SYMBOL STRUCTURES IN CONNECTIONIST NETWORKS: FIVE PROPERTIES AND TWO ARCHITECTURES

Technical Report AIP - 20

D. S. Touretzky & M. A. Derthick

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa. 15213

The Artificial Intelligence and Psychology Project

Departments of
Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

SYMBOL STRUCTURES IN CONNECTIONIST NETWORKS: FIVE PROPERTIES AND TWO ARCHITECTURES

Technical Report AIP - 20

D. S. Touretzky & M. A. Derthick

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa. 15213

29 September 1987

This research was supported by the Computer Sciences Division, Office of Naval Research and DARPA under Contract Number N00014-86-K-0678; National Science Foundation grant IST-8516330 and an ONR graduate fellowship to the second author. We thank Geoff Hinton for helpful discussions about distributed representations. Reproduction in whole or in part is permitted for purposes of the United States Government. Approved for public release; distribution unlimited.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AIP - 20	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University · 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research (Code 1133)
6c. ADDRESS (City, State, and ZIP Code) Department of Psychology Pittsburgh, Pennsylvania 15213	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, Virginia 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Same as Monitoring Organization · 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS p400005ub201/7-4-86

PROGRAM ELEMENT NO	PROJECT NO	TASK NO.	WORK UNIT ACCESSION NO
N/A	N/A	N/A	N/A

11. TITLE (Include Security Classification)
Symbol Structures in Connectionist Networks: Five Properties and Two Architectures

12. PERSONAL AUTHOR(S)
D.S. Touretzky & M.A. Derthick

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 86Sept15 TO 91Sept14	87 September 29	5

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	> Artificial intelligence, connectionism, symbols, computer [illegible], [illegible], Lisp on vonNeumann machines, BoltzCONS, micro-KLONE

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

We define five properties of symbol representations (called the Five M's) that together provide a powerful and flexible symbol processing facility. Our ideal symbol structures are Mobile, Memorable, Meaningful, Malleable, and Modifiable. Conventional symbol processing architectures, such as Lisp on von Neumann machines, support some of these properties, but others seem to require flexibility and an ability to generalize that is unique to connectionist models. Two recently-developed connectionist architectures, BoltzCONS and micro-KLONE, are described in some detail. Representations used in these two architectures satisfy different subsets of the Five M's.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b. TELEPHONE (Include Area Code) (202) 696-4302 · 22c. OFFICE SYMBOL N00014

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

Proceedings of IEEE COMPCON, Spring, 1987.
February 1987. San Francisco.

Symbol Structures in Connectionist Networks: Five Properties and Two Architectures

David S. Touretzky and Mark A. Derthick

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

We define five properties of symbol representations (called the Five M's) that together provide a powerful and flexible symbol processing facility. Our ideal symbol structures are Mobile, Memorable, Meaningful, Malleable, and Modifiable. Conventional symbol processing architectures, such as Lisp on von Neumann machines, support some of these properties, but others seem to require flexibility and an ability to generalize that is unique to connectionist models. Two recently-developed connectionist architectures, BoltzCONS and micro-KLONE, are described in some detail. The representations used in these two architectures satisfy different subsets of the Five M's.

1. Introduction

Intelligence seems to require the ability to build complex structures and to refer to them with simpler objects that may be passed among processes easily. In this paper we use "symbol" informally to denote such objects, rather than any of the more technical definitions for symbols that have been proposed. (Derthick and Plaut[1] compare Newell and Simon's definition of symbol with what can be achieved in connectionist systems.) Symbols, and structures built from symbols, can be represented in a variety of ways. In this paper we list five properties of symbolic representations (called the Five M's) that together provide a powerful and flexible symbol processing facility. Conventional symbol processing architectures, such as Lisp on von Neumann machines, support some of these properties, but others seem to require flexibility and an ability to generalize that is unique to connectionist models.[2] Two recently-developed connectionist architectures, BoltzCONS and micro-KLONE, are described in some detail. The representations used in these two architectures satisfy different subsets of the Five M's. The prospects for implementing all five properties in a single system appear to be good.

2. The Five M's

Symbols should be Mobile, Meaningful, Memorable, Malleable, and Modifiable. The five properties are described below. Together they make for a very powerful symbol processing architecture.

1. **Mobile.** A symbol is not fixed in a particular place, but rather can occur in any number of places, and possibly several at once. In order to represent "John loves Mary, but Mary only loves chocolate," one must be able to instantiate the symbol MARY as both the object of the first loves relation and the agent of the second. Symbols in Lisp are mobile because their addresses are easily copied and passed around. In connectionist models where symbols are identified with activity in particular units, symbols are not mobile. Not all connectionist models have this flaw, though: symbols in BoltzCONS (described in a following section) are mobile.

2. **Meaningful.** Some connectionist models -- those based on microfeature representations -- assign similar bit patterns to symbols with similar meanings. The bits that two symbols have in common presumably code for properties they share; e.g., the patterns for CAT and DOG would both include bits for *animate*, *mammal*, *quadruped*, and *carnivore*. One can therefore impose (or have the system learn) constraints on bit patterns, such as "If the *mammal* bit is on, the *animate* bit should also be on," which help the system settle into states representing valid symbols. During relaxation, which is the dominant form of computation in connectionist models and is used to implement associative retrieval, the system tries to satisfy as many of these sub-symbolic constraints as possible. In doing so, it makes what are known as "micro-inferences."

In Lisp, symbols are atomic; they cannot be meaningfully decomposed because their bit patterns (addresses) are arbitrary. The symbol CAT in Lisp is no more similar to DOG than to CONCEPTO.

Besides the ability to do associative retrieval, another advantage of connectionist models over non-connectionist, Lisp-based ones is their ability to automatically generalize from a collection of examples so as to handle novel inputs correctly, without any extra machinery. Generalization is possible as long as the novel input is similar to one of the cases the network was trained on; it is a natural consequence of performing many parallel microinferences.

Another level of meaning in symbol structures, suggested by Hinton (personal communication), is known as reduced descriptions. If a symbol contains or is associated with some other symbols, then it should include their microfeature descriptions in its own representation, but at a reduced level of detail. Thus, for example, the bit pattern for the symbol HORSE might contain features referring to hooves, mane, and tail, but these features are less detailed than the full bit patterns for the symbols HOOF, MANE, and TAIL considered individually. Use of symbols built from reduced descriptions would allow a system to reason at one level of detail (the horse) while still paying attention to important features at lower levels (the parts of the horse), and to easily move back and forth between levels.

3. **Memorable.** Symbol structures must have some sort of static representation that allows them to be stored in memory for later use. A von Neumann machine can store things easily enough, but it has a hard time retrieving them; truly powerful symbol processing requires rapid access to the symbol memory via associative recall.

Connectionist models remember things in one of two ways. The most common technique is to represent a set of symbols by a set of weights on connections between units comprising an associative memory[3,4]. Individual symbols are then retrieved by having the memory settle into particular states. Another technique, used in BoltzCONS, is to represent both symbols and sets of symbols as activity patterns. Individual symbols are retrieved from the set via special machinery known as pullout networks and bind spaces.

4. **Malleable.** The matching of symbol structures should be a flexible sort of matching rather than the rigid atom-for-atom matching built into Lisp; patterns may need to be reshaped slightly to satisfy a query. For example, a query about a bird's nose should succeed even though strictly speaking birds don't have noses, they have *beaks*. Since beaks are the closest thing birds have to mouths or noses (they share many microfeatures of both), a flexible matcher should automatically adjust the slightly off-base query to best fit the available data. Note that this does not depend on any predetermined rule that queries about noses should be mapped to beaks; it is an emergent phenomenon resulting from the fact that NOSE and BEAK share microfeatures. This is the sort of inexact reasoning micro-KLONE addresses.

5. **Modifiable:** networks must be able to create new symbol structures on the fly, and modify existing ones. Without this ability, complex sequential reasoning is not possible; facts can be retrieved, but nothing new can be learned as a result of the network's own experiences. Lisp data structures are easily modifiable (e.g., by RPLACA and RPLACD), but many connectionist models are not. Activity patterns are easily changed, but information stored in weights is harder to manipulate

3. The BoltzCONS Architecture

BoltzCONS, shown in figure 1, represents and manipulates recursive symbol structures such as stacks and trees[5]; it has been used to perform simple parse tree transformations[6]. The model employs a coarse coded, distributed representation[7] in which trees are represented as activity patterns over a set of 2,000 binary state units called a Cons Memory. Briefly, trees such as the one shown in Figure 2a are encoded as Lisp cons cell structures (Figure 2b), and these in turn are encoded as a set of triples of form (**tag car cdr**), as in Figure 2c. Each triple has a corresponding activity pattern in Cons Memory consisting of roughly 28 active units out of the 2,000; the representation of a set of triples defining a tree is obtained by superimposing the activity patterns of the various triples.

BoltzCONS uses a 25 symbol alphabet, giving 25^3 or 15,625 possible triples, of which only a dozen or so will be stored in memory at any one time. Each Cons Memory unit has a

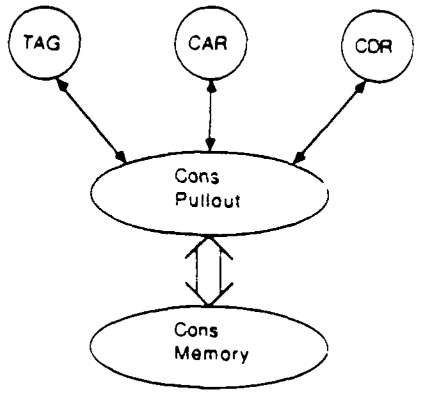
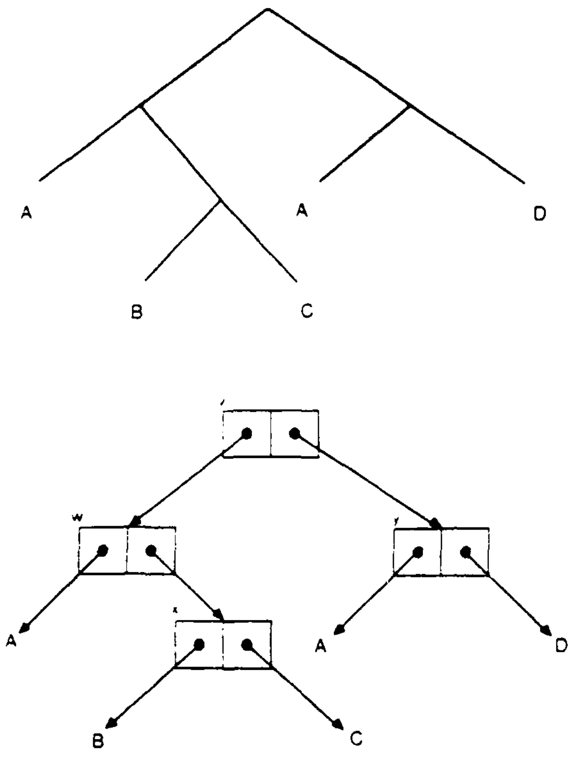


Figure 1: The BoltzCONS architecture.

randomly-generated 6×3 receptive field table that defines the triples the unit becomes active for; one of these 2,000 tables is shown in Figure 3. The receptive field of a unit is the crossproduct of the letters in the three columns, so that each unit codes for 6^3=216 triples. Conversely, each triple falls in the receptive field of $(6/25)^3\times2000$ or roughly 28 units. The representation is termed "distributed" because an external observer cannot tell by looking at a single active unit what triple it represents; triples are only discernible as patterns of activity over a set of units.

Coarse coded memories have a number of interesting properties, including fault tolerance and insensitivity to noise. The capacity of a coarse coded memory depends in part on the similarity of the items to be stored. The memory exhibits an interesting local blurring effect: if a large number of related items are stored, such as (**F A A**), (**F A B**), (**F A C**), (**F A D**), etc., the memory may be unable to distinguish between stored triples and others such as (**F A W**), but it can still discriminate unrelated triples such as (**G K Q**). Hinton has suggested that the famous "seven plus or minus two" limitation on human short term memory may be due to coarse coding effects of this sort[8].

BoltzCONS uses a device called a pullout network to retrieve individual triples from memory, and a set of three "bind spaces" to extract the individual components of a triple; the details of these mechanisms are given elsewhere[9,8]. Basically, the Cons Pullout space consists of 2,000 units in a one-one excitatory mapping with Cons Memory, but Cons Pullout units interact via mutual inhibition so that only about 28 at a time can be on -- just enough to represent a single triple. Bind spaces consist of 600 units whose activity patterns code for individual symbols rather than triples: the wiring patterns between bind units and cons pullout units map the Tag, Car, and Cdr spaces into the respective components of the triple represented in Cons Pullout space, e.g., if the pullout network is clamped to a state representing the triple (**F A B**), the Tag space units will settle into the pattern for F.



Tag	Car	Cdr
(v	w	y)
(w	A	x)
(x	B	C)
(y	A	D)

Figure 2: (a) A binary tree; (b) its cons cell representation; (c) cons cells represented as triples.

C	A	A
F	B	D
H	G	J
P	K	M
S	N	P
W	Y	R

Figure 3: A receptive field table.

The basic operation in BoltzCONS is associative retrieval; it uses this to follow pointers the way Lisp does in computing CARs and CDRs, but it can also follow pointers backwards, which conventional von Neumann machines cannot do. For example, to find the cons cell in Figure 2b whose cdr points to the cell labeled **x**, **x** is clamped into Cdr space and the Tag, Car, and Cons Pullout spaces are cleared. When the simulated annealing associative retrieval algorithm is run, Cdr space exerts an influence on pullout space that causes it to select the pattern for the triple (**w A x**); simultaneously, **w** appears in Tag space and **A** in Car space.

BoltzCONS can create new trees by storing new triples in Cons Memory. First the components of the triple are clamped into Tag, Car, and Cdr spaces; then the three spaces act in concert to create an activity pattern in Cons Pullout space for the triple to be stored. Finally, a gated connection from Cons Pullout to Cons Memory causes the corresponding Cons Memory cells to become active, thus storing the triple. Triples are deleted by turning the selected Cons Memory cells off instead of on. Transformations on trees can be expressed as sequences of additions and deletions of triples.

The BoltzCONS architecture satisfies properties 1, 3, and 4 above. Its symbols are **mobile** because they can exist as components of more than one triple, and because the three bind spaces can be used to extract them and move them around. Its symbols are **memorable** because they are static activity patterns in Cons Memory and can be associatively retrieved through the Cons Pullout net. They do not have more permanent representations as weights in an auto-associative memory, though. Symbol structures in BoltzCONS are clearly **modifiable**, since triples can be added and deleted dynamically by changing the activity pattern in Cons Memory.

BoltzCONS lacks properties 2 and 5 because its patterns have no internal structure to them; the patterns for symbols or triples of symbols are arbitrary, not meaningful. Also, there is no notion of similarity of symbols, or of closest match. Coarse coded activity patterns are not incompatible with microfeature representations, though: St. John has developed a model that employs both techniques[10], as does micro-KLONE.

4. The Micro-KLONE Architecture

Micro-KLONE[11] is a connectionist implementation of a frame-based knowledge representation system similar to KL-TWO[12]. There are three categories of objects understood by the system: *Individuals* are particular entities in the world, such as CLYDE; *Concepts* are classes of individuals, such as ELEPHANT; and *Roles* are classes of relations among individuals, such as the HAS-OCCUPATION relation from animals to jobs. That Clyde is a circus performer is an instance of the HAS-OCCUPATION role. Concepts and roles are organized into an ISA hierarchy (see figure 4). Two concepts or roles are linked if instances of the lower term are necessarily instances of the upper term. For example, DOG is linked to ANIMAL because anything that is a dog must be an animal.

Once a knowledge base has been defined and compiled into a connectionist network, it may be queried. In micro-KLONE as well as in KL-TWO, information that is only implicit in the

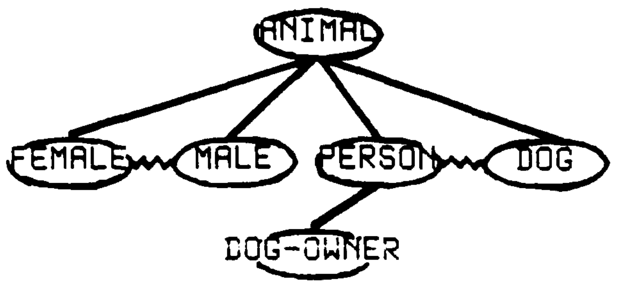


Figure 4: In this concept taxonomy, solid lines represent ISA relations; jagged lines represent disjointness relations.

knowledge base is used in answering queries. For instance, the system can conclude that Fido is an animal from the knowledge that Fido is a dog and that all dogs are animals. It can also conclude that Doug is a dog-owner from the facts that Doug is a person and Fido is his pet. The goal for micro-KLONE is to go beyond strictly deductive inference, however. The simple domain of figure 4 is not rich enough to demonstrate near-miss matching, but a more extensive knowledge base is being studied now. In it, John is a computer scientist, and it is known that computer scientists have no hobbies. A query asking in effect "If John had a hobby, what would it be?" will hopefully be answered "computer science," since that is the sort of thing that can be a hobby, and since jobs are similar to (share microfeatures with) hobbies.

All micro-KLONE queries are about a single individual or concept and its relationships. The architecture directly reflects this: in figure 5 the group labeled "Concept" represents a single concept, and "INDV" represents a single individual. The group labeled "vr×role" represents the restrictions on the types of fillers each role may have (known as value restrictions), while the group labeled "tag×role" represents the actual fillers of every individual's roles. These latter two groups use a coarse coded representation, with information relating to different roles superimposed. In order to make sense of these patterns, they must be separated and decoded. That is the job of the groups labeled "pullout" in figure 5.

To find John's hobby, the known reduced description of John as a computer scientist would be clamped into the concept group and expanded internally so that more of his properties become explicit. A subset of this pattern is shown in figure 6. Each micro-feature unit in the concept group is connected to overlapping subsets of units in the vr×role group, which is analogous to the Cons Memory in BoltzCONS. Simultaneously, the non-distributed pattern representing John is clamped into the INDV group, which causes an overlapped, coarse coded representation of all the role/filler pairs for John to appear in the tag×role group. Because the role patterns for JOB and HOBBY are similar, the distributed representations for John's job and John's hobby share many of the same units. The pullout group will try to assemble a consistent answer from the subset of tag×role units representing hobbies, but no legal activity pattern is represented exactly. There will be a weak tendency to pull out computer science due to the partial overlap between hobby and job units; normally this would not be strong enough to produce an output, but when the query forces something to be represented in the pullout group, computer science is chosen.

In contrast to the uniform organization of BoltzCONS's Cons memory, in which any of the representable triples is equally easy to activate, micro-KLONE's tag×role space is customized for the domain. Since jobs and hobbies are always activities, and never animals, the representation for John's having the job "Fido," though it exists, is highly unstable.

Though the conclusion that John's hobby, if he had one, would be computer science can be explained in terms of micro-inferences, such as that hobbies have to be activities, even this paints an inappropriately deterministic and explicit picture. There are tens of thousands of weights in the system, of which hundreds are responsible for the seemingly atomic micro-inference about hobbies. The fact that domain-specific knowledge is built into the structure of the network is responsible for the meaningfulness and the malleability of the symbols it uses. Each bit is meaningful because its semantics are wired into it. The malleability results from the coarseness of the receptive fields of the units. Since each is simultaneously participating in many micro-inferences, when there are conflicts a weighted majority rule is in effect. If an overwhelming majority of interior units can find a consistent interpretation only by ignoring some of the input, they are capable of doing that.

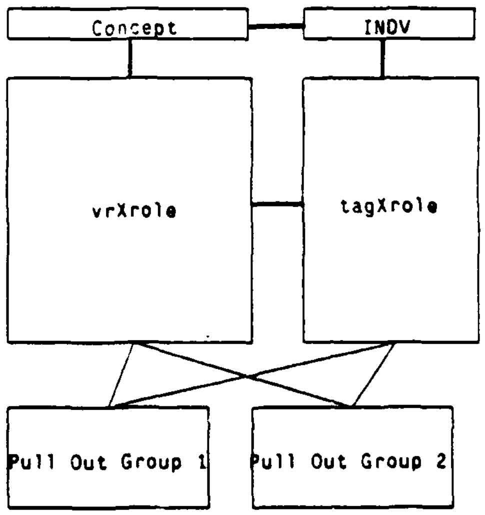


Figure 5: The micro-KLONE architecture. The concept and individual the query is about are represented in the two top groups. The value restrictions for the concept are all represented in the vr×role group, and the individual's role fillers are represented in the tag×role group. There can be an arbitrary number of pullout groups, each of which decodes the value restriction and fillers of a single role. Solid lines indicate paths of information flow between groups.

Bit	Value	Meaning
1	1	Computer Scientists are Animals
2	1	Computer Scientists are Persons
3	0	Computer Scientists are not Activities
4	0	Computer Scientists may be Males
5	0	Computer Scientists may be Females
6	1	Computer Scientist's Siblings are Persons
7	0	Computer Scientist's Siblings are not Dogs
8	1	Computer Scientists may have Pets
9	0	Computer Scientists have no Hobbies

Figure 6: Concepts are represented internally as vectors of properties. For the domain discussed above, these vectors are 43 bits long. A subset of the vector for COMPUTER-SCIENTIST is shown, along with the meaning of each bit.

There is some amount of mobility exhibited by the symbols used. John can have more than one relationship to computer science, as illustrated above. Because the network is designed to consider a single concept at a time, its symbols are not memorable. To be useful, micro-KLONE would have to be interfaced to a system capable of storing many concepts simultaneously, such as BoltzCONS. Even this is not enough to store those symbol structures for which micro-KLONE cannot produce a satisfactory reduced description. Storing novel symbol structures requires modifiable symbols, which would be quite hard to do in this system. The knowledge contained by the system is distributed throughout its weights. This is not conducive to rule based updates, because of the complexity of the rules that would be required. On the other hand, automatic learning procedures for connectionist nets are extremely slow, and are not capable of assimilating new information in one presentation. Overcoming this hurdle is an important challenge not only for micro-KLONE, but for all connectionist systems.

5. Conclusions

We would not go so far as to say the Five M's constitute sufficient or even necessary conditions for intelligent symbol processing. However, they provide a goal to strive for as we design the next generation of connectionist symbol processing models. The Five M's also help us define for non-connectionists what it is about these models that we find so powerful and so psychologically compelling. Until recently it was a matter of debate whether connectionist models could do symbol processing at all. Today there is a growing suspicion that conventional architectures alone may be inadequate for human-like symbol processing behavior; the properties of connectionist models may be indispensable.

Acknowledgements

This research was supported by National Science Foundation grant IST-8516330, and by an ONR graduate fellowship to the second author. We thank Geoff Hinton for helpful discussions about distributed representations.

References

1. Derthick, M. A., and Plaut, D. C., "Is distributed connectionism compatible with the physical symbol system hypothesis?", *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, August 1986, pp. 639-644.

2. Rumelhart, D. E., McClelland, J. L., and the PDP research group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Bradford Books/MIT Press, Cambridge, MA, 1986.

3. Hinton, G. E., and Anderson, J. A., editors, *Parallel Models of Associative Memory*, Lawrence Earlbaum Associates, Inc., Hillsdale, NJ, 1981.

4. Kohonen, T., *Self-Organization and Associative Memory (2nd edition)*, Springer-Verlag, New York, 1985.

5. Touretzky, D. S., "BoltzCONS: Reconciling connectionism with the recursive nature of stacks and trees", *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, August 1986, pp. 522-530.

6. Touretzky, D. S., "Representing and transforming recursive objects in a neural network, or 'Trees *do* grow on Boltzmann machines'", *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics*, Atlanta, GA, October 1986, pp. 12-16.

7. Hinton, G. E., McClelland, J. L., and Rumelhart, D. E., "Distributed representations", in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.*, Rumelhart, D. E., McClelland, J. L., and the PDP research group, eds., Bradford Books/MIT Press, 1986, ch. 3.

8. Touretzky, D. S., and Hinton, G. E., "Neural Network Production Systems", Submitted for publication.

9. Touretzky, D. S., and Hinton, G. E., "Symbols among the neurons: details of a connectionist inference architecture", *Proceedings of the Ninth IJCAI*, Los Angeles, CA, August 1985, pp. 238-243.

10. St. John, M. F., "Reconstructive memory for sentences", Unpublished manuscript.

11. Derthick, M. A., "A connectionist knowledge representation system", Unpublished thesis proposal, Computer Science Department, Carnegie Mellon University, 1986.

12. Vilain, M., "The restricted language architecture of a hybrid representation system", *Proceedings of the Ninth IJCAI*, Los Angeles, CA, August 1985, pp. 546-551.