Neural-Net Implementation of Complex Symbol-Processing in a Mental Model Approach to Syllogistic Reasoning

John A. Barnden*

Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003

Abstract

A neural net system called "Conposit" is described. Conposit performs rule-based manipulation of very short term, complex symbolic data structures. This paper concentrates on a simulated version of Conposit that embodies core aspects of Johnson-Laird's mental model theory of syllogistic reasoning. This Conposit version is not intended to be a psychological theory, but rather to act as a test and demonstration of the power and flexibility of Conposit's unusual connectionist techniques for encoding the structure of data.

1 Introduction

The challenge presented to conncctionism and neural networks studies by high-level cognitive processing — which includes commonsense reasoning, planning, and some aspects of natural language understanding — is gaining increasing recognition. The main technical difficulties are discussed in Barnden (1984), Fodor & Pylyshyn (1988) and elsewhere in the connectionist literature, and include the well-known variable-binding problem and the problem of accounting for complex, temporary, novel structures of short-term information. Conposit is a neural net system that is aimed at high-level cognitive processing and is able to manipulate neural-net implementations of traditional, complex, symbolic data structures by means of production rules implemented as neural subnetworks [Barnden 1988a,b, and elsewhere]. Conposit, in its current simulated versions, is therefore an exercise in "implementational connectionism". See Bamden [1988b] for a discussion of the value this type of connectionist research. Currently simulated versions of Conposit are concerned only with short-term processing: they have no adaptive learning capability, and long-term memory consists entirely of the fixed set of production rules.

In this paper 1 describe Conposit-3, a version of Conposit that engages in commonscnse syllogistic reasoning, by embodying some core aspects of the Johnson-Laird's "mental model" theory [Johnson-Laird & Bara 1984]. This theory requires complex symbolic structures to be manipulated in a variety of ways. The production of a connectionist/ncural-net *psychological theory* is not, however, an immediate goal of Conposit research. The purpose of applying Conposit to an existing psychological theory was to provide a more objective test of the flexibility and generality of Conposit's symbol-processing techniques, which were originally developed for other symbol processing tasks. The mental-model implementation has been straightforward and natural, and has required no ad hoc representational or processing techniques. The implementation also suggests that Conposit could, in the future, be a route by which to map complex psychological symbol-processing theories down to the neural network level. Conposit is, in fact, intended as a highly tentative, partial and idealized model of how biological neural networks in human cortex could effect symbolic information processing, but this biological aspect is beyond the scope of this paper.

Section 2 summarizes Johnson-Laird's approach. Sections 3 and 4 outline Conposit-3\s embodiment of it, at an abstract level of description in which Conposit-3 is viewed as a machine manipulating the symbolic contents of registers. Section 5 sketches the neural-net implementation of this register machine. Section 6 concludes. It is impossible in a paper of this length to describe a system of the complexity of Conposit in detail. Further information can be found in Barnden (1988b) and Srinivas & Barnden (1989).

2 Syllogisms and Mental Models

Consider the syllogism

Some athletes are beekeepers.
All beekeepers are chemists.
Therefore, some athletes are chemists.

* This research is being supported in part by USAF grant AFOSR-88-0215, by supercomputer facilities through NASA's MPP Working Group, and by grant NAGW-1592 under the Innovative Research Program of the NASA Office of Space Science and Applications.

Johnson-Laird maintains that we make such a syllogistic inference by constructing one or more *'mental models*' conforming to the premises, and then try to read a conclusion off a mental model. A mental model is an abstract data structure made up of atomic *tokens* and *identity links* between tokens. For our example, there are arbitrarily selected number of tokens standing for athletes, beekeepers, or chemists. An arbitrarily selected proper non-empty subset of the athlete tokens are related by identity links to some beekeeper tokens, and all beekeeper tokens are so linked to chemist tokens. Some tokens may be marked as being optional, a practice which cuts down the number of models that needs to be considered. The conclusion that some athletes are chemists arises from noticing that some athlete tokens are linked by chains of identity links to chemist tokens. The mental model serves as a sort of internalized, highly abstract "example situation" conforming to the premises of the English syllogism. Naturally, the "conclusion" read off from a mental model might merely be an artifact of the particular example it embodies, and therefore be invalid. In response to this, Johnson-Laird postulates that the system attempts to construct several different mental models conforming to the premises, in an attempt to falsify any particular putative conclusion. The attempted-falsification process will fail in the present case, but would have succeeded if the above syllogism had contained "some beekeepers" rather than "all beekeepers".

3 Mental Models in Conposit

Conposit-3 straightforwardly represents the Johnson-Laird mental models, and constructs them from symbolic data structures (fragments of semantic network) that encode syllogism premises. Conposit-3 does not yet address the following aspects of Johnson-Laird's approach: (i) the understanding or generation of natural language; (ii) a thorough attempted-falsification process — Conposit-3 is given the conclusion, and merely checks its validity with a single model randomly generated from the premises; or (iii) negative premises and conclusions ("no X are Y" and "some X are not Y"). The correction of the last two deficiencies is not difficult, however, and will be described elsewhere.

Conposit-3's representation of the above example syllogism is illustrated in Figure 1. What this shows is the starting state of an 8x8 region within a 32x32 array of *registers* called the *configuration matrix* (CM) of Conposit-3. The CM is Conposit-3's working memory holding the short-term data structures on which production rules act. Each square in the figure illustrates a register. The values in registers may change rapidly as a result of symbol processing. A register's state consists abstractly of a *symbol* and a vector of binary *highlighting flag* values (each is ON or OFF). The following shows what the various items in a square in the Figures

THEN ●				S2 *g*			S3 *g*
∇			S1 *r*	∇		THEN ●	∇
r δ	*g* S1		● THEN				*r* S2
OLAP ●			SUBC ●				ATHS *r*
S1 ∇	BKRS *g*			S2 ∇		S3 ∇	● OLAP
r ATHS			*r* BKRS	*g* CHMS			*g* CHMS

Figure 1: Conposit-3 statement of example syllogism.

signify as regards the current state of the register illustrated by the square:

capitalized word or letter:
occurrence of (non-null) symbol

∇: ON value of "white" highlighting flag

•: ON value of "black" highlighting flag

r, *g*: ON value of "red", "green" flags resp.

δ: ON value of highlighting flag named "done".

A square not showing any symbol illustrates a register containing a special "null" symbol. If a square does not show an ON value for a highlighting flag, then that flag is OFF in the register. When, say, the black highlighting flag is ON in a register, we say that the register is highlighted in black.

A non-null symbol may have a specific representational function, such as denoting a particular object or class of objects in the world, or a particular type of relationship among things. For instance, the ATMS, BKRS and CHMS symbols in the Figure denote the classes of all conceivable athletes, beekeepers and chemists respectively. The OLAP and SUBC symbols denote the classes of all conceivable class-overlap situations and subclass situations respectively. The S1, S2 and S3 symbols will be discussed below. (None of the symbols mentioned is dedicated to syllogistic reasoning.) A register containing a non-null symbol is thought of as currently denoting whatever that symbol denotes. For instance, in the figure there are registers that — temporarily — denote the athlete class and the class of conceivable class-overlap situations.

The group of squares at the bottom middle of Figure 1 illustrates a possible "subconfiguration" of register states that acts as a representation of the second premise of the syllogism, namely that all beekeepers are chemists. Conposit-3 takes a white-highlighted register to denote a member of the class denoted by *any adjacent* black-highlighted register. Therefore the white-highlighted register in the bottom middle subconfiguration denotes some class-inclusion situation. Further, if a register denotes a class-inclusion situation, then Conposit-3 takes *any adjacent* red-highlighted register (here, the one containing BKRS) to denote the included class, and any adjacent green-highlighted register (here, the one containing CHMS) to denote the including class.

The bottom left and right subconfigurations in Figure 1 are very similar to the middle one, and encode the syllogism's first premise and the conclusion respectively. The SI, S2 and S3 symbols are arbitrary, distinct, variable-like *unassigned symbols.* Unassigned symbols have no permanent denotation, but can be viewed as attaining temporary denotations by virtue of their appearance within data structures in the CM. For example, since the register at the bottom left containing S1 denotes the athletcs-overlap-bcckcepcrs situation, the symbol SI is taken to denote this temporarily. Now, *Conposit takes all registers containing the same non-null symbol to denote the same thing;* so, by this *symbol-sharing* principle, the two registers in the upper half of the Figure containing SI also denote the athletes-ovcrlap-bcekeepers situation. The registers containing S2 and S3 arc analogously interpreted.

Note that the *absolute* positions of the symbols and highlighting states are irrelevant, as are the *directions* of the adjacency relationships. Moreover, a subconfiguration such as the ones mentioned can be split up into pieces by means of symbol sharing tBarnden 1988b], so that the predecessor and successor role fillers could be specified at widely separated positions in the CM.

The three subconfigurations in the upper half of Figure 1 specify the order in which the situations represented in the bottom half should be considered, corresponding to the order of the statements in the syllogism. (The Johnson-Laird theory respects this order.) For instance, the upper middle subconfiguration states that the situation denoted by SI is to be considered before the situation denoted by S2. The interpretation of the upper three subconfigurations is analogous to that of the lower three: the symbol THEN denotes the class of possible processing-ordering situations, and, for such situations, red and green highlighting mark the predecessor and successor respectively. (As a special case, the left-hand upper subconfiguration states that the SI situation is the first to be considered. The purpose of the 5 sign will be explained later.)

In a way to be described below, production rules that fire in response to ordering and premise subconfigurations shown in Figure 1 create the subsiate shown in Figure 2, which illustrates a different region of the CM. Actually, the new data structures in that Figure could be spread around randomly over the CM, but they arc shown in a regimented arrangement for illustrative clarity.

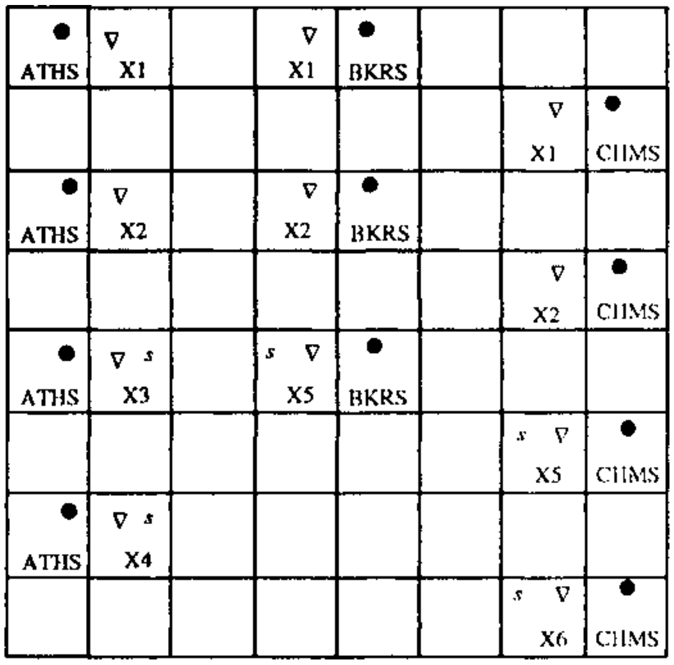

Figure 2: CM version of a syllogistic mental model.

The XI to X6 are distinct unassigned symbols. Each Johnson-Laird token for a class of persons is implemented as a pair of adjacent CM registers, one *of* which (the black one) temporarily represents a class of person, and the other of which (the white one, containing an Xi symbol) therefore represents a particular though indefinite member of the class. Each Xi symbol is thereby considered to denote a person for the time being. The function of Johnson-Laird's identity links is taken over by symbol-sharing, since symbol-sharing registers represent the same thing. In the Figure an 'S' indicates special highlighting signifying that the token is optional. (Optionality could also be indicated by a more elaborate data structure.)

The construction of the mental model has two main phases. First, a production rule called "Rule_Some" detects the subconfiguration for the first premise (Figure 1, bottom left), and constructs, in random positions in the CM, the athlete and beekeeper tokens in Figure 2. It creates randomly many athlete tokens, six on average, then constructs beekeeper tokens using the same unassigned symbols as in a random subset of the athlete tokens, and, finally, randomly constructs three extra beekeeper tokens on average. Secondly, another, similar, rule called "Rule_All" detects the subconfiguration for the second premise and constructs some chemist tokens with the same unassigned symbols as in the beekeeper tokens, and then constructs some extra chemist tokens.

To check the conclusion, Rule_Some comes into play again by detecting the conclusion subconfiguration (Figure 1, bottom right) and checking that there is at least one athlete token and chemist token sharing a symbol. In cases where the conclusion of a syllogism is invalid, Conposit-3 sometimes does and sometimes does not construct a mental model consistent with the conclusion, because of the randomness. It would be trivial to get Conposit-3 to repeat the whole process in an attempt to alight randomly on a falsifying model.

Rule_Some, Rule_AH and the remaining rule, Notc_Ncxt (see below), work with *any* classes in syllogisms, not just the athlete, beekeeper and chemist classes. There is no replication of rule circuitry for the different classes.

4 Conposit-3's Production Rules

A Conposit-3 rule consists of a condition part and an action part. At any time, only one action part can be executing, but the CM state detection implied by condition parts is done in parallel across all rules. Conflict resolution among simultaneously satisfied rules is based on a fixed total priority ordering, but this will be relaxed in later Conposit versions.

The condition part of a rule tests for the presence or absence of specific sorts of CM state configuration, by means of a highly parallel detection mechanism that will be sketched in a moment. The action part manipulates CM data structures by changing the symbol and/or highlighting states of registers. The registers subjected to these changes are chosen "associatively" on the basis only of their current highlighting/symbol states, and the states of their immediate neighbors. An action part is an arbitrarily complex flowchart, possibly involving branching and looping, each node of which sends a *command signal* that causes a group of elementary changes at some CM registers.

The following will describe the subconfiguration-detection and state-change mechanisms in the context mainly of "Note_Next", the simplest of the three rules in Conposit-3. Its condition part detects the presence of a THEN subconfiguration (see top of Figure 1) whose predecessor register (the red one) has been marked by ONness of the "done" highlighting flag (5). (In the initial state of the CM in simulation runs, the "done" highlighting is at the predecessor register of the "first" THEN subconfiguration.) The goal of Note_Ncxt is to place "now" highlighting at the successor register R (the green one) in the detected THEN subconfiguration, and to spread this now-highlighting also to all registers having the same symbol as R. In Figure 1, the effect of a Note_Next execution is to mark with "now" highlighting the two registers both containing the SI symbol, or both containing the S2 symbol, or both containing the S3 symbol. This serves to mark the class-overlap or subclass situation that should now be attended to; the now-highlighting contributes to the firing of "Rulc_Somc" or "Rule_AU" as appropriate.

Note_Next's detection mechanism is as follows, at the register-machine level of description. The mechanism relies on register arrays, called *location matrices* (LMs), that are isomorphic to the CM. A location matrix register has one of two stales, ON or OFF. An ON value at a register in a particular LM indicates the presence of a particular sort of subconfiguration based at the corresponding register in the CM. For the purpose of Note__Ncxt, the LMs are ones called *LMr.THEN, LM ..member THEN* and *LM::done_pred.* The first acquires an ON value at every position corresponding to a CM register containing the THEN symbol. The second LM acquires an ON value at every position corresponding to a CM register that is white-highlighted and is adjacent to a black-highlighted register containing the THEN symbol. Recall that a white register of this sort denotes a member of the class of possible succession situations. The LM also acquires ON values at the positions where CM registers contain the same symbol as a white one of the sort just described. That is, ON values in this (and every other) LM are "spread by symbol-sharing". The third LM, *LM::done_pred,* acquires an ON value at the positions of "done-predecessor" CM registers: CM registers that arc done-highlighted and are acting as the predecessor register of a THEN subconfiguration. In fact, therefore, *LMr.done_pred* acquires an ON value at every position that is adjacent to the position of an ON register in *LM:.member_'/'I/EN* and that corresponds to a CM register that is both done-highlighted and red-highlighted. Also, ONness is spread by symbol-sharing in this LM as in any other.

The starting node of the action-part flowchart of Note_Next is stimulated into action if there is an ON value anywhere in *LM::done_pred.* The action pari is a linear flowchart with four nodes. The starling node sends a command signal to *LMr.donejyred,* telling every register in it that is ON to send a signal to its corresponding CM register. The CM registers receiving such a signal turn their "detected" highlighting flags ON. Thus, the effect is to mark every "done predecessor" register in the CM with "detected" highlighting. Once this effect is complete, the second flowchart node is activated, and sends a command signal to the CM, telling every white register that has a neighbor highlighted in both red and * detected" to turn its "moving" highlighting flag ON. The signal also tells the CM to add "moving" highlighting to every register having the same symbol as one that is already "moving"-highlighted. (This is "spreading by symbol-sharing" of changes within the CM.) Then, the third flowchart node is activated, and sends a command signal to the CM telling every green register that has a moving-highlighted neighbor to turn its "now" flag ON. Again, the signal also causes spreading of the now-highlighting by symbol sharing. Finally, the fourth flowchart node sends a command signal to the CM telling every register to turn OFF its "done" and "moving" highlighting flags.

A full description of Rule_Some or Rulc_AU would be too long to be included here, but they work analogously to Note_Ncxt on the basis of LMs and command signals very like those used in Note_Next. However, their action parts are more complex, involving branching and looping, both of which arc controlled by detection of ON values in certain LMs or specific highlighting states in the CM. Rule_Some detects the presence of a now-highlighted OLAP (i.e. overlap) situation in the CM. If mental model tokens of both classes involved already exist, as will be the case if Rule_Some is operating on the bottom right OLAP situation in Figure 1, then the rule merely checks for the existence of a token of one the classes that shares a symbol with a token of the other class. In more detail, the rule marks all the while registers in athlete and chemist tokens (as in Figure 2) with special highlighting flags "membcr_of_classl" and "member_of_class2" respectively. Part of this marking process is to spread such highlighting by symbol-sharing. All that is left to do is to detect the presence of some register marked with both "mcmbcr_of_classl" and "mcmbcr__of_class2". We have here a traditional marker passing process, but working over highly temporary data structures. When tokens of both classes do not already exist, Rule_Some creates randomly many tokens of one or both classes, as appropriate, establishing sharing of some symbols Xi between some tokens of one class an some tokens of the other. Rule_All is very similar to Rule_Some.

A rule operates on the CM in a fashion that is highly, but not completely, *SIMD-like, register-local, and parallel.* Each command signal sent to the CM or to an LM is identically distributed to all the registers in the matrix. As we have seen, different registers react differently, according to their own current states and (in the CM case) those of their immediate neighbors. Often, a command signal sent to the CM requires the CM to confine its initial reaction to just one, arbitrarily selected, register. Arbitrary selection is used, for instance, in register recruitment during data structure building. However, even when arbitrary selection is in operation, the command signal can require every register that has the same symbol as one that is already chosen as reactive to react also. This is "spreading by symbol-sharing". Alternatively, the reaction required from reacting registers can include the broadcasting of their (assumed identical) symbols to all registers in the CM; the next command signal can then predicate reaction on whether registers' symbols equal the one broadcast at the previous step. Apart from the sequencing implicit in spreading by symbol-sharing, reacting registers work in parallel.

5 Neural-Net Implementation of Con posit

This section briefly sketches a simple neural net implementation of the Conposit register machine. As stated earlier, each CM register is implemented as a neural subnetwork. Part of this carries the symbol and highlighting slates of the register, which are just temporary activation patterns. For each highlighting flag there is a small cyclic subnetwork that can maintain one of two states of recurrent activity. A symbol can be thought of a a bit vector, each bit being carried by a similar cyclic subnetwork.

The rest of the circuitry in the register is concerned with the register's reaction to command signals. Each CM register subnetwork is connected to the subnetworks for immediately neighboring registers, in order to service the sensitivity to neighboring-register state required by many command signals. Each register subnetwork is also connected to the so-called *parallel distributor* (PD) for the CM. The PD is what directly receives command signals from rule action part flowchart nodes, and one of its main functions is to distribute each signal identically to all the CM registers. Command signals are conveyed on cables of connections, each connection in a cable conveying a binary or ternary value. Connections in a cable serve purposes such as providing new values for symbol bits or highlighting flags, specifying ON/OFF/don't-carc highlighting-flag values for the purpose of selection of reacting registers, and specifying whether the command signal requires arbitrary selection, spreading by symbol sharing, or symbol broadcasting.

A second important function of the PD is to perform the arbitrary selection, when required, among registers satisfying the symbol/highlighting conditions specified in a command signal. (A succession of arbitrary-selection steps is also used to serialize the symbol broadcastings needed in the spreading-by-symbol-sharing option.) Arbitrary-selection is performed by a *temporal winner-take-all* contention-resolution mechanism in the PD [Srinivas & Barnden 1989]. This mechanism receives "ready" signals on special connections from all registers among which selection is to take place. The mechanism has the effect of detecting the first such signal to arrive and telling the register that sent it that it has been chosen. However, if the first ready signal is followed within a certain time window by another ready signal, all the sending registers are told to try again. The mechanism relies on small random differences in the times taken by CM registers to decide to send a ready signal, and on differences between the travel times of signals on different connections. Thus, Conposit at the implemenlational level brings in relative timing of signals in an important way, and is therefore different from most other connectionist systems.

LM registers are implemented similarly to CM registers, and each LM also has a PD very like the CM's. LMs are connected together, and to the CM, in a simple way to reflect the functional dependencies described earlier. Each LM register receives input connections only from the *corresponding* and *neighboring* registers in the CM and/or other LMs. Simple AND/OR logic, implemented in neural units acting as logic gates, is then all that is required for introducing ON values into LMs.

Rule flowcharts map easily onto neural networks, each node being realized as a small subnetwork. Each node receives enablement connections from its predecessor flowchart nodes, and/or connections conveying information about the presence of highlighting values values (anywhere) in the CM and/or ON values (anywhere) in some LMs. Each non-starting node also receives an enablement connection from the CM's PD, so that a node only becomes activated when the previously activated node's work has been completed.

6 Conclusion

Conposit-3 is a connectionist/neural-network system with the ability to manipulate complex symbolic data structures, based on those proposed in an existing, sophisticated psychological theory. Also, Conposit could easily effect straightforward deduction of a syllogism conclusion from the premises, without going through mental models. Conposit versions reported elsewhere extend the demonstration of its powers. Its ability to do constrained variable binding is shown by a version that embodies the rule: *if A loves B, B loves C, and C is not A, then A is jealous of C* [Barnden 1988b]. Also, a type of variable-binding is implicit in Conposit-3, in that the rules do not involve different pieces of circuitry to cope with different combinations of classes in syllogisms.

Although the symbolic manipulations in running Conposit simulations have been simple from the point of view of traditional AI, they are readily extensible to much more complex cases. Further, very few other fully connectionist systems can even come near to what Conposit does, the systems of Shastri & Ajjanagadde (1989) and Touretzky & Hinton (1988) being two important exceptions. Conposit's data structuring flexibility comes from its use of *registers,* and from its *relative-position encoding* (RPE) and *pattern-similarity association* (PSA) techniques. In Conposit, PSA takes the form of symbol-sharing: the use of several occurrences of the same symbol in different registers to achieve a linking power similar to that of linking by associative addressing in computers. RPE appeals to the idea that one can put things into association with each other rapidly and flexibly by putting them next to each other (cf. the sequential-allocation method used for data structuring in computers.) The relatively computer-like appearance of Conposit makes it unusual as a connectionist/ncural-nct system. However, RPE and PSA have loose but reasonably clear connections to techniques used in other connectionist systems [Barnden 1988a,b], and there arc many aspects of future Conposit versions that ally it to more traditional connectionist/neural systems [Barndcn 1988b].

Acknowledgments

I thank K. Srinivas for help with simulation work.

References

Barnden, J.A. (1984). On short-term information processing in connectionist theories. *Cognition and Brain Theory, 7* (1), pp.25-59.

Barnden, J.A. (1988a). The right of free association: relative-position encoding for connectionist data structures. *Procs. 10th Annual Conf of the Cognitive Science Soc.* Hillsdale, N.J.: Lawrence Erlbaum.

Barnden, J.A. (1988b). Conposit, a neural net system for high-level symbolic processing: the register-machine level. *Memoranda in Computer and Cognitive Science,* MCCS-88-145, Computing Research Laboratory, New Mexico State University. Modified version to appear in J.A. Barndcn & J.B. Pollack (Eds), *Advances in Connectionist and Neural Computation Theory, Vol. 1,* Norwood, N.J.: Ablex.

Fodor, J.A. & Pylyshyn, Z.W. (1988). Conncclionism and cognitive architecture: a critical analysis. In S. Pinker & J. Mehler (Eds.), *Connections and symbols,* Cambridge, Mass.: MIT Press.

Johnson-Laird, P.N. & Bara, B.G. (1984). Syllogistic inference. *Cognition,* 76 (1), pp. 1-61.

Shastri, L. & Ajjanagadde, V. (1989). A connectionist system for rule-based reasoning with multi-place predicates and variables. Report MS-CIS-8905, Computer and Information Science Dept, University of Pennsylvania.

Srinivas, K. & Barnden, J.A. (1989). Temporal-winncr-take-all networks for arbitrary selection in connectionist and neural networks. Manuscript for poster at *1st Int. Joint Conf on Neural Networks,* 1989.

Touretzky, D.S. & Hinton, G.E. (1988). A distributed connectionist production system. *Cognitive Science, 12* (3), pp.423-466.