the attributes), the representation of objects within an attribute network is wholly localist. The effect of the binding procedure is to selectively "expand" some of these "partial" subnetworks into a "whole", enabling a full, distributed representation of its value, which is another unique object (or schema). Thus in a context denoted by *loves(X,Y)*, there will be two copies of the IAC network, one representing *X* and one representing *Y*. The binding procedure will provide a mapping between the *lovee* attribute in the *X* network and the entire *Y* network. This expansion is not strictly hierarchical, as in Hinton's discussion, since there will be an additional mapping between the *lover* subnetwork in the *Y* network and the entire *X* network (i.e., each network is an expansion of a part of the other network).

The first step in determining how to fill in the binding matrices is to map them out in a similar manner to the IAC object-representing network

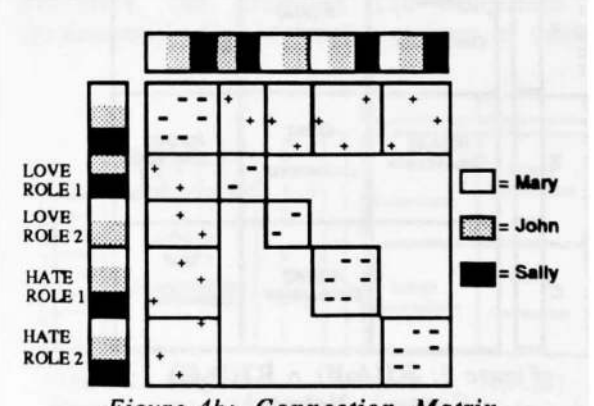

Figure 4b: Connection Matrix

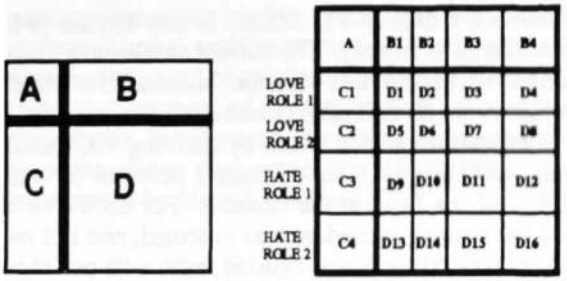

Figure 5 & 6: **Binding Regions & SubRegions**

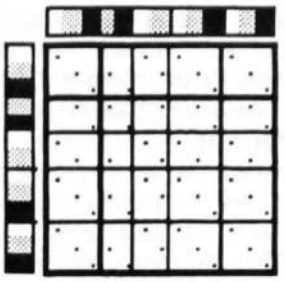

Figure 7: **Binding Connections**

connection matrices (Figure 4). This exposes that the mapping between objects can be seen in terms of mappings between aspects of the representations of objects. Figure 5 shows how each binding matrix can be divided into four "regions": **A** (a mapping between localist representations), **B** and **C** (mappings between localist representations and distributed representations, and *vice versa*), and **D** (a mapping between distributed representations). A further level of structure exposes the individual submappings in each of the regions, involving specific attributes, as shown for the example domain in Figure 6.

Next we consider which of these binding matrix regions and subregions will receive non-zero weights. To accomplish the part/whole expansion of the simple binding procedure, only subregions in the **B** and **C** regions will have non-zero weights. Specifically, two subregions in each binding matrix (one in region **B** and one in region **C**) will be eligible, corresponding to the mappings involving the expanded attribute in these regions. In the example domain (Figure 4), for the context *loves(X,Y)*, subregions **B1** and **C2** will be eligible in the upper-right binding matrix (Figure 2), as will subregions **B2** and **C1** in the lower-left binding matrix.

The issue of which specific connections *within* these subregions will receive non-zero weights is determined by a notion of *co-reference* (motivated in part by Fodor and Pylyshyn's own discussion of "the role of labels in connectionist theories", particularly footnote 12). If we consider two basic sets, one of *units* and one of *labels*, and define a *reference function* that maps from units to labels (such that each unit references a label), then a notion of *co-reference* can be defined as a relation between units which is true iff they reference the same label.

This enables a definition of a rule of co-reference: *Only co-referencing units may be inter-connected at bind time*. Figure 7 shows all of the potential connections in each binding matrix for the example domain. The co-reference rule combined with the subregional breakdown of the binding matrices allows a definition of the simple binding procedure:

Simple Binding Procedure: *To bind two objects in a relation **R**, use two copies of the object-representation network (forming a context network). Positively connect co-referencing units between the instance subnetwork of the first object network and the attribute subnetwork representing the first role of relation **R** in the second object network. Likewise, positively connect co-referencing units between the instance subnetwork of the second object network and the attribute subnetwork representing the second role of relation **R** in the first object network.*

Given a complete description of a domain, contexts of arbitrary complexity (limited only by resources) may be constructed using repeated application of this binding procedure (as in Figure 3). Given information about at least one object in the context, and appropriate network dynamics, the context network will settle in a state whereby all unclamped object-representing subnetworks will represent "solution" objects (or schemas) appropriate for the context. If the set of observed domain facts is complete, *the solutions will be the same as would have been derived by traditional means* (e.g., by Prolog).

5. Training and Generalization

The simple binding procedure, while demonstrating the basic power of the interactive strategy to perform variable instantiation within conjunctively specified contexts, requires complete domain information in order to derive solutions. Although the strategy allows for each object network to represent any object *in principle*, the simple binding procedure will not result in object networks settling on representations of objects that have not been observed in a specified role of a relation. This section outlines a procedure for *learning* binding matrix connection weights and allowing a greater degree of generalization.

The procedure is simple, and follows that outlined near the end of Section 2. The binding matrices are learned using a two-network context (as in Figure 2). A pair of binding matrices are learned for each relation, using as a training set a set of vectors obtained by concatenating the vector representations for each pair of objects observed to be in the relation. Only co-referencing connections are learned (see Figure 7), and the object-representing connections are "frozen" (so as to allow object networks to settle on any object, not just ones observed in the given role of the given relation). Any learning rule may potentially be used.

Generalization is achieved by applying a special "generalization" rule to each resultant matrix. The effect of this rule is to "combine" all of the learned connections in each subregion (Figure 6) into a *single* value, thus forming a *correlation matrix* which represents a generalized version of the binding weights for a relation. Any of a variety of generalization rules may be used, such as taking the mean.

When the binding procedure is invoked, it will supply the weights for each of the potential connections in each of the binding matrices by finding a uniform value for each connection using a normalization formula applied to the values in the appropriate correlation matrix. Thus every eligible connection in a given subregion will have the same weight after binding, even if the learning rule derived different weights for each connection.

Analysis of binding matrices in terms of subregions allows each inter-attribute mapping to be considered separately. As attribute values are mapped at run time, the inherent content-addressable properties of the IAC architecture allow each individual object-representing subnetwork to settle on an object representation that is consistent with the other object representations being derived in the context network.

The co-reference rule provides a means of raising the power of learned mappings by considering not just whether an object in one role is likely to have a specific value for an attribute given that an object in another role has a given attribute value, but whether two objects are likely to have the *same* value for any of their attributes. The correlation matrix subregions can be interpreted in terms of rules governing the objects in the relation. Region **A** contains a single reflexive rule, while regions **B** and **C** encode symmetry rules. Thus in the example domain these regions will encode the rule

$$\texttt{loves}(O_1, O_2) \rightarrow \texttt{hates}(O_2, O_1).$$

More complex rules involving third parties are handled in each of the **D** subregions; e.g. subregion **D 3** (Figure 6) in the lower-left correlation matrix in the example domain encodes the rule

$$\texttt{loves}(X, O_1) \rightarrow \texttt{hates}(X, O_2)$$

(which is always true in the example domain when O_1 and O_2 are bound in the relation `loves`, as the rule asserts that if the `lover` [O_1] is loved by someone [X], that person [X] hates the `lovee` [O_2]).

6. Experimental Results

An implementation of this architecture (described in the Appendix) has been used as the basis for an experimental study of various aspects of the architecture, using the "Family Tree" domain of Hinton (1990) (and others, e.g. Quinlan, 1990; Melz & Holyoak, 1991). This domain of kinship relations consists of twenty-four individuals and twelve relations, organized in two isomorphic "family trees". There are a total of 112 "facts" in this domain (when considered as triples). The current architecture is well suited to handle this domain because the kinship relations are all definable in terms of other relations.

Generalization was tested by deriving 400 training sets, such that each set contained between 60% and 100% of the facts in the domain. For each training set, the training procedure was executed, and 172 two-object contexts were constructed, each with one object known (68 contexts in which the object in the first role was known, and 104 contexts in which the second role object was known). After settling, the objects in the missing role were derived by examining the activation of units in the instance subnetwork of the unclamped object network (see the Appendix and Stark, 1992). Perfect performance was indicated by the proper set of 224 objects (112 in each role) being determined by the settled context networks.

Figure 8 show the results of the basic test. The X axis represents the percentage of the domain facts present in each training set, and the Y axis represents the percentage of unknown objects that were derived correctly. The diagonal line indicates expected performance if no generalization took place (i.e., X = Y). In the graph, training sets with the same number of missing facts are grouped together, and their max, min, and mean plotted.

Hinton (1990) reports variable results with 4% of the facts missing, as did Quinlan (1991). The current system performs perfectly on training sets with nearly 20% of the facts missing, and can still retrieve over 90% of missing role fillers in cases where 40% of the facts are missing from the training set. The current experiment, involving 400 different data sets, shows the importance of exactly *which* facts are missing (as can be seen in the variance between the max and min figures for each test set group).

Other experiments show that the effect is quite robust, demonstrating considerable parameter insensitivity and tolerance to "lesions" in the binding matrices. In addition, contexts with up to a dozen objects have been tested and found to perform well, especially when a high percentage of the domain facts are known.

The fact that relational correlations are stored independently of specific objects, coupled with the

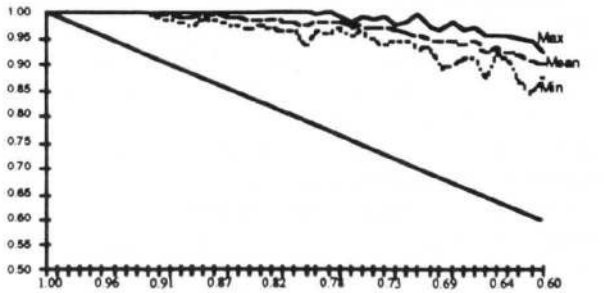

Figure 8: Basic Generalization Test Results

dynamic nature of the binding procedure driven by the co-reference rule, enables a more powerful type of generalization: a set of relations once learned may be applied to a new set of objects *without retraining* provided the regularities governing the relations remain the same, thus achieving effects similar to those reported by Melz & Holyoak (1991).

7. Discussion

This system, and particularly its approach to the binding problem, differs from other similar ones primarily in its goals. Rather than focussing on a universal encoding scheme (such as, e.g. Smolensky, 1990; Pollack, 1990), the emphasis is placed on learning relations and propagating bindings in order to perform variable instantiation in contexts describable by a generative predicate-based syntax.

While Ajjanagadde and Shastri's (1991) temporal binding system does focus on propagation of bindings in a variable instantiation task, and indeed offers provably correct inference in this domain, it does not offer a training procedure, does not address the issue of shared variables (compositionality), and is limited to localist or quasi-localist representation of objects. Hinton, McClelland, and Rumelhart (1986) suggest the possibility of solving the binding problem by making "multiple copies", but express concern about the *implementation* of copies. Temporal binding may indeed provide an implementation mechanism for my scheme, which is dependant on some form of "copies" of a basic network, but the current work focuses not on how copies are made, but rather examines when they are needed and how they should interact.

The current system has a number of important limitations. These include a treatment only of two-place predicates, and the ability to perform only first order bindings; it would be useful to be able to determine what relation two objects are in, given their roles. Perhaps the most important limitation is the lack of hidden units. This lack is partially motivated by an interest in seeing how far one could go in solving problems such as the Family Tree without using hidden units. An extended architecture that exploits hidden units has been developed and will be the subject of future experimentation.

Appendix (Implementation)

Unit activation function:

$$U_j = \max\ (0,\ \tanh(\sum_i U_i W_{ij}))$$

Learning Rule:

$$W_{ij} = p(U_j|U_i) = \frac{(U_i \wedge U_j)}{U_i}$$

Generalization Rule:

$C_r = \mathbf{mean}(W_{ij} > 0)$ (for each subregion r).

Normalization Rule:

$$W_{ij} = 0.1\ C_r$$

Constant weights in each IAC network were set to 0.1 and -0.1. The network was allowed 20 cycles to settle.

The localist competition subnetwork in each object network was assumed to be a "K-winner-take-all" network, where K was equal to the number of objects in the solution (this was assumed to be information supplied to the system). The weights for each localist subnetwork were set according to the formula

$$W = \frac{0.3}{K^2}$$

After settling, the K units with the highest activations (above zero) were taken as solution. No units with zero activation were considered solutions.

References

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

Fodor, J. A. and Pylyshyn, Z. W. (1988) "Connectionism and Cognitive Architecture: A Critical Analysis," *Cognition*, 28:3-71.

Hinton, G.E. (1990) "Mapping Part-Whole Hierarchies into Connectionist Networks," *Artificial Intelligence* 46:47-75

Hinton, G.E. McClelland, J.L., and Rumelhart,D.E. (1986) "Distributed Representations," in Rumelhart, D.E., McClelland, J.L. and the PDP Research Group, eds., *Parallel Distributed Processing*, MIT Press, Cambridge, MA.

McClelland, J.L. (1981): "Retrieving General and Specific Knowledge from Stored Knowledge of Specifics," *Proceedings of the Third Annual Conference of the Cognitive Science Society*, Berkeley, CA.

Melz, E., and Holyoak, K. (1991) "Analogical transfer by constraint satisfaction," *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, IL.

Pollack, J.B. (1990) "Recursive Distributed Representations," *Artificial Intelligence* 46: 77-107.

Smolensky, P. (1990) "Tensor product variable binding and the representation of symbolic structures in connectionist systems," *Artificial Intelligence* 46:5-46.

Stark, R. (1992) "A Symbolic/Subsymbolic Interface for Variable Instantiation," to appear in *Artificial Neural Networks II: Proceedings of the International Conference on Artificial Neural Networks*, Elsevier.

Quinlan, J.R. (1990) "Learning Logical Definitions from Relations," *Machine Learning* 5:239-266.