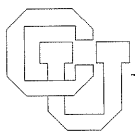
On Variable Binding and the Representation Of Symbolic Structures in Connectionist System

Paul Smolensky

CU-CS-355-87 February 1987



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

On variable binding and the representation of symbolic structures in connectionist systems

Paul Smolensky

CU-CS-355-87 February, 1987

Department of Computer Science &
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309–0430
(303) 492–8991
smolensky@boulder.csnet

Abstract

A general method, the *tensor product representation*, is defined for the connectionist representation of value/variable bindings. The method allows the fully distributed representation of bindings and symbolic structures. Fully and partially localized special cases of the tensor product representation reduce to existing cases of connectionist representations of structured data. The representation rests on a principled analysis of structure; it saturates gracefully as larger structures are represented; it permits recursive construction of complex representations from simpler ones; it respects the independence of the capacities to generate and maintain multiple bindings in parallel; it extends naturally to continuous structures and continuous representational patterns; it permits values to also serve as variables; it enables analysis of the interference of symbolic structures stored in associative memories; and it leads to characterization of optimal distributed representations of structural roles and a recirculation algorithm for learning them.

Submitted to *Artificial Intelligence*

1. Introduction

There has been a recent surge of interest in the "connectionist" approach to artificial intelligence. In this approach, AI models are large, massively interconnected networks of simple parallel processors, each possessing a numerical "activation value" which it computes from the activation values of its neighbors according to some simple numerical formula. (For recent collections, see Feldman, Ballard, Brown, & Dell, 1985; McClelland, Rumelhart, & the PDP Group, 1986; Rumelhart, McClelland, & the PDP Group, 1986) The recent interest in connectionist research has been fueled by significant conceptual and technical advances in the approach, by a rapidly increasing number of demonstrations of the importance of connectionist computation in the detailed modeling of human cognitive behavior, by growing interest in the power of various parallel architectures, and by dramatic progress in neuroscience demanding a theoretical understanding of the properties of neural-like computation.

Connectionist models rely on parallel numerical computation rather than the serial symbolic computation of traditional AI models, and with the inroads of connectionism has come considerable debate about the roles these two forms of computation should play in AI. While some presume the approaches to be diametrically opposed, and argue that one or the other should be abandoned, others argue that the two approaches are so compatible that in fact connectionist models should just be viewed as implementations of symbolic systems.

In (Smolensky, forthcoming; also 1986a, 1987) I have argued at considerable length for a more complex view of the roles of connectionist and symbolic computation in cognitive science. A one-sentence summary of the implications of this view for AI is this: connectionist models may well offer an opportunity to escape the brittleness of symbolic AI systems, a chance to develop more human-like intelligent systems—but only if we can find ways of naturally instantiating the sources of power of symbolic computation within fully connectionist systems. If we ignore the connectionist approach, we miss our current best hope for formally capturing the subtlety, robustness, and flexibility of human cognition. If we ignore the symbolic approach, we throw out tremendous insights into the nature of the problems that must be solved in creating intelligent systems, and of techniques for solving these problems; we probably doom the connectionist approach to forever grappling with simple cognitive tasks that fall far short of the true capacity of human intelligence. If we use connectionist systems merely to implement symbolic systems, we might get AI systems that are faster and more tolerant of hardware faults, but they will be just as brittle.

The present paper is part of an effort to extend the connectionist framework to naturally incorporate, without losing the virtues of connectionist computation, the ingredients essential to the power of symbolic computation. This extended version of connectionist computation would integrate, in an intimate collaboration, the discrete mathematics of symbolic computation and the continuous mathematics of connectionist computation. This paper offers an example of what such a collaboration might look like.

One domain where connectionist computation has much to gain by incorporating some of the power of symbolic computation is language. The problems here are extremely fundamental. Natural connectionist representation of a structured object like a phrase structure tree—or even a simple sequence of words or phonemes—poses serious conceptual difficulties, as I will shortly discuss. The problem can be traced back to difficulties with the elementary operation of binding a value to a variable. It is this basic problem that is addressed in this paper.

I begin in Section 1.1 by discussing why natural connectionist representation of structured objects is a problem. I list several properties of the solution to this problem that is presented in this paper. In Section 1.2 I respond to the possible connectionist criticism that it is misguided to even try to solve this problem. Then in Section 1.3 I outline the rest of the paper.

Before proceeding it is worth commenting on where the research reported here fits into an overall scheme of connectionist AI. As in the traditional approach, in the connectionist approach several components must be put together in constructing a model. Elements of the task domain must be represented, a network architecture must be designed, and a processing algorithm must be specified. If the knowledge in the model is to be provided by the designer, a set of connections must be designed to perform the task. If the model is to aquire its knowledge through learning, a learning algorithm for adapting the connections must be specified, and a training set must be designed (eg., a set of input/output pairs). For most of these aspects of connectionist modeling, there exists considerable formal literature analyzing the problem and offering solutions. There is one glaring exception: the representation component. This is a crucial component, for a poor representation will often doom the model to failure, and an excessively generous representation may essentially solve the problem in advance. Representation is particularly critical to understanding the relation between connectionist and symbolic computation, for the representation often embodies most of the relation between a symbolically characterized problem (eg. a linguistic task) and a connectionist solution.

Not only is the connectionist representation problem a central one, it is also a problem that is amenable to formal analysis. In this paper the problem will be characterized as finding a mapping from a set of structured objects (eg. trees) to a vector space, the set of states of the part of a connectionist network representing those objects. The mélange of discrete and continuous mathematics that results is reminiscent of a related classical area of mathematics: the problem of representing abstract groups as collections of linear operators on a vector space. The discrete aspects of group theory and the continuous aspects of vector space theory interact in a most constructive way. Group representation theory, with its application to quantum physics, in fact offers a useful analogy for the connectionist representation of symbolic structures. The world of elementary particles involves a discrete set of particle species whose properties exhibit many symmetries, both exact and approximate, that are described by group theory. Yet the underlying elementary particle state spaces are continuous vector spaces, in which the discrete structure is imbedded. In the view that guides the research reported here, in human language processing, the discrete symbolic structures that describe linguistic objects are actually imbedded in a continuous connectionist system that operates on them with flexible, robust processes that can only be approximated by discrete symbol manipulations.

One final note on terminology. In most of this paper the structures being represented will be refered to as *symbolic structures*, because the principal cases of interest will be objects like strings and trees. Except for the consideration of particular symbolic structures, however, the analysis presented here is of structured objects in general; it therefore applies equally well to objects like images and speech trains which are not typically considered "symbolic structures." With this understood, in general discussions I will indiscriminately refer to objects being represented as "structures," "structured objects," or "symbolic structures."

1.1. Distributed representation and variable binding in connectionist systems

I have called the problem considered in this paper that of finding "natural" connectionist representation of structured objects and variable bindings. In fact what I refer to is the problem of finding connectionist representations that are *fully distributed.* The notion of *distributed representation* is central to the power of the connectionist approach (eg., Anderson & Hinton, 1981; Hinton, McClelland & Rumelhart, 1986; Smolensky forthcoming, 1986b; for an opposing view, see Feldman, 1986). To illustrate the idea of distributed representation, consider the NETtalk system, a connectionist network that learns to pronounce written English (Sejnowski & Rosenberg, 1986; see Fig. 1). Each output of this network is a phonetic segment, eg. the vowel [*i*] in the pronunciation of the word *we*. Each phonetic segment is represented in terms of phonetic features; for [*i*], we have: front = 1, tensed = 1, high-frequency = 1, back = 0, stop = 0, nasal = 0, and so forth. There is one output processor in the network for each of the phonetic features, and its numerical value indicates whether that feature is present (1) or absent (0). Each phonetic segment is therefore represented by a *pattern of activity* over the numerous output processors, and each output processor participates in the representation of many different outputs. This defines a distributed representation of the output phonetic segment.

At the opposite end of a connectionist representational spectrum are *local representations.* These too are illustrated by NETtalk; this time, in the input. Each NETtalk input consists of a letter to be pronounced together with the three preceding and three following letters to provide some context. For each of these seven letters there is a separate pool of input processors in the network, and within each pool there is a separate processor for each letter. In the input representation, in each of the seven pools the single processor corresponding to the letter present is assigned activity 1, and the remaining processors in the pool are all assigned activity 0. This representation is local in two senses. Most obviously, different letters are represented by activity in disjoint localities—in single processing units. Unlike the output activity, there is no overlap of the activity representing alternative values. The other sense of locality is that the activity representing different letter *positions* are all disjoint: each pool localizes the activity representing one letter position.

The input representation in NETtalk illustrates the problems of representing structures and of variable binding in connectionist networks. The input is a string of seven characters (of which the middle one is to be pronounced). There is a pool of processing units dedicated to representing each item in this string. Each pool can be viewed as a slot in the structure: a variable. The value of each variable is a pattern of activity residing in its pool of units. In

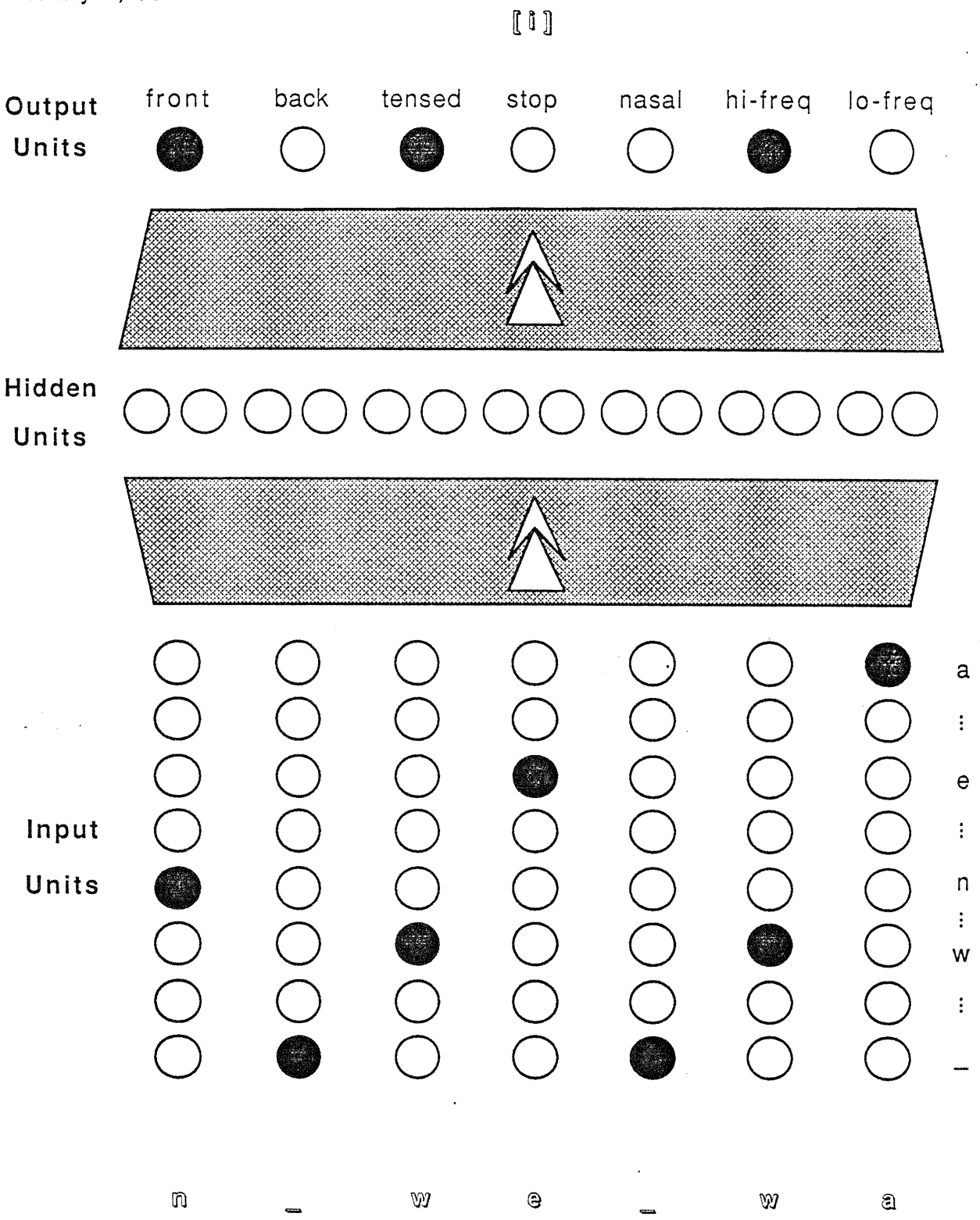


Figure 1. The NETtalk system of Sejnowski & Rosenberg (1986) illustrates both distributed and local connectionist representations.

NETtalk this pattern is localized; we will later consider examples of models in which the corresponding pattern is distributed throughout the pool. Regardless of the patterns used to represent the values, in such systems *the variables are localized regions of the network.* These variables are in fact *registers*, and nearly all connectionist systems have represented structured data using them. Yet registers are hardly natural or desirable within connectionist models. In order to make available in the processing of structured data the full power of connectionist computation that derives from distributed representation, we need to use *distributed representations of variables* in addition to distributed representations of values.

In this paper a completely distributed representational scheme is proposed for variable binding: the *tensor product representation.* In this representation both the variables and the values can be arbitrarily nonlocal. Applications of the tensor product scheme to the connectionist representation of complex structured objects is explored. Features of the tensor product representation, most of which distinguish it from existing representations, include the following (corresponding section numbers are indicated in parentheses):

- The representation rests on a principled and general analysis of structure: role decomposition (2.2.1).
- A fully distributed representation of a structured object is built from distributed representations of both the structure's constituents and the structure's roles (2.2.4).
- Nearly all previous connectionist representations of structured data, employing varying degrees of localization, are special cases (2.3).
- If a structure does not saturate the capacity of a connectionist network that represents it, the components of the structure can be extracted with complete accuracy (3.1).
- Structures of unbounded size can be represented in a fixed connectionist network, and the representation will saturate gracefully (3.2).
- The representation applies to continuous structures and to infinite networks as naturally as to the discrete and finite cases (3.3).
- The binding mechanisms can be simply performed in a connectionist network (3.4).
- The representation respects the independence of two aspects of parallelism in variable binding: generating vs. maintaining bindings (3.4.1).
- The components of structures can be simply extracted in a connectionist network (3.4.2).
- A value bound to one variable can itself be used as a variable (3.6).
- Connectionist representations of operations on symbolic structures, and recursive data types, can be naturally analyzed (3.7).
- Retrieval of representations of structured data stored in connectionist memories can be formally analyzed (3.8).
- A general sense of optimality for activity patterns representing roles in structures can be defined and analyzed (3.9.1).
- A connectionist "recirculation" learning algorithm can be derived for finding these optimal representations (3.9.2).

1.2. Connectionist representation of symbolic structures

The general issue behind the research reported here is the representation in connectionist systems of symbolic structures. What are computationally adequate connectionist representations of strings, trees, sentences?

This section is addressed to connectionists who may find this question misguided. The essence of the connectionist approach, they might say, is to expunge symbolic structures from models of the mind. I must agree that the connectionist approach is rather far from a "language of thought" view of cognition in which all mental states are formalized as symbolic structures. However there still remains in connectionism an important role to be played by language and symbolic structures, even if that role is substantially reduced relative to its counterpart in the traditional radically symbolic approach. I have argued this point in some detail in Smolensky (forthcoming), and will only summarize the relevant conclusions here.

Any connectionist model of natural language processing must cope with the questions of how linguistic structures are represented in connectionist models. A reasonable starting point would seem to be to take linguistic analysis of the structure of linguistic objects seriously, and to find a way of representing this structure in a connectionist system. Since the majority of existing representations of linguistic structure employ structures like trees and strings, it is important to find adequate connectionist representations of these symbolic structures. It may well turn out that once such representations are understood, new connectionist representations of linguistic structures will be developed that are not truly representations of symbolic structures but which are more adequate according to the criteria of linguistics, computational linguistics, psycholinguistics, or neurolinguistics. It seems likely, however, that such improvements will rest on prior understanding of connectionist representations of existing symbolic descriptions of linguistic structure.

The importance to the connectionist approach of representing linguistic structures goes well beyond models of natural language processing. Once adequate connectionist representations are found for linguistic structures, then these can serve as the basis for connectionist models of conscious, serial, rule-guided behavior. This behavior can be modeled as explicit (connectionist) retrieval and interpretation of linguistically structured rules. Adequate connectionist models of such behavior are important for connectionist models of higher thought processes.

One line of thought in the connectionist approach implies that analyses of connectionist representations of symbolic structures are unnecessary. The argument goes something like this. Just as a child somehow learns to internally represent sentences with no explicit instruction on how to do so, so a connectionist system with the right learning rule will somehow learn the appropriate internal representations. The problem of linguistic representation is not to be solved by a connectionist theorist but rather a connectionist network.

In response to this argument I have five points.

(1) In the short term, at least, our learning rules and network simulators do not seem powerful enough to make network learning of linguistic representation feasible.

(2) Even if such learning is feasible at some future point, we will still need to *explain* how the representation is done. There are two empirical reasons to believe that such explanation will require the kind of analysis begun in this paper: explanation of the computation of real neural networks has turned out to require much analysis, as mere observation has proved woefully inadequate; the same has turned out to be true even of the self-organized connectionist networks that perform computations vastly simpler than most of natural language processing.

(3) It is important to try to build bridges as soon as possible between connectionist accounts of language processing and existing accounts; the problem is just too difficult to start all over again from scratch.

(4) We would like to be able to experiment *now* with connectionist learning models of rather complex linguistic skills (eg. parsing, anaphoric resolution, and semantic interpretation, all in complex sentences). For now, at least, such experiments require connectionist representation of linguistic structures to serve as inputs and outputs. We want to study the learning of the operations performed on linguistic structures without waiting many years for the completion of the study of the learning of the linguistic representations themselves.

(5) Language is more than just a domain for building models, it is a foundation on which the entire traditional theory of computation rests. To understand the computational implications of connectionism, it is crucial to know how the basic concepts of symbolic computation and formal language theory relate to connectionist computation.

Of course exploiting connectionist representations of the sort of symbolic structures used in symbolic AI by no means commits one to a full connectionist implementation of symbolic AI, which, as stated earlier, would miss most of the point of the connectionist approach. The semantic processing of a connectionist representation of a parse tree should not be performed by a connectionist implementation of serially applied symbolic rules that manipulate the tree; rather, the processing should be of the usual connectionist sort: massively parallel satisfaction of multiple soft constraints involving the micro-elements forming the distributed representation of the parse tree. Thus in this paper connectionist representations of *pop* and *cdr* will be mathematical relations between patterns of activity, not

processes carried out over time in a connectionist network as part of an extended serial computation (in contrast to Touretzky, 1986). The view behind the present research is not that mental operations are always serial symbol manipluations (although a few are); rather the view is that the information processed often has useful symbolic *descriptions*, and that these descriptions should be taken seriously. (This view is spelled out in detail in Smolensky, forthcoming).

1.3. Outline of the paper

In Section 2, the notion of connectionist representation is formally defined and the tensor product representation is constructed. Examples are considered, and the various special cases that reduce to previous connectionist representations are discussed. In Section 3, a number of properties of the tensor product representation are proved and several extensions discussed. The connectionist representation of symbolic operations is defined, and examples for stacks and trees are considered. Retrieval of symbolic structures represented in connectionist memories by the tensor product representation is analyzed. Finally, a sense of optimality for patterns representing roles in structures is defined and explored, and a connectionist "recirculation" algorithm is derived for learning these optimal representations. Section 4 is a summary and conclusion.

The entire paper centers around the vector operation of tensor product. This operation is simple to define numerically but considerably more subtle to characterize abstractly. Because the tensor product is central here but often omitted in treatments of linear algebra, a brief exposition of its abstract characterization is offered in the main Apppendix, Section 5. In the process of characterizing the tensor product, certain other vector space concepts are introduced that are also drawn upon in the paper. Section 6 is a small Appendix containing a calculation deferred from a proof.

2. Connectionist representation and tensor product binding: Definition and examples

In this section I first formally characterize the notion of connectionist representation. Next, the problem of representing structured objects is reduced to three subproblems: decomposing the structures via roles, representing conjunctions, and representing variable/value bindings. First role decompositions are discussed, and then I define the superpositional representation of conjunction and the tensor product representation for variable/value bindings. Next I show how various special cases of the tensor product representation yield the previous connectionist representations of structured data.

2.1. Connectionist representation

The question of how to represent symbolic structures in connectionist systems will be treated formally in this paper in the following way.

Connectionist representations are patterns of activity over connectionist networks; these patterns can extend over many processors in the network, as in distributed representations, or be localized to a single processor, as in a local representation. Such a pattern is a collection of activation values: a vector with one numerical component for every network processor. The space of representational states of a connectionist network thus lies in a vector space, with a dimension equal to the number of processors in the network. Each processor corresponds to an independent basis vector; this forms a distinguished basis for the space. In many connectionist networks the processor's values are restricted in some way; such restrictions are important for consideration of the dynamics of the network but are not central to the representational issues considered here, and they will be ignored. (For expositions of the application of vector space theory—linear algebra—to connectionist systems, see, e.g., Jordan, 1986; Smolensky, 1986b.)

> DEFINITION 2.1.1: The *activity states of a connectionist network* are the elements of a vector space V which has a distinguished basis $\{\hat{\mathbf{v}}_i\}$.

Whenever I speak of a vector space representing the states of a connectionist network, a distinguished basis will be implicitly assumed. Rarely will it be necessary to deal explicitly with this basis. Sometimes it will be useful to use the canonical inner product associated with the distinguished basis: the one in which the basis vectors are orthogonal and of unit norm. (Equivalently, this inner product of two vectors can be computed as the sum of the products of

corresponding vector components with respect to the distinguished basis.) Whenever I speak of activity patterns being orthogonal, or of their norm, these concepts are taken to be defined with respect to this canonical inner product; the inner product of vectors **u** and **v** will be denoted **u**·**v**.

DEFINITION 2.1.2: A *connectionist representation* of the symbolic structures in a set S is a mapping ψ from S to a vector space V:

$$\psi\colon S \to V$$

Of central interest are the images under the mapping ψ of the relations between symbolic structures and their constituents, and the images of the operations transforming symbolic structures into other structures. Also important are basic questions about the representation mapping such as whether distinguishable symbolic structures have distinguishable representations:

DEFINITION 2.1.3: A connectionist representation ψ is *faithful* iff it maps no structure to the zero vector $\mathbf{0} \in V$ and is one-to-one:

$$s_1 \neq s_2 \Rightarrow \psi(s_1) \neq \psi(s_2)$$

2.2. Tensor product representation: Definition

The representation of structured objects explored in this paper requires first that structures be viewed as possessing a number (possibly unbounded) of *roles* which, for particular instances of the structure, are individually bound to particular *fillers*. For example, a string may be viewed as possessing an infinite set of roles $\{r_1, r_2, \cdots\}$ where r_i is the role of the i^{th} element in the string. A particular string of length n involves binding the first n roles to particular fillers. For example, the string *aba* involves the bindings $\{a/r_1, b/r_2, a/r_3\}$, using a notation in which f/r denotes the binding of filler f to role r; in this string, the roles r_i for $i>3$ are all unbound. Now note that the structure has been characterized as the *conjunction* of an unordered set of variable bindings. The problem of representing the structure has been reduced to the problems of

(1) representing the structure as a conjunction of filler/role bindings;
(2) representing the conjunction operation;
(3) representing the bindings in a connectionist network.

These problems are respectively considered in Sections 2.2.1 through 2.2.3 and brought together in Section 2.2.4.

2.2.1. Role decompositions of symbolic structures

As a formal definition of roles and fillers, I will take the following:

DEFINITION 2.2.1.1: Let S be a set of symbolic structures. A *role decomposition* F/R for S is a pair of sets (F, R), the sets of *fillers* and *roles*, respectively, and a mapping

$$\mu_{F/R}: F \times R \to Pred(S); (f, r) \mapsto f/r$$

For any pair $f \in F, r \in R$, the predicate on S $\mu_{F/R}(f, r) = f/r$ is expressed: *f fills role r*.

The role decomposition has *single-valued roles* iff for any $s \in S$ and $r \in R$, there is at most one $f \in F$ such that $f/r(s)$ holds.

The role decomposition is *recursive* iff $F = S$.

A role decomposition determines a mapping

$$\beta: S \rightarrow 2^{F \times R}; s \mapsto \{(f, r) | f/r(s)\}$$

The set $\beta(s)$ will be called the *filler/role bindings in* s, and the mapping β will be called the *filler/role representation* of S induced by the role decomposition.

The role decomposition is *faithful* iff β is one-to-one.

The role decomposition is *finite* iff for each $s \in S$, the set of bindings in s, $\beta(s)$, is finite.

Throughout this paper all role decompositions will be assumed to be finite, except in sections where the infinite case is explicitly considered.

Recursive role decompositions are heavily used in the standard description of symbolic structures. For example, the description of a LISP S-expression as a structure whose *car* and *cdr* are both S-expressions is a recursive decomposition via the roles *car* and *cdr*. The tensor product representation to be presented shortly cannot be *defined* using recursive role decompositions; recursive role decompositions can however be *analyzed*, as will be done in Section 3.7.

Faithful role decompositions are particularly useful because the filler/role representations they induce allow us to identify each symbolic structure with a predicate having a simple conjunctive form:

THEOREM 2.2.1.2: Let F/R be a role decomposition of S. For each $s_0 \in S$, define the predicate π_{s_0} by:

$$\pi_{s_0}(s) = \bigwedge_{(f,r) \in \beta(s_0)} f/r(s)$$

where $\wedge$ denotes conjunction. Then if the role decomposition is faithful, the structure s_0 can be recovered from the predicate π_{s_0}.

PROOF: This result follows immediately from the following lemma.

LEMMA 2.2.1.3: The mapping β of the role decomposition maps elements of S into subsets of $F \times R$. These subsets possess a partial order, set inclusion $\subseteq$, which can be pulled back to S via β:

$$s_1 \leq s_2 \text{ iff } \beta(s_1) \subseteq \beta(s_2)$$

Suppose F/R is faithful. Then with respect to the partial order $\leq$, the set of elements of S for which the predicate π_{s_0} holds has a unique least element, which is s_0. In this way s_0 can be recovered from its corresponding predicate π_{s_0}.

PROOF OF LEMMA 2.2.1.3: Since $\beta(s)$ is the set of filler/role bindings in s, $s_1 \leq s_2$ iff the bindings in s_1 are a subset of those of s_2:

$$s_1 \leq s_2 \text{ iff } [\text{for all } f \in F \text{ and } r \in R, f/r(s_1) \Rightarrow f/r(s_2)]$$

for all $f \in F$ and $r \in R$. Now consider the set of elements s satisfying the predicate π_{s_0}:

$$S(\pi_{s_0}) := \{s \in S \mid \pi_{s_0}(s)\} = \{s \in S \mid \text{for all } f \in F \text{ and } r \in R, f/r(s_0) \Rightarrow f/r(s)\}$$

$$= \{s \in S \mid s_0 \leq s\}$$

This set contains s_0, and s_0 is a least element; it remains to show that there is no other least element. Consider any other element s_1 in $S(\pi_{s_0})$. Since μ is faithful and $s_1 \neq s_0$, there is at least one binding f/r not shared by s_0 and s_1. Since $s_1 \in S(\pi_{s_0})$ and s_0 is a least element of $S(\pi_{s_0})$, we must have $f/r(s_1) \wedge \neg f/r(s_0)$. This implies $\neg(s_1 \leq s_0)$ so s_1 cannot be a least element in $S(\pi_{s_0})$. []

[]

2.2.2. Connectionist representation of conjunction

The representation of conjunction in connectionist models has traditionally been performed with pattern superposition, i.e. vector addition. If two propositions are each represented in a connectionist system by some pattern of activity, the representation of the conjunction of those propositions is the pattern resulting from superimposing the individual patterns. This paper adopts this method. In terms of the representation mapping ψ, we can write:

DEFINITION 2.2.2.1: A connectionist representation ψ employs the *superpositional representation of conjunction* iff:

$$\psi(\bigwedge_i p_i) = \sum_i \psi(p_i)$$

The representation of the conjunction of a collection of propositions is the sum of the representations of the individual propositions.

Note that, like conjunction, vector addition is an operation possessing the properties of associativity and commutativity. Were this not so, vector addition could not be used to represent conjunction.

Applying the superpositional representation of conjunction to the case at hand:

DEFINITION 2.2.2.2: Suppose S is a set of symbolic structures and F/R is a role decomposition of S with fillers F and roles R. Suppose further that ψ_b is a connectionist representation of the filler/role bindings:

$$\psi_b : \{f/r \mid f \in F, r \in R\} \rightarrow V$$

where V is a vector space. Then $\psi_{F/R}$, the connectionist representation of S induced by F/R, the superpositional representation of conjunction, and ψ_b, is

$$\psi_{F/R} : S \rightarrow V; s \mapsto \sum_{(f,r) \in \beta(s)} \psi_b(f/r)$$

The use of vector addition to represent conjunction has pervasive implications for the faithfulness of representations. If the representations of $a \wedge b$ and $c \wedge d$ are to be distinguishable, then **a**+**b** and **c**+**d** must be different. This constrains the possible patterns **a**, **b**, **c** and **d** that can represent a, b, c and d. It will be guaranteed that **a**+**b** and **c**+**d** will be different if the vectors **a**, **b**, **c** and **d** are all *linearly independent*: no one can be expressed as a weighted superposition of the others. In order to guarantee the faithfulness of representations, it will often be necessary to impose the restriction of linearly independent representing patterns for the constituents. This restriction is an expensive one, however, since to have n linearly independent patterns one must have at least n nodes in the network.

2.2.3. Connectionist representation of variable binding

It remains to consider the representation of filler/role bindings; this section introduces the tensor product representation.

The tensor product representation of a value/variable binding is quite simple to define (see Figure 2). To bind a filler f to a role r we first represent f as a pattern of activity $\mathbf{f}$ over a set of "filler" units $\{\tilde{f}_\phi\}$ and represent r as a pattern of activity $\mathbf{r}$ over a set of "role" units $\{\tilde{r}_\rho\}$. The binding f/r is represented by a pattern of activity $\mathbf{f/r}$ over a set of "binding" units $\{\tilde{b}_{\phi\rho}\}$, of which there is one for each pair of filler and role units. The activity of the binding unit $\tilde{b}_{\phi\rho}$ is the activity of the filler unit $\tilde{f}_\phi$ in the pattern $\mathbf{f}$ times the activity of the role unit $\tilde{r}_\rho$ in the pattern $\mathbf{r}$.

This procedure is readily characterizable in vector space terminology. The representation of the role r is a vector $\mathbf{r}$ in a vector space V_R. V_R is a real vector space with dimension equal to the number of units $\tilde{r}_\rho$. The representation of the filler f is a vector $\mathbf{f}$ in a vector space V_F, a real vector space with dimension equal to the number of units $\tilde{f}_\phi$. The representation of the binding f/r is the *tensor product* vector $\mathbf{f/r} = \mathbf{f} \otimes \mathbf{r}$ in the tensor product vector space $V_B = V_F \otimes V_R$ (see the Appendix in Section 5). V_B is a real vector space with dimension equal to the product of the dimensions of V_F and V_R. The components of the vector $\mathbf{f/r}$ are related to the components of $\mathbf{f}$ and $\mathbf{r}$ as follows. Each filler unit $\tilde{f}_\phi$ corresponds to a vector $\hat{\mathbf{f}}_\phi$ in V_F (the vector representing the pattern of activity in which that unit has activity 1 and all other units have activity zero). The complete set of vectors $\{\hat{\mathbf{f}}_\phi\}$ forms the distinguished basis for V_F and any vector $\mathbf{f}$ in V_F can be expressed in terms of this basis as a sequence of real numbers; these are the activities of all the units in the pattern corresponding to $\mathbf{f}$. Exactly the same story holds for the roles. Then the tensor product space $V_B = V_F \otimes V_R$ has as a basis the set of vectors $\{\hat{\mathbf{b}}_{\phi\rho} = \hat{\mathbf{f}}_\phi \otimes \hat{\mathbf{r}}_\rho\}$. The $\phi\rho$ component ($b_{\phi\rho} = f/r_{\phi\rho}$) of the vector $\mathbf{b} = \mathbf{f/r} = \mathbf{f} \otimes \mathbf{r}$ representing the binding is the product of the ϕ component of $\mathbf{f}$ (f_ϕ) and the ρ component of $\mathbf{r}$ (r_ρ):

$$b_{\phi\rho} = f/r_{\phi\rho} = f_\phi r_\rho$$

DEFINITION 2.2.3.1: Let F/R be a role decomposition of S. Let ψ_F and ψ_R be connectionist representations of the fillers and roles:

$$\psi_F : F \to V_F$$
$$\psi_R : R \to V_R$$

Then the *tensor product representation of the filler/role bindings* induced by ψ_F and ψ_R is the mapping:

$$\psi_b : \{f/r \mid f \in F, r \in R\} \to V_F \otimes V_R ; f/r \mapsto \psi_F(f) \otimes \psi_R(r)$$

Figure 3 shows an example specially chosen for visual transparency. Consider an application to speech processing, and imagine that we are representing the amount of energy in a particular formant over time. For the roles here we take a series of time points and for the fillers the amount of energy in the formant. In Figure 3, the roles are represented as patterns of activity over five units. Each role r_ρ is a time point and is represented as a peaked pattern centered at unit ρ; the Figure shows the case $\rho = 4$. Each filler f_ϕ is an energy level; in Figure 3 this is represented as a pattern of activity over four units: a single peak centered at the energy level being represented. The binding pattern is a two-dimensional peak centered at the point whose x- and y-coordinates are the time and energy values being bound together.

The example of Figure 3 is visually transparent because of the simple geometrical structure of the patterns. Of course there is nothing in the binding mechanism itself that requires this. The distributed representation of roles and fillers can be arbitrary patterns and in general the tensor product of these patterns will be even more visually opaque than are the patterns for the roles and fillers: see Fig. 4. However the mathematical simplicity of tensor product binding makes the general case as easy to analyze as special cases like that of Figure 3.

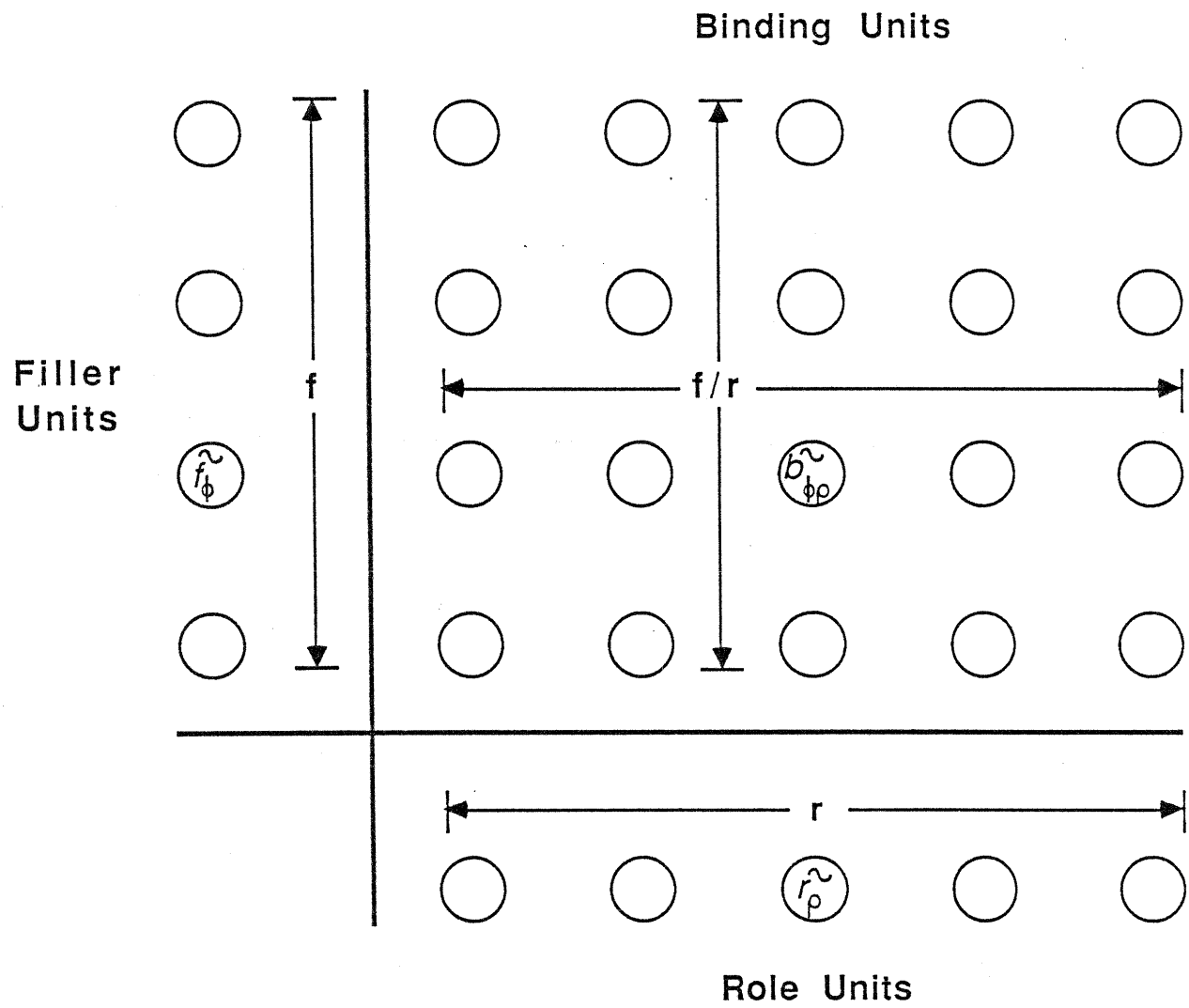


Figure 2. The tensor product representation for filler/role bindings.

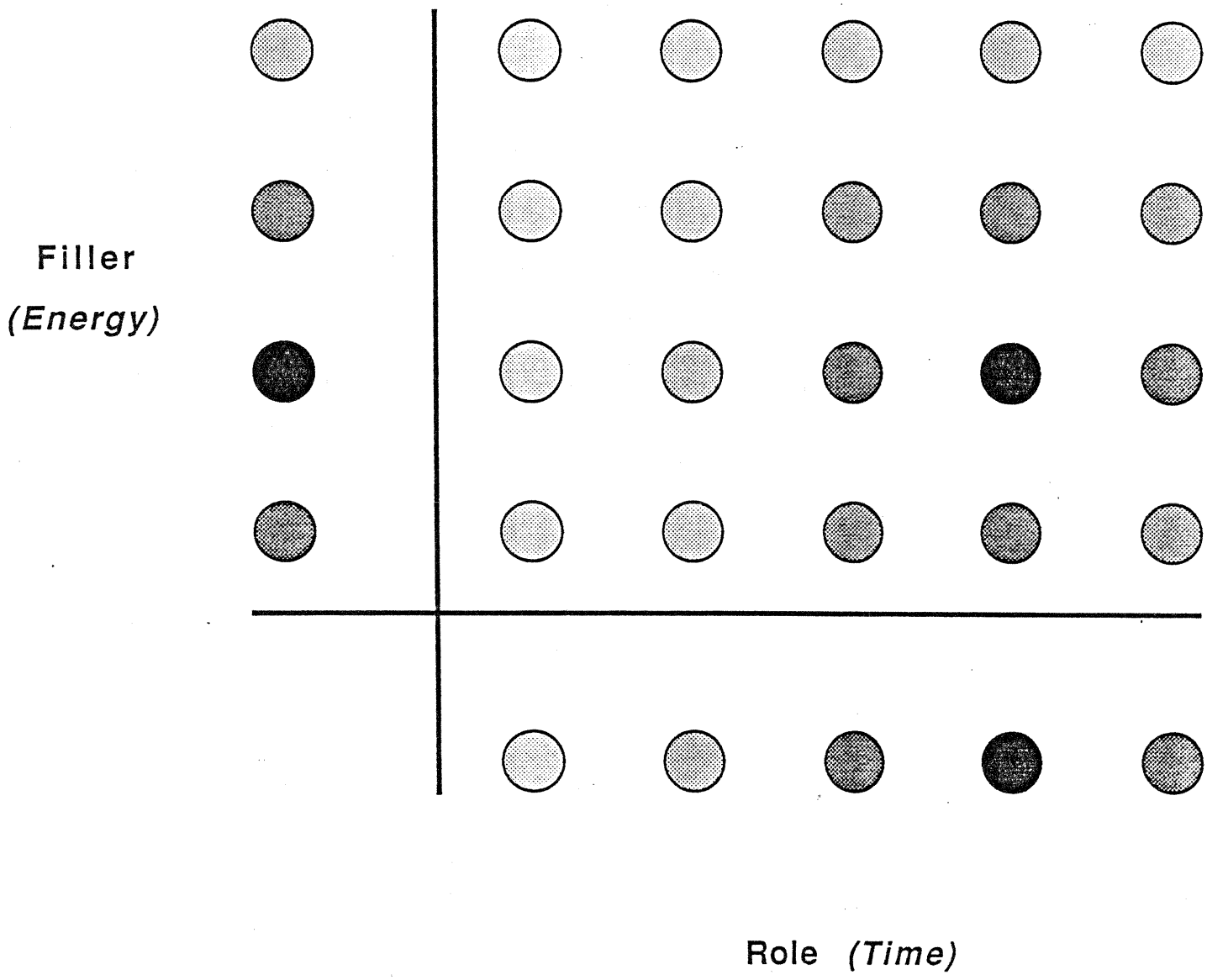


Figure 3. A visually transparent example of the tensor product representation of a role/filler binding. Darker units are more active.

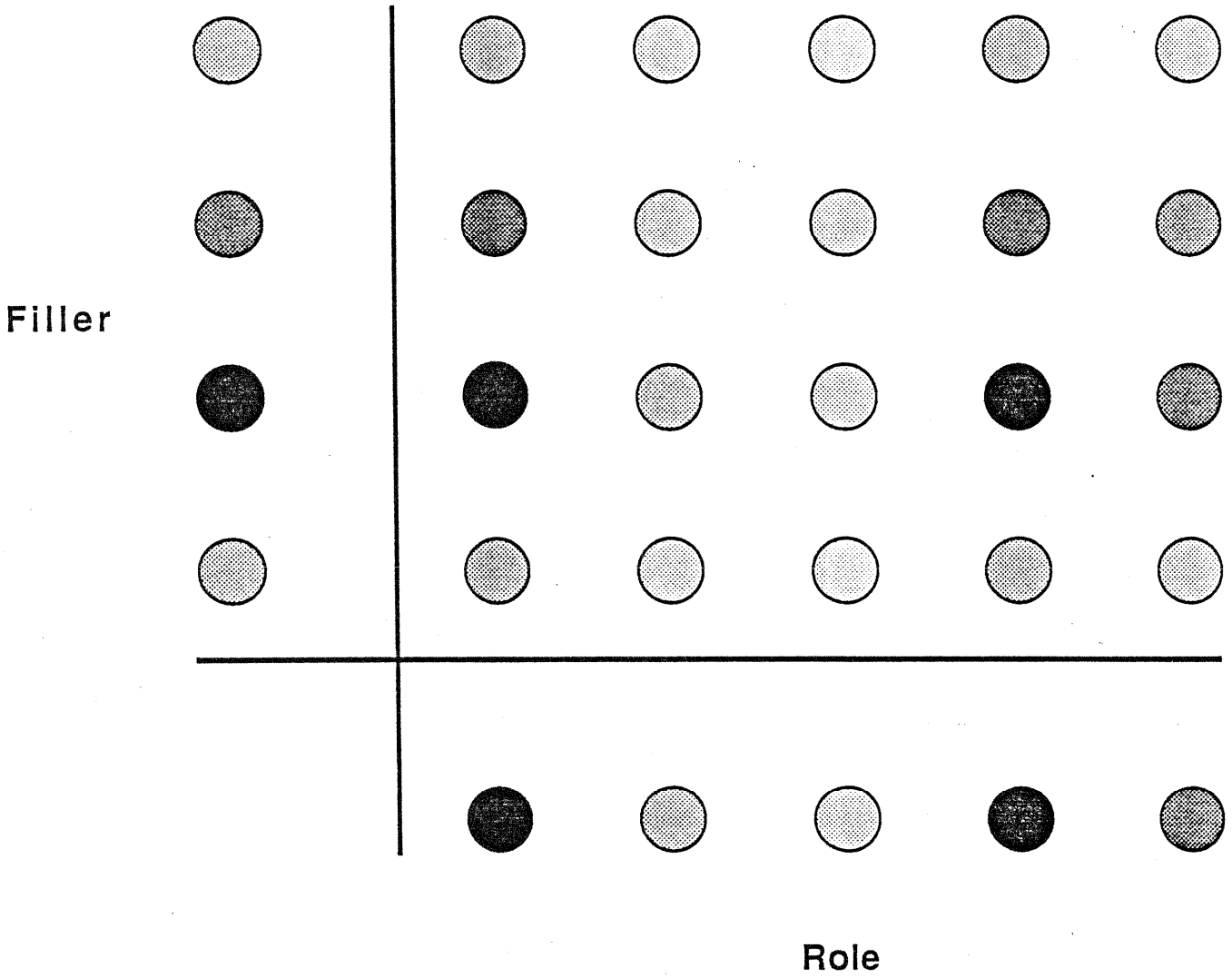


Figure 4. A generic example of the tensor product representation of a role/filler binding.

2.2.4. Tensor product representation

Putting together the previous representations, we have:

DEFINITION 2.2.4.1: Let F/R be a role decomposition of S, and let ψ_F and ψ_R be connectionist representations of the fillers and roles. Then the corresponding *tensor product representation* of S is

$$\psi: S \rightarrow V_F \otimes V_R;\ s \mapsto \sum_{(f,r) \in \beta(s)} \psi_F(f) \otimes \psi_R(r)$$

If we identify s with the conjunction of the bindings it contains, and if we let $\mathbf{f} = \psi_F(f)$ and $\mathbf{r} = \psi_R(r)$, we can write this in the more transparent form:

$$\psi(\bigwedge_i f_i/r_i) = \sum_i \mathbf{f}_i \otimes \mathbf{r}_i$$

The interpretation of the activity of binding units in the tensor product representation depends on the interpretation of the feature and role units. If the filler or role representations are local, then each unit individually represents a particular filler or role. In the filler or role representation is distributed, the activation of an individual node may indicate the presence of an identifiable feature in the entity being represented. This was true of the example given in Section 1.1: each of the output units represents a phonetic feature in the phonetic segment output by the network. For expository simplicity, we can consider a local representation to be one where a given "feature" is present in exactly one represented object, and a given object possesses exactly one "feature." Then if the binding unit $\tilde{b}_{\phi\rho}$ is active in the tensor product representation of a structure s, the interpretation is that the feature represented by $\tilde{f}_\phi$ is present in a filler of a role possessing the feature $\tilde{r}_\rho$. In this sense, $\tilde{b}_{\phi\rho}$ *represents the conjunction of the features represented by* $\tilde{f}_\phi$ *and* $\tilde{r}_\rho$.[1] By iterating the tensor product representation, we can produce conjunctions of more than two features; this will be considered in Section 3.7.3.

2.3. Previous representations and special cases of tensor product representation

Section 3 analyzes the general properties of the tensor product representation. Before proceeding to this general analysis, it is useful to examine a number of special cases of the tensor product representation because these turn out to include nearly all previous cases of connectionist representations of structured objects.

2.3.1. Role decompositions

The examples of previous connectionist representations of structured objects that we shall consider employ only a few role decompositions.

DEFINITION 2.3.1.1: Suppose S is the set of strings of length no more than n from an alphabet A. Let $F = A$, and let $R = \{r_i\}_{i=1}^n$, where r_i is the role "occupies the i^{th} position in the string". Then F/R is the *positional role decomposition* of S.

This is the example given above in Section 2.2, in which the string *aba* is represented by bindings $\{a/r_1, b/r_2, a/r_3\}$. This decomposition is finite, has single-valued roles, and is faithful. This decomposition is the most obvious one, and the one most often used in previous connectionist systems.

---

1. For a more precise formulation, consider a simple case where the activity of unit $\tilde{f}_\phi$ is 1 or 0, and indicates the truth value of the proposition "there exists x among the represented objects such that the predicate $\tilde{f}_\phi$ holds of x"; and suppose $\tilde{r}_\rho$ can be similarly interpreted. Then $\tilde{b}_{\phi\rho}$ indicates the truth value of the proposition "there exists x among the represented objects such that both predicates $\tilde{f}_\phi$ and $\tilde{r}_\rho$ hold of x."

The positional decomposition has an obvious extension to the case of finite strings of arbitrary length, where the set of roles becomes infinite; I will treat this as the case of the above definition with $n = \infty$. In the infinite case the decomposition is still faithful, still has single-valued roles, and is still finite, since the strings are all of finite length. The infinite case will be used later to explore saturation of the tensor product representation.

There is a less obvious role decomposition of strings that is used, as we shall shortly see, to great advantage by Rumelhart and McClelland (1986):

> DEFINITION 2.3.1.2: Suppose S is the set of strings of length no more than n from an alphabet A. Let $F = A \cup \{<, >\}$, where $<$ and $>$ are two new symbols meaning "left string boundary" and "right string boundary" respectively. Let $R = \{r_{x\_y} \mid x \in F, y \in F\}$, where r_{x_y} is the role "is immediately preceded by x and immediately followed by y". F/R is a role decomposition of S called the *1-neighbor context decomposition.*

Under this decomposition, the string *aba* becomes the set of bindings $\{a/r_{<\_b}, b/r_{a\_a}, a/r_{b\_>}\}$. This decomposition does not have single-valued roles and is not faithful if $n \geq 4$ (the strings a^3 and a^4 can't be distinguished). There is an obvious generalization to the k-neighbor context decomposition: this is faithful if $n < 2k+2$.[2]

There are also obvious generalizations of the 1-neighbor context decomposition to differing size contexts on the left and right. A special case is the representation of pairs, say strings with $n = 2$, where the roles are $R = \{r_{\_x} \mid x \in F\}$: the right-neighbor context. The pair *ab* is represented as the single binding $a/r_{\_b}$. This role decomposition, we shall see, is used in a powerful technique called *conjunctive coding.*

While it is true that the positional role decomposition is more faithful than context decompositions for the representation of a *single* structure, it turns out that if multiple structures are to be simultaneously represented, the positional decomposition can be *less* faithful than the context decomposition. Suppose we are to represent the conjunction of *ab* and *cd* by superimposing the representation of the two pairs. What gets represented is the union of the binding sets of the two structures. In the case of positional roles, this union is $\{a/r_1, b/r_2, c/r_1, d/r_2\}$; now it is impossible to distinguish what is being represented from the conjunction of *ad* and *cb*. However, with the right-neighbor context decomposition, the union of the binding sets is $\{a/r_{\_b}, c/r_{\_d}\}$, which is not at all confusable with the conjunction of *ad* and *cb*. With context decompositions confusions can of course also result; these decompositions are not even faithful for representing single structures, when the same fillers appear multiple times in the same context.

An additional virtue of context decompositions is that they give rise to connectionist representations that give the network direct access to the kind of information needed to capture the regularities in many context-sensitive tasks; we shall discuss this below for the specific example of the Rumelhart and McClelland (1986) model..

2.3.2. Connectionist representations

Having discussed a few of the role decompositions that have been used in connectionist representations of structures, we can now consider a number of examples of such representations. These are grouped according to the degree of locality in the representations of roles and fillers; we therefore start by distinguishing local and distributed connectionist representations in general, and then examine the degree of locality of various exisiting representations of structured objects.

---

2. This decomposition gives the initial and final substrings of length up to $2k$, and all internal substrings of length $2k+1$. These substrings uniquely determine strings of length no more than $2k+1$. The strings a^{2k+1} and a^{2k+2} can't be distinguished, however, so the decomposition is not faithful if $n > 2k+1$.

2.3.2.1. Local and distributed representations

Local representations dedicate an individual processor to each item represented. In terms of the vector space of network states, these individual processors correspond to the members of the distinguished basis. Thus:

DEFINITION 2.3.2.1.1: Let ψ be a connectionist representation of a set X in a vector space V with distinguished basis $\{\hat{v}_i\}$. ψ is a *local representation* iff it is a one-to-one mapping of the elements of X onto the set of basis vectors $\{\hat{v}_i\}$.

A connectionist representation that is not a local representation is a *distributed representation.*

2.3.2.2. Purely local representations of symbolic structures

The first special case of the tensor product representation is the most local one.

DEFINITION 2.3.2.2.1: Let $\psi_{F/R}$ be the tensor product representation of S induced by a role decomposition F/R of S and two connectionist representations ψ_F and ψ_R. Then $\psi_{F/R}$ is a *purely local tensor product representation* if ψ_F and ψ_R are both local representations.

This case is illustrated for the representation of strings in Fig. 5. If the filler and role patterns both involve the activity of only a single processor, then the tensor product pattern representing their binding will also involve only a single unit. In other words, if ψ_F and ψ_R are both local representations, then so too is $\psi_{F/R}$.

Purely local tensor product representations have been used along with the positional role decomposition of strings in many connectionist models; for example:

- As was already mentioned in Section 1.1 and illustrated in Fig. 1, NETtalk uses the purely local representation of Fig. 5 to represent seven-letter input strings.
- The interactive activation model of the perception of letters in words (McClelland & Rumelhart, 1981, Rumelhart & McClelland, 1982) uses the representation shown in Fig. 5 for representing four-letter strings, at its intermediate or "letter" level of representation. This too is a purely local tensor product representation.
- The TRACE model of speech perception (McClelland & Elman, 1986) uses a purely local representation of strings of phonemes, although some of the positional roles involve overlapping time intervals.
- Fanty's (1985) parser uses a purely local tensor product representation involving a positional role decomposition of trees.
- Feldman's (1985) connectionist system for visual processing uses a representation that includes the tensor product of a local representation for visual features (including color, size, and shape) and a local representation for position in the visual field.

2.3.2.3. Semi-local representations of symbolic structures

The next most local special case is this.

DEFINITION 2.3.2.3.1: Let $\psi_{F/R}$ be the tensor product representation of S induced by a role decomposition F/R of S and two connectionist representations ψ_F and ψ_R. If ψ_F is a distributed representation and ψ_R is a local representation then $\psi_{F/R}$ is a *semi-local tensor product representation* or a *role register representation.*

If the filler representation is a distributed pattern and the role representation involves the activity of a single unit, the result is a copy of the filler pattern in a pool of units dedicated to the role: see Fig. 6.

Semi-local tensor product representations have been widely used in conjunction with positional role decompositions:

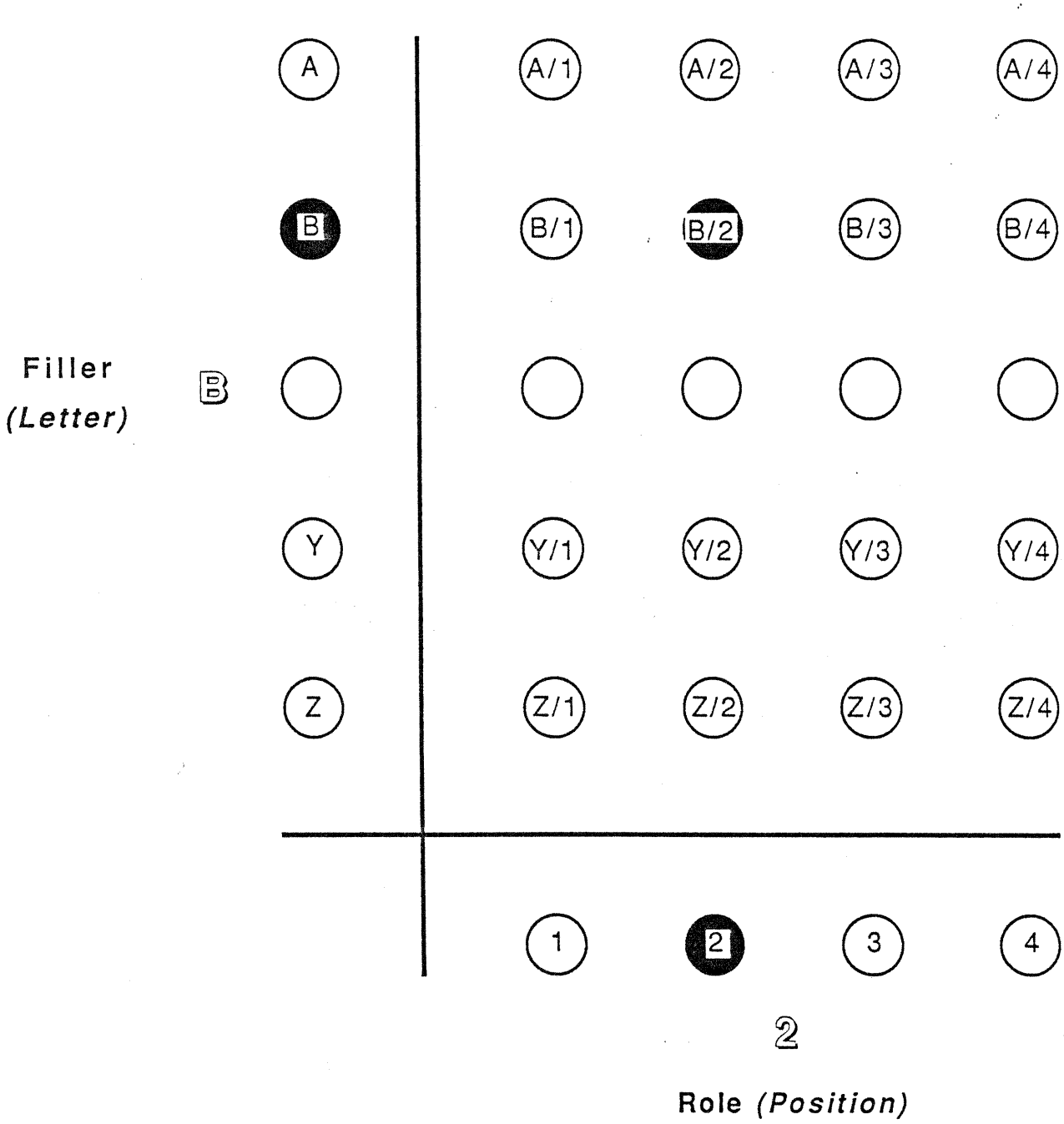


Figure 5. A purely local tensor product representation of four-letter strings.

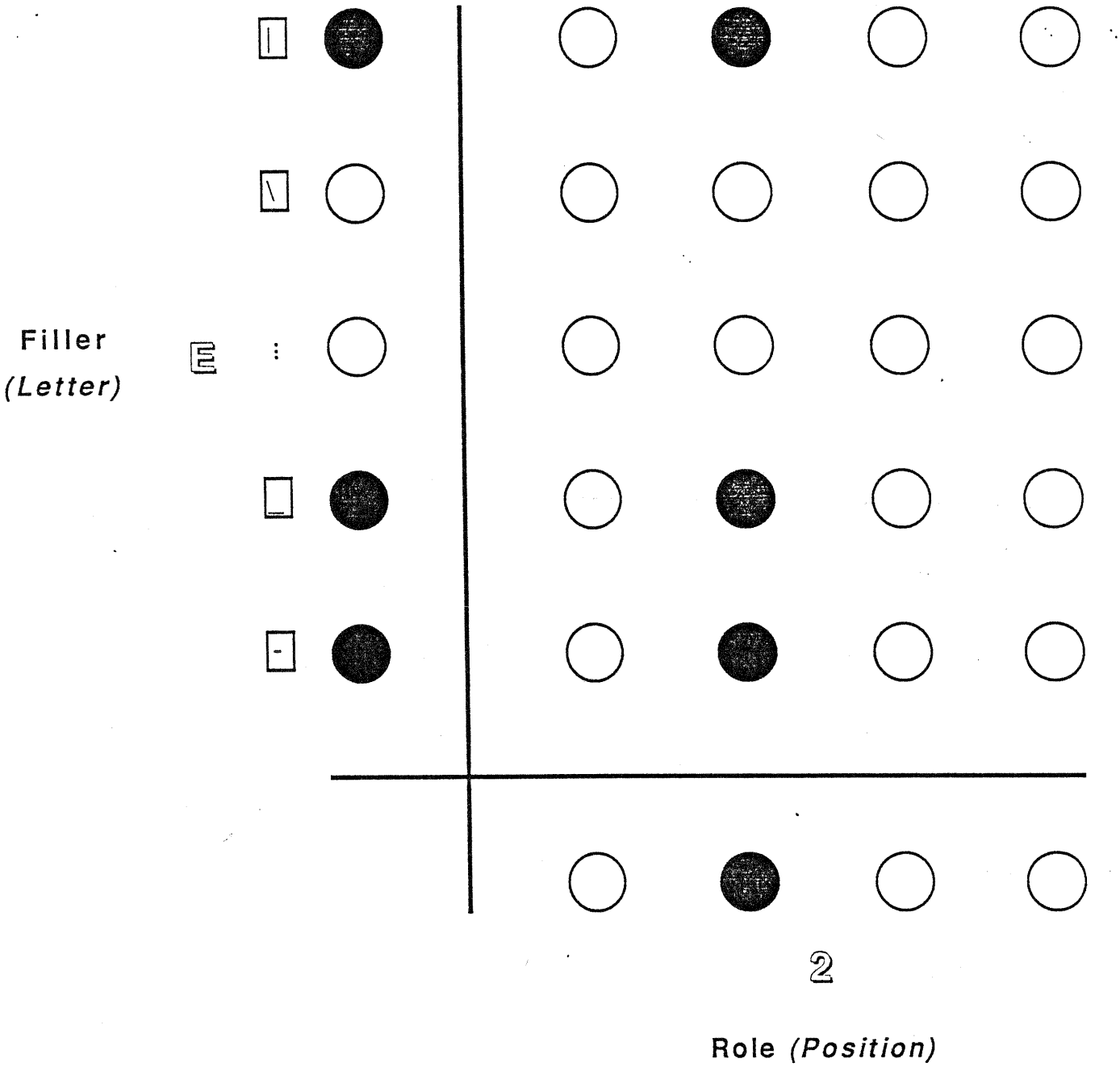


Figure 6. A semi-local tensor product representation of four-letter strings.

- The letter perception model (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982) uses a semi-local representation of letters at its lowest or "letter feature" level; this is the example shown in Fig. 6. A set of units is dedicated to the representation of the first letter's features; a letter is represented as a pattern of activity over these units, where each unit indicated whether a particular line segment is or is not present in the first letter. There were identical copies of this "first letter register" for the second, third, and fourth letter.
- An early version of NETtalk (Charles Rosenberg, unpublished communication, 1985) used a semi-local representation for the input string: the i^{th} letter was represented by a pattern of activity over a set of units dedicated to the i^{th} position, and each unit indicated whether a particular orthographic feature (eg., closed loop, ascending line) was present in that letter.
- In Hinton's (1981b) semantic net model, relationships of the form $R(x, y)$ (eg., *has_color*(*clyde*, *grey*)), are represented by placing three distributed patterns of activity representing the fillers of the roles R, x, and y in pools dedicated to those roles. (There is an additional pool as well.)
- The model of Riley & Smolensky (1984) that answers qualitative questions about a fixed simple electric circuit also uses a semi-local representation. Each role is a circuit variable (eg., the current, or the resistance of one of the resistors) and the fillers are the qualitative values *increases, decreases, stays_constant*. Each filler is represented as a small pattern in a pool of two units dedicated to the corresponding role.
- Touretzky and Hinton's (1985) connectionist production system interpreter uses productions with two symbolic triples on the condition side; each triple is represented by a pattern of activity in a separate pool of units. (The representation of the triples themselves are considered in the next section.)
- The McClelland and Kawamoto (1986) model that learns to assign case to the nouns appearing in a standard sentence frame uses a semi-local input representation. Each input is an instance of the frame: *The* N_1 V *the* N_2 *with the* N_3. The roles here are the three nouns and the verb, and each filler is represented by a pattern of activity in a pool of units dedicated to the corresponding role.

2.3.2.4. Fully distributed representations of symbolic structures

Now we come to the most distributed case:

DEFINITION 2.3.2.4.1: Let $\psi_{F/R}$ be the tensor product representation of S induced by a role decomposition F/R of S and two connectionist representations ψ_F and ψ_R. If ψ_F and ψ_R are both distributed representations, then $\psi_{F/R}$ is a *fully distributed tensor product representation.*

Examples of fully distributed representations are few.

- A visually transparent example of a fully distributed tensor product representation using the positional role decomposition was given in Fig. 3. The patterns representing roles here are examples of *coarse coding* representations described in Hinton, McClelland, & Rumelhart (1986). It is traditional to focus on the numerous positions (roles) that activate a particular role unit (its "receptive field"); the formulation here focuses on the numerous role units activated by a particular positional role. These are merely two perspectives on the many-to-many mapping between positions and units.
- The McClelland and Kawamoto (1986) model mentioned earlier can be viewed as using a fully distributed representation of the output. Each output is a set of bindings of noun fillers to the case-frame slots of the verb. This output can be viewed as having roles like *loves-agent, loves-patient, eat-instrument, break-patient*, and so on; these roles can in turn be viewed as structured objects with two sub-roles: *verb* and *case-role*. The patterns representing the overall roles are the tensor product of a distributed pattern representing the verb (built from semantic verb features) and a local representation of the case-role. The representation of the overall roles is thus semi-local. The representation of the

output as a whole is the tensor product of this distributed (albeit semi-local) representation of the roles and a distributed representation of the fillers (built of semantic noun features of nouns). This is an example of the kind of iterated tensor product representation that will be discussed in Section 3.7.3. Because of this iterated structure, the output units in this model represent three-way conjunctions of features for nouns, verbs, and semantic roles. (The "features" of semantic roles used in the model are of the local type mentioned in Section 2.3.2.4: they are in one-to-one correspondence with the semantic roles. A more distributed version of this model would employ real features of semantic roles, where each semantic role is a distributed pattern of features. Then the roles in the output as a whole would have fully distributed representations instead of semi-local ones.)

- An example of a fully distributed representation employing the 1-neighbor context decomposition is the Rumelhart and McClelland (1986) model that learns to form the past tense of English verbs; see Fig. 7. In this model, elements of S are strings of phonetic segments. The word "we" corrsponds to the string $[w][i]$ which has the bindings $\{w/r_{<\_i}, i/r_{w\_>}\}$. The representation of this string is thus

$$\mathbf{w} \otimes \mathbf{r}_{<\_i} + \mathbf{i} \otimes \mathbf{r}_{w\_>}$$

The filler vectors (eg. $\mathbf{w}$) are distributed patterns over a set of units representing phonetic features (eg., *rounded*, *front*, *stop*). The role vectors (eg. $\mathbf{r}_{<\_i}$) are patterns of activity over a set of units each of which represents the conjunction of a feature of the left neighbor (<) and a feature of the right neighbor (w). (In this model, both < and > possess the single feature *word_boundary*.) As in the previous example, since the roles are composite objects, they are in fact themselves further decomposed into sub-roles. The pair of phonetic segments defining the context is decomposed using the right-neighbor context decomposition, and the pattern representing the role r_{a_b} is the tensor product of patterns of phonetic features for $[a]$ and $[b]$. To reduce the number of units in the network, many of the units arising in this further decomposition of the roles were in fact discarded. The overall structure of the representation of the roles can still be productively viewed as a tensor product from which some units have been thrown away.

- Touretzky and Hinton's (1985) representation of triples of letters can be viewed as the same sort of third-order tensor product as in the last example, but in which even more binding units are discarded. Their representation involves a set of units $\alpha = 1, \cdots, N$, each of which responds to three groups of letters $(L_\alpha^{(1)}, L_\alpha^{(2)}, L_\alpha^{(3)})$: unit α is active in the representation of $(l^{(1)}, l^{(2)}, l^{(3)})$ iff $l^{(i)} \in L_\alpha^{(i)}$ for $i = 1, 2$, and 3. To relate this to the tensor product representation, imagine three pools of N units, one pool for each letter in the triple. In the i^{th} pool, unit $\tilde{u}_\alpha^{(i)}$ is active iff $l^{(i)} \in L_\alpha^{(i)}$: each letter is represented by a pattern in the corresponding pool. Create a binding unit for each triple of units, one from each pool; it is active iff the corresponding three units are active. This is the tensor product representation of the triples induced by the 1-neighbor context decomposition, with the roles further decomposed by the right-neighbor context decomposition, as in the previous example. Now if we throw out all the binding units except the N "diagonal" ones corresponding to $(\tilde{u}_\alpha^{(1)}, \tilde{u}_\alpha^{(2)}, \tilde{u}_\alpha^{(3)})$, we get Touretzky and Hinton's representation.

Having mentioned Rumelhart and McClelland's (1986) use of context decompositions, it is worth elaborating on remarks of Section 2.3.1 about the advantages of context decompositions over simpler positional decompositions. Many regularities in language depend on the context in which a constituent finds itself, rather than its absolute position. This is particularly true in phonology; the regularities that must be learned in order to form the past tenses of English verbs typically depend on neighbor relations: for example, the rule for "regular" verbs involves replacing $x/r_{y\_>}$ by the bindings $\{x/r_{y\_d}, d/r_{x\_>}\}$ if x has feature *voiced*, and by the binding $\{x/r_{y\_t}, t/r_{x\_>}\}$ if x does not have feature *voiced*. Thus the featural representation of phonetic segments together with the context decomposition of the string provides the network with just the kind of representation of phonetic strings that it needs in order to learn the regularities characterizing this task.

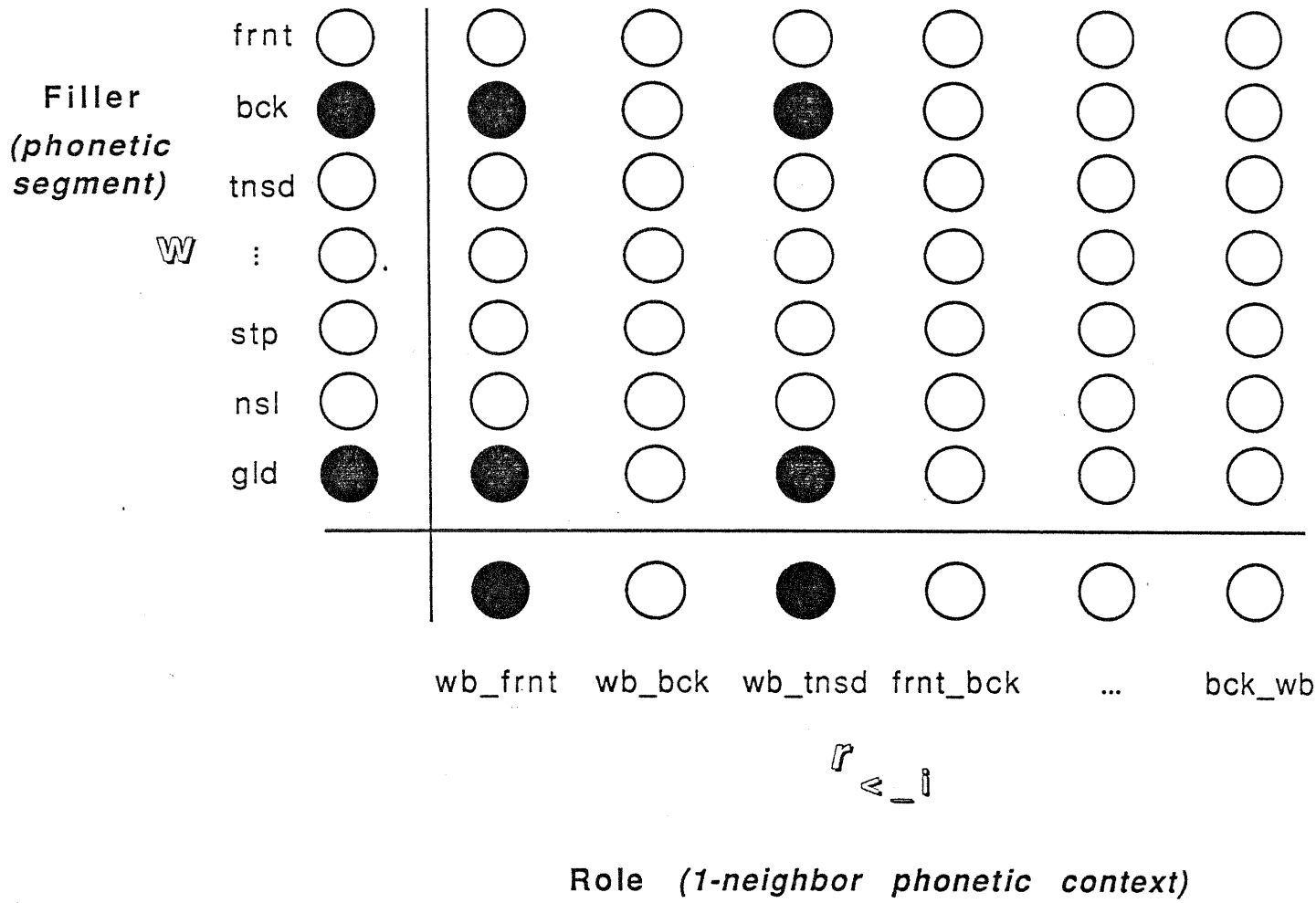


Figure 7. The principal representation used in Rumelhart and McClelland (1986) for phonetic strings. The abbreviations used are *wb = word_boundary, frnt = front, bck = back, tnsd = tensed, stp = stop; nsl = nasal; gld = glide.*

2.4. Relations among purely local, semi-local, and fully distributed representations

Purely local, semi-local and fully distributed representations look quite different on the surface. Are they really as different as they seem? According to the definitions, the only difference is the relation between the representation vectors and the distinguished basis vectors indicating the individual processing units. Does this really matter?

As discussed at length in Smolensky 1986b, the answer depends on the dynamics driving the connectionist network, and not solely on the representations themselves. If the dynamics is linear, so that the activity of every unit is exactly a weighted sum of the activity of its neighbors in the network, then networks using purely local, semi-local and fully distributed representations will have exactly isomorphic behavior, subject to a few qualifications. Under the linear transformations that map these three cases into each other, locality is not preserved, so that local damage to the networks will have different effects, and what can be learned via the usual local connectionist learning procedures will be different. If the network contains nonlinear units, the isomorphism fails. Also, assuming finite networks, the local case accomodates only a fixed, finite set of fillers and roles; the semi-local case allows an unlimited number of fillers but only a finite set of roles. The fully distributed case, however, can accomodate an infinite set of fillers and roles in a finite network, as will be discussed in Section 3.2.

3. Tensor product representation: Properties

In Section 2 I defined the tensor product representation and showed how a number of representations used in previous connectionist models are various special cases of the tensor product representation. In this section I will discuss a number of general properties of this representation. The case of interest is fully distributed representation; while most of the results apply also to the more localized special cases, in these cases they become rather trivial.

3.1. Unbinding

Until now I have ignored a crucial and obvious question: if the representations of all the variable bindings necessary for a particular structure are superimposed on top of each other in a single set of binding units, how can we be sure the binding information is all kept straight? In this section we explore this question via the *unbinding* process: taking the tensor product representation for a complex structure and extracting from it the filler for a particular role. Under what conditions can we perform this unbinding operation accurately?

THEOREM 3.1.1: Let $\psi_{F/R}$ be a tensor product representation induced by a role decomposition with single-valued roles. Suppose the vectors representing the roles bound in a structure s are all linearly independent. Then each role can be unbound with complete accuracy: for each bound role r_i there is an operation which takes the vector $\psi_{F/R}(s)$ representing s into the vector $\mathbf{f}_i$ representing the filler f_i bound to r_i.

PROOF: If the role vectors $\{\mathbf{r}_i\}$ being used are linearly independent, then they form a basis for the subspace of V_R that they span. To this basis there corresponds a *dual basis* $\{U_i\}$ (see Section 5.1 of the Appendix). Each element in this dual basis is a linear mapping from V_R into the real numbers with the property that

$$U_i(\mathbf{r}_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

That is, U_i maps the single role vector $\mathbf{r}_i$ to 1 and all other role vectors to 0. If we make use of the canonical inner product on the vector space V_R, then the dual vector U_i can be expressed as the operation of taking the

inner product with respect to some vector $\mathbf{u}_i$ in V_R:

$$U_i(\mathbf{v}) = \mathbf{v}\cdot\mathbf{u}_i$$

for all v in V_R; this is shown in Theorem 5.1.5 of the Appendix. Call $\{\mathbf{u}_i\}$ the *unbinding vectors* for roles $\{r_i\}$. Now let s be the tensor product representation of a structure in which the roles $\{r_i\}$ are bound to the fillers $\{f_i\}$. Then we can extract f_i from s, or unbind r_i, by taking a partial inner product of s with the unbinding vector $\mathbf{u}_i$:

$$\mathbf{s}\cdot\mathbf{u}_i = (\sum_j \mathbf{f}_j \otimes \mathbf{r}_j)\cdot\mathbf{u}_i := \sum_j \mathbf{f}_j(\mathbf{r}_j\cdot\mathbf{u}_i) = \sum_j \mathbf{f}_j \delta_{ij} = \mathbf{f}_i$$

∎

DEFINITION 3.1.2: The procedure defined in the preceding proof is the *exact unbinding procedure*.

Let unbinding of role r_i be performed as in the previous proof, but in place of the unbinding vector $\mathbf{u}_i$ use the role vector $\mathbf{r}_i$ itself. This is the *self-addressing unbinding procedure*.

Unlike the exact binding procedure, the self-addressing unbinding procedure is defined for any set of role vectors, even if they are not linearly independent.

THEOREM 3.1.3: Suppose the self-addressing procedure is used to unbind roles. If the role vectors are all orthogonal, the correct filler pattern will be generated, apart from an overall magnitude factor. Otherwise, the pattern generated will be a weighted superposition of the pattern of the correct filler, $\mathbf{f}_i$, and all the other fillers, $\{\mathbf{f}_j\}_{j\neq i}$. In this superposition, the weight of each erroneous pattern $\mathbf{f}_j$ relative to the correct pattern $\mathbf{f}_i$, the *intrusion of role j into role i*, is

$$\frac{\mathbf{r}_i\cdot\mathbf{r}_j}{\|\mathbf{r}_i\|^2} = \cos\theta_{ji}\,\frac{\|\mathbf{r}_j\|}{\|\mathbf{r}_i\|}$$

where θ_{ji} is the angle between the vectors $\mathbf{r}_j$ and $\mathbf{r}_i$.

PROOF:

$$\mathbf{s}\cdot\mathbf{r}_i = (\sum_j \mathbf{f}_j \otimes \mathbf{r}_j)\cdot\mathbf{r}_i = \sum_j \mathbf{f}_j(\mathbf{r}_j\cdot\mathbf{r}_i) = (\mathbf{r}_i\cdot\mathbf{r}_i)\mathbf{f}_i + \sum_{j\neq i}(\mathbf{r}_j\cdot\mathbf{r}_i)\mathbf{f}_j$$

In this weighted superposition, the ratio of the coefficient of each incorrect filler $\mathbf{f}_j$ to that of the correct filler $\mathbf{f}_i$ is

$$\frac{\mathbf{r}_j\cdot\mathbf{r}_i}{\mathbf{r}_i\cdot\mathbf{r}_i}$$

The denominator is $\|\mathbf{r}_i\|^2$ and the numerator is $\cos\theta_{ji}\,\|\mathbf{r}_j\|\,\|\mathbf{r}_i\|$, giving the claimed result. ∎

Since the tensor product binding representation is symmetric between role and filler, the unbinding procedures given above can also be used to retrieve a role pattern from the filler pattern to which it is bound. While there is no asymmetry between role and filler in the representation of a single binding, an assymmetry may however result from the combination of many bindings in the representation of a structured object. For while role decompositions often involve single-valued roles, it is uncommon to encounter single-valued *fillers*. Thus while there will often be a unique filler indexed by a given role, there will often be several roles associated with a single filler. In the latter case, an unbinding that is performed using the filler pattern as an index will generate the superposition of all the role vectors bound to that filler.

3.2. Graceful saturation

Like a digital memory with n registers, a connectionist system that uses n pools of units to represent a structure with n roles has a discrete saturation point. Structures with no more than n roles filled can be represented precisely, but for larger structures some information must be omitted entirely. The form of saturation characteristic of connectionist systems (eg., connectionist memories) is less discrete than this; this is one aspect of the "graceful degradation" advertised for connectionist systems.

Aspects of the graceful degradation notion can be formally characterized as follows.

DEFINITION 3.2.1: Let F/R be a role decomposition of S. A connectionist representation ψ of S has *unbounded sensitivity* with respect to F/R if for arbitrarily large n,

$$\psi(\bigwedge_{i=1}^{n} f_i/r_i)$$

varies as f_i varies, for each $i = 1, 2, \cdots, n$.

If for sufficiently large n the representation of structures containing n filler/role bindings is not faithful, then ψ *saturates*.

If ψ saturates and has unbounded sensitivity then ψ possesses *graceful saturation*.

The tensor product representation, unlike local and role register representations, can exhibit graceful saturation. To show this, I now consider an example that also illustrates how fully distributed tensor product representations can be used to represent an infinite number of roles in a finite-dimensional vector space corresponding to a finite connectionist network.

THEOREM 3.2.2: Suppose S is the set of finite strings with unbounded length, and let $\{r_i\}_{i=1}^{\infty}$ be the positional roles. Let the vectors $\{\mathbf{r}_i\}_{i=1}^{\infty}$ be unit vectors in N-dimensional space, randomly chosen according to the uniform distribution. Then this tensor product representation possesses graceful saturation. The expected value of the magnitude of the intrusion of role i into role j is proportional to $N^{1/2}$. The number of bindings n that can be stored before the expected total magnitude of intrusions equals the magnitude of the correct pattern increases as $N^{1/2}$.

PROOF: Since all role vectors have unit length, the expected value of the magnitude of the intrusion is

$$\mathbf{EI} = \frac{1}{V_{N-1}} \int_0^{\pi} |\cos\theta_{ji}| V(\theta_{ji})\, d\theta_{ji}$$

Here V_{N-1} is the $N-1$-dimensional volume of the unit sphere in N-space, and $V(\theta_{ji})$ is the volume of the subset of the unit spere in N-space consisting of all vectors having angle θ_{ji} with the vector $\mathbf{r}_i$. This subset is in fact a sphere in $N-1$-space with radius $\sin\theta_{ji}$. [To see this, choose a Cartesian coordinate system in N-space in which the first coordinate direction lies along $\mathbf{r}_i$. Then the first coordinate x_1 of all points in the subset is $\cos\theta_{ji}$. Since all points lie on the unit spere, we have

$$1 = \sum_{i=1}^{N} x_i^2 = \cos^2\theta_{ji} + \sum_{i=2}^{N} x_i^2$$

which implies

$$\sum_{i=2}^{N} x_i^2 = 1 - \cos^2\theta_{ji} = \sin^2\theta_{ji}$$

Thus the subset is a sphere in N–1-space with radius $\sin\theta_{ji}$.] Therefore

$$V(\theta_{ji}) = V_{N-2} \sin^{N-2}\theta_{ji}$$

Thus the expected intrusion is

$$\mathrm{EI} = \frac{V_{N-2}}{V_{N-1}} 2\int_0^{\pi/2} \sin^{N-2}\theta \cos\theta \, d\theta = \frac{V_{N-2}}{V_{N-1}} 2\int_0^1 z^{N-2} dz = \frac{V_{N-2}}{V_{N-1}} \frac{2}{N-1}$$

(using the substitution $z = \cos\theta$ which implies $dz = -\sin\theta \, d\theta$). As shown in the Appendix in Section 6, the ratio of volumes of spheres of successive dimensions V_{N-2}/V_{N-1} is a complex expression taking different forms depending on whether N is odd or even. Since these details are quite irrelevant to the general behavior as N increases, we can look at the mean of two successive such ratios (using the geometric mean since the quantities are ratios) which is given by the simple expression:[3]

$$\sqrt{(N-1)/2\pi}$$

The result then is

$$\mathrm{EI} = \sqrt{\frac{2}{\pi(N-1)}}$$

As claimed, the expected interference falls as $N^{1/2}$.

For a structure involving n bindings, the expected total magnitude of intrusions of all $\{r_j\}_{j \neq i}$ into r_i is $(n-1)$EI. This equals unity at

$$n = \sqrt{\pi/2} \, (N-1)^{1/2} + 1$$

which increases as the square-root of N. ∎

The estimate of interference given in the preceding theorem is a very conservative one, since it computes the expected sum of the *absolute values* of all intrusions. In fact, for any given component of the desired filler, the errors caused by intrusions will be of both signs, producing a net error much smaller than the worst case analyzed above.

3.3. Continuous structures and infinite-dimensional representations

Certain structures are characterized by a continuum of roles. Strings, for example, have a natural extension to a continuum of positions. Examples of such continuous one-dimensional "strings" include speech input and motor output; a two-dimensional example is an image.

---

3. There is a rough calculation that suggests that, as the dimension N grows, the expected inner product of role vectors should decrease with the square root of N. Suppose for the first N role vectors we chose an orthonormal basis. For the next vector, suppose we choose one that is equidistant from all the others; an example is the vector whose components in the orthonormal basis are $N^{-1}(1, 1, \cdots, 1)$. In order for this vector to have unit length, the normalization constant N must be $\sqrt{N}$. Now the inner product of this vector with any of the others is $N^{-1} = N^{-1/2}$.

The tensor product representation extends naturally to the case of a continuum of roles. The representation of the conjunction of bindings extends naturally from the sum over a discrete set of bindings to an integral over a continuum of bindings.

DEFINITION 3.3.1: Let F/R be a role decomposition of S, not necessarily finite, and let $d\mu(r)$ be a measure on R. Let $\mathrm{supp}_R(s)$ be the subset of R containing roles which are bound in s, and suppose F/R has single-valued roles. Suppose given the connectionist representations

$$\psi_F : F \to V_F; f \mapsto \mathbf{f}$$
$$\psi_R : R \to V_R; r \mapsto \mathbf{r}$$

and assume these functions are measurable with respect to $d\mu(r)$. Then the corresponding tensor product representation of S is

$$\psi_{F/R}(s) = \int_{\mathrm{supp}_R(s)} \mathbf{f}(r) \otimes \mathbf{r}\, d\mu(r)$$

$\psi_{F/R}(s)$ is defined only for those s for which the integral is well-defined: $\mathrm{supp}_R(s)$ must be a measurable set and the integral must converge.

If the role decomposition is finite, and $d\mu$ is counting measure, then this reduces to the previous definition of the tensor product representation.

In the case of a continuous string, we can take the roles to be $r(t)$ for a continuous time index t. For the measure we can use ordinary Lebesgue measure on t. Then if each is represented by a pattern $\mathbf{r}(t)$ and the fillers by the patterns $\mathbf{f}(t)$, the entire continuous string is represented by $\int_t \mathbf{f}(t) \otimes \mathbf{r}(t)\, dt$ This representation of the continuous structure goes over exactly to the discrete case if it happens that the fillers are discrete step-functions of time. Suppose the filler $\mathbf{f}(t)$ is constant over the interval $[t_i, t_{i+1}]$ with value $\mathbf{f}_i$. Then the representation of the string is

$$\int_t \mathbf{f}(t) \otimes \mathbf{r}(t)\, dt = \sum_i \int_{t_i}^{t_{i+1}} \mathbf{f}(t) \otimes \mathbf{r}(t)\, dt = \sum_i \int_{t_i}^{t_{i+1}} \mathbf{f}_i \otimes \mathbf{r}(t)\, dt = \sum_i \mathbf{f}_i \otimes \int_{t_i}^{t_{i+1}} \mathbf{r}(t)\, dt = \sum_i \mathbf{f}_i \otimes \mathbf{r}_i$$

where the vector representing the discrete role for the time slot $[t_i, t_{i+1}]$ is the integral of the vectors representing the time points in the slot:

$$\mathbf{r}_i := \int_{t_i}^{t_{i+1}} \mathbf{r}(t)\, dt$$

The representation of a continuous string can be visualized with the help of the example illustrated in Fig. 3, which shows a tensor product binding between a time and the energy level of a speech formant. The patterns representing the energy level and time are peaks centered at the values being represented; this can apply to continuous represented values as well as discrete ones. The pattern $\mathbf{r}_4$ representing time $i = 4$ (shown in Figure 3) is a peak centered on the third role unit; a pattern $\mathbf{r}(4.2)$ representing time $t = 4.2$ would be derived by taking a peak on the continuous line centered at 4.2 and evaluating it at the integer values $i = 1, 2, \cdots, 5$. One can similarly generate patterns representing continuous energy levels $\mathbf{f}(t)$. As in the discrete case, the tensor product representation of the binding $f(t)/r(t)$ then becomes a two-dimensional peak centered at $(t, f(t))$, evaluated at points with integer coordinates. Superimposing the representation of the bindings for all t, we get the representation of the continuous string of energy levels: it resembles a smeared-out version of the graph of energy versus time, the activity of each unit in the grid of Fig. 3 being greater the closer it lies to the actual graph.

In the representation illustrated in Fig. 3, the role and filler vector spaces have finite dimensions (5 and 4, respectively). In such a case it is of course impossible for all the role vectors to be linearly independent; that would require an infinite-dimensional role vector space. The tensor product representation applies as well to infinite-dimensional vector spaces as to finite-dimensional ones. In that case the patterns representing roles (and possibly

also fillers) would not be patterns defined by a finite number of values as shown in Fig. 3 but could rather be curves defined over a continuous segment. The peaked patterns representing energy levels and times could be smooth Gaussians over a fixed interval, with mean equal to the quantity being represented and with variance, say, some fixed value. Then the representation of each binding would be a two-dimensional smooth Gaussian with mean at the point with x- and y-coordinates equal to the time and energy values, respectively.

If the role space is infinite-dimensional then so too will be the binding space. To view this space as the states of a connectionist network would require postulating an infinite number of units, one for each dimension of the space. The infinite-dimensional case is of interest not for computer simulation but for analysis; patterns which are functions of a continuous variable pose no particular difficulty for analysis relative to patterns which are finite-dimensional vectors.

It is significant that the tensor product representation extends so naturally to continuous collections of roles, continuous sets of fillers, and vectors for representing roles and fillers that are continuous patterns. As I have argued elsewhere (Smolensky, forthcoming), it is useful to hypothesize that a defining characteristic of connectionist computation is the existence of an underlying continuous model. Thus a well-motivated connectionist representational scheme should have a natural continuous extension, even if particular simulation models take advantage only of the discrete case.

3.4. Connectionist mechanisms for binding and unbinding

The tensor product representation has so far been characterized mathematically, without any discussion of how such a representation might be set up and used in a connectionist system. In this section I consider first the creation of bindings and then unbinding.

3.4.1. Parallel binding in connectionist systems

The most immediate application of the tensor product representation is to models learning to map some structured input to structured output; for example, the surface form of a sentence to its parsed form. Here it is not the job of the network to set up the tensor product representations: in presenting the input/output pairs to the network during training, the modeler must convert the symbolic inputs and outputs to their vector representations, and this can be done directly by using the mathematical definition of the tensor product representation.

In more complex applications, a network might be so constructed as to internally perform variable binding via the tensor product. A convenient way to achieve this is to use so-called *sigma-pi* processing units (Rumelhart, Hinton, & McClelland, 1986). Such a unit has a number of input sites at each of which connections from a number of other processors converge. For each site σ, the sigma-pi unit takes the *product* of all the inputs $\{I_{\sigma i}\}_i$ there; it then adopts as its value v a weighted *sum* over all sites, with one weight w_σ per site:

$$v = \sum_\sigma w_\sigma \prod_i I_{\sigma i}$$

Using sigma-pi units, tensor product binding can be easily achieved in a connectionist network: see Fig. 8. The network consists of a set of filler units $\tilde{f}_\phi$, a set of role units $\tilde{r}_\rho$, and set of binding units $\tilde{b}_{\phi\rho}$, one for each pair of filler and role units. Each binding unit is a sigma-pi unit with a single site with unit weight. Converging on the site of the binding unit $\tilde{b}_{\phi\rho}$ are two connections, one from $\tilde{f}_\phi$ and one from $\tilde{r}_\rho$. Then if the filler and role patterns **f** and **r** are set up on the filler and role units, the binding units will set up the representation of the binding **f**/**r**.

Figure 9 shows a network equivalent to the one shown in Fig. 8. Here the product occurs not at the unit but at the junction of two connections; the two activities entering the triangular junction (of Hinton, 1981a) from the filler and role units are multiplied together and the result is sent along the third line to the binding unit.

The representation of complex structures requires superimposing multiple filler/role bindings. There are two obvious ways of doing this: sequentially and in parallel. In the sequential case, one binding is performed at a time, and the binding units accumulate their activity over time. This can be achieved with the network shown in Fig. 8 if

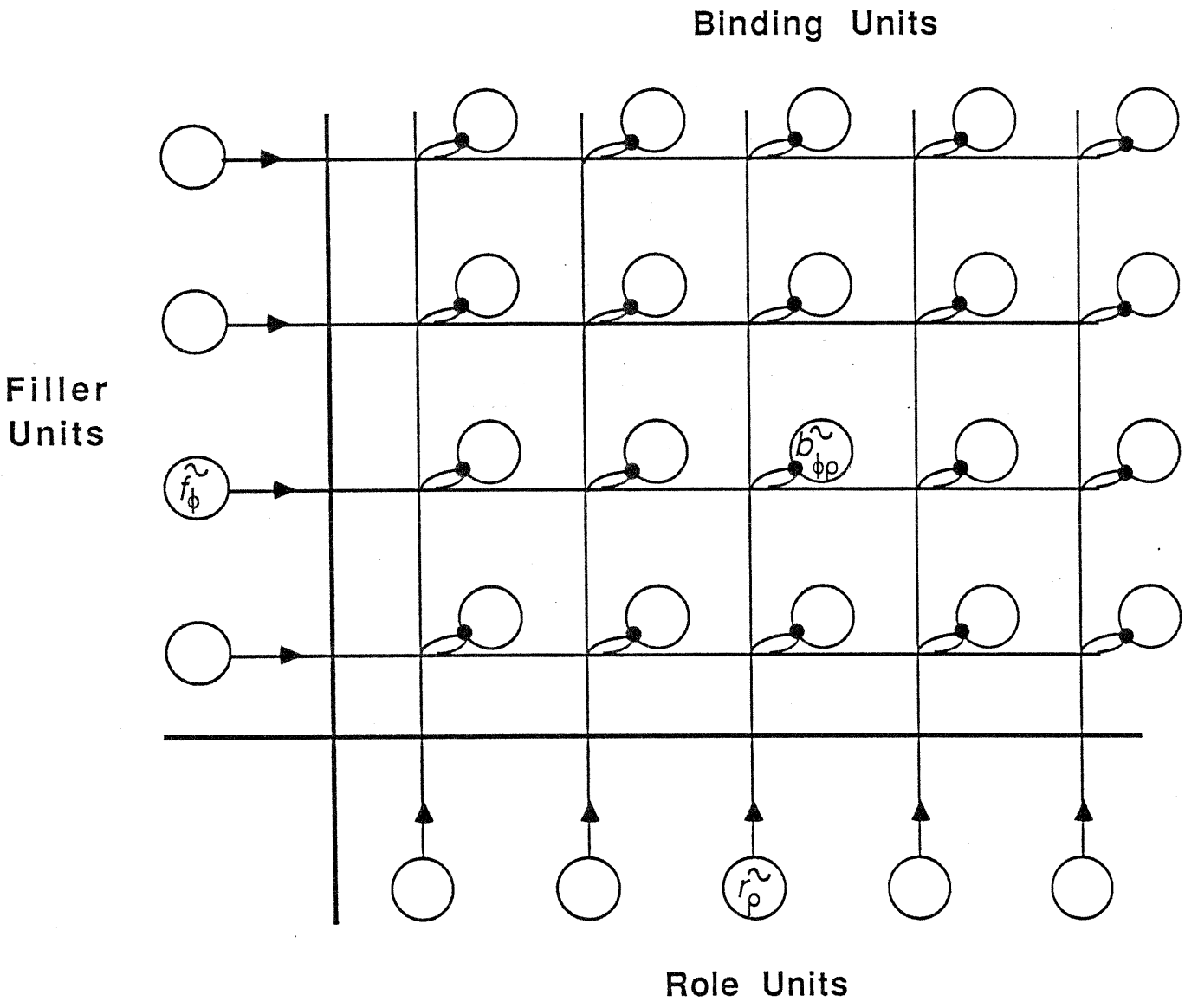


Figure 8. A network using sigma-pi binding units to perform tensor product binding.

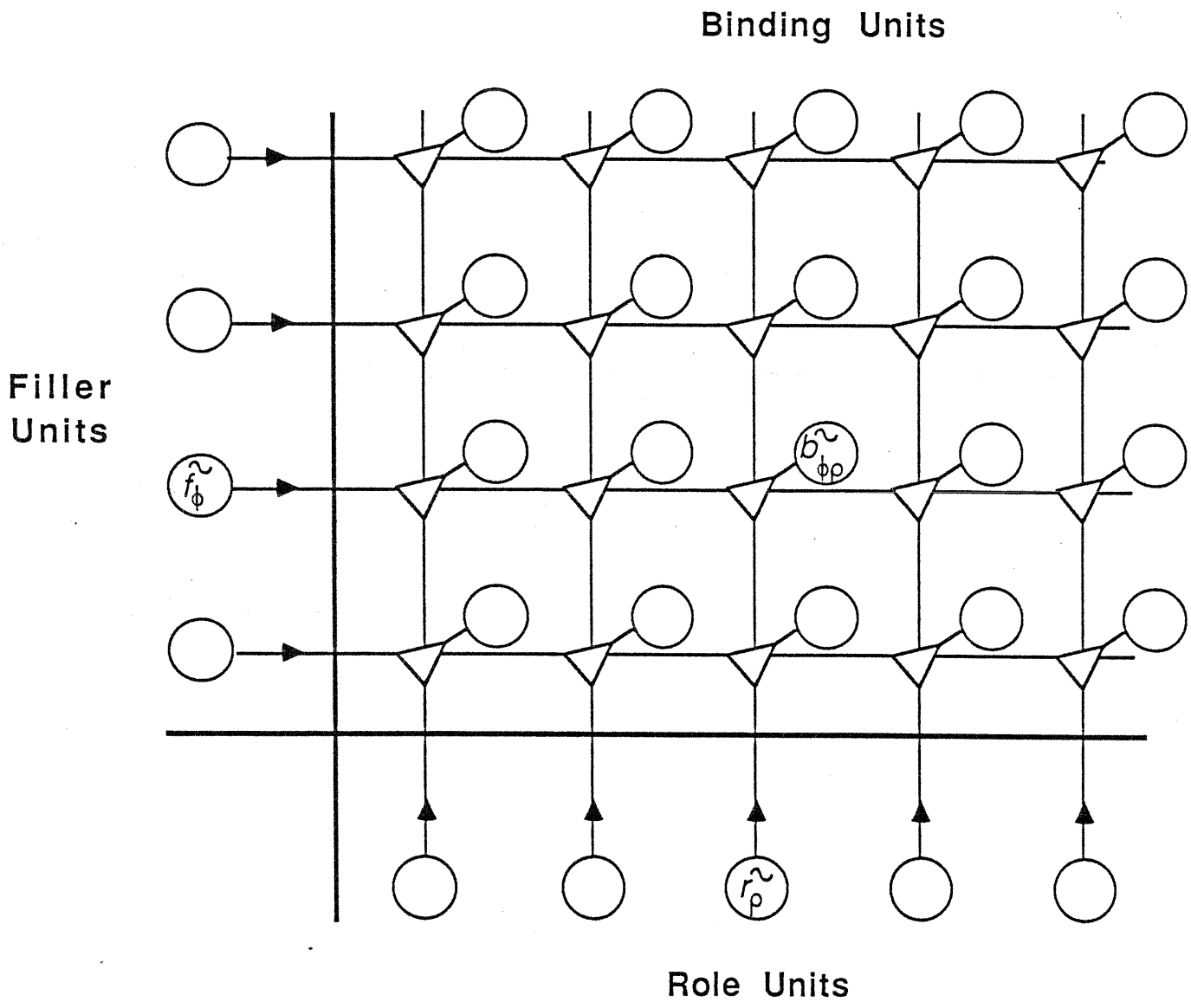


Figure 9. A network using multiplicative junctions to perform tensor product binding.

we use accumulating sigma-pi binding units obeying:

$$\frac{dv}{dt} = \sum_{\sigma} w_{\sigma} \prod_{i} I_{\sigma i}$$

Equivalently, serial binding can be performed by the network of Fig. 9 if the binding units accumulate activity over time.

In order to superimpose all N bindings in parallel, we need to extend the network shown in Fig. 8, creating nodes $\{f_{\phi}^{(\sigma)}\}_{\sigma=1}^{N}$ and $\{r_{\rho}^{(\sigma)}\}_{\sigma=1}^{N}$: see Fig. 10, which illustrates the simplest case, $N = 2$. Now each sigma-pi binding unit has N sites instead of one; each site has unit weight. Each site σ on binding unit $\tilde{b}_{\phi\rho}$ receives a pair of connections from the nodes $\tilde{f}_{\phi}^{(\sigma)}$ and $\tilde{r}_{\rho}^{(\sigma)}$. Now we can bind N pairs of roles and fillers in parallel. In the σ^{th} filler pool we set up the pattern $\mathbf{f}_{\sigma}$ representing f_{σ} and on the σ^{th} role pool we set up the pattern $\mathbf{r}_{\sigma}$ representing r_{σ}. The value of binding unit $\tilde{b}_{\phi\rho}$ is then

$$\tilde{b}_{\phi\rho} = \sum_{\sigma} 1 \prod \{\tilde{f}_{\phi}^{(\sigma)}, \tilde{r}_{\rho}^{(\sigma)}\} = \sum_{\sigma} (\mathbf{f}_{\sigma})_{\phi} (\mathbf{r}_{\sigma})_{\rho}$$

The pattern of activity on the binding units is thus the correct tensor product representation of the structure. Fig. 11 is the equivalent of Fig. 10 using multiplicative junctions instead of sigma-pi units.

There is no need to perform *all* the binding serially or in parallel; the mechanisms of sequential and parallel combination of bindings are independent, and can be combined. If there are N pools of filler and role units, N bindings can be established in parallel, and if the binding units accumulate activity over time, further bindings can be added sequentially, up to N at a time.

There are two senses in which bindings are occuring in parallel here. Bindings are *generated* in parallel, N at a time; the *generative capacity* is sharply defined by N. At the same time, multiple bindings are being *maintained* in parallel; the binding units can simultaneously support multiple bindings superimposed on each other. The *maintenance capacity* of the representation is not sharply defined, due to the graceful saturation of the representation. The scale of the maintenance capacity is, however, set by n, the number of role units in each of the N sets.

For the network shown in Fig. 10, the generative and maintenance capacities are independent; this contrasts with most existing connectionist systems. For example, the McClelland & Rumelhart letter perception model processes exclusively four letter words. Strings of length $n = 4$ can be represented; the maintenance capacity is precisely defined at 4 letters. The binding of all four letters to their positions are all performed in parallel; the generative capacity is also $N = 4$. If different roles correspond to different regions of a parallel network, as in local and semi-local representations, it is natural that these roles should all be sent activation in parallel. If the different roles share a common set of units, as in fully distributed representations, there comes the space/time trade-off we have seen above: duplicate machinery to permit parallel binding, or wait while multiple bindings are performed serially.

It seems intuitive that the two binding capacities ought to be independent characteristics of the degree of parallelism in a processing system. In many human cognitive processes, for example, the generative capacity of binding appears to be much smaller than the maintenance capacity: $N \ll n$. In visual perception we are able to maintain rich percepts involving a huge number of bindings of properties to locations, but it turns out that at any one time (requiring approximately 50 msec) our visual systems can only establish the bindings for a small region of the visual field (Treisman & Schmidt, 1982). The large number of bindings that we maintain in parallel are generated a small fraction at a time through an extended sequential process. In discourse processing, syntactic and semantic processes seem to indicate that many constituents in complex structures are being maintained and processed in parallel, yet only a small fraction of these constituent/role bindings are generated at once. If one looks at the processing of small linguistic and/or visual items whose size fits within the binding generative capacity (eg. four-letter words), the distinction between the generative and maintenance capacities does not assert itself. However, connectionist models of more complex, extended tasks such as reading whole passages must respect the distinction between these two aspects of parallelism; the tensor product representation offers a natural way to do so.

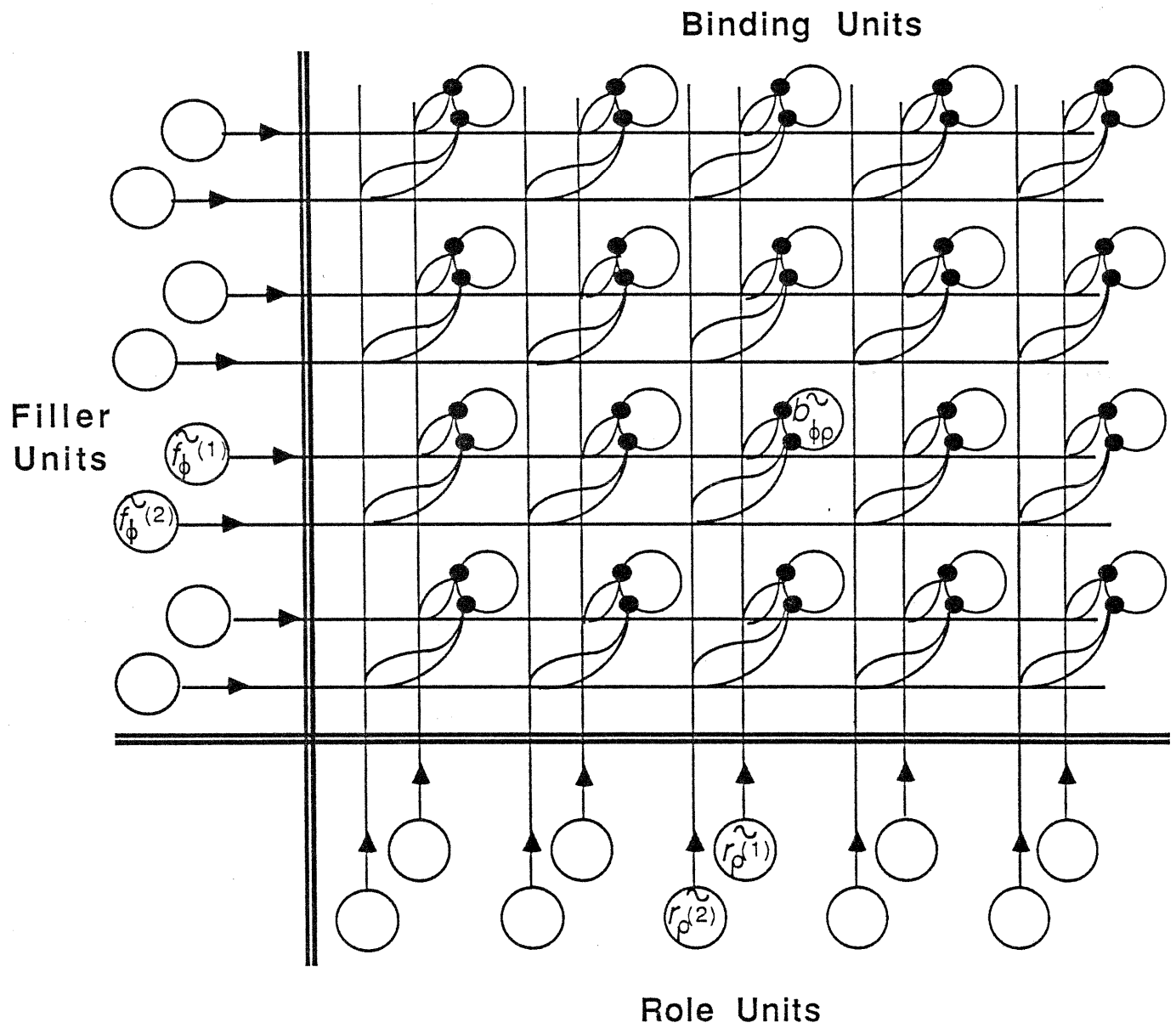


Figure 10. An extension of the network of Fig. 8 that can perform two variable bindings in parallel.

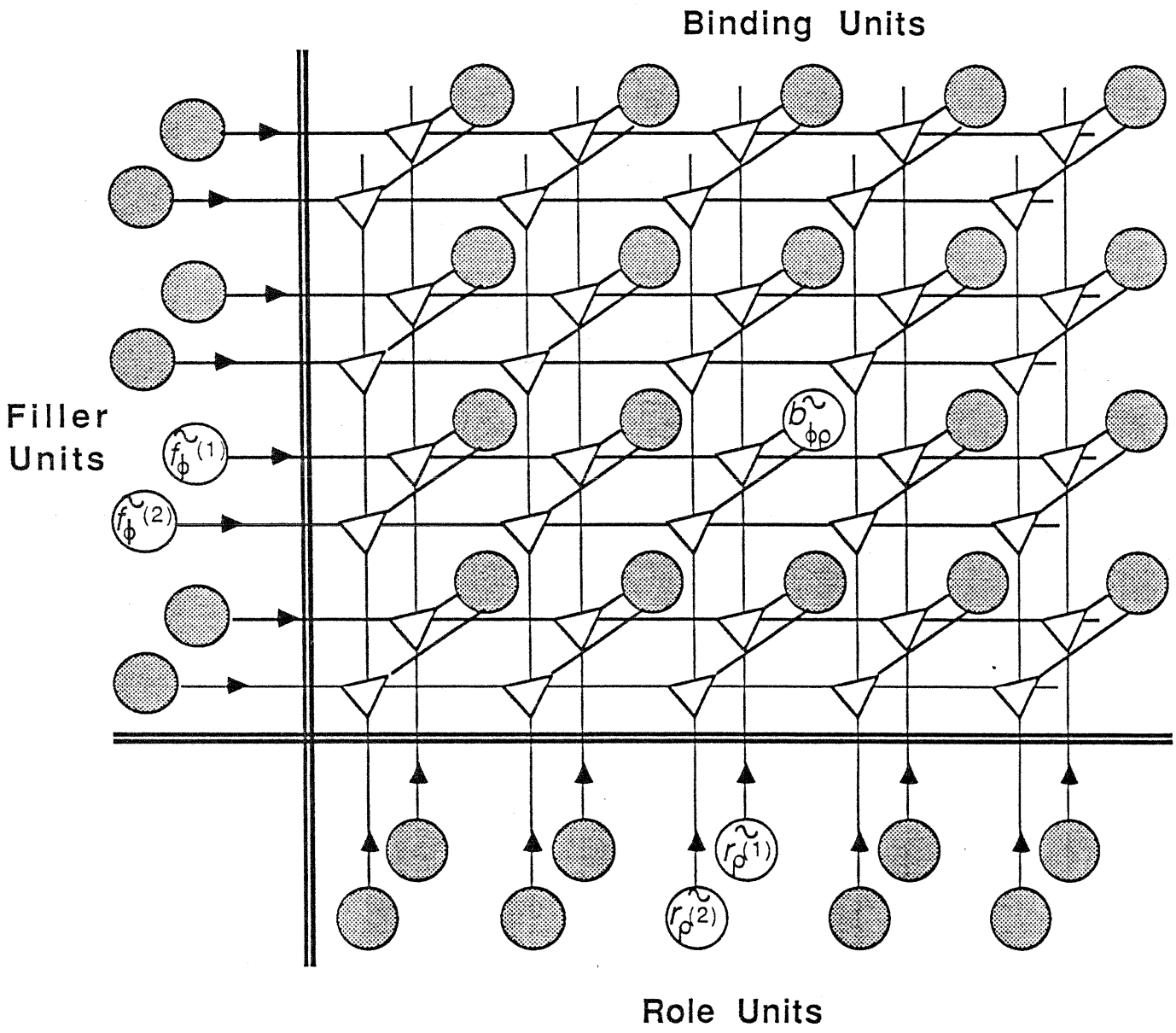


Figure 11. An extension of the network of Fig. 9 that can perform two variable bindings in parallel.

3.4.2. Connectionist unbinding mechanisms

The mathematics of the unbinding procedure was described in section 3.1. It is easy to implement this procedure in a connectionist network; in fact, the network of Fig. 9 can be used for unbinding as well as for binding. We presume that the binding units are supporting a pattern of activity which is the tensor product representation of a structure. To unbind role r_i, a pattern of activity is first set up on the role units: for the exact unbinding procedure, this pattern should be that of the unbinding vector $\mathbf{u}_i$; for the self-addressing unbinding procedure, the pattern should be $\mathbf{r}_i$. As a result of the activity in the role and binding units, a pattern of actitivity arises on the filler units. At each triangular junction, the activity of the connected role and binder units are multiplied together and sent to the connected filler unit, which adds up all the inputs it so receives. Thus the activity of filler unit $\tilde{f}_\phi$ is

$$\tilde{f}_\phi = \sum_\rho \tilde{r}_\rho \tilde{b}_{\phi\rho}$$

This is the correct activity to implement the unbinding procedures of Section 3.1. With the extended network shown in Fig. 11, N roles can be unbound simultaneously.

This procedure has been defined for retrieving a filler from a role. By interchanging roles and fillers, it can also be used to retrieve a role from a filler, subject to the caveat of section 3.1 about non-single-valuedness.

3.5. Binding unit activities as connection weights

In Section 3.4.1 we discussed one way of generating the tensor product representation of a structure: sequentially representing individual filler/role pairs on the role and filler units, while each binding unit takes the product of the activities of its corresponding pair of role and filler units. These products then accumulate on the binding units as the individual pairs are presented. This procedure is formally identical to the *Hebbian learning procedure* for storing the associations between roles and corresponding fillers: each binding unit plays the role of the *connection* between a role and filler unit, and its activity plays the role of the weight or strength of that connection. Furthermore, the self-addressing unbinding mechanism described in Section 3.4.2 is formally identical to the use of the Hebbian weight matrix to associate a pattern over the role units with the corresponding pattern on the filler units.

This relationship between binding units and connections suggests avenues for further exploration, two of which will now be briefly described.

3.5.1. From Hebbian to Widrow-Hoff weights

In section 3.1 it was pointed out that pattern needed for exact unbinding of role r_i, the unbinding vector $\mathbf{u}_i$, is not in general equal to the role vector $\mathbf{r}_i$; the retrieval and role patterns are equal only if the role vectors are all orthogonal. This corresponds to a well-known property of the Hebbian weight matrix: associations will be correctly formed by the Hebbian learning procedure if and only if the input patterns are orthogonal. There is a more complex learning procedure than the Hebbian one which produces a matrix with better retrieval capability than the Hebbian matrix: the Widrow-Hoff (1960) or delta rule (Rumelhart, Hinton, & McClelland, 1986) This suggests replacing the Hebbian matrix corresponding to the tensor product representation with the Widrow-Hoff matrix. With this new representation, the self-addressing unbinding procedure would produce correct results as long as the role vectors are *linearly independent*: orthogonality is not required. Unfortunately, this Widrow-Hoff representation is considerably more difficult to write down, analyze, and actually construct in a connectionist network. For example, the Widrow-Hoff learning procedure, unlike the Hebbian one, requires repeated presentations of the set of items to be stored.

3.5.2. Relation to Connection Information Distribution

The relation between tensor product binding units and Hebbian weights suggests another development of the present analysis. In McClelland's (1986) Connection Information Distribution (CID) scheme, the activity of certain units determine the weights between others. Unbinding could be naturally carried out in a CID as follows. The represented structure would be active in a set of binder units which would set the weights between role and filler units. This would create a machine that transforms roles patterns to filler patterns (to the approximation to which retrieval vectors equal role vectors). Fig. 11 can be viewed as a CID in which the binder units are setting weights in a collection of N role/filler associators.

Despite the intimate relation between tensor product binding units and connection weights, it should emphasized that the primary purpose of the tensor product representation is not to serve as an apparatus for filler/role associations: it is rather to provide a pattern of activity representing a structured object which can then be used to process the object as a whole. This is the reason the elements of the tensor product representation have been viewed as the activities of units rather than the strength of connections. The CID allows us to use unit activities as connection strengths, giving us simultaneous access to both aspects of the representation.

3.6. Values as variables

It is often important for the value bound to a variable to in fact itself be a variable to which a value is to be bound. The tensor product binding representation allows for this in the following way. Out of the representation for the variable/value binding can be extracted the pattern of activity that represents the value. This pattern can in turn be used as the pattern representing a variable, and used in another binding on other binding units where it is bound to a value. The situation is depicted in Fig. 12.

3.7. Representation of symbolic operations; recursive decompositions

So far we have not considered the representation of symbolic *operations*: mappings from S to itself. Examples that will now be considered are the stack operations *push* and *pop* and the LISP operators *car*, *cdr*, and *cons*. Understanding such operations are important for treating recursive role decompositions, since in such a decomposition each role is in fact an operator mapping S into S.

The definition we need to get started is

DEFINITION 3.7.1: Let O be an operator on S:

$$O: S \rightarrow S; s \mapsto O(s)$$

Suppose $\psi: S \rightarrow V$ is a connectionist representation of S. Then a corresponding *representation of* O is an operator

$$\mathbf{O}: V \rightarrow V; \mathbf{v} \mapsto \mathbf{O}\mathbf{v}$$

with the property

$$\psi(O(s)) = \mathbf{O}\psi(s)$$

3.7.1. Stack operations: push and pop

In this section we consider the basic stack operations, *push* and *pop*. To keep complications to a minimum, two simplifications will be made. In place of a stack containing complex elements, simple strings from a fixed alphabet will be used to model the essential stack structure of linear ordered elements with a first element. The second simplification will be to consider an infinite stack, i.e., no limit to the length of the strings modeling the stack.

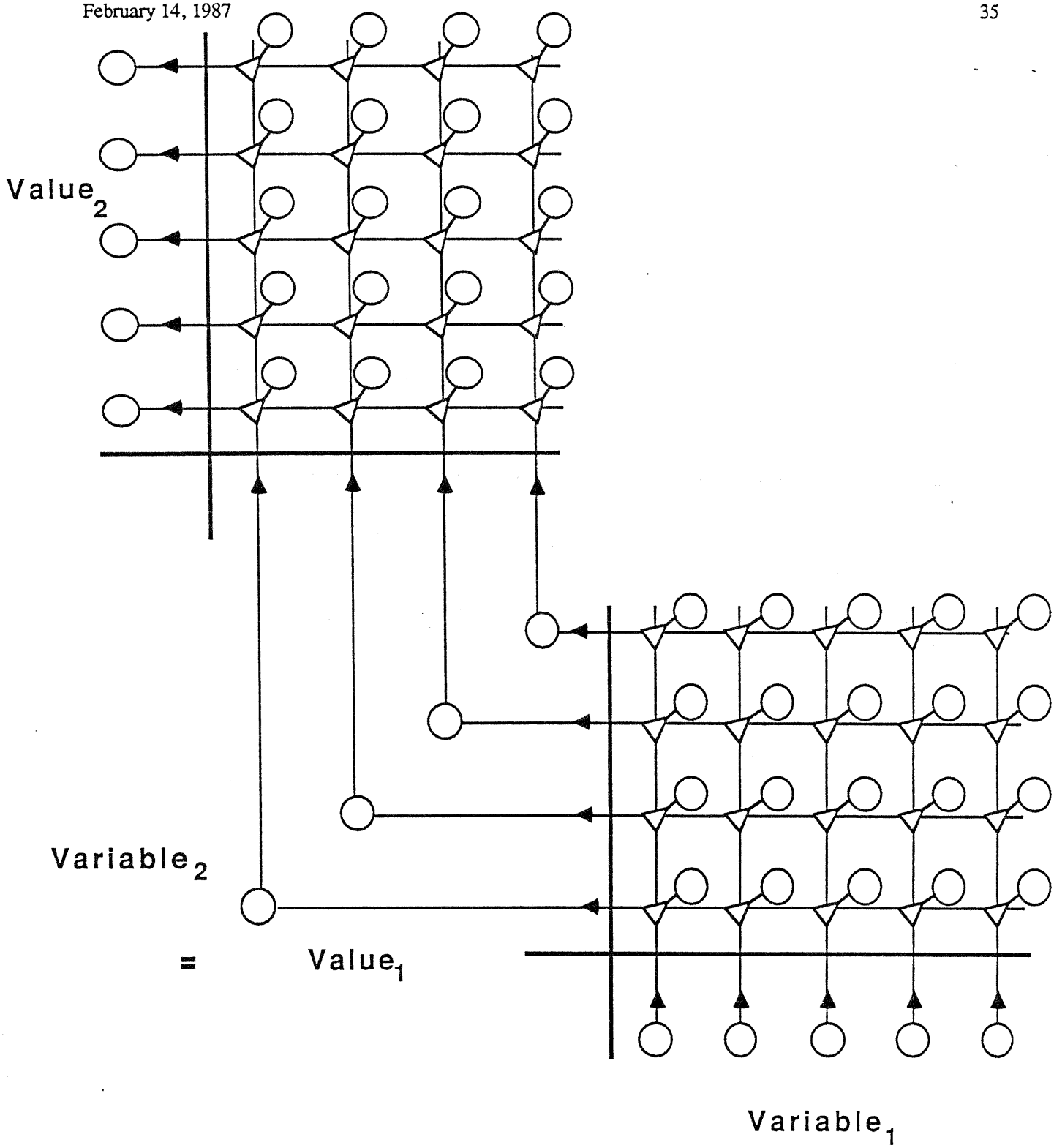


Figure 12. A network capable of representing two value/variable bindings in which the same entity—the pattern of activity over the diagonally-aligned units—serves as the value in the first binding ($Value_1$) and the variable in the second binding ($Variable_2$).

Let S be the set of finite-length strings from an alphabet A. Let F/R be the positional role decomposition of definition 2.3.1.1. Let ψ_F be a faithful representation of F, and let ψ_R be a representation of R in which the role vectors $\{\mathbf{r}_i\}_{i=0}^{\infty}$ representing the positional roles $\{r_i\}_{i=0}^{\infty}$ are all linearly independent. This means that V_R is an infinite-dimensional space. (The analysis can easily be modified to strings of length no more than n, in which case V_R can be n-dimensional; the finite case just introduces uninteresting complications.) For simplicity, assume that the role vectors span the space V_R and therefore form a basis.

The positional role decomposition has the property that if r_i is unbound, so is r_j if $j>i$. Thus the representations of strings are all in a restricted subspace of V:

DEFINITION 3.7.1.1: The *string subset* of V is

$$V_S = \{\sum_i \mathbf{f}_i \otimes \mathbf{r}_i \mid \text{for all } i, \text{ if } \mathbf{f}_i = \mathbf{0} \text{ then for all } j>i, \mathbf{f}_j = \mathbf{0}\}$$

THEOREM 3.7.1.2: The *pop* operation on S is represented by a linear transformation **pop** on V:

$$\mathbf{pop}: V \to V; \sum_i \mathbf{f}_i \otimes \mathbf{r}_i \mapsto \sum_i \mathbf{f}_i \otimes \mathbf{r}_{i-1}$$

The operation $push_a$ on S is represented by an affine transformation $\mathbf{push_a}$ on V:

$$\mathbf{push_a}: V \to V; \sum_i \mathbf{f}_i \otimes \mathbf{r}_i \mapsto \mathbf{a} \otimes \mathbf{r}_0 + \sum_i \mathbf{f}_i \otimes \mathbf{r}_{i+1}$$

Both **pop** and $\mathbf{push_a}$ map V_S into V_S, for all $\mathbf{a} \neq \mathbf{0}$.

PROOF: First note that the definitions of **pop** and **push** given in the Theorem are adequate because, as shown in Theorem 5.3.4 (b) of the Appendix, every vector in $V = V_F \otimes V_R$ can be uniquely expressed in the form $\sum_i \mathbf{f}_i \otimes \mathbf{r}_i$ since $\{\mathbf{r}_i\}$ is a basis of V_R. That **pop** is linear and **push** is affine are easily checked.

Suppose s is the string $a_0 a_1 \cdots a_n$ and that the characters have representations $\psi_F(a_i) = \mathbf{a}_i$. Then

$$\psi(pop(s)) = \psi(a_1 a_2 \cdots a_{n-1}) = \sum_{i=0}^{n-1} \mathbf{a}_i \otimes \mathbf{r}_{i-1} = \mathbf{pop} \sum_{i=0}^{n} \mathbf{a}_i \otimes \mathbf{r}_i = \mathbf{pop}\,\psi(s)$$

Thus **pop** is a representation of *pop*. Similarly, $\mathbf{push_a}$ is a representation of $push_a$.

$$\psi(push_a(s)) = \psi(a a_0 a_1 \cdots a_n) = \mathbf{a} \otimes \mathbf{r}_0 + \sum_{i=0}^{n} \mathbf{a}_i \otimes \mathbf{r}_{i+1} = \mathbf{push_a} \sum_{i=0}^{n} \mathbf{a}_i \otimes \mathbf{r}_i = \mathbf{push_a}\,\psi(s)$$

∎

3.7.2. LISP binary tree operations: car, cdr, and cons

Let S be the set of LISP S-expressions built from a set of atoms A. We define a role decomposition as follows. For the fillers, take $F = A$. A typical role, r_{011011}, is defined as follows. The predicate a/r_{011011} is "the *caddaddr* is the atom a". The roles are indexed by finite bit strings, and correspond to compositions of *car* and *cdr* operations, with 0 indicating *car* and 1 indicating *cdr*. Note that these roles are to be filled only by *atoms*. Thus, for example, the S-expression $s = (a\ (b\ .\ c))$ contains the bindings $\{a/r_0, b/r_{01}, c/r_{11}\}$; the role r_1 is unbound—not because s has no *cdr*, but because the *cdr* is not an atom. The role indexed by the empty string ε is special: the predicate a/r_ε is "is the atom a."

This decomposition is faithful and has single-valued roles. If objects like circular lists are considered valid S-expressions, then the decomposition is not finite.

This role decomposition has the property that if r_x is bound, then r_{yx} is unbound, where yx is the concatenation of the bit strings y and x. In particular, if r_ε is bound, no other role can be; this is exactly the case for atoms. Lists are S-expressions for which the *cdr* is never a non-nil atom, at all levels of imbedding; in other words, for all bit strings x, r_{1x} is unbound or bound to *nil*.

Let ψ_R map each r_x into a corresponding vector $\mathbf{r}_x$ in a basis of an infinite-dimensional vector space V_R. Let ψ_F be a faithful representation of $F = A$ in V_F, and let $\mathbf{nil}:=\psi_F(nil)$. Now we investigate the properties of the induced tensor product representation ψ.

DEFINITION 3.7.2.1: The *atomic subspace* of V is

$$V_a = \{\mathbf{f}\otimes\mathbf{r}_\varepsilon \,|\, \mathbf{f}\in V_F\} = V_F\otimes \mathrm{span}(\{\mathbf{r}_\varepsilon\})$$

The *non-atomic subspace* of V is

$$V_{na} = \{\sum_{x\neq\varepsilon}\mathbf{f}_x\otimes\mathbf{r}_x \,|\, \mathbf{f}_x\in V_F\} = V_F\otimes \mathrm{span}(\{\mathbf{r}_x \,|\, x\neq\varepsilon\})$$

The *S-subset* of V is

$$V_S = \{\sum_x \mathbf{f}_x\otimes\mathbf{r}_x \,|\, \text{for all } x, \text{ if } \mathbf{f}_x = \mathbf{0} \text{ then for all } y, \mathbf{f}_{yx} = \mathbf{0}\}$$

The *list subset* of V is

$$V_l = \{\sum_x \mathbf{f}_x\otimes\mathbf{r}_x \in V_S \,|\, \text{for all } x, \mathbf{f}_{1x}\neq\mathbf{0} \Rightarrow \mathbf{f}_{1x} = \mathbf{nil}\}$$

Note that V_S is not closed under vector addition. For example, $\psi((a)) + \psi(((b)))$ corresponds to a mixture of two list structures; it possesses the bindings $\{a/r_0, b/r_{00}\}$, violating the condition defining V_S. Thus V_S is not a vector space. The same example also shows that V_l is not a vector space.

Now we are ready for representations of the operators *car*, *cdr*, and *cons*.

DEFINITION 3.7.2.2: Define two linear transformations $\mathbf{T}_0$ and $\mathbf{T}_1$ on V_R by the following actions on the basis $\{\mathbf{r}_x\}$:

$$\mathbf{T}_0\colon V_R\to V_R;\ \mathbf{r}_{x0}\mapsto\mathbf{r}_x;\ \mathbf{r}_{x1}\mapsto\mathbf{0};\ \mathbf{r}_\varepsilon\mapsto\mathbf{0}$$

$$\mathbf{T}_1\colon V_R\to V_R;\ \mathbf{r}_{x1}\mapsto\mathbf{r}_x;\ \mathbf{r}_{x0}\mapsto\mathbf{0};\ \mathbf{r}_\varepsilon\mapsto\mathbf{0}$$

THEOREM 3.7.2.3: The following linear transformations on V are representations of the operators *car* and *cdr*:

$$\mathbf{car}\colon \sum_x \mathbf{f}_x\otimes\mathbf{r}_x \mapsto \sum_x \mathbf{f}_x\otimes\mathbf{T}_0\mathbf{r}_x$$

$$\mathbf{cdr}: \sum_x \mathbf{f}_x \otimes \mathbf{r}_x \mapsto \sum_x \mathbf{f}_x \otimes \mathbf{T}_1 \mathbf{r}_x$$

PROOF: The representation of an S-expression s can be written

$$\psi(s) = \sum_y \mathbf{f}_y \otimes \mathbf{r}_y = \mathbf{f}_\varepsilon \otimes \mathbf{r}_\varepsilon + \sum_x \mathbf{f}_{x0} \otimes \mathbf{r}_{x0} + \sum_x \mathbf{f}_{x1} \otimes \mathbf{r}_{x1}$$

Now $f_{x0} = cxr(car(s))$, where cxr denotes the composition of *car*s and *cdr*s corresponding to the bit string x. So if $t = car(s)$, then $f_{x0} = cxr(t)$. Thus the filler of r_{x0} in s is the filler of r_x in $t = car(s)$. Conversely, any filler of r_x in t is a filler of r_{x0} in s. Thus the representation of $car(s)$ is

$$\psi(car(s)) = \sum_x \mathbf{f}_{x0} \otimes \mathbf{r}_x = \mathbf{car}\left[\mathbf{f}_\varepsilon \otimes \mathbf{r}_\varepsilon + \sum_x \mathbf{f}_{x0} \otimes \mathbf{r}_{x0} + \sum_x \mathbf{f}_{x1} \otimes \mathbf{r}_{x1}\right] = \mathbf{car}\,\psi(s)$$

This shows that **car** represents *car*. By replaying this argument with *car* replacing *cdr* and with 0 and 1 interchanged, we see that **cdr** represents *cdr*. The linearity of **car** and **cdr** are immediate consequences of the linearity of $\mathbf{T}_0$ and $\mathbf{T}_1$.

NOTE: The operators **car** and **cdr** treat **nil** like all other atoms: they map it to **0**. This corresponds to the *car* and *cdr* of all atoms, including *nil*, being *undefined*. If *car* and *cdr* are defined to be *undefined* on all non-nil atoms, but to take *nil* to *nil*, then the above definitions of **car** and **cdr** have to be changed if they are to represent *car* and *cdr*: the definitions must include the *ad hoc* stipulation that $\mathbf{nil} \otimes \mathbf{r}_\varepsilon$ is mapped to itself, while $a \otimes \mathbf{r}_\varepsilon$ is mapped to **0** for all vectors a representing non-nil atoms. This does not destroy the linearity of **car** and **cdr** as long as the vector **nil** is linearly independent of the representations of all non-nil atoms. It does destroy the property that $\mathbf{f} \otimes \mathbf{r} \mapsto \mathbf{r} \otimes \mathbf{Tr}$, where the transformation of the role is independent of its filler. []

THEOREM 3.7.2.4: Let $\mathbf{u}_0$ and $\mathbf{u}_1$ be two vectors in V. Then there is a unique vector $\mathbf{v}$ in V_{na} such that

$$\mathbf{car}\ \mathbf{v} = \mathbf{u}_0$$
$$\mathbf{cdr}\ \mathbf{v} = \mathbf{u}_1$$

Define

$$\mathbf{cons}: V \times V \to V_{na};\ (\mathbf{u}_0, \mathbf{u}_1) \mapsto \mathbf{v}$$

Then this function is:

$$\mathbf{cons}: \left(\sum_x \mathbf{f}_x \otimes \mathbf{r}_x, \sum_y \mathbf{f}'_y \otimes \mathbf{r}_y\right) \mapsto \sum_x \mathbf{f}_x \otimes \mathbf{r}_{x0} + \sum_y \mathbf{f}'_y \otimes \mathbf{r}_{y1}$$

cons is a representation of the *cons* function on S.

PROOF: Let

$$\mathbf{u}_0 = \sum_x \mathbf{f}_x \otimes \mathbf{r}_x$$

$$\mathbf{u}_1 = \sum_y \mathbf{f}'_y \otimes \mathbf{r}_y$$
$$\mathbf{v} = \sum_x \mathbf{f}_x \otimes \mathbf{r}_{x0} + \sum_y \mathbf{f}'_y \otimes \mathbf{r}_{y1}$$

Then

$$\mathbf{car}\ \mathbf{v} = \mathbf{car}\left[\sum_x \mathbf{f}_x \otimes \mathbf{r}_{x0} + \sum_y \mathbf{f}'_y \otimes \mathbf{r}_{y1}\right] = \sum_x \mathbf{f}_x \otimes \mathbf{r}_x = \mathbf{u}_0$$

and

$$\mathbf{cdr}\ \mathbf{v} = \mathbf{cdr}\left[\sum_x \mathbf{f}_x \otimes \mathbf{r}_{x0} + \sum_y \mathbf{f}'_y \otimes \mathbf{r}_{y1}\right] = \sum_y \mathbf{f}'_y \otimes \mathbf{r}_y = \mathbf{u}_1$$

Furthermore, $\mathbf{v} \in V_{na}$ so v satisfies the required conditions. These conditions completely determine v: the **car** condition determines the fillers of all $\{r_{x0}\}$, the **cdr** condition determines the fillers of all $\{r_{x1}\}$, and the condition that v be in V_{na} implies that the only remaining role, r_ε, must be unfilled.

Since **car** and **cdr** represent *car* and *cdr*, it follows that **cons** represents *cons*. To see this, let

$$s = cons(s_0, s_1)$$
$$\mathbf{u}_0 = \psi(s_0)$$
$$\mathbf{u}_1 = \psi(s_1)$$

Then, since **car** represents *car*, and $car(s) = s_0$,

$$\mathbf{car}\ \psi(s) = \psi(car(s)) = \psi(s_0) = \mathbf{u}_0$$

and similarly

$$\mathbf{cdr}\ \psi(s) = \mathbf{u}_1$$

By the previous part of the proof, this implies that

$$\psi(s) = \mathbf{cons}(\mathbf{u}_0, \mathbf{u}_1)$$

In other words,

$$\psi(cons(s_0, s_1)) = \mathbf{cons}(\psi(s_0), \psi(s_1))$$

Thus **cons** represents *cons*. ∎

Just as complex structures in S can be constructed from atoms by successive applications of *cons*, so the tensor product representation of these items can similarly be constructed by successive applications of **cons** on the vectors representing atoms:

$$\psi(a) = \psi_F(a) \otimes \mathbf{r}_\varepsilon$$

Using **cons** to build up complex representations from simpler ones allows us to exploit the recursive role decomposition of S provided by *car* and *cdr*.

The analysis of strings in Section 3.7.1 can be viewed as a subset of this analysis of S-expressions. The alphabet is identified with the set of atoms, and the i^{th} positional role r_i of the string is identified with r_{0i_u}, where i_u is the unary representation of i: $i_u = 11\cdots1$ (i times). The operator *pop* becomes *cdr* and $push_a(s)$ becomes $cons(a, s)$.

3.7.3. Iterated tensor product representations

Related to recursive decomposition is the simpler case of *iterated decomposition*. This occurs when the fillers or roles are themselves structures that are decomposed by a new role decomposition. In other words, having decomposed S in terms of F and R, we now take F or R as a new S' and decompose it in terms of new fillers F' and roles R'. Consider the case of decomposition of R. If the role decomposition of R is F'/R', then the binding f/r is itself a set of bindings $f/(f'/r')$. The tensor product representation of such a finer-grained binding is then

$$\mathbf{f}\otimes(\mathbf{f}'\otimes\mathbf{r}')$$

In this case we are led to third-order (or, by further iteration, even higher-order) tensor products. The binding units can be interpreted as representing third- (or higher-) order conjunctions of features.

This iterative structure is just what we see in the Rumelhart and McClelland (1986) past-tense learning model. Here the original role decomposition of phonetic strings is the 1-neighbor context decomposition. Each role r_{x_y} is itself a structured object, whose structure is determined by the pair (x, y). These pairs can be decomposed by the right-neighbor role decomposition, in which x fills the role *has right neighbor* y, $r'_{\_y}$. Thus the binding i/r_{w_k} (the vowel in *week*) becomes $i/(w/r'_{\_k})$ and the final representation is the third-order tensor product:

$$\mathbf{i}\otimes\mathbf{w}\otimes\mathbf{r}'_{\_k}$$

(In fact, in this model, this is just $\mathbf{i}\otimes\mathbf{w}\otimes\mathbf{k}$, since the role vector $\mathbf{r}'_{\_k}$ is just $\mathbf{k}$.)

3.8. Storage of structured data in connectionist memories

One of the primary uses of connectionist representations is as objects of associations in associtive memories. Because of its mathematical simplicity it is possible to analyze the use of tensor product representations in such memories. Here I analyze the case of pair association since it is simpler than the content-addressed auto-association case which is perhaps a purer example of connectionist "memory" (see Rumelhart, Hinton, & McClelland, 1986).

We start with the simplest possible case.

THEOREM 3.8.1: Suppose $\psi_{F/R}$ is a tensor product representation of S induced by a decomposition with single-valed roles, with representations of fillers and roles in which all filler vectors are mutually orthogonal as are all role vectors. Let $\{s^{(k)}\}$ be a subset of S, and let the vectors representing these structures, $\{\mathbf{s}^{(k)}\}$, be associated in a connectionist network using the Hebb rule with the patterns $\{\mathbf{t}^{(k)}\}$. Then if the structures $\{s^{(k)}\}$ share no common fillers (i.e., for each role, all structures have different fillers), the associator will function perfectly; otherwise there will be cross-talk that is monotonic in the degree of shared fillers. In particular, the output associated with $\mathbf{s}^{(l)}$ is proportional to

$$\mathbf{t}^{(l)} + \sum_{k\neq l}\mu_{lk}\,\mathbf{t}^{(k)}$$

where

$$\mu_{lk} = \frac{\sum_{i:f_i^{(l)}=f_i^{(k)}} \|\mathbf{f}_i\|^2 \|\mathbf{r}_i\|^2}{\sum_i \|\mathbf{f}_i\|^2 \|\mathbf{r}_i\|^2}$$

PROOF: The Hebbian weights are

$$\mathbf{W} = \sum_k \mathbf{t}^{(k)} \mathbf{s}^{(k)T}$$

Thus the output generated from the input representing $s^{(l)}$ is (using Theorem 0 (c) from the Appendix):

$$\begin{aligned}
\mathbf{W}\mathbf{s}^{(l)} &= \sum_k \mathbf{t}^{(k)} \mathbf{s}^{(k)T} \mathbf{s}^{(l)} \\
&= \sum_k \mathbf{t}^{(k)} \left[\sum_i \mathbf{f}_i^{(k)} \otimes \mathbf{r}_i\right] \cdot \left[\sum_j \mathbf{f}_j^{(l)} \otimes \mathbf{r}_j\right] \\
&= \sum_k \mathbf{t}^{(k)} \sum_i \sum_j (\mathbf{f}_i^{(k)} \cdot \mathbf{f}_j^{(l)})(\mathbf{r}_i \cdot \mathbf{r}_j) \\
&= \sum_k \mathbf{t}^{(k)} \sum_i \sum_j (\delta_{f_i^{(k)}, f_j^{(l)}} \|\mathbf{f}_i^{(k)}\|^2)(\delta_{ij} \|\mathbf{r}_i\|^2) \\
&= \sum_k \mathbf{t}^{(k)} \sum_i \delta_{f_i^{(k)}, r_j^{(l)}} \|\mathbf{f}_i^{(k)}\|^2 \|\mathbf{r}_i\|^2 \\
&= \left[\sum_i \|\mathbf{f}_i\|^2 \|\mathbf{r}_i^{(l)}\|^2\right] \mathbf{t}^{(l)} + \sum_{k \neq l} \left[\sum_i \|\mathbf{f}_i\|^2 \|\mathbf{r}_i^{(l)}\|^2 \delta_{f_i^{(k)}, f_j^{(l)}}\right] \mathbf{t}^{(k)}
\end{aligned}$$

The first term here is the correct associate $\mathbf{t}^{(l)}$ weighted by a positive coefficient. The second term is a sum of all other (incorrect) associates $\{\mathbf{t}^{(k)}\}_{k \neq l}$, each weighted by a non-negative coefficient. These coefficients will all vanish if there are no common fillers. Taking the ratio of the coefficient of $\mathbf{t}^{(k)}$ to that of $\mathbf{t}^{(l)}$ gives the desired result. []

The Hebb rule is capable of accurately learning associations to patterns that are orthogonal. If the patterns are not necessarily orthogonal but are still linearly independent, the associations can be accurately stored in a connectionist memory using the more complex Widrow-Hoff (1960) or delta learning procedure (Rumelhart, Hinton, & McClelland, 1986). So the question is, what collection of symbolic structures have linearly independent representations under the tensor product representation? To answer this question, it turns out to be important to define the following concept:

DEFINITION 3.8.2: Let F/R be a role decomposition of S and let $k \mapsto s^{(k)}$ be a sequence of elements in S. An *annihilator* of $k \mapsto s^{(k)}$ with respect to R/F is a sequence of real numbers $k \mapsto \alpha^{(k)}$, not all zero, such that, for

all fillers $f \in F$, and all roles $r \in R$,

$$\sum_{k:\, f/r \in \beta(s^{(k)})} \alpha^{(k)} = 0.$$

For example, consider the sequence of strings (ax, bx, ay, by). With respect to the positional role decomposition, this has a total annihilator $(+1, -1, -1, +1)$, since for each filler/role binding in $\{a/r_1, b/r_1, x/r_2, y/r_2\}$, the corresponding annihilator elements are $\{+1, -1\}$, which sum to zero.

THEOREM 3.8.3: Suppose ψ is a tensor product representation of the structures S, and that $k \mapsto s^{(k)}$ is a sequence of distinct elements in S. Suppose that the filler vectors $\mathbf{f}$ representing the fillers bound in the elements $\{s^{(k)}\}$ are all linearly independent, and that the same is true of the role vectors $\mathbf{r}$ representing the roles bound in the elements $s^{(k)}$. If $k \mapsto s^{(k)}$ has no annihilator with respect to F/R, then associations to the tensor product representations $\{\psi(s_i)\}$ can all be simultaneously and accurately stored in a connectionist memory by using the Widrow-Hoff learning rule.

PROOF: Let

$$\psi(s^{(k)}) = \sum_i \mathbf{f}_i^{(k)} \otimes \mathbf{r}_i$$

Here we use the same set of roles $\{\mathbf{r}_i\}$ for all structures $\{s^{(k)}\}$; by Theorem 0 (b), this can always be done provided we allow the filler vector $\mathbf{f}_i^{(k)}$ to equal the zero vector whenever the role r_i is unbound in structure $s^{(k)}$.

By the remarks immediately preceding Definition 3.8.2, it is sufficient to show that the patterns $\{\psi(s^{(k)})\}$ are all linearly independent. Suppose on the contrary that there are coefficients $\{\alpha^{(k)}\}$, not all zero, such that

$$\mathbf{0} = \sum_k \alpha^{(k)} \psi(s^{(k)}) = \sum_k \alpha^{(k)} \left[\sum_i \mathbf{f}_i^{(k)} \otimes \mathbf{r}_i \right] = \sum_i \left[\sum_k \alpha^{(k)} \mathbf{f}_i^{(k)} \right] \otimes \mathbf{r}_i$$

Then, as shown in Theorem 5.3.4 (a) of the Appendix, because the role vectors $\{\mathbf{r}_i\}$ are linearly independent, this implies that for all i,

$$\sum_k \alpha^{(k)} \mathbf{f}_i^{(k)} = \mathbf{0}$$

Now we rewrite this as a sum over all distinct filler vectors:

$$\sum_\gamma \mathbf{f}_\gamma \sum_{k:\, \mathbf{f}_i^{(k)} = \mathbf{f}_\gamma} \alpha^{(k)} = \mathbf{0}$$

But since the filler vectors $\{\mathbf{f}_\gamma\}$ are linearly independent, this implies, for all i and for all γ,

$$\sum_{k:\, \mathbf{f}_i^{(k)} = \mathbf{f}_\gamma} \alpha^{(k)} = 0$$

This means exactly that $\{\alpha^{(k)}\}$ is an annihilator of the sequence of structures $k \mapsto s^{(k)}$. Since by hypothesis such an annihilator does not exist, it must be that the representations $\{\psi(s^{(k)})\}$ are linearly independent. ∎

It was remarked above that the strings $\{ax, bx, ay, by\}$ possess an annihilator with respect to the positional role decomposition. This means that the tensor product representations of these strings are not linearly independent, even under the preceding theorem's assumptions of linearly independent filler and role vectors. They cannot therefore be accurately associated with arbitrary patterns even using the Widrow-Hoff learning rule. On the other

hand, it is easy to see that the strings $\{ax, bx, ay\}$ do *not* possess an annhilator; the preceding theorem shows that they can therefore be accurately associated with any patterns.

3.9. Learning optimal role representations

The tensor product representation is constructed from connectionist representations of fillers and roles. As indicated in Section 2.3.2.3, distributed representation of fillers has been used in many connectionist models for some time; usually, these representations are built from an analysis of the fillers in terms of features relevant for the task being performed. But what about distributed representations of roles? This is basically a new problem raised by the tensor product representation. For many applications, it is easy to imagine task-appropriate features for roles that could serve well as the basis for distributed role representations. For example, Fig. 3 shows a distributed representation of positional roles with the useful property that nearby positions are represented by similar patterns.

In this section I will examine the question of distributed representations for roles from a domain-independent perspective. I will characterize one rather general sense in which a set of role vectors might be considered "optimal." Then I will analyse this criterion, and finally show how a connectionist network could learn such optimal representations itself.

3.9.1. Optimal role representations

Suppose that F/R is a faithful role decomposition of S with a finite set of roles, $\{r_i\}_{i=1}^{N}$. Suppose we are to represent these roles using a set of n role units, where $n<N$. What role vectors in the n-dimensional role space V_R should be used to represent the N roles? What we have is essentially a compression of information, and the question is: How can the n-dimensional space be used to represent N roles with the minimal loss of information?

How can we measure the information available to a connectionist system using the tensor product representation with a given representation of the roles? One way is to have a network of connections attempt to extract the N roles out of the pattern, and see how close it can come to the original. Fig. 13 illustrates this method, which is a variation on approaches that have been pursued by a number of connectionist researchers. Grossberg has for a long time used similar approaches for studying the learning of codes in connectionist systems (see, eg., Grossberg, 1982). Closely related methods have been embodied in Boltzmann "encoder" networks (see, eg., Ackley, Hinton & Sejnowski, 1985) and further explored with the back-propogation learning rule (Rumelhart, Hinton, & Williams, 1986). The verb-learning model of Rumelhart and McClelland (1986) also involved a similar scheme. The study by Williams (1985) is directly relevant: certain special cases of results from that study are used as lemmas below. (The theorems presented below are, to the best of my knowledge, new.)

Consider the tensor product representation of s. It is a pattern of activity over a matrix of connectionist units, as in Fig. 3. Since all filler units are treated equally, for the purposes of analyzing role representations we can just arbitrarily pick one, $\tilde{f}_{\phi_0}$, and focus exclusively on it. This amounts to focussing on a single row of the matrix of binding units, $\{\tilde{b}_{\phi_0\rho}\}_{\rho=1}^{n}$. This is the middle row of Fig. 13. In the representation of s, the pattern of activity on this row of binding units can be constructed as follows. For each role r_i there is a corresponding "input" unit $\tilde{i}_i$ in the bottom row of units in Fig. 13. The activity of input unit $\tilde{i}_i$ is the activity of the chosen filler unit $\tilde{f}_{\phi_0}$ in the pattern $\mathbf{f}_i$ representing the filler of role r_i

$$\tilde{i}_i = (\mathbf{f}_i)_{\phi_0}$$

The components of the role vectors determine the strengths of the connections between the input units and the binding units in Fig. 13. In particular, the strength $W_{\rho i}$ of the connection from $\tilde{i}_i$ to $\tilde{b}_{\phi_0\rho}$ is the ρ-component of the

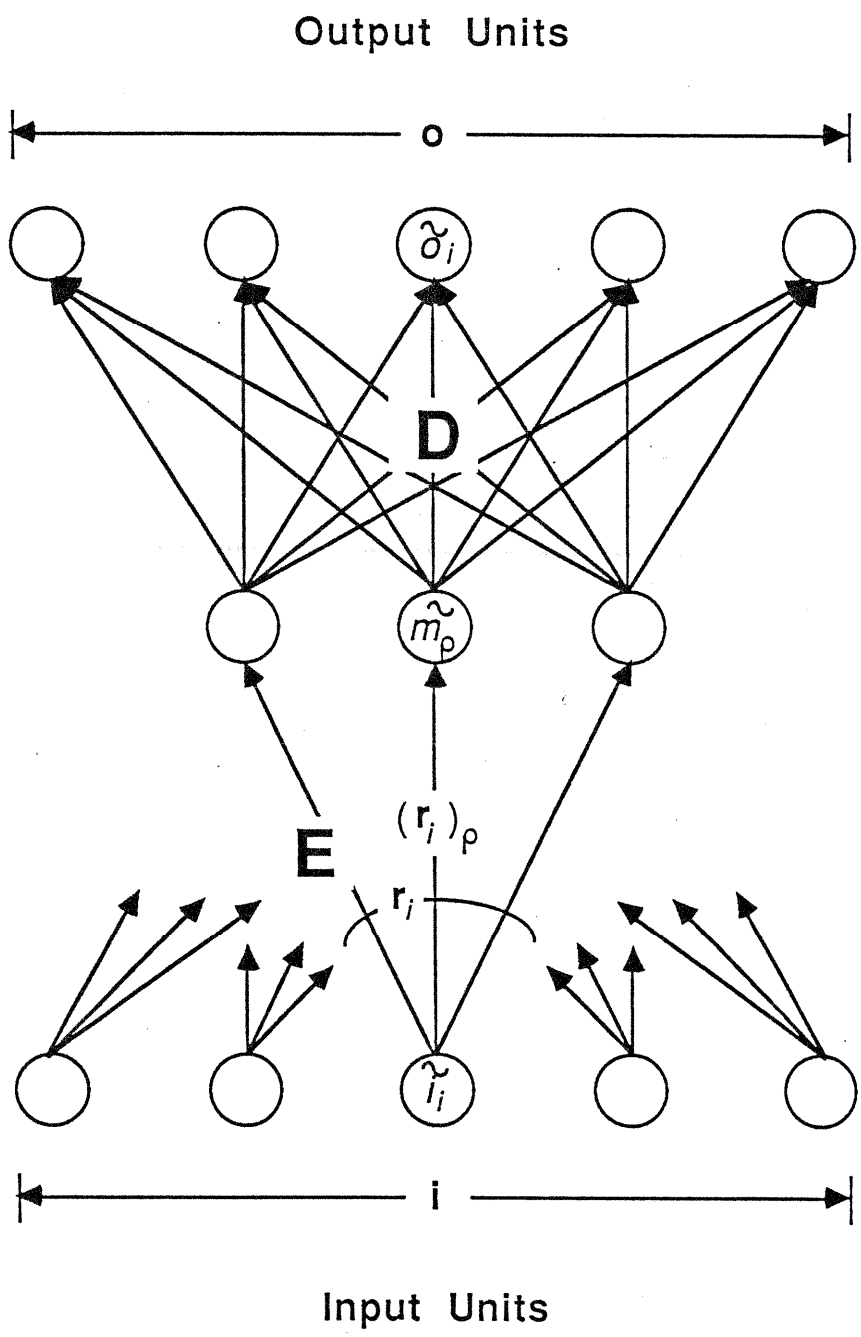


Figure 13. A network for studying the information available to connectionist processing in tensor product representations using a set of role vectors $\{\mathbf{r}_i\}$.

vector $\mathbf{r}_i$ representing role r_i:

$$W_{\rho i} = (\mathbf{r}_i)_\rho$$

The units in the middle layer of Fig. 13 are linear units whose value is the weighted sum of their inputs. Therefore the activity of the ρ^{th} middle unit, $\tilde{m}_\rho$, is:

$$\tilde{m}_\rho = \sum_i W_{\rho i}\tilde{i}_i = \sum_i (\mathbf{r}_i)_\rho (\mathbf{f}_i)_{\phi_0} = \left[\sum_i \mathbf{f}_i \otimes \mathbf{r}_i\right]_{\phi_0 \rho} = \tilde{b}_{\phi_0 \rho}$$

Thus we see that the activity of the middle layer is indeed the pattern of activity of row ϕ_0 in the tensor product representation. (In fact, the network of Fig. 13 is equivalent to the ϕ_0 portion of the network of Fig. 10. Here we assume there are enough pools of units in Fig. 10 to dedicate one pool to each role and thereby to generate all bindings in parallel. Unit $\tilde{i}_i$ in Fig. 13 corresponds to unit $\tilde{f}_{\phi_0^{(i)}}$ of Fig. 10. With a given set of role vectors, each placed in a fixed pool of role units, the activity of the units $\tilde{r}_\rho^{(i)}$ in Fig. 10 is constant, and instead of using these activities as multiplication factors at the sites of the sigma-pi units, they are used as connection weights in Fig. 13.)

The weights between the lower and middle layers of Fig. 13 use the role vectors to set up (the ϕ_0 part of) the tensor product representation of s in the middle layer. This matrix of weights determined by the role vectors will be called the *encoding matrix* **E**. In order to measure the amount of information in the tensor product representation that is available to connectionist processing, there is a second layer of connections in Fig. 13 that attempts to extract the original information. The matrix of weights from the middle to the upper layer will be called the *decoding matrix* **D**. Each unit $\tilde{o}_i$ in the upper layer of Fig. 13 attempts to compute the activity of the corresponding input unit $\tilde{i}_i$ in the lower layer. Perfect performance would occur if the pattern of activity of the top layer exactly matched that of the bottom layer. In this case, the compression of information through the middle layer would not lose any information. As a measure of the amount of information lost, it is natural to compute the sum-squared error in the output pattern **o** as a copy of the input pattern **i**:

$$E(\mathbf{i}) = \sum_{i=1}^{N} (\tilde{o}_i - \tilde{i}_i)^2 = \|\mathbf{o}-\mathbf{i}\|^2$$

This error, of course, depends on the input pattern i; as an overall measure of the error incurred in the encoding/decoding process, we can average over input patterns.

THEOREM 3.9.1.1: Suppose we randomly generate input patterns **i**, with the activities of the input units independent, identically distributed random variables with mean 0 and variance v^2. Then the expected value of $E(\mathbf{i})$ is

$$v^2 \sum_{i=1}^{N} \|\mathbf{DE}\hat{\mathbf{e}}_i - \hat{\mathbf{e}}_i\|^2$$

Here, $\hat{\mathbf{e}}_i$ is the input pattern in which input unit $\tilde{i}_i$ has activity 1 while all other input units have activity 0.

PROOF: Expand the input vector **i** in the basis $\hat{\mathbf{e}}_i$:

$$\mathbf{i} = \sum_i v_i \hat{\mathbf{e}}_i$$

Then

$$
\begin{aligned}
E(\mathbf{i}) &= \|\mathbf{o}-\mathbf{i}\|^2 \\
&= \|\mathbf{DEi}-\mathbf{i}\|^2 \\
&= \|(\mathbf{DE}-\mathbf{1})\mathbf{i}\|^2 \\
&= \|(\mathbf{DE}-\mathbf{1})\sum_{i=1}^{N} v_i \hat{\mathbf{e}}_i\|^2 \\
&= \|\sum_{i=1}^{N} v_i (\mathbf{DE}-\mathbf{1})\hat{\mathbf{e}}_i\|^2 \\
&= \left[\sum_{i=1}^{N} v_i (\mathbf{DE}-\mathbf{1})\hat{\mathbf{e}}_i\right] \cdot \left[\sum_{j=1}^{N} v_j (\mathbf{DE}-\mathbf{1})\hat{\mathbf{e}}_j\right] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} v_i v_j (\mathbf{DE}-\mathbf{1})\hat{\mathbf{e}}_i \cdot (\mathbf{DE}-\mathbf{1})\hat{\mathbf{e}}_j
\end{aligned}
$$

When this quantity is averaged, the terms with $i \neq j$ vanish: since the random variables v_i are independent, the expected value of $v_i v_j$ if $i \neq j$ is the expected value of v_i times that of v_j; both these are 0 since all v_i have mean 0. What remains are the terms with $i = j$; for these terms, the expected value of $v_i v_j$ is the variance v^2 (again, because all v_i have mean 0), and the dot-product of the vectors is just their squared length. This gives the desired result. []

The preceding result motivates the following definition:

DEFINITION 3.9.1.2: The *expected error* for the network of Fig. 13 is defined to be

$$E = \sum_{i=1}^{N} \|\mathbf{DE}\hat{\mathbf{e}}_i - \hat{\mathbf{e}}_i\|^2$$

The expected error depends both on the encoding weights **E** and the decoding weights **D**. For present purposes, I will consider a set of role vectors to be "optimal" if it permits the least possible expected error.

DEFINITION 3.9.1.3: A set of role vectors determines an encoding matrix **E**. For a given **E**, let the *minimal error* be the smallest value of E that can be obtained by varying the decoding matrix **D**. A set of role vectors is *optimal* iff no other set of role vectors has a lower minimal error.

Before proceeding to the analysis of optimal role vectors, it is helpful to change the network of Fig. 13 to the one shown in Fig. 14. The upper layer of Fig. 13 has been identified with the lower layer, so that instead of a three-layer, feed-forward network we now have a two-layer, feed-back network. The weights are unchanged: **E** is the matrix of weights from the lower layer to the upper layer of Fig. 14, and **D** is the matrix of weights from the upper to the lower layer. I will refer to networks like those of Fig. 14 as *encoding/decoding networks*.

The definition of the expected error E is formally unchanged, although the interpretation is slightly different: the matrix **DE** now represents the passage of activity up from the lower to the upper layer and then back down to the lower layer again.

THEOREM 3.9.1.4: It is always possible to find a set of optimal role vectors for which the connections in the corresponding encoding/decoding network are symmetric.

PROOF: For the purposes of this proof and the following one, let **L** = **DE**. **L** is so named because it is the

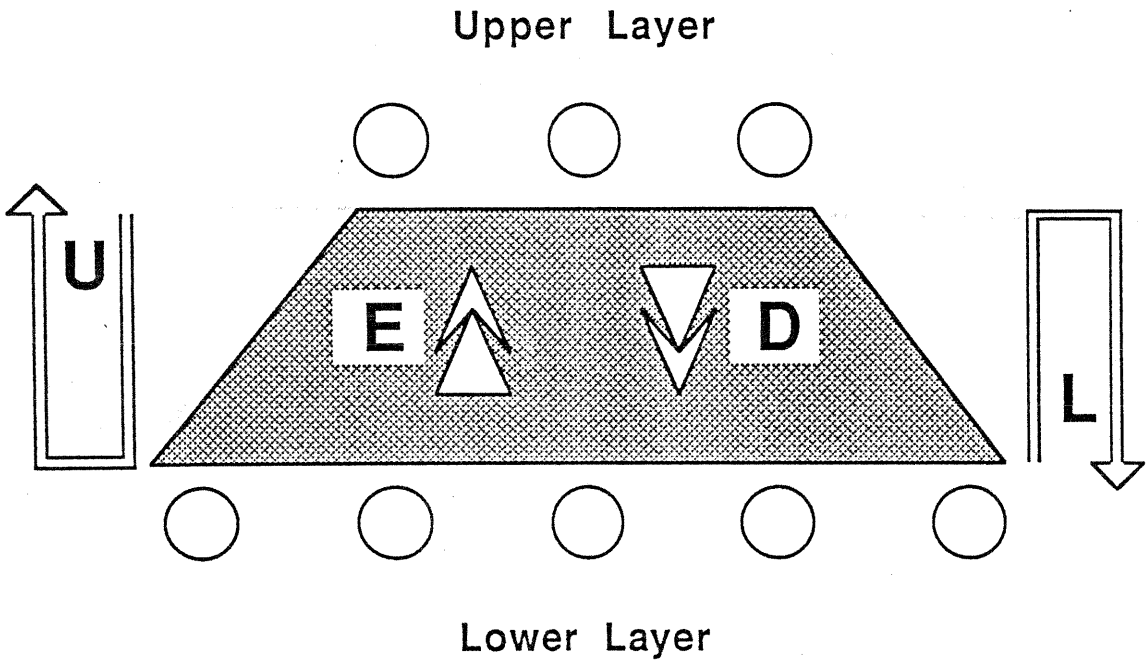


Figure 14. The network of Fig. 13 with corresponding input and output units identified: an encoding/decoding network.

lower cycle: it takes activity from the lower layer, passes it to the upper layer, and then back down again to the lower one. This cycle determines the expected error, so the expected error can be written in terms of **L**:

$$E = \sum_{i=1}^{N} \| \mathbf{L}\hat{\mathbf{e}}_i - \hat{\mathbf{e}}_i \|^2$$

$$= \sum_i \| (\mathbf{L}-1)\hat{\mathbf{e}}_i \|^2$$

$$= \sum_i (\mathbf{L}-1)\hat{\mathbf{e}}_i \cdot (\mathbf{L}-1)\hat{\mathbf{e}}_i$$

$$= \sum_i \hat{\mathbf{e}}_i \cdot (\mathbf{L}-1)^T (\mathbf{L}-1)\hat{\mathbf{e}}_i$$

$$= \mathrm{Tr}(\mathbf{L}-1)^T (\mathbf{L}-1)$$

Here Tr denote the trace operation. What we show next is that for optimal role vectors, **L** is symmetric. To this end we express **L** in terms of its symmetric and anti-symmetric parts S and A:

$$\mathbf{S} := 1/2(\mathbf{L} + \mathbf{L}^T);\ \mathbf{A} := 1/2(\mathbf{L} - \mathbf{L}^T) \Rightarrow \mathbf{L} = \mathbf{S} + \mathbf{A};\ \mathbf{S}^T = \mathbf{S};\ \mathbf{A}^T = -\mathbf{A};\ \mathbf{SA} = \mathbf{AS}$$

Now we express the error in terms of **S** and **A**:

$$E = \mathrm{Tr}(\mathbf{L}-1)^T (\mathbf{L}-1)$$

$$= \mathrm{Tr}(\mathbf{L}^T\mathbf{L} - [\mathbf{L}^T + \mathbf{L}] + 1)$$

$$= \mathrm{Tr}([\mathbf{S}^T+\mathbf{A}^T][\mathbf{S}+\mathbf{A}] - 2\mathbf{S} + 1)$$

$$= \mathrm{Tr}(\mathbf{S}^T\mathbf{S}+[\mathbf{A}^T\mathbf{S}+\mathbf{S}^T\mathbf{A}]+\mathbf{A}^T\mathbf{A} - 2\mathbf{S} + 1)$$

$$= \mathrm{Tr}(\mathbf{S}^T\mathbf{S}+[-\mathbf{AS}+\mathbf{SA}]+\mathbf{A}^T\mathbf{A} - 2\mathbf{S} + 1)$$

$$= \mathrm{Tr}(\mathbf{S}^T\mathbf{S}+\mathbf{S}[-\mathbf{A}+\mathbf{A}]+\mathbf{A}^T\mathbf{A} - 2\mathbf{S} + 1)$$

$$= \mathrm{Tr}(\mathbf{S}^T\mathbf{S}+\mathbf{A}^T\mathbf{A} - 2\mathbf{S} + 1)$$

$$= \mathrm{Tr}(\mathbf{S}^T\mathbf{S} - 2\mathbf{S} + 1) + \mathrm{Tr}\mathbf{A}^T\mathbf{A}$$

Thus the error is the sum of a term depending on the symmetric part of **L** and a term depending on the antisymmetric part. The latter is positive-definite:

$$\mathrm{Tr}\mathbf{A}^T\mathbf{A} = \sum_i \sum_j (\mathrm{A}_{ij})^2$$

Thus if **L** is not already symmetric, replacing **L** by its symmetric part **S**—i.e. setting the antisymmetric part **A** to zero—will lower the error. Hence for optimal role vectors, **L** must be symmetric.

From the symmetry of **L** follows a Lemma appearing in Williams (1985):

LEMMA 3.9.1.5: The minimal possible error is $N-n$. The error will be minimized iff **L** is an orthogonal projection onto a subspace of dimension n.

PROOF: Since $\mathbf{L}$ is symmetric:

$$E = \mathrm{Tr}(\mathbf{L}-\mathbf{1})^T(\mathbf{L}-\mathbf{1}) = \mathrm{Tr}(\mathbf{L}-\mathbf{1})^2$$

This trace can be evaluated in any orthonormal basis we like. We can choose a basis in which $\mathbf{L}$ is diagonal; this basis is an orthonormal set of eigenvectors of $\mathbf{L}$, which exists because $\mathbf{L}$ is symmetric. In this basis we have

$$E = \sum_{i=1}^{N}(L_{ii}-1)^2$$

To minimize the error, we therefore want as many as possible of the diagonal elements of $\mathbf{L}$ to equal 1. The maximum number possible is n, because $\mathbf{L} = \mathbf{DE}$, and $\mathbf{E}$ is a linear map into an n-dimensional space; thus the range of $\mathbf{L}$ is at most n dimensional (the rank of $\mathbf{L}$ is at most n). The remaining $N-n$ diagonal elements of $\mathbf{L}$ must be zero. It follows that $E = N-n$. We now know what $\mathbf{L}$ looks like. In the basis in which it is diagonal, there are n basis vectors on which $\mathbf{L}$ is the identity operator and $N-n$ basis vectors on which $\mathbf{L}$ is the zero operator. Thus $\mathbf{L}$ is an orthogonal projection onto an n-dimensional subspace. ∎

A geometrical picture of Lemma 3.9.1.5 (Ron Williams, personal communication, 1986) is illustrated in Fig. 15. Each term in the error is the squared-length of the difference vector between $\hat{\mathbf{e}}_i$ and its image under $\mathbf{L}$. The image is constrained to lie in the range of $\mathbf{L}$, a subspace of dimension at most n. To minimize the error, the range of $\mathbf{L}$ should be as large as possible: of dimension n; also, the image of $\hat{\mathbf{e}}_i$ should be as close as possible to $\hat{\mathbf{e}}_i$ while still in the range of $\mathbf{L}$: the image should be the orthogonal projection of $\hat{\mathbf{e}}_i$ onto the subspace. Thus $\mathbf{L}$ should be an orthogonal projection onto an n-dimensional subspace.

Now that we know $\mathbf{L}$ is an orthogonal projection, it follows that $\mathbf{L}$ is a non-negative definite symmetric matrix of rank n. Any such matrix has a decomposition

$$\mathbf{L} = \mathbf{W}^T\mathbf{W}$$

where $\mathbf{W}$ has rank n. By choosing n basis vectors in the range of $\mathbf{W}$ we can take $\mathbf{W}$ to be an $n\times N$ matrix. Thus the matrix $\mathbf{L} = \mathbf{DE}$ can alternately be decomposed $\mathbf{L} = \mathbf{W}^T\mathbf{W}$, i.e. with the $n\times N$ encoding matrix $\mathbf{E}$ replaced by $\mathbf{W}$ and the $N\times n$ decoding matrix $\mathbf{D}$ replaced by $\mathbf{W}^T$. Since the new $\mathbf{E} = \mathbf{W}$ and $\mathbf{D} = \mathbf{W}^T$ are transposes of each other,

$$\mathbf{E}_{pi} = \mathbf{D}_{ip}$$

and the new encoding/decoding network has symmetric connections. The new network has the same expected error as the old one, since $\mathbf{L}$ is unchanged and E is determined by $\mathbf{L}$. ∎

Because of this theorem, I will assume henceforth that symmetry is part of the definition of an encoding/decoding network.

The next result gives a geometrical characterization of the optimality condition. It relies on viewing the role vectors as defining a coordinate system in the role pattern space $\mathbb{R}^n$, the space of patterns in the upper layer of the encoding/decoding network. In an ordinary Cartesian coordinate system defined by an orthonormal basis $\hat{\mathbf{e}}_i$ (eg., the basis of that name defined above for the lower layer pattern space $\mathbb{R}^N$), the i^{th} coordinate of the point at the tip

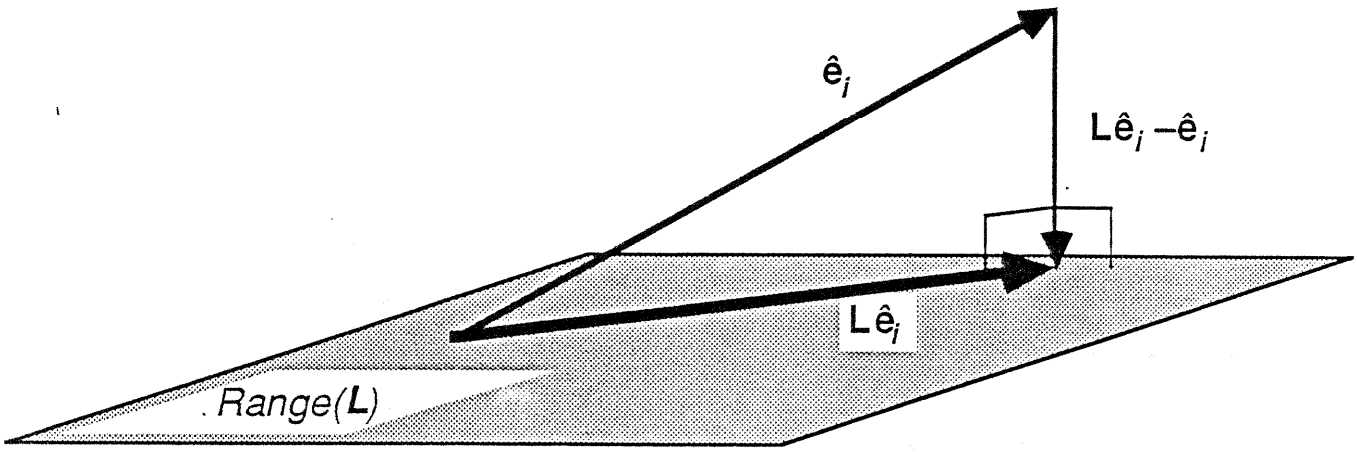


Figure 15. The geometrical picture of Lemma 3.9.1.5.

of the vector $\mathbf{v}$ is $x_i = \mathbf{v}\cdot\hat{\mathbf{e}}_i$. Suppose we use the N role vectors $\{\mathbf{r}_i\}_{i=1}^N$ in this way to define N coordinates $\{\xi_i\}_{i=1}^N$:

$$\xi_i(\mathbf{v}) = \mathbf{v}\cdot\mathbf{r}_i$$

These coordinates allow us to describe patterns in terms of their orthogonal projections along all the role directions. Of course these coordinates are not all independent: there are N of them, but only n dimensions in the space of role patterns. Using the coordinates $\{\xi_i\}$, we can describe the optimality condition:

THEOREM 3.9.1.6: Using the coordinates $\{\xi_i\}_{i=1}^N$, define a putative inner product on $\mathbb{R}^n$ by

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{N} \xi_i(\mathbf{u})\xi_i(\mathbf{v})$$

Then the role vectors are optimal iff this putative inner product equals the canonical inner product. Equivalently, the role vectors are optimal iff the rows of the matrix $\mathbf{E}$ are orthonormal.

PROOF: First we need a second lemma appearing in Williams (1985). Define $\mathbf{U} = \mathbf{ED}$; $\mathbf{U}$ is so called because it is the *upper cycle*: it takes activity at the upper layer, sends it down to the lower layer, and then sends it back up the upper layer.

LEMMA 3.9.1.7: The error is minimized iff $\mathbf{U} = \mathbf{1}$. This holds iff the rows of $\mathbf{E}$ are orthonormal.

PROOF: By Lemma 3.9.1.5, $\mathbf{L}$ is a projection, so we have $\mathbf{L}^2 = \mathbf{L}$. (This is trivially verified using the basis in which $\mathbf{L}$ is diagonal: each diagonal element is either 0 or 1.) Thus:

$$\mathbf{DE} = \mathbf{L} = \mathbf{L}^2 = \mathbf{DEDE} = \mathbf{DUE}$$

From this we show $\mathbf{U} = \mathbf{1}$. By Lemma 3.9.1.5, $\mathbf{L}$ is of rank n, so the range of $\mathbf{E}$ is all of $\mathbb{R}^n$. Thus any vector $\mathbf{v}$ in $\mathbb{R}^n$ (the upper layer pattern space) is the image of some $\mathbf{u}$ in $\mathbb{R}^N$ (the lower layer pattern space):

$$\mathbf{v} = \mathbf{Eu} \Rightarrow \mathbf{Uv} = \mathbf{UEu}$$

Now it follows from the previous two equations that both $\mathbf{v}$ and $\mathbf{Uv}$ have the same image under $\mathbf{D}$:

$$\mathbf{D(v)} = \mathbf{D(Eu)} = \mathbf{(DE)u} = \mathbf{(DUE)u} = \mathbf{D(UEu)} = \mathbf{D(Uv)}$$

It follows that $\mathbf{v}$ and $\mathbf{Uv}$ must be the same vector, since for any vector $\mathbf{w}$ in $\mathbb{R}^N$:

$$\mathbf{(Uv-v)\cdot Ew} = \mathbf{E}^T\mathbf{(Uv-v)} = \mathbf{D(Uv-v)\cdot w} = \mathbf{0\cdot w} = 0$$

Thus $\mathbf{Uv-v}$ is orthogonal to the entire range of $\mathbf{E}$, which is all of $\mathbb{R}^n$: it must therefore be $\mathbf{0}$. Since this is true for all $\mathbf{v} \in \mathbb{R}^n$, it follows that

$$\mathbf{U} = \mathbf{1}$$

Evaluating the previous equation to see its significance for **E**, we find:

$$\delta_{\rho\rho'} = \mathbf{U}_{\rho\rho'} = (\mathbf{ED})_{\rho\rho'} = \sum_i (\mathbf{E})_{\rho i}(\mathbf{D})_{i\rho'} = \sum_i (\mathbf{E})_{\rho i}(\mathbf{E}^T)_{i\rho'} = \sum_i \mathrm{E}_{\rho i}\mathrm{E}_{\rho' i}$$

For $\rho \neq \rho'$, this says that the ρ and ρ' rows of **E** are orthogonal; for $\rho = \rho'$, it says that row ρ has norm one. In other words, the rows of **E** are orthonormal. ▯

From Lemma 3.9.1.7 it follows that the inner product defined by $\{\xi_i\}$ is the same as the canonical one:

$$\begin{aligned}
\langle \mathbf{v}, \mathbf{w} \rangle &= \sum_{i=1}^{N} \xi_i(\mathbf{v})\xi_i(\mathbf{w}) \\
&= \sum_{i=1}^{N} \mathbf{v}\cdot\mathbf{r}_i\, \mathbf{w}\cdot\mathbf{r}_i \\
&= \sum_i \Big[\sum_\rho \mathrm{v}_\rho \mathrm{E}_{\rho i}\Big]\Big[\sum_{\rho'} \mathrm{w}_{\rho'} \mathrm{E}_{\rho' i}\Big] \\
&= \sum_\rho \sum_{\rho'} \mathrm{v}_\rho \mathrm{w}_{\rho'} \sum_i \mathrm{E}_{\rho i}\mathrm{E}_{\rho' i} \\
&= \sum_\rho \sum_{\rho'} \mathrm{v}_\rho \mathrm{w}_{\rho'} \delta_{\rho\rho'} \\
&= \sum_\rho \mathrm{v}_\rho \mathrm{w}_\rho \\
&= \mathbf{v}\cdot\mathbf{w}
\end{aligned}$$

▯

The previous result can be paraphrased as follows. If the role space were large enough to accomodate all the role vectors, if n were greater than N, we could choose the role vectors to be orthonormal, and then base a Cartesian coordinate system on them; with respect to these Cartesian coordinates, the canonical inner product would be the usual sum of products of corresponding coordinates. But in the present case, we are trying to squeeze too many role vectors into the role space: $n < N$, and we can't choose orthonormal role vectors. Yet if the role vectors we choose are optimal, we can still go ahead and define a coordinate system using them, and correctly compute the inner product by the sum of products of corresponding coordinates.

Several examples of optimal role vectors in $\mathbb{R}^2$ are shown in Fig. 16. In each case, the corresponding encoding matrix **E** is also shown; it can be verified that the rows are orthonormal. In one case, the coordinates $\{\xi_i\}$ are also shown.

3.9.2. Connectionist learning of optimal role representations by recirculation

The encoding/decoding network of the previous section imbeds the role vectors as weights in a connectionist network. This makes it possible for the network to learn these vectors through a weight-change procedure.

DEFINITION 3.9.2.1: Let an encoding/decoding network be given as in Fig. 14. Then the *recirculation algorithm* for modifying the weights is as follows. The first unit in the lower layer is given activity 1, and all other units are given activity 0: this is the "input pattern." This pattern of activity is passed to the upper layer through the connections, establishing a pattern $\mathbf{p}_0$. This pattern is passed back to the lower layer and then up again to the upper layer, forming pattern $\mathbf{p}_1$. Then the Widrow-Hoff (1960) or delta rule (Rumelhart, Hinton, & McClelland, 1986) is used to compute a weight change for each connection: $\mathbf{p}_0$ is treated as the "teaching"

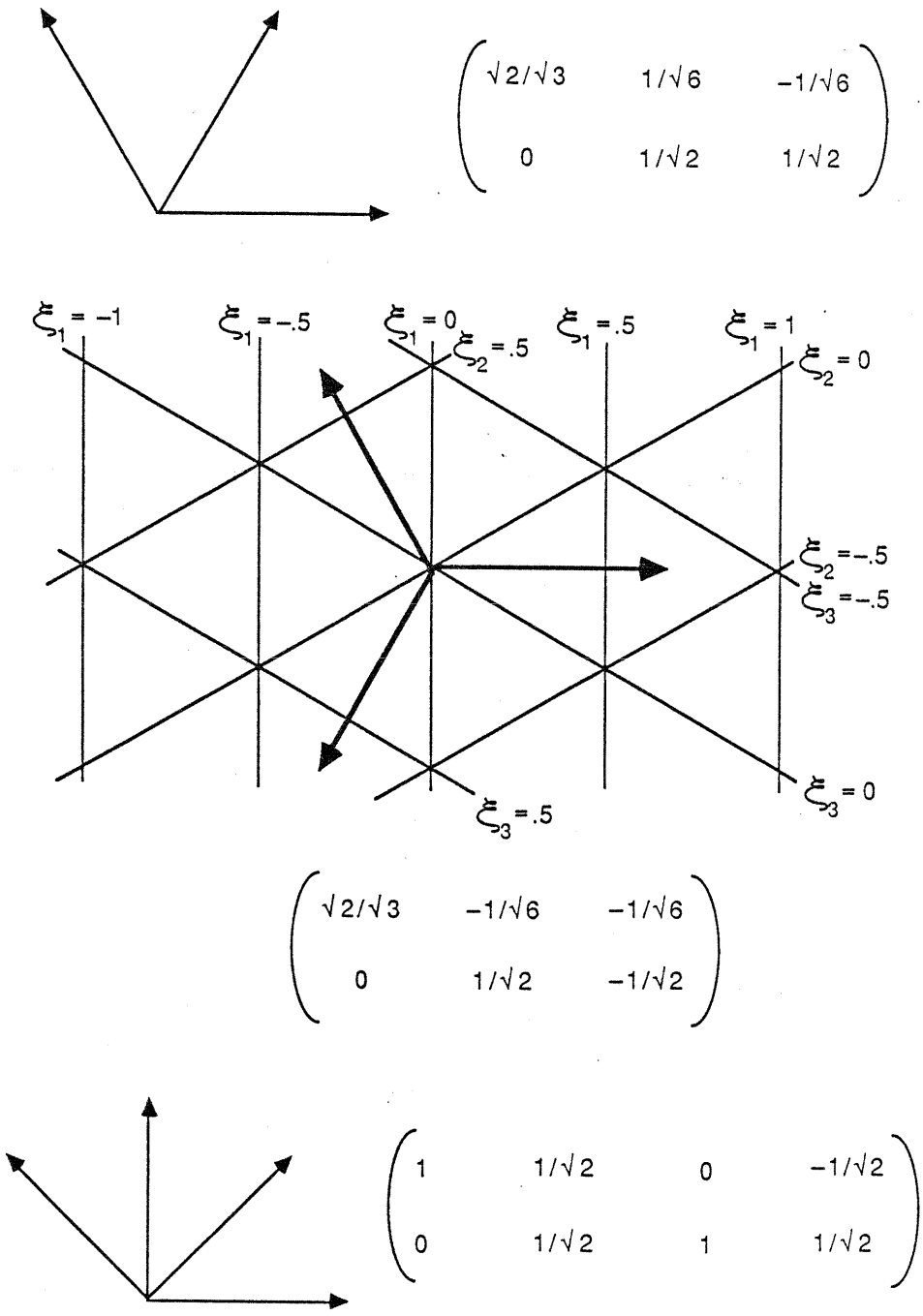


Figure 16. Examples of optimal sets of role vectors in $\mathbb{R}^2$. The role vectors are ordered counterclockwise starting with the vector pointing directly to the right.

pattern, $\mathbf{p}_1$ as the "current output" pattern, and the original "input pattern" is used. This weight change is recorded but not yet carried out. Next the second unit of the lower layer is individually activated, and the process repeated, and so on until all N lower units have been individually activated. Then the weight changes that have been recorded for each connection are added to the current weight and the whole process is repeated.

There is a stochastic counterpart to this recirculation algorithm, in which instead of activating an individual lower unit, a random initial pattern is generated for the entire lower layer. The recirculation and weight change is done as before, but now the weights changed on each trial. I will analyze only the non-stochastic version; however, under appropriate circumstances the stochastic algorithm can be expected to approximate the non-stochastic one since, as I now show, the non-stochastic algorithm performs gradient descent in E, and, as shown in Theorem 3.9.1.1, E is the expected value of the error arising from randomly chosen input patterns.

THEOREM 3.9.2.2: The recirculation algorithm performs a gradient descent in the expected error E. With sufficiently small learning coefficient it will converge to a locally optimal set of role vectors (a local minimum of E). If this set of role vectors spans all of $\mathbb{R}^n$ then it is a truly optimal set (a global minimum of E).

PROOF:

$$E = \sum_i \| \mathbf{L}\hat{\mathbf{e}}_i - \hat{\mathbf{e}}_i \|^2 = \sum_i \sum_j (\sum_\rho \mathrm{E}_{\rho j} \mathrm{E}_{\rho i} - \delta_{ij})^2$$

The component of the gradient of E in the direction of the weight $\mathrm{E}_{\rho' i'}$ is

$$\frac{\partial E}{\partial \mathrm{E}_{\rho' i'}} = 2 \sum_i \sum_j (\sum_\rho \mathrm{E}_{\rho j} \mathrm{E}_{\rho i} - \delta_{ij}) \frac{\partial}{\partial \mathrm{E}_{\rho' i'}} (\sum_{\rho''} \mathrm{E}_{\rho'' j} \mathrm{E}_{\rho'' i} - \delta_{ij})$$

$$= 2 \sum_i \sum_j (\sum_\rho \mathrm{E}_{\rho j} \mathrm{E}_{\rho i} - \delta_{ij}) (\mathrm{E}_{\rho' i} \delta_{i' j} + \mathrm{E}_{\rho' j} \delta_{i' i})$$

$$= 4 \sum_k \sum_\rho \mathrm{E}_{\rho i'} \mathrm{E}_{\rho k} \mathrm{E}_{\rho' k} - 4 \mathrm{E}_{\rho' i'}$$

$$= 4 \sum_k \sum_\rho \mathrm{E}_{\rho' k} \mathrm{D}_{k \rho} \mathrm{E}_{\rho i'} - 4 \mathrm{E}_{\rho' i'}$$

These two terms can be related to activities produced in the recirculation algorithm. Suppose we have activated unit i' in the lower layer with activity 1, and given all other lower units zero activation. Then the first term is the activity at unit ρ' in pattern $\mathbf{p}_1$, $(\mathbf{p}_1)_{\rho'}$, and the second term is the activity at unit ρ' in pattern $\mathbf{p}_0$, $(\mathbf{p}_0)_{\rho'}$. Thus we can write

$$-\frac{\partial E}{\partial \mathrm{E}_{\rho' i'}} = (\mathbf{p}_0)_{\rho'} - (\mathbf{p}_1)_{\rho'}$$

This can be identified as the Widrow-Hoff weight change for weight $\mathrm{E}_{\rho' i'}$,

$$\Delta \mathrm{E}_{\rho' i'} = g\,[(\textit{teaching pattern})_{\rho'} - (\textit{current output})_{\rho'}](\textit{input})_{i'}$$

if $\mathbf{p}_0$ is taken to be the teaching pattern and $\mathbf{p}_1$ the current output; for this input, $(\textit{input})_{i'} = 1$. (Here g is the learning coefficient.) In fact, since $(\textit{input})_j = 0$ if $j \neq i'$, the Widrow-Hoff rule produces zero weight change for all weights $\mathrm{E}_{\rho j}$ where $j \neq i'$. In other words, as we cycle through the units in the lower layer, individually activating each one, the Widrow-Hoff rule gives us weight changes for the connections emanating from the active unit, and these weight changes are along the direction of gradient descent in E. By waiting until the end of the complete sweep before adding in all the weight changes and updating the weights, we ensure that

the change is in the true gradient direction.

Writing the gradient in matrix notation, we have

$$¼\nabla E = \mathbf{EDE}-\mathbf{E} = (\mathbf{ED}-\mathbf{1})\mathbf{E} = (\mathbf{U}-\mathbf{1})\mathbf{E}$$

Assuming a sufficiently small learning coefficient g, the gradient descent will approach a local minimum of E where $\nabla E = \mathbf{0}$. If the role vectors span $\mathbb{R}^n$, $\mathbf{E}$ will be of full rank, and, as in the proof to Lemma 3.9.1.7, this allows us to conclude from $\mathbf{0} = \nabla E = (\mathbf{U}-\mathbf{1})\mathbf{E}$ that $\mathbf{U} = \mathbf{1}$. By Lemma 3.9.1.7, this shows the role vectors are optimal. Conversely, if the algorithm converges to a set of optimal role vectors, we also know from Lemma 3.9.1.7 that these role vectors must span $\mathbb{R}^n$. Thus the local minimum of E to which the algorithm converges will provide an optimal set of role vectors iff they span $\mathbb{R}^n$. □

4. Conclusion

The limitations of the results reported here are many. The theoretical analyses of role decompositions, graceful saturation, connectionist representations of symbolic operators and recursive structures, retrieval of tensor product representations in connectionist memories, and optimal role vectors have just begun. An analysis is needed of the consequences of throwing away binding units to control the potentially prohibitive growth in their number. A further analysis is needed of the possibility of having a value for one variable serve as another variable, without an unbinding of the first variable. The relations between tensor product binding units and connection weights, briefly considered in Section 3.5, need to be pursued. The tensor product representation needs to be tested out in real connectionist models in a variety of domains to see if the theoretical virtues of the representation can be cashed in practice. The recirculation algorithm for finding "optimal" role vectors needs to be explored to see whether it can really serve a valuable role within an actual model.

Nonetheless, the tensor product representation enables truly distributed representations of complex symbolic structures in connectionist systems, in a natural way that generalizes existing representations and is simple enough to permit analyses of a number of properties. Tensor product representations are determined by a number of constituents which can be productively analyzed separately: the role decomposition of the structures being represented, the method for connectionist representation of conjunction, and the connectionist representations of fillers and roles being used. Such conceptual tools for analyzing alternative connectionist representations are necessary if we are to deepen our understanding of the representational component of connectionist modeling. Most importantly, the tensor product representation allows a crucial element of symbolic computation, the binding of values to variables, to be incorporated into the connectionist approach in a natural way that adds to the power of connectionist computation without sacrificing its advantages.

5. Appendix: The Tensor Product

It is extremely simple to define the tensor product with respect to a basis. Suppose V and W are two vector spaces, with dimensions N_V and N_W and bases $\{\hat{\mathbf{v}}_i\}$ and $\{\hat{\mathbf{w}}_j\}$ respectively. Then the tensor product space $V \otimes W$ has dimension $N_V N_W$. The tensor product operation takes a vector $\mathbf{v}$ in V and a vector $\mathbf{w}$ in W to a vector $\mathbf{v} \otimes \mathbf{w}$ in $V \otimes W$. The vectors $\{\hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j\}$ form a basis for $V \otimes W$. If the components of $\mathbf{v}$ with respect to the basis $\{\hat{\mathbf{v}}_i\}$ are $\{\mathrm{v}_i\}$ and the components of $\mathbf{w}$ with respect to the basis $\{\hat{\mathbf{w}}_j\}$ are $\{\mathrm{w}_j\}$ then the components of $\mathbf{v} \otimes \mathbf{w}$ with respect to the basis $\hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j$ are $\mathrm{v}_i \mathrm{w}_j$.

It is often useful to have a definition of vector operations independent of the choice of any basis. This gives a deeper understanding of the operations from which the surface manifestations relative to particular bases can be *derived* rather than *stipulated*. In the case of the tensor product, the basis-independent definition is rather subtle.

The tensor product is one generalization of multiplication to the vector space setting. The definition is performed in three steps. First, a correspondence called *duality* is defined between vector spaces and certain spaces of functions. Next, the notion of tensor product is defined for these function spaces. Finally, the definition of tensor product is transfered back to the original vector spaces by again using the dual operation. The situation is summed

up in Fig. 17. The concepts introduced along the way, dual vector spaces and bilinear functionals, are quite important ones that merit attention in their own right. Dual vectors in particular are useful for performing unbinding in the tensor product representation of variable binding.

5.1. Dual vector spaces

The first concept we need is that of two vector spaces being *duals*. This derives from the general concept of the duality between a set X and a set of functions F on X. The key observation is that just as each function f in F is a mapping from X to Y:

$$f: x \mapsto f(x)$$

so each point x in X can be viewed as a mapping $\bar{x}$ from F to Y

$$\bar{x}: f \mapsto f(x)$$

The traditional notation $f(x)$ for function evaluation tends to hide this duality by treating f and x assymetrically; the duality can be brought out better by using the more symmetrical LISP notation $(f\ x)$.[4] In the remainder of this Appendix I will use the LISP notation at appropriate points.

The notion of duality, then, can be captured by this semi-formal definition:

DEFINITION 5.1.1: Suppose F is the set of all functions from X to Y satisfying some property p:

$$F := \{f: X \to Y; x \mapsto (f\ x) \,|\, p(f)\}$$

Now consider the functions

$$\bar{X} := \{\bar{x}: F \to Y \,|\, f \mapsto (f\ x)\}$$

Then F and X are *dual spaces* if $\bar{X}$ is characterized by the same property p that defines F:

$$\bar{X} = \{\bar{x}: F \to Y \,|\, p(\bar{x})\}$$

The relevant case here is that of vector spaces. The function space F will possess the vector space operations of addition and scalar multiplication if the range of the functions, Y, has these operations. The simplest case is $Y = \mathbb{R}$, the real numbers; in this case the functions in F are called *functionals*. Then, for any set X, F inherits the vector space operations from $\mathbb{R}$:

$$\alpha f_1 + \beta f_2: X \to Y; x \mapsto \alpha(f_1\ x) + \beta(f_2\ x)$$

4. It is interesting to note that one can view the starting point of object-oriented programming as the stategy of associating evaluation procedures for pairs $(f\ x)$ not with the operation symbol f but rather with the data symbol x (that is, with the data type of x).

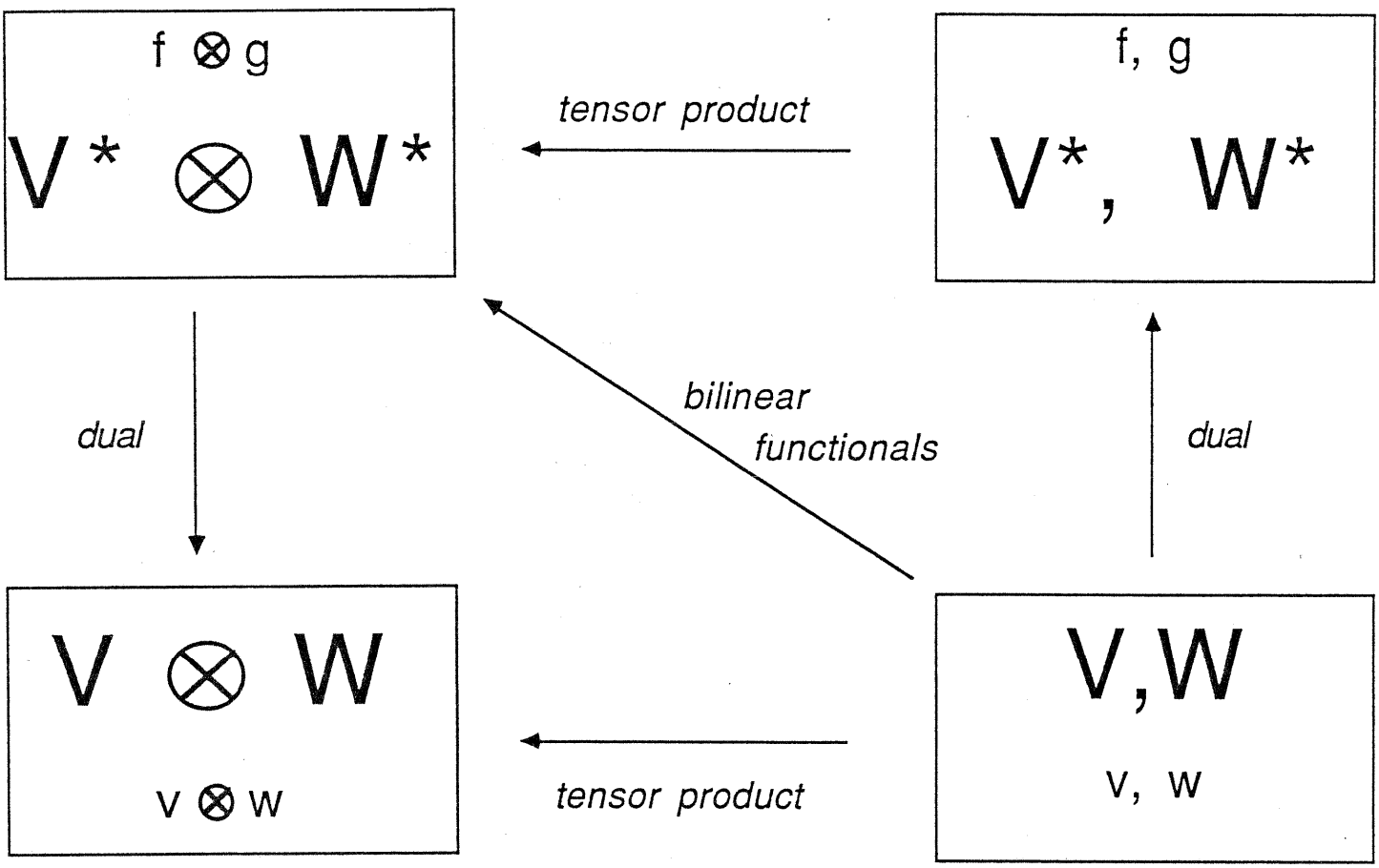


Figure 17. The basis-independent definition of tensor product.

Now note that the dual functionals $\bar{x}$ have a special property: they are *linear functionals on F*:

$$(\alpha f_1+\beta f_2\ x) = \alpha(f_1\ x)+\beta(f_2\ x)$$

Thus in order for F and X to be dual spaces, the functionals in F, like those in $\bar{X}$, must be linear:

$$(f\ \alpha x_1+\beta x_2) = \alpha(f\ x_1)+\beta(f\ x_2)$$

This only makes sense if X itself is a vector space (so that $\alpha x_1+\beta x_2$ is defined). Thus:

DEFINITION 5.1.2: Let V be a vector space. The *dual vector space* V^* is the set of linear functions from V to $\mathbb{R}$:

$$V^* = \{f\colon V \to \mathbb{R} \mid (f\ \alpha \mathbf{v}_1+\beta \mathbf{v}_2) = \alpha(f\ \mathbf{v}_1)+\beta(f\ \mathbf{v}_2)\}$$

This definition was motivated by the condition that V and its dual share the property of linearity; in fact, as required by the notion of duality, V and V^* share *all* their structural properties:

THEOREM 5.1.3: V and V^* are isomorphic vector spaces, although there is no canonical isomorphism. $(V^*)^*$ and V are canonically isomorphic.

The proof of Theorem 5.1.3 makes use of a concept which is also important for unbinding tensor product bindings (Section 3.1):

LEMMA 5.1.4: Let $\{\hat{\mathbf{v}}_i\}_{i=1}^n$ be a basis of V. Define the functionals $\{\hat{v}_i\}_{i=1}^n$ by

$$\hat{v}_i(\hat{\mathbf{v}}_j) = \delta_{ij}$$

Then $\{\hat{v}_i\}$ is a basis for V^*: the *dual basis* of $\{\hat{\mathbf{v}}_i\}$. Given any $f \in V^*$, its components with respect to $\{\hat{v}_i\}$ are

$$\mathrm{f}_i = f(\hat{\mathbf{v}}_i)$$

PROOF OF LEMMA 5.1.4: Let $\mathbf{v}$ be any vector in V; expand it in the basis $\{\hat{\mathbf{v}}_i\}$:

$$\mathbf{v} = \sum_i \mathrm{v}_i \hat{\mathbf{v}}_i$$

Now note that

$$\hat{v}_i(\mathbf{v}) = \hat{v}_i\left(\sum_j \mathrm{v}_j \hat{\mathbf{v}}_j\right) = \sum_j \mathrm{v}_j \hat{v}_i(\hat{\mathbf{v}}_j) = \sum_j \mathrm{v}_j \delta_{ij} = \mathrm{v}_i$$

Thus $\hat{v}_i$ is the functional that extracts the i^{th} component of any vector, with respect to the basis $\{\hat{\mathbf{v}}_i\}$.

Now let f be any functional in V^*. Since f is linear,

$$f(\mathbf{v}) = f\left(\sum_i \mathrm{v}_i \hat{\mathbf{v}}_i\right) = \sum_i \mathrm{v}_i f(\hat{\mathbf{v}}_i) = \sum_i \hat{v}_i(\mathbf{v})\mathrm{f}_i = \left[\sum_i \mathrm{f}_i \hat{v}_i\right](\mathbf{v})$$

Since this is true for any $\mathbf{v} \in V$, it follows that

$$f = \sum_i f_i \hat{v}_i$$

Thus we have shown that any $f \in V^*$ is a linear combination of the functionals $\{\hat{v}_k\}$; in other words, these functionals span V^*.

It remains to show that these functionals are linearly independent; it then follows that they are a basis of V^* and the lemma is proved. Suppose that

$$\sum_i \alpha_i \hat{v}_i = 0$$

This then implies, for all $\mathbf{v} \in V$, that

$$0 = \left[\sum_i \alpha_i \hat{v}_i \right] (\mathbf{v}) = \sum_i \alpha_i \hat{v}_i(\mathbf{v}) = \sum_i \alpha_i v_i$$

Since $\{v_i\}$ are the components of $\mathbf{v}$ with respect to the basis $\{\hat{\mathbf{v}}_i\}$, they are completely independent; since $\mathbf{v}$ is arbitrary, these numbers can be anything. The only way that the previous equation can hold is if all α_i vanish. This shows that the functionals $\{\hat{v}_i\}$ are linearly independent. ∎

PROOF OF THEOREM 5.1.3: As Lemma 5.1.4 shows, both V and V^* are real vector spaces of the same dimension, and are therefore isomorphic. To specify an isomorphism (a linear mapping) taking vectors in V into functionals in V^*, it suffices to map corresponding elements of a basis for V into the elements of a basis for V^*. It might seem that a canonical way to do this is to map a basis $\{\hat{\mathbf{v}}_i\}$ onto its dual basis $\{\hat{v}_i\}$. However the resulting mapping will vary depending on the choice of basis $\{\hat{\mathbf{v}}_i\}$. This can be easily demonstrated by comparing the mapping determined by $\{\hat{\mathbf{v}}_i\}$ with that determined by a basis $\{\hat{\mathbf{v}}'_i\}$ in which all the basis vectors of $\{\hat{\mathbf{v}}_i\}$ have simply been multiplied by 2:

$$\hat{\mathbf{v}}'_i = 2\hat{\mathbf{v}}_i$$

If a vector $\mathbf{v}$ is expanded in the two bases, we see that its two sets of components are related by

$$v'_i = \frac{1}{2} v_i$$

Since the dual basis elements $\{\hat{v}'_i\}$ give the components with respect to the basis $\{\hat{\mathbf{v}}'_i\}$, this last equation implies

$$\hat{v}'_i = \frac{1}{2} \hat{v}_i$$

Now the mapping from V to V^* determined by $\{\hat{\mathbf{v}}_i\}$ is

$$\mathbf{v} = \sum_i v_i \hat{\mathbf{v}}_i \mapsto \sum_i v_i \hat{v}_i$$

while the mapping from V to V^* determined by $\{\hat{\mathbf{v}}'_i\}$ is

$$\mathbf{v} = \sum_i v'_i \hat{\mathbf{v}}'_i \mapsto \sum_i v'_i \hat{v}'_i = \sum_i \frac{1}{2} v_i \frac{1}{2} \hat{v}_i = \frac{1}{4} \sum_i v_i \hat{v}_i$$

As the basis expands, the components of $\mathbf{v}$ and the dual functionals $\{\hat{v}_i\}$ *both* contract; rather than compensating each other, they compound the change, with the result that the image of $\mathbf{v}$ under the mapping corresponding to the expanded basis is only one quarter of that under the original basis. For transformations of basis that are more involved than simple rescaling, the situation is worse: in general, there is no simple relation at all between the images of $\mathbf{v}$ under the two mappings corresponding to the two bases. The conclusion is that while V and V^* are isomorphic, there is no canonical isomorphism.

The situation is different for V and $(V^*)^* = V^{**}$. Since isomorphism is transitive, it follows from the preceding conclusion that V^{**} and V are isomorphic; but now there is a canonical isomorphism. This isomorphism is exactly the duality mapping $x \mapsto \bar{x}$ with which this section began. For any $\mathbf{v} \in V$, associate the functional $\bar{\mathbf{v}} \in V^{**}$ defined by

$$\bar{\mathbf{v}}: V^* \to \mathbb{R}; f \mapsto f(\mathbf{v})$$

This is a linear transformation that requires no basis or other arbitrary choice for its definition; it is canonical. Using this canonical isomorphism we can identify $\mathbf{v}$ with $\bar{\mathbf{v}}$, identify V with V^{**}, and thereby regard duality as a symmetrical relation going both directions between V and V^*. ▯

There is one final fact about dual bases that is needed in Section 3.1:

THEOREM 5.1.5: Each dual basis functional $\hat{v}_i \in V^*$ is equivalent to the inner product with some vector $\mathbf{u}_i \in V$:

$$\hat{v}_i(\mathbf{v}) = \mathbf{u}_i \cdot \mathbf{v}$$

for all $\mathbf{v} \in V$.

PROOF: Let V_i be the subspace of V spanned by all the basis vectors other than $\hat{\mathbf{v}}_i$:

$$V_{\neg i} = \text{span}(\{\hat{\mathbf{v}}_j \mid j \neq i\})$$

This is a subspace of dimension $n-1$, since V has dimension n. There is a one-dimensional subspace orthogonal to $V_{\neg i}$; call it V_i, and let $\mathbf{w}_i$ be any non-zero vector in it. Define

$$\mathbf{u}_i = \frac{\mathbf{w}_i}{\mathbf{w}_i \cdot \hat{\mathbf{v}}_i}$$

(The denominator here cannot be zero: if it were, $\hat{\mathbf{v}}_i$ would be orthogonal to V_i and therefore in $V_{\neg i}$; this would mean it is a linear combination of the other basis vectors, which is impossible.) Now we can see that

$$\mathbf{u}_i \cdot \hat{\mathbf{v}}_i = \frac{\mathbf{w}_i \cdot \hat{\mathbf{v}}_i}{\mathbf{w}_i \cdot \hat{\mathbf{v}}_i} = 1$$

On the other hand, if $j \neq i$, $\mathbf{u}_i \cdot \hat{\mathbf{v}}_j = 0$ since $\mathbf{u}_i \in V_i$, $\hat{\mathbf{v}}_j \in V_{\neg i}$, and V_i is orthogonal to $V_{\neg i}$. Thus

$$f_i : V \rightarrow \mathbb{R};\ \mathbf{v} \mapsto \mathbf{u}_i \cdot \mathbf{v}$$

is a linear functional with the following values on the basis vectors:

$$f_i(\hat{\mathbf{v}}_j) = \delta_{ij}$$

Thus $f_i = \hat{v}_i$, completing the proof. ∎

5.2. Tensor product of functionals

Using the dual space V^* of V we can define a sort of product of two functionals. Just as functions into $\mathbb{R}$ inherit the operations of addition and scalar multipication, they also inherit another sort of multiplication:

DEFINITION 5.2.1: Let $f \in V^*$ and $g \in W^*$. The *tensor product* of f and g is the functional $f \otimes g$ defined by

$$f \otimes g : V \times W \rightarrow \mathbb{R};\ (\mathbf{v}, \mathbf{w}) \mapsto f(\mathbf{v}) g(\mathbf{w})$$

The tensor product uses multiplication in $\mathbb{R}$ to combine two functions of one variable into one function of two variables.

The linear properties of f and g and the properties of multiplication of real numbers combine to produce the following properties for the tensor product $\pi = f \otimes g$ function:

$$\pi(\alpha\mathbf{v}_1 + \beta\mathbf{v}_2, \mathbf{w}) = \alpha\pi(\mathbf{v}_1, \mathbf{w}) + \beta\pi(\mathbf{v}_2, \mathbf{w})$$
$$\pi(\mathbf{v}, \alpha\mathbf{w}_1 + \beta\mathbf{w}_2) = \alpha\pi(\mathbf{v}, \mathbf{w}_1) + \beta\pi(\mathbf{v}, \mathbf{w}_2)$$

The function space which these tensor products inhabit is therefore defined as follows:

DEFINITION 5.2.2: Let V and W be vector spaces. A function ϕ from $V \times W$ to $\mathbb{R}$ is a *bilinear functional* if

$$\phi(\alpha\mathbf{v}_1 + \beta\mathbf{v}_2, \mathbf{w}) = \alpha\phi(\mathbf{v}_1, \mathbf{w}) + \beta\phi(\mathbf{v}_2, \mathbf{w})$$
$$\phi(\mathbf{v}, \alpha\mathbf{w}_1 + \beta\mathbf{w}_2) = \alpha\phi(\mathbf{v}, \mathbf{w}_1) + \beta\phi(\mathbf{v}, \mathbf{w}_2)$$

The space of all such bilinear functionals is denoted $V^* \otimes W^*$.

We have already noted that functionals inherit vector space operations from $\mathbb{R}$; it follows that $V^* \otimes W^*$ is a vector space, since the conditions that functions in this space must satisfy are preserved by addition and scalar multiplication.

5.3. Tensor product of vectors

It is now a simple matter to define the tensor product of V and W. We have already defined the tensor product of their dual spaces, $V^* \otimes W^*$; all we need to do is take its dual, which exists since $V^* \otimes W^*$ is a vector space.

DEFINITION 5.3.1: Let V and W be vector spaces. The *tensor product space* $V \otimes W$ is $(V^* \otimes W^*)^*$. The

tensor product of two vectors $\mathbf{v} \in V$ and $\mathbf{w} \in W$ is

$$\mathbf{v} \otimes \mathbf{w}: V^* \otimes W^* \rightarrow \mathbb{R};\ \phi \mapsto \phi(\mathbf{v}, \mathbf{w})$$

Given this basis-independent definition of the tensor product, we can prove the basic result that was stipulated in the first paragraph of this Appendix:

THEOREM 5.3.2: Suppose $\{\hat{\mathbf{v}}_i\}$ is a basis of V and $\{\hat{\mathbf{w}}_j\}$ is a basis of W. Then $\{\hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j\}$ is a basis of $V \otimes W$. Thus if V and W have dimensions N_V and N_W, $V \otimes W$ has dimension $N_V N_W$. If the components of $\mathbf{v}$ with respect to $\{\hat{\mathbf{v}}_i\}$ are $\{\mathbf{v}_i\}$ and the components of $\mathbf{w}$ with respect to $\{\hat{\mathbf{w}}_j\}$ are $\{\mathbf{w}_j\}$ then the components of $\mathbf{v} \otimes \mathbf{w}$ with respect to $\{\hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j\}$ are $\{\mathbf{v}_i \mathbf{w}_j\}$.

Before proving this result, here is an extremely useful lemma:

LEMMA 5.3.3: $\left[\sum_i \alpha_i \mathbf{v}_i\right] \otimes \left[\sum_j \beta_j \mathbf{w}_j\right] = \sum_i \sum_j \alpha_i \beta_j \mathbf{v}_i \otimes \mathbf{w}_j$

PROOF OF LEMMA 5.3.3: Using definitions 5.3.1 and 5.2.2, the calculation is straightforward:

$$\left[\sum_i \alpha_i \mathbf{v}_i\right] \otimes \left[\sum_j \beta_j \mathbf{w}_j\right] : \phi \mapsto \phi\left(\sum_i \alpha_i \mathbf{v}_i, \sum_j \beta_j \mathbf{w}_j\right) = \sum_i \sum_j \alpha_i \beta_j \phi(\mathbf{v}_i, \mathbf{w}_j) = \left[\sum_i \sum_j \alpha_i \beta_j \mathbf{v}_i \otimes \mathbf{w}_j\right](\phi)$$

∎

PROOF OF THEOREM 5.3.2: First we show that $\{\hat{v}_i \otimes \hat{w}_j\}$ is a basis of $V^* \otimes W^*$, where $\{\hat{v}_i\}$ is the basis of V^* dual to $\{\hat{\mathbf{v}}_i\}$ and $\{\hat{w}_j\}$ is the basis of W^* dual to $\{\hat{\mathbf{w}}_j\}$. For any $\phi \in V^* \otimes W^*$, define

$$\phi_{ij} = \phi(\hat{\mathbf{v}}_i, \hat{\mathbf{w}}_j)$$

Then

$$\phi(\mathbf{v}, \mathbf{w}) = \phi\left(\sum_i \mathbf{v}_i \hat{\mathbf{v}}_i, \sum_j \mathbf{w}_j \hat{\mathbf{w}}_j\right) = \sum_i \sum_j \mathbf{v}_i \mathbf{w}_j \phi(\hat{\mathbf{v}}_i, \hat{\mathbf{w}}_j) = \sum_i \sum_j \mathbf{v}_i \mathbf{w}_j \phi_{ij} = \sum_i \sum_j \hat{v}_i(\mathbf{v}) \hat{w}_j(\mathbf{w}) \phi_{ij}$$

$$= \left[\sum_i \sum_j \phi_{ij} \hat{v}_i \otimes \hat{w}_j\right](\mathbf{v}, \mathbf{w})$$

Thus

$$\phi = \sum_i \sum_j \phi_{ij} \hat{v}_i \otimes \hat{w}_j$$

and the functionals $\{\hat{v}_i \otimes \hat{w}_j\}$ span $V^* \otimes W^*$. The linear independence of these functionals follows exactly as in the proof of Lemma 5.1.4; they therefore form a basis of $V^* \otimes W^*$.

Having established that $\{\hat{v}_i \otimes \hat{w}_j\}$ is a basis of $V^* \otimes W^*$, we can use its dual basis as a basis of $(V^* \otimes W^*)^* = V \otimes W$. Call this dual basis $\{f_{ij}\}$. Recall from the proof of Lemma 5.1.4 that f_{ij} is the functional that extracts the component ϕ_{ij} of any $\phi \in V^* \otimes W^*$ with respect to the basis $\{\hat{v}_i \otimes \hat{w}_j\}$:

$$f_{ij}: \phi \mapsto \phi_{ij} = \phi(\hat{\mathbf{v}}_i, \hat{\mathbf{w}}_j) = [\hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j](\phi)$$

In other words, $f_{ij} = \hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j$, and the set $\{\hat{\mathbf{v}}_i \otimes \hat{\mathbf{w}}_j\}$ is exactly the basis of $V \otimes W$ dual to $\{\hat{v}_i \otimes \hat{w}_j\}$.

It remains to verify that the tensor product multiplies vector components. Using Lemma 5.3.3, this is straightforward:

$$\mathbf{v}\otimes\mathbf{w} = \left[\sum_i \mathrm{v}_i\hat{\mathbf{v}}_i\right]\otimes\left[\sum_j \mathrm{w}_j\hat{\mathbf{w}}_j\right] = \sum_i\sum_j(\mathrm{v}_i\,\mathrm{w}_j)(\hat{\mathbf{v}}_i\otimes\hat{\mathbf{w}}_j)$$

Thus the ij component of $\mathbf{v}\otimes\mathbf{w}$ with respect to the basis $\{\hat{\mathbf{v}}_i\otimes\hat{\mathbf{w}}_j\}$ is simply $\mathrm{v}_i\,\mathrm{w}_j$. []

There are a few remaining facts about the tensor product that are used in the course of this paper; they are collected together in the following theorem.

THEOREM 5.3.4:

(a) If a set $\{\mathbf{w}_j\}$ is linearly independent, and $\sum_j \mathbf{v}_j\otimes\mathbf{w}_j = \mathbf{0}$, then every $\mathbf{v}_j = \mathbf{0}$.

(b) Let $\{\hat{\mathbf{w}}_j\}$ be a basis of W. Then every vector in $V\otimes W$ can be uniquely written: $\sum_j \mathbf{v}_j\otimes\hat{\mathbf{w}}_j$

(c) $(\mathbf{v}\otimes\mathbf{w})\cdot(\mathbf{v}'\otimes\mathbf{w}') = (\mathbf{v}\cdot\mathbf{v}')(\mathbf{w}\cdot\mathbf{w}')$

PROOF:

(a) Expanding each vector $\mathbf{v}_j$ in a basis $\{\mathbf{v}_i\}$ for V, we have

$$\mathbf{0} = \sum_j \mathbf{v}_j\otimes\mathbf{w}_j = \sum_j\left[\sum_i \mathrm{v}_{ji}\hat{\mathbf{v}}_j\right]\otimes\mathbf{w}_j = \sum_i\sum_j \mathrm{v}_{ji}\hat{\mathbf{v}}_i\otimes\mathbf{w}_j$$

(using Lemma 5.3.3). Since $\{\mathbf{w}_j\}$ is linearly independent, by adding additional vectors if necessary we can expand it to a basis $\{\hat{\mathbf{w}}_j\}$ for W. Then the preceding equation gives an expression for the zero vector in $V\otimes W$ in terms of the basis $\{\hat{\mathbf{v}}_i\otimes\hat{\mathbf{w}}_j\}$. In such an expression the coefficient of every basis vector must vanish, so we conclude that all $\mathrm{v}_{ij} = 0$, i.e., that all $\mathbf{v}_j = 0$.

(b) Since $\{\hat{\mathbf{v}}_i\otimes\hat{\mathbf{w}}_j\}$ is a basis of $V\otimes W$, every vector $\mathbf{u}$ in $V\otimes W$ can be written

$$\sum_i\sum_j \mathrm{u}_{ij}\hat{\mathbf{v}}_i\otimes\hat{\mathbf{w}}_j = \sum_j\left[\sum_i \mathrm{u}_{ij}\hat{\mathbf{v}}_i\right]\otimes\hat{\mathbf{w}}_j$$

Thus for $\mathbf{v}_j$ in the result to be proved we can choose $\mathbf{v}_j = \sum_i \mathrm{u}_{ij}\hat{\mathbf{v}}_i$. The uniqueness of $\mathbf{v}_j$ follows from (a); for if

$$\sum_j \mathbf{v}_j\otimes\mathbf{w}_j = \sum_j \mathbf{v}'_j\otimes\mathbf{w}_j$$

then

$$\sum_j(\mathbf{v}_j-\mathbf{v}'_j)\otimes\mathbf{w}_j = \mathbf{0}$$

and by (a) we have that all $\mathbf{v}_j-\mathbf{v}'_j = \mathbf{0}$.

(c) If V and W have inner products, there is a canonical inner product on $V\otimes W$ in which the basis

$\{\hat{v}_i \otimes \hat{w}_j\}$ is orthonormal if $\{\hat{v}_i\}$ and $\{\hat{w}_j\}$ are. With this inner product,

$$(v \otimes w)\cdot(v' \otimes w') = \sum_i \sum_j (v \otimes w)_{ij} (v' \otimes w')_{ij} = \sum_i \sum_j v_i w_j v'_i w'_j = \left[\sum_i v_i v'_i\right]\left[\sum_j w_j w'_j\right]$$

$$= (v \cdot v')(w \cdot w')$$

This result shows that this inner product on $V \otimes W$ is independent of the choices of basis for V and W, and depends only on their inner products.

∎

6. Appendix: Volumes of N-spheres

This Appendix contains a calculation deferred from the proof of Theorem 3.2.2: the N-dimensional volume of the unit sphere in $N+1$-space.

Using the notation of the proof of Theorem 3.2.2, this volume is

$$V_N = 2\int_0^{\pi/2} V_{N-1} \sin^{N-1}\theta \, d\theta =: 2V_{N-1}C_{N-1}$$

The integrals C_N can be related by an integration by parts:

$$C_N = \int_0^{\pi/2} [\sin^{N-1}\theta] \sin\theta \, d\theta = [\sin^{N-1}\theta](-\cos\theta)\Big|_0^{\pi/2} - \int_0^{\pi/2} [(N-1)\sin^{N-2}\theta \cos\theta](-\cos\theta)\, d\theta$$

$$= 0 + (N-1)\int_0^{\pi/2} \sin^{N-2}\theta \, (1-\sin^2\theta)\, d\theta = (N-1)[C_{N-2} - C_N]$$

Thus we get a recursion relation

$$C_N = \frac{N-1}{N} C_{N-2}$$

The first two integrals can be trivially computed:

$$C_0 = \pi/2$$
$$C_1 = 1$$

Thus for N even we have:

$$C_N = \frac{\pi}{2}\frac{1}{2}\frac{3}{4}\frac{5}{6}\cdots\frac{N-1}{N} = \frac{\pi}{2}\frac{N!}{2^N[(N/2)!]^2}$$

while for N odd we get:

$$C_N = \frac{2}{3}\frac{4}{5}\frac{6}{7}\cdots\frac{N-1}{N} = \frac{2^{N-1}\left[\frac{N-1}{2}!\right]^2}{N!}$$

It can readily be shown by induction or direct multiplication of the preceding expressions that the product of two

successive integrals can be simply expressed:

$$C_N C_{N-1} = \frac{\pi}{2}\frac{1}{N}$$

Returning to the volumes, we have the recursion relation

$$V_N = 2V_{N-1}C_{N-1} = 2(2V_{N-2}C_{N-2})C_{N-1} = 4V_{N-2}(C_{N-1}C_{N-2}) = 4V_{N-2}\frac{\pi}{2}\frac{1}{N-1} = \frac{2\pi}{N-1}V_{N-2}$$

Since $V_1 = 2\pi$, we have, for N odd,

$$V_N = 2\pi\frac{\pi^{\frac{N-1}{2}}}{\frac{N-1}{2}!}$$

and for N even,

$$V_N = 2(4\pi)^{N/2}(N/2)!/N!$$

The ratio of successive volumes,

$$\frac{V_N}{V_{N-1}} = 2C_{N-1}$$

follows the complex formulae given above for C_{N-1}, but the product of two successive ratios is simply expressed:

$$\frac{V_{N+1}}{V_N}\frac{V_N}{V_{N-1}} = (2C_N)(2C_{N-1}) = 4(\frac{\pi}{2}\frac{1}{N})$$

Therefore the geometric mean of the two ratios involving N is

$$\left[\frac{V_{N+1}}{V_N}\frac{V_N}{V_{N-1}}\right]^{1/2} = \sqrt{\frac{2\pi}{N}}$$

Acknowledgements

This paper attempts to formalize and analyze ideas on distributed representation that have been articulated and exploited in various ways by a number of connectionist researchers. I have benefitted in particular from many ideas of Geoff Hinton, both published and personally communicated. The responsibility for the formulation pursued here is of course entirely my own.

I am most grateful to Ron Williams for a conversation that was important for the development of the material in Section 3.9.1.

This research has been supported by NSF grant IST-8609599 and by the Department of Computer Science and Institute of Cognitive Science at the University of Colorado at Boulder.

References

Ackley, D.H., Hinton, G.E., & Sejnowski, T.J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 147–169.

Anderson, J.A. & Hinton, G.E. (1981). Models of information processing in the brain. In G. E. Hinton and J. A. Anderson, Eds., *Parallel models of associative memory*. Hillsdale, NJ: Erlbaum.

Fanty, M. (1985). Context-free parsing in connectionist networks. Technical Report 174, Department of Computer Science, University of Rochester.

Feldman, J.A. (1985). Four frames suffice: A provisional model of vision and space. *The Behavioral and Brain Sciences* 8, 265–289.

Feldman, J.A. (1986). Neural representation of conceptual knowledge. Technical Report 189, Department of Computer Science, University of Rochester.

Feldman, J.A., Ballard, D.H., Brown, C.M., & Dell, G.S. (1985). Rochester connectionist papers: 1979–1985. Technical Report 172, Department of Computer Science, University of Rochester.

Grossberg, S. (1982). *Studies of mind and brain*. Boston: Kluwer.

Hinton, G.E. (1981a). A parallel computation that assigns canonical object-based frames of reference. In the *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*. Vancouver, British Columbia.

Hinton, G.E. (1981b). Implementing semantic networks in parallel hardware. In Hinton, G.E. and Anderson, J.A., Eds., *Parallel models of associative memory*. Hillsdale, NJ: Erlbaum.

Hinton, G.E., Mclelland, J.L., & Rumelhart, D.E. (1986). Distributed representations. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models*. Cambridge, MA: MIT Press/Bradford Books.

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 distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*. Cambridge, MA: MIT Press/Bradford Books.

McClelland, J.L. (1986). The programmable blackboard model of reading. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models*. Cambridge, MA: MIT Press/Bradford Books.

McClelland, J.L. & Elman, J.L. (1986). Interactive processes in speech perception: The TRACE model. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models*. Cambridge, MA: MIT Press/Bradford Books.

McClelland, J.L. & Kawamoto, A.H. (1986). Mechanisms of sentence processing: Assigning roles to constituents. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel distributed processing:*

Explorations in the microstructure of cognition. Volume 2: Psychological and biological models. Cambridge, MA: MIT Press/Bradford Books.

McClelland, J.L. & Rumelhart, D.E. (1981). An interactive activation model of context effects in letter perception: Part 1. An account of the basic findings. *Psychological Review* 88, 375–407.

McClelland, J.L., Rumelhart, D.E., & the PDP Research Group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models.* Cambridge, MA: MIT Press/Bradford Books.

Riley, M.S. & Smolensky, P. (1984). A parallel model of (sequential) problem solving. *Proceedings of the Sixth Annual Conference of the Cognitive Science Society.* Boulder, CO.

Rumelhart, D.E., Hinton, G.E., & McClelland, J.L. (1986). A general framework for parallel distributed processing. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propogation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Rumelhart, D.E. & McClelland, J.L. (1982). An interactive activation model of context effects in letter perception: Part 2. The contextual enhancement effect and some tests and extensions of the model. *Psychological Review* 89, 60–94.

Rumelhart, D.E. & McClelland, J.L. (1986). On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models.* Cambridge, MA: MIT Press/Bradford Books.

Rumelhart, D.E., McClelland, J.L., & the PDP Research Group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Sejnowski, T.J. & Rosenberg, C.R. (1986). NETtalk: A parallel network that learns to read aloud. Technical Report JHU/EECS-86/01, Department of Electrical Engineering and Computer Science, The Johns Hopkins University.

Smolensky, P. (1986a). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Smolensky, P. (1986b). Neural and conceptual interpretations of parallel distributed processing models. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2: Psychological and biological models.* Cambridge, MA: MIT Press/Bradford Books.

Smolensky, P. (1987). Connectionist AI, symbolic AI, and the brain. *AI Review*, special issue on the foundations of AI..

Smolensky, P. (forthcoming). On the proper treatment of connectionism.

Touretzky, D.S. (1986). BoltzCONS: Reconciling connectionism with the recursive nature of stacks and trees. *Proceedings of the 8th Conference of the Cognitive Science Society.* Amherst, MA.

Touretzky, D.S. & Hinton, G.E. (1985). Symbols among the neurons: Details of a connectionist inference architecture. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 238-243.

Treisman, A.M. & Schmidt, H. (1982). Illusory conjunctions in the perception of objects. *Cognitive Psychology*, **14**, 107–141.

Widrow, G. & Hoff, M.E. (1960) Adaptive switching circuits. *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*, 96–104.

Williams, R.J. (1985). Feature discovery through error-correction learning. Technical Report 8501, Institute of Cognitive Science, University of California, San Diego.