WILLIAM BECHTEL

NATURAL DEDUCTION IN CONNECTIONIST SYSTEMS

ABSTRACT. The relation between logic and thought has long been controversial, but has recently influenced theorizing about the nature of mental processes in cognitive science. One prominent tradition argues that to explain the systematicity of thought we must posit syntactically structured representations inside the cognitive system which can be operated upon by structure sensitive rules similar to those employed in systems of natural deduction. I have argued elsewhere that the systematicity of human thought might better be explained as resulting from the fact that we have learned natural languages which are themselves syntactically structured. According to this view, symbols of natural language are external to the cognitive processing system and what the cognitive system must learn to do is produce and comprehend such symbols. In this paper I pursue that idea by arguing that ability in natural deduction itself may rely on pattern recognition abilities that enable us to operate on external symbols rather than encodings of rules that might be applied to internal representations. To support this suggestion, I present a series of experiments with connectionist networks that have been trained to construct simple natural deductions in sentential logic. These networks not only succeed in reconstructing the derivations on which they have been trained, but in constructing new derivations that are only similar to the ones on which they have been trained.

1. LOGICAL STRUCTURE AND COGNITION

The relation between logic and thought has long been controversial. This is particularly true of rules for natural deduction. Most philosophers construe such inference rules as normative: they are truth preserving rules which, if adhered to, would insure that if a person started with true beliefs, he or she would not end up believing falsehoods. The emergence of cognitive science, however, has had inconsistent implications for the status of such inference rules. On the one hand, there is now empirical evidence that humans often find it difficult to apply such basic rules of inference as *modus tollens*; in some situations they are more likely to apply the invalid principle of affirming the consequent instead (Wason and Johnson-Laird 1972). On the other hand, a number of theorists have adapted natural deduction rules to explain human deductive reasoning (Braine, Reiser and Rumain 1984; Osherson 1975, 1976; Rips 1983, 1990). In these accounts, humans are

Synthese **101**: 433–463, 1994.

thought to apply formal rules to mental representations of propositions so as to reach desired or interesting conclusions.

The view that humans employ mental rules comparable to those employed in natural deduction systems fits well with Fodor's (1975 1987) claim that there is a language of thought. According to Fodor, cognitive performance requires an internal system of language-like representations and formal syntactic operations which can be applied to these representations. That is, language provides the metaphor by which theorists can understand and model cognition. If the cognitive system has an overall language-like architecture, it makes sense to model deductive reasoning by specifying mental rules (comparable to the inference rules of natural deduction) that operate upon language-like mental representations. This linguistic metaphor has dominated psychological models of reasoning (e.g., Smith, Langston and Nisbett 1992).

Fodor and Pylyshyn (1988) defended the linguistic metaphor as being compelled by three properties of thought: systematicity, productivity and inferential coherence. In justifying the first of these properties, Fodor and Pylyshyn noted that beliefs (or other thoughts) are systematically related to one another. For example, any cognitive system that has the capacity to believe that Mary loved the florist is capable as well of believing that the florist loved Mary. To account for this systematicity, Fodor and Pylyshyn claimed that one must employ a representational system with compositional syntax and semantics (in other words, a representational system comparable to that found in natural languages). In this system, each atomic thought is represented by a symbol, and formal operations are performed on these symbols. If the representation of a thought such as *Mary loved the florist* were constructed by compositional operations from representations of *Mary*, *loved*, and *florist*, then the system could create, using the same operations, a representation of *The florist loved Mary*. Similar arguments were offered for the other two properties. The result is a language-like system in which the compositional structure of symbol strings is specified by formal rules.

The hegemony of the linguistic representation/formal rules approach to modeling cognition has been challenged in the last decade by the reemergence of connectionist models (see especially Rumelhart, McClelland and the PDP Research Group 1986, and the introduction to this approach provided by Bechtel and Abrahamsen 1991). In connectionist networks, there are no formal rules that operate upon symbolic representations. Instead, connectionist networks consist of numerous neuron-like units. These receive input from other units, take activation values based on those inputs, and send outputs to other units based on their own activation values. The propagation of activation is governed by weights on the connections between pairs of units and by mathematical functions that relate input activations and weights to output activations. The metaphor that inspires connectionist models is neural rather than linguistic.

Connectionist networks can naturally be interpreted as engaging in pattern recognition. A feedforward network, for example, responds to a pattern of activation across its input units by producing another pattern of activation across its output units. The output pattern can be construed as representing a category to which the input pattern has been assigned. Consequently, connectionist networks have been widely accepted as useful models for cognitive abilities that clearly involve pattern recognition (e.g., perceptual identification or categorization tasks), which have proven difficult to model within a classical framework. But are connectionist networks capable of handling the broader range of tasks, including reasoning tasks, that we ordinarily think of as cognitive? If so, reasoning might be reconstrued as a kind of pattern recognition, and all of cognition might be brought under the same theoretical umbrella. Indeed, even from a nonconnectionist perspective, Margolis (1987) has offered an extensive argument for the claim that pattern recognition is pervasive in mental life and is sufficient to explain cognition. Connectionist networks provide both additional motivation and a medium for exploring this bold idea.

How does a connectionist pattern-recognizer successfully process sentences or sentence-like thoughts and beliefs? The models available to date are only suggestive, because they are designed to handle only small subsets of the range of vocabulary and syntactic structures found in a natural language. As one example, Elman (1992) designed a recurrent network in which the input layer receives a localist encoding of each successive word of a sentence. A context layer provides a distributed encoding of the part of the sentence so far encountered, and the network's task is to propagate activations through its four layers of connections so as to produce a prediction of the next word on its output layer. Following training on several 10,000-sentence corpora generated by a phrase-structure grammar, the network produced appropriate predictions for a new corpus. To do so, it had to show sensitivity to subject-verb agreement and verb argument structures in simple sentences as well as sentences with relative clauses. It did not, however, explicitly represent agreement, argument structures, and constituent structure, nor did it operate on such representations with structure-sensitive rules. Rather, it relied on the statistical behavior of its weighted connections between units to act appropriately on the sequence of symbols with which it was presented. The input to the network is similar to the input to a symbolic parser (a sequence of words), and the output may also be similar, but what happens in getting from input to output is quite different. The network propagates activations to achieve a distributed representation of its input; the symbolic parser applies rules that utilize and retain the constitutent structure of its input.

Fodor and Pylyshyn found the lack of explicit compositional structure and rules in connectionist networks quite problematic. If a network really lacks a compositional syntax, they claim, it cannot account for such properties as systematicity. If it does account for systematicity, it must be merely implementing a compositional syntax despite its superficially different architecture. One strategy that connectionists have employed to answer Fodor and Pylyshyn is to claim that the internal states of successful networks exhibit functional compositionality (the information is there) but not explicit compositionality (the components are not components in the distributed representations on hidden layers); for variations on this argument, see van Gelder (1990), Pollack (1990) and Smolensky (1990). Elsewhere (Bechtel, in press), I have argued for exploring a different strategy. The property of systematicity, and the compositional syntax and semantics that underlie that property, might best be attributed to natural languages themselves but not to the mental mechanisms involved in language use. On this view, it is enough that a network pick up whatever information allows it to respect the constituent structure of sentences presented to it. How the network does this might be quite different than we would expect from our knowledge of the grammar that characterizes the sentence's syntactic and semantic structure. That is, cognitive systems might be able to interact with symbolic representations, such as those of natural language, without having to build up and operate on internal representations that are themselves comparable to those of natural language.

According to this view, linguistic representations would be *external* to the cognitive processing system. These representations might be presented to the system for processing by means of a language-like localist encoding on an input layer, but the system's internal procedures accomplish a kind of transduction into quite different, distributed representations. Furthermore, the procedures may rely on the shifting availability of parts of the sentence within a temporal window (as simulated by limiting what parts of a sentence are available on the input units at a given time). Since external symbol strings can be written, spoken, signed, or even silently generated as inner speech,[1] the temporal characteristics and memory requirements that must be simulated will vary. Grammars, in contrast, give an atemporal account of sentence structure that may be suitable for describing the language itself but not for specifying people's internal procedures for operating on stretches of external encodings. Networks can learn to respect the syntax of external strings that grammars describe, without themselves employing compositional internal representations. A network in action and the string it acts on may jointly exhibit features such as systematicity and productivity, providing a connectionist response to Fodor and Pylyshyn.

It is especially interesting in this regard to design simulations in which symbols provide an external memory that acts in partnership with a network that relies on that external memory. The idea of training networks to use external symbols in this way was originally suggested by Rumelhart, Smolensky, McClelland and Hinton (1986; see also the discussion by Clark 1989 and Rumelhart 1989), who used arithmetic problems to illustrate this idea. In solving all but the simplest arithmetic problems, we rely on writing symbols in canonical fashion in an external medium, and then use those enscribings as input for further processing. For example, if we want to multiply the following two three-digit numbers, we begin by writing them in the following canonical form:

$$\begin{array}{r}343\\ \underline{822}\end{array}$$

Writing the problem in this way permits us to decompose it into several simple component tasks, each of which can be solved mentally if we have learned the multiplication tables. As soon as we *recognize* one of the simple problems, the answer is brought to mind. The first task in the above problem is 2 × 3, whose answer we write in the appropriate position:

$$\begin{array}{r} 343 \\ \underline{822} \\ 6 \end{array}$$

We then turn to the next task of multiplying 2 × 4. What we have learned is a routine for dealing with a complex problem in a step-by-step manner, with each step requiring limited cognitive effort. As a result, a problem that would be quite difficult if external symbols were not available is rendered much simpler with these symbols. (Margolis 1987, also suggests that cognitive problems might be solved by repeated processes of pattern recognition, although he does not develop his analysis within a connectionist framework.)

The arithmetic example shows not only how use of external symbols can facilitate decomposition of complex problems, but also how productive and systematic performance can result from use of external symbols. The rules of arithmetic are normative principles governing the use of arithmetic symbols by cognitive agents. As long as the cognitive system conforms to these norms in its behavior it can, by working with these external symbols, manifest productivity and systematicity. The same internal procedures can enable it to solve new problems, and by virtue of being able to solve certain problems, it is able to solve others. In adopting this view, we might grant Fodor that syntax is what explains systematicity, but limit syntax to external symbols. A cognitive system or network which can interact with such symbols will then exhibit productivity and systematicity. By dividing the labor between external symbols which must conform to syntactical principles and a cognitive system which is sensitive to those constraints without itself employing syntactically structured representations, one can perhaps explain the systematicity and productivity of cognitive performance.

The division of labor envisaged here can be illustrated with a Turing machine. There are two crucial components to a Turing machine. The first is a finite state device which is capable of reading the symbol on a particular square of a tape and of following rules which specify, depending upon the symbol read and the particular state the device is in, either the writing of a new symbol, moving right or left, or stopping. The second component is a potentially infinite tape on which symbols are stored. It is only by supplementing the finite state device with the infinite tape that the Turing machine achieves its capabilities, including exhibiting systematicity. Systematicity will result when the set of procedures constituting the finite state device are so configured that it produces syntactically structured symbol sequences on the tape. For example, a set of procedures for constructing SVO sentences from specifications of case roles would generate the representation of either "Mary loves the florist" or "The florist loves Mary" depending upon whether *Mary* was specified as the agent and *the florist* as the patient, or vice versa. What I am proposing is that the human mind need not incorporate the entire Turing machine: in a given domain, it may be sufficient that only the finite state device be internally realized if it interacts with (reads and writes) externally-provided symbols. The joint operation of the external symbols and internal device would exhibit systematicity.

If appealing to external symbols is even remotely plausible as part of an account of systematicity, it suggests a different perspective on principles of natural deduction from that adopted by theorists such as Braine et al. (1984), Osherson (1975, 1976) and Rips (1983), who propose that we employ rules like those for natural deduction to manipulate internal symbols. Under the alternative account, principles of natural deduction would not govern the processing of internal symbols, but would be applied to the symbols of natural languages, whether spoken, written, or privately rehearsed. Thus, we might recognize an argument uttered by another as an instance of *modus ponens*, and recognize that if we accepted the premises as true, we have to accept the conclusion. Moreover, the ability to apply these principles constitutes a learned skill. Learning this skill requires learning to recognize particular patterns in arguments.

The idea that one has to learn to recognize such patterns was suggested to me by observing students in an elementary logic course as they learned to evaluate simple arguments of sentential logic. After introducing various valid and invalid forms and giving examples of natural language arguments using each form, I would assign homework which asked students to identify the form found in a natural language argument. The answers on these exercises often revealed that the students had not identified what made an argument an instance of *modus ponens* rather than an instance of affirming the consequent. Pointing out the distinctive features of each form in lectures did not seem to suffice to teach students to recognize instances of the form. In an attempt to improve student performance, I developed a set of computer-based exercises which required students (a) to identify argument forms for natural language arguments and assess their validity or (b) to complete natural language enthymemes (arguments in which one of the premises or the conclusion has been omitted). In performing the computer exercises students often began by writing out templates of the different argument forms we had covered, and explicitly matching parts of a natural language argument to the different templates. After considerable practice, most students gradually weaned themselves from reliance on the templates and were able to identify forms or complete enthymemes reliably.

My diagnosis of the situation was that the students had not previously been sensitive to the specific syntactic features that are required for an argument to exemplify a particular logical form. While they had made inferences much of their lives, they were likely guided primarily by the semantics of the information involved, not the syntax of the representation. To become sensitive to syntax, students had to learn a new way of seeing arguments; they had to learn to see patterns where they had not seen them previously. This suggested that identification of logical structure, far from being a native component of ordinary reasoning with natural language, is a learned capacity to recognize patterns. Since connectionist networks are good at pattern recognition tasks, it seemed plausible that a connectionist network could learn to recognize the forms of sentential arguments and assess their validity, or complete enthymemes in a valid manner.

In my first exploration of this idea, I constructed a small set of logic problems and two very simple feedforward networks that might master them by treating them as problems of pattern recognition (see Bechtel and Abrahamsen 1991, pp. 167ff). The first network was trained to identify and assess the validity of arguments constructed using the 12 argument forms that are shown in Table 1. Six of these were valid argument forms, each of which was deformed to obtain a corresponding invalid argument form. For example, the second premise and conclusion of the *modus ponens* form were reversed to obtain an invalid form, generally known as *affirming the consequent*. By substituting either A, B, C or D or their negations for p and q in these schemas, a set of 576 different arguments was constructed. The network shown in Figure 1 was trained using the backpropagation algorithm to judge the form and validity of these arguments.[2] The arguments were encoded on an input layer of 14 units. These were connected through two layers of ten hidden units each to three output units, on which the network specified

Table 1. Argument forms used to train networks to evaluate arguments and complete enthymemes. There were six valid forms, each of which was deformed to create a corresponding invalid form (Bechtel and Abrahamsen 1991, p. 167).

Valid forms	Corresponding invalid forms
Moden ponens	Deformed *Modus ponens*
	(affirming the consequent)
If *p*, then *q*	If *p*, then *q*
p	*q*
∴*q*	∴*p*
Modus tollens	Deformed *modus tollens*
	(denying the antecedent)
If *p*, then *q*	If *p*, then *q*
not *q*	not *p*
∴not *p*	∴not *q*
Alternative syllogism	Deformed alternative syllogism
p or *q*	*p* or *q*
not *p*	*p*
∴*q*	∴not *p*
p or *q*	*p* or *q*
not *q*	*q*
∴*p*	∴not *p*
Disjunctive syllogism	Deformed disjunctive syllogism
Not both *p* and *q*	Not both *p* and *q*
p	not *p*
not ∴*q*	∴*q*
Not both *p* and *q*	Not both *p* and *q*
q	not *q*
not ∴*p*	∴*p*

the form and validity of the argument. All arguments generated from one pair of forms (original and distorted) were assigned the same encoding for form on the first two output units; the original (valid) forms were distinguished from the corresponding distorted (invalid) forms by the validity encoding on the third unit. For example, the network was trained to encode all instances of affirming the consequent as *modus ponens* (the form from which they had been derived by distortion) and as invalid (to mark the logical effect of the distortion).

Initially 192 of the arguments were used to train the network. These were chosen to include at least one valid and one invalid example of each basic problem type, e.g., there was at least one valid instance of *modus ponens* with *A* or ~*A* as the antecedent and *B* or ~B as the

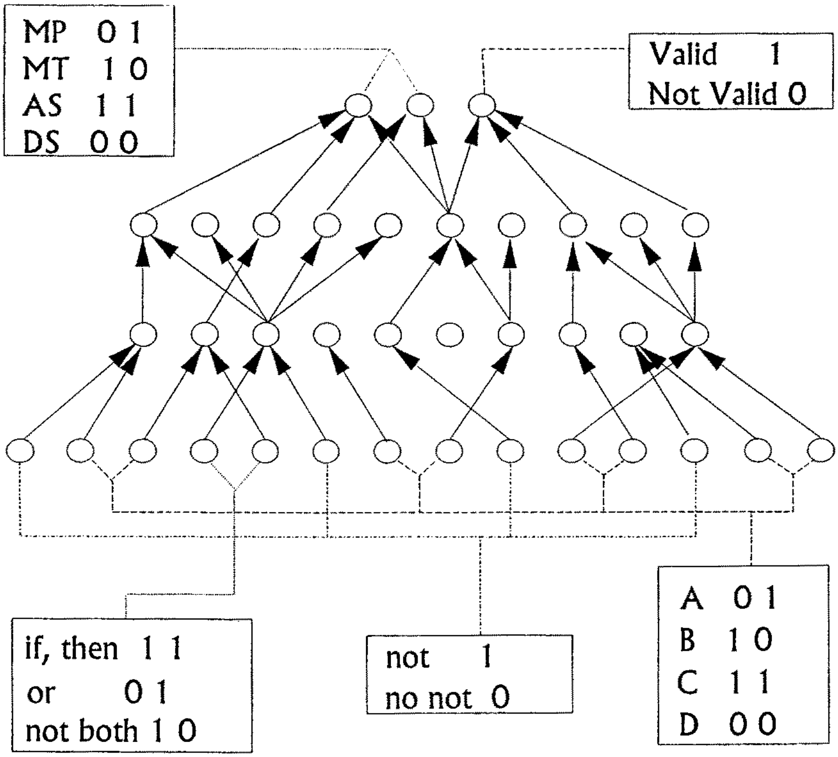

Fig. 1. Multi-layer network used for evaluating simple argument forms in sentential logic. The interpretation of each unit in the input and output layers is shown in one of the boxes. The network includes all possible connections between adjoining layers; only some of these are shown (from Bechtel and Abrahamsen 1991, p. 169).

consequent. After 3,000 presentations of the training set, the network had learned to correctly judge the form and validity of all 192 arguments in that set. To assess generalization, it was tested on 192 additional arguments which it had not previously encountered, and was correct on 139 (76%). It was then trained further on the combined set of 384 arguments for 5,000 presentations, after which it was correct on all but four. Finally, generalization was assessed on an additional set of 192 arguments and was correct on 161 (84%).

The second network was trained to complete enthymemes employing the same set of arguments as above. In this case, there were input units for the complete argument together with the designation of the name of the form and the validity of the argument. To make the argument into an enthymeme, neutral activations (0.5) were presented on the units representing either the second premise, the conclusion, or the name and validity of the argument; the rest of the units received non-neutral activations (0 or 1 as appropriate). The output units corresponded to the input units; the network's task was to reconstruct on the output units the complete argument, with the proper substitutions supplied for the neutral values on the input units. Hence, the task was one of pattern completion. The network was trained on 384 of the argument forms using 30,000 presentations of each form. At that point the network responded correctly on 380 of the problems. Generalization was then tested using the remaining 192 arguments. On 128 of these the network was required to supply either the second premise or the conclusion. The network correctly completed 125 (97.6%). On the remaining 64 problems the network was required to specify the form and validity of the argument. It made numerous errors on this task, misidentifying the form 14 times, misjudging the validity 18 times, and erring on both form and validity an additional 2 times. Here performance was far less impressive, although still far better than chance. The relatively poor performance on the arguments that required specification of form and validity was surprising since the previous network performed much better on this task. The fact that the network did so well on those problems that required supplying the missing premise or conclusion, though, indicates that networks are capable of completing simple enthymemes in sentential logic.

These two investigations confirm that simple problems in logic can be successfully treated as problems of pattern recognition and completion that can be solved by a feedforward network. Moreover, the syntactically ordered symbolic representations of the arguments remain external to the network in the sense employed in this paper: they are encoded only on the input and output layers of the network and are not themselves manipulated in the internal processing. There are several ways in which we might extend these initial feasibility studies. First, the problem set could be enlarged by adding additional argument forms and atomic propositions. Second, if the number of additional atomic propositions in the learning set were very large or if the network were pushed to generalize to problems with new atomic propositions, the issue of variables and variable-binding would quickly become a high priority. Third, if problems with compound propositions were included, the ability of networks to handle constituent structure would receive a more challenging test. Fourth, semantics could be added so that we would have to grapple with how to keep the network from relying on semantics rather than syntax.

At least some of these extensions ultimately will be needed to convincingly complete the demonstration that logic problems can be solved by processes of pattern recognition using the distributed representations of connectionist networks. It is not yet clear what kinds of network models will best fulfil this agenda, but the challenges are not unique to logic and a variety of avenues are being pursued by connectionist investigators. See, for example, Pollack's (1990) RAAM networks and Smolensky's (1990) tensor product representations.

Although extensions like these may ultimately be crucial to the story of how we do logic, it is also important to probe the capabilities of simple feedforward networks. The next section describes how I did this by posing a different kind of logic problem to another feedforward network.

2. SIMULATING NATURAL DEDUCTION IN NETWORKS

One might argue that while evaluating simple sentential arguments and completing enthymemes can be easily construed as a task of pattern recognition, and thus within the power of simple feedforward networks, natural deduction is not. Natural deduction requires the deployment of simple argument forms, now construed as principles[3] of inference, in a sequential manner so as to progress from premises to the conclusion, construed as a goal state. This is a much more difficult task. One cannot just deploy principles that are applicable given the premises and the steps derived so far; rather, one must formulate and implement a strategy, maintaining a focus on the goal and adding steps to the argument that are likely to help reach the goal. Often this requires reasoning backwards from the goal, determining that if I were able to establish this statement, then I would be able to employ this principle to reach my conclusion.

What does such means-ends reasoning require? One possibility is that one learns a variety of strategies specifying what one should do in particular kinds of circumstances. In learning natural deduction students often seek such strategies. If the instructor develops a derivation for them, students often ask about a particular line in the derivation: "Why did you do that?" For example, in the following derivation, a student might ask why on step 3, I derived F rather than B, which was equally derivable by and-elim from step 2.

1.	$D \& (F \supset A)$	:premise
2.	$B \& F$	:premise
3.	F	:2 &-elim
4.	$F \supset A$	:1 &-elim
5.	A	:4,3 $\supset$ -elim

I, and I suspect nearly all logic instructors, answer such queries by offering reasons that could be compiled into strategies. For example, we might note that the conclusion was the consequence of a conditional statement found in a premise, and if we could derive the antecedent of the conditional, we could use $\supset$-elim to derive the conclusion. Some such strategies are even abstracted and discussed in logic textbooks or used in computer programs that generate derivations.[4] But when I engage in offering such explanations, I strongly suspect that I am confabulating. Having done innumerable derivations, when I first look at natural deduction problems, especially relatively simple ones, I immediately *see* what to do. This is a contentious claim which I cannot demonstrate. One piece of evidence for it, however, is that the strategies I present to students are underspecified; there are circumstances in which it is inappropriate to apply the strategy I have specified, and when these arise and I do not follow the strategy, a student who has heard me enunciate it will then ask why I did not follow it.

There is an alternative to the strategy account that is worthy of exploration. Expertise in natural deduction may be viewed as a particular case of Dreyfus and Dreyfus's (1986) analysis of expertise. In their account, rules arise at the stage of competent performance, but when true expertise arises, what they term *intuition* predominates. The term *intuition* suggests something mysterious, but that is not what is intended. Rather, the term refers to the ability to recognize directly that a particular situation is comparable to one experienced previously, and to use the solution to the previous situation as a basis for dealing with the current one. By the time one teaches natural deduction, one typically has a great deal of experience developing derivations, and on this basis is able to recognize immediately that a particular problem is comparable to previous cases. In some domains of expertise, such as nursing diagnosis (Benner 1984), these previous cases may be explicitly remembered. But in others, such as natural deduction, we may simply experience awareness that a particular strategy is likely to succeed for a particular kind of problem.

The proposal that expertise in natural deduction might rely on recognizing patterns on the basis of considerable experience in constructing derivations motivates the attempt to simulate such expertise in connectionist networks. In what follows I describe a simulation in which a feedforward network is trained by backpropagation to construct a restricted set of natural deductions from sentential logic. Like the networks for evaluating arguments and completing enthymemes discussed above, this network is trained with a limited set of problem types constructed from particular atomic propositions. A different architecture might be needed to incorporate variable binding or to solve novel problem types. There is, however, one important change in design relative to the earlier networks. Natural deduction is viewed as a cooperative effort between internal processing and external symbols which keep track of the completed lines of the derivation. To simulate this, each line of the derivation is assigned to a different set of input units, and several passes are made through the network to construct the entire derivation (one pass per added line). On each pass the correct encoding of the lines completed so far is provided on the input units, and the network must try to generate the next line. Hence, memory for completed work is externally available, reducing and changing what is required internally. In this way, the simulation explores the strategy of dividing labor between external structured representations and a pattern-recognizing connectionist system.

The natural deductions used by the network followed fourteen derivation patterns (Table 2). These derivation patterns involved derivations using either three or four premises and either three, four, or five inferential steps. A line in the derivation could consist of one or two atomic sentences, possibly negated and one two-place connective. Five inference rules were employed: v-intro, v-elim, &-intro, &-elim, and ⊃-elim. In constructing the problem set, three permutations of the order of the first three premises were employed for all 14 derivations patterns: 1, 2, 3; 2, 1, 3; and 3, 1, 2.[5] Twenty-four derivations were constructed from each of these schemas using the sentential constants *A*, *B*, *C*, *D* in all possible permutations to replace the sentential vari-

Table 2. The 14 derivation schemas used with the natural deduction networks.

I		II	
1. $p \vee q$	:pr	1. $p \vee q$	:pr
2. $q \supset r$	:pr	2. $q \supset r$	:pr
3. $\sim p$	:pr	3. $\sim p$	:pr
4. q	:1,3 v-elim	4. q	:1,3 v-elim
5. r	:2,4 ⊃ -elim	5. r	:2,4 ⊃ -elim
6. $r \vee s$	:5 v-intro	6. $r \,\&\, q$	:5 &-intro
III		IV	
1. $p \supset q$	:pr	1. $p \supset q$	:pr
2. $q \supset r$	:pr	2. $q \supset r$	:pr
3. p	:pr	3. p	:pr
4. q	:1,3 ⊃ -elim	4. q	:1,3 ⊃ -elim
5. r	:2,4 ⊃ -elim	5. r	:2,4 ⊃ -elim
6. $r \vee s$	:5 v-intro	6. $r \,\&\, q$	:5;4 &-intro
V		VI	
1. $p \vee q$	:pr	1. $p \supset q$	:pr
2. $q \supset r$	:pr	2. $q \supset r$	:pr
3. $\sim p \,\&\, s$	:pr	3. $p \,\&\, q$	:pr
4. $\sim p$	:3 &-elim	4. p	:3 &-elim
5. q	:1,4 v-elim	5. q	:1,4 ⊃ -elim
6. r	:2,5 ⊃ -elim	6. r	:2,5 ⊃ -elim
VII		VIII	
1. $p \,\&\, q$	:pr	1. $p \,\&\, q$	:pr
2. $p \supset \sim r$	:pr	2. $p \supset r$	:pr
3. $r \vee s$	:pr	3. $q \supset s$	:pr
4. p	:1 &-elim	4. p	:1 &-elim
5. $\sim r$	:2,4 ⊃-elim	5. r	:2,4 ⊃-elim
6. s	:3,5 v-elim	6. q	:1 &-elim
7. q	:1 &-elim	7. s	:3,6 ⊃-elim
8. $s \,\&\, q$	:6,7 &-intro	8. $r \,\&\, s$	:5,7 &-intro
IX		X	
1. $\sim p \,\&\, q$	:pr	1. $\sim p \,\&\, q$	:pr
2. $p \vee r$	:pr	2. $p \vee \sim r$	:pr
3. $r \supset s$	:pr	3. $r \vee \sim s$	:pr
4. $\sim p$	:1 &-elim	4. $\sim p$	:1 &-elim
5. r	:2,4 ⊃-elim	5. $\sim r$	:2,4 v-elim
6. s	:3,5 ⊃-elim	6. s	:3,5 v-elim
7. q	:1 &-elim	7. q	:1 &-elim
8. $s \,\&\, q$	:6,7 &-intro	8. $s \,\&\, q$	:5,7 &-intro

Table 2. Continued.

XI		XII	
1. $p \supset q$	:pr	1. $p \vee \sim q$	:pr
2. $p \supset r$	:pr	2. $p \vee \sim r$	:pr
3. $r \supset s$	:pr	3. $r \vee s$	:pr
4. p	:pr	4. $\sim p$	:pr
5. q	:1,4 $\supset$-elim	5. $\sim q$	:1,4 v-elim
6. r	:2,5 $\supset$-elim	6. $\sim r$	:2,5 v-elim
7. s	:3,6 $\supset$-elim	7. s	:3,6 v-elim
8. s & q	:7,5 &-intro	8. s & $\sim r$	:7,6 &-intro
XIII		XIV	
1. $p \vee q$	:pr	1. $p \supset \sim q$	:pr
2. $q \supset r$	:pr	2. $q \vee \sim r$	:pr
3. $r \supset s$	:pr	3. $r \vee s$	:pr
4. $\sim p$	:pr	4. p	:pr
5. q	:1,4 v-elim	5. $\sim q$	:1,4 $\supset$-elim
6. r	:2,5 $\supset$-elim	6. $\sim r$	:2,5 v-elim
7. s	:3,6 $\supset$-elim	7. s	:3,6 v-elim

ables *p*, *q*, *r* and *s*. I shall refer to a set of derivations which share a particular assignment of sentential constants to the sentential variables as a *substitution set*. Derivations constructed from three-fourths of the substitution sets were used as the training set; accordingly, the training set consisted of 756 derivations. Derivations from the remaining one-fourth of the substitution sets (chosen to be representative of the overall distribution) were reserved for testing generalization. These totalled 252.

A feedforward connectionist network with one layer of hidden units was used for the simulation (Figure 2). The input layer consisted of 104 units, each of which received an activation of 0 or 1. Each derivation was presented to the input units in steps. On the first step the network was provided with an encoding of the final conclusion to be constructed and the premises. On subsequent steps it would also be presented with the correct steps in the derivation up to that point. As noted above, the representation of all of this material on the input layer is intended to emulate the ability of a person to take in as visual input the written representation of the problem and the steps completed so far. The input units fed into a layer of hidden units. Experimentally it was determined that no more than twenty hidden units were needed to learn this task. The output layer consisted of 13 units, on which the

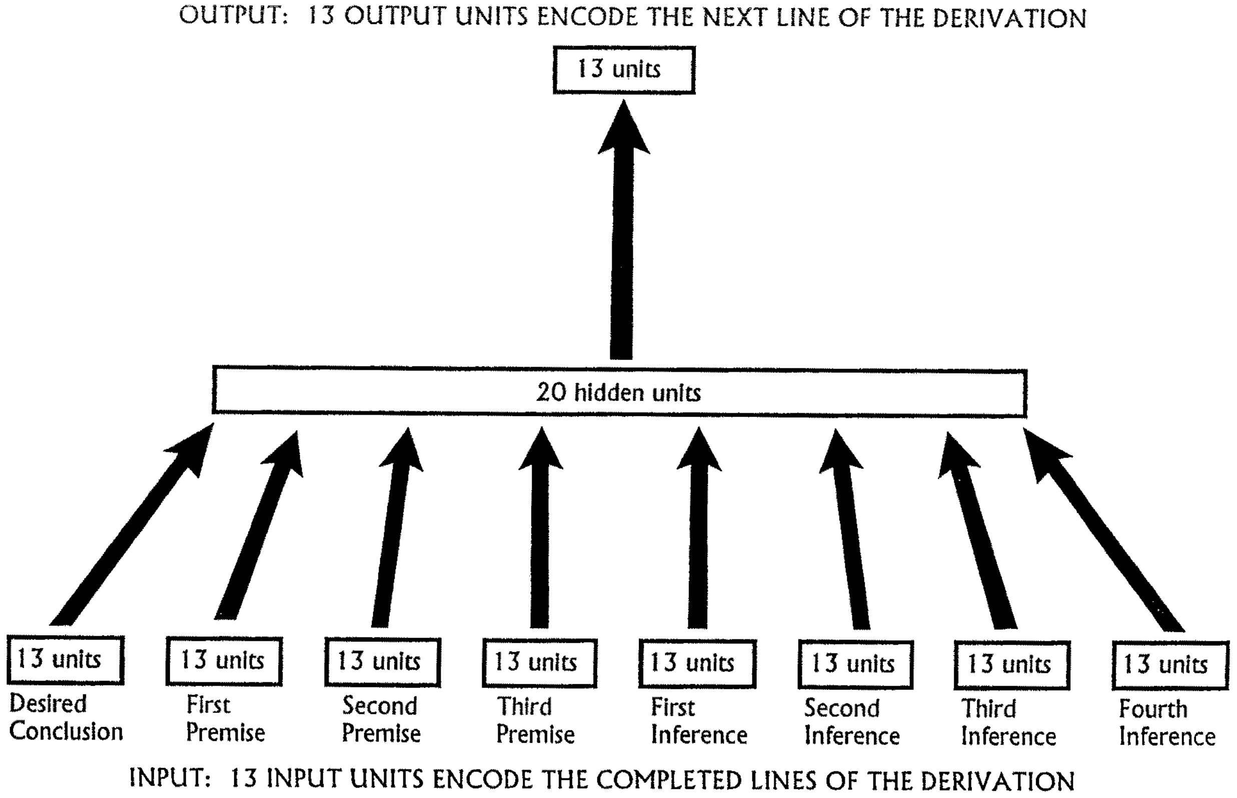

Fig. 2. Network trained to construct natural deductions from three premises. If there are four premises, the group here designated 'First Inference' encodes the last premise, and the first inference is presented on the next set of units to the right, etc. Each arrow represents 260 connections (e.g., each of the 13 units for the first premise has a connection to each of the 20 hidden units). Units in groups for lines not yet completed are all left blank.

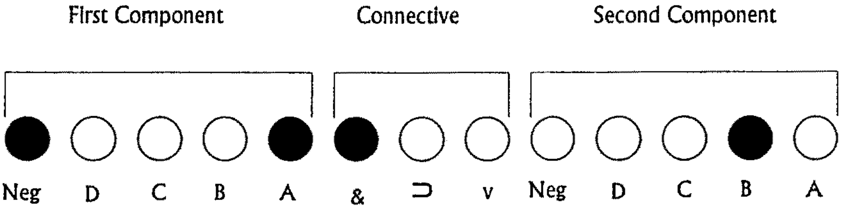

Fig. 3. Encoding schema used to encode one line in a derivation on 13 units. Each unit represents a negation, a sentential constant, or a connective. The units active in this case represent the statement ~A and B.

network was trained to produce an encoding of the next step in the derivation. Although the hidden units and output units could take continuous activations between 0 and 1, the target outputs were patterns of 0s and 1s. On the input and output layers 13 units sufficed to encode each line of the derivation in such a manner that a unit having an activation of 1 or close to 1 represented the presence of a negation, sentential constant, or connective assigned to that unit. Activation of unit 1 served to negate the first sentential constant; activation of unit 9 negated the second sentential constant. Units 2–5 and 10–13 specified the first and second sentential constants; the first unit in each of these clusters signified *D*, the second *C*, etc. Units 6–8 encoded the connective (unit 6 represented &, unit 7 ⊃, and unit 8 *v*). Figure 3 shows the manner in which the proposition *~A & B* was encoded. If the line consisted simply of an atomic letter or its negation, the first 8 units were assigned an activation of 0.

To present a complete derivation problem to the network, the appropriate activation pattern was presented to the three or four sets of units encoding the premises as well as the set of units encoding the desired conclusion, and the remaining sets of units were left off. The network was required to construct the proper next line on the output units (that is, it had to make the first inference). Then, regardless of what pattern was generated by the network on the output units, the desired pattern (the correct first inference) became part of the input on the next step. That is, the network was supplied with the premises, desired conclusion and correct first inference. From this input the network's job was to generate the second inference. This process was repeated until the network had constructed all of the steps of the derivation. (Note that the only representation of the previously completed steps is on the input layer, which corresponds to the visual input a human might have who has written the problem and the steps completed so far on paper.)

To train each derivation, the network was presented with each step in the appropriate sequence. For each step, the network produced its answer on the output units, was informed of the correct answer, and, in accord with the backpropagation learning algorithm, used the difference between its answer and the correct answer to revise the weights on the connections in the network. An epoch of training consisted of one complete pass through each of the derivations in the training set. After only a few epochs of training the network began to get most of the steps correct. After 500 training epochs the network had mastered the training set. Although technical factors such as the number of hidden units influence the rate of learning, the fact that the network learned these problems so fast suggests that natural deduction was not actually as challenging a task as it appeared at the outset. The incorporation of an external memory for the completed lines of the derivation presumably played a major role in rendering the task tractable.

The 252 derivations reserved for testing generalization involved a total of 936 inferences. The network was judged to have made the inference correctly if all units that had values of 1 in the target had activations greater than 0.5 and units that were 0 in the target had activations less than 0.5.[6] The network was correct on 767 of these inferences (81.9%). All of the errors the network made involved the sentential constants. There were no errors involving either the negation operators or the main logical connectives. The errors involving the sentential constants fell into three categories: (a) no unit representing a sentential constant was activated above 0.5 (interpretable as a failure to respond); (b) both the correct and one incorrect unit were activated above 0.5 (interpretable as uncertainty as to the correct response); or (c) an incorrect unit was activated above 0.5 (interpretable as the simple error of producing the wrong sentential constant).

In those cases in which either no unit for a sentential constant was activated above 0.5 or where more than one was, we can examine which unit had the highest activation. In 80 of the 169 cases in which errors were made, the correct unit was the one with the greatest activation. There is a clean-up technique that can be used to raise the activation of the winning unit and suppress the activation of all of the other units in a cluster (this involves adding excitatory connections from a unit to itself and inhibitory connections to all other units in the cluster). If such a technique had been employed, the percentage correct would have risen to 90.5%.

The errors the network made were not randomly distributed. Using the initial, more conservative criterion of correctness, 76 of the errors occurred on instances of derivation patterns I–IV, resulting in a percentage correct on those problems of only 73.6%. On the other hand, only 15 errors were made on problem types V, VI, XIII and XIV combined, yielding a percentage correct of 94.7%. An examination of derivation patterns I–IV reveals that they are extremely similar to one another. As a result, when an error occurred in one of these problems, it generally occurred in several other problems. Thus, in two of the substitution sets in the test set, *q* was replaced by *D*. Under this substitution the first inferential step in all four derivation patterns should be *D*. In 14 of the 24 derivations within these two substitution sets the network made an error on this step, either providing no response above 0.5, activating the unit representing *A*, or in one case activating the units for both *A* and *D*. However, on the eight cases in which the premises appeared in the order 2, 1, 3 the network made no errors. Thus, the network had learned to make this inference when the disjunction was the second premise and the negation of the first disjunct was the third premise, but not in the other cases. A similar pattern of consistent errors was exhibited in one of the substitution sets in which *r* was replaced by *B*. Here the network made an error on the second derivation step in 10 of the 12 instances of patterns VII–X in which the correct response should have been either *B* or ∼*B*.

These are the most extreme cases in which a pattern can be found in the errors, but there were other less dramatic cases in which the same type of error was made on several similar problems. This indicates that many of the errors are highly localized and that rather than simply making random errors, the network learned incorrect generalizations of specific inference patterns. In the training set, though, the network had encountered numerous instances of each of these types of inferences involving the same substitution for the sentential variable in question, and had learned to produce all of these correctly. Thus, the problem was one of learning to generalize and make the same inference when required in problems in which the other sentential variables were replaced with different sentential constants. More significantly, the fact that the network makes systematic errors and is highly accurate on the other cases suggests that the network has learned to generalize patterns of inference.

Although the network had no problem with proper use of negations on this first task, the derivation sets were constructed using only *A*, *B*, *C*, or *D*, not their negations, as substitutions for the sentential constants *p*, *q*, *r* and *s*. In a second simulation, the previous set of derivation sets was supplemented with three additional sets in which the substitution for either *q*, for *p* and *r*, or for *q*, *r* and *s* were negated. (Thus, in the first case, if *A* originally was to be substituted for *q*, ~*A* would be substituted instead; double negations were automatically removed.) This introduction of additional substitution sets required the network to be extremely sensitive to the proper placement of the negation signs. Because of the increased complexity of the stimulus set, the number of hidden units was increased to 40. In one version of the experiment with this expanded problem set, one-fourth of the derivations were selected for the test set in such a manner that if a substitution set in which one assignment of sentential constants to sentential variables was chosen for the test set, then none of the other substitution sets differing only in assignments of negations was chosen. These other permutations using the same atomic sentences but differing in negation were included in the training set. The network learned this task rapidly, and after 100 epochs made no errors on the training set. On the test for generalization the network was also perfect. Apparently proper placement of negation signs was not a challenge in this case.

In a second variation with this expanded problem set the principle for selecting the test set was the reverse: if a substitution set in which one assignment of sentential constants to sentential letters was chosen for the test set, the other three substitution sets differing in assignments of negations were also chosen. Thus, if the substitution set in which *A*, *D*, *C*, *B* were chosen to replace *p*, *q*, *r*, *s* was selected, then so were the substitution sets in which *A*, ~*D*, *C*, *B*; ~*A*, *D*, ~*C*, *B*; and *A*, ~*D*, ~*C*, ~*B* served as the replacements. Four such sets of four substitution sets each (a total of 16 substitution sets) were selected for the test set, leaving 80 substitution sets in the training set. The network was again trained for 100 epochs and learned all of the cases. On the test for generalization the network was correct on 2,112 out of 2,496 inferential steps it was required to make (84.6% correct). In this variation of the experiment, five errors arose when the network failed to supply the correct negation sign. The rest of the errors, as in the experiment without the extra negations, involved activating the unit for an incorrect sentential constant, failing to activate any unit for an atomic sentences above 0.5, or activating the units for two atomic sentences. (When the more lax criterion requiring only that the unit representing the correct atomic sentence be the most active in the group was employed, the network was correct on 2,336 inferences, yielding 93.6% correct.)

There was once again a significant degree of consistency in the errors the network made so that if an error were made on an inference with a particular substitution set, the same error was likely to occur on other inferences in the substitution set that had the same form. Moreover, in this version there were four sets of four substitution sets that were identical except for negation signs. If an error appeared on an inferential step in a particular derivation using one of the substitution sets, it was very likely to also appear in the other three. This is a further indication that the network has acquired sensitivity to the logical structure of the problems: most of the errors the network makes reflect inferential patterns that the network has not learned to generalize correctly and the same error shows up repeatedly. On the other cases, which are the vast majority, the network has learned the pattern and the errors that occur are very sporadic.

In all of the tests of generalization reported so far, the network was required only to generalize to a substitution instance of a derivation pattern it had already experienced. The network was not required to generalize to derivations in which the inferential steps would be put together in different orders. In a further test of the generalization capacities of this last network, I tested it on a new set of derivations constructed following the patterns shown in Table 3. These derivation patterns are modelled on those used in the training, but introduce variations. For example, schema XV follows the general pattern of schemas I and II, but employs two disjuncts in the first two premises rather than one or two conditionals. Once again, three permutations of the first three premises were employed. Four substitution sets were constructed, substituting for p, q, r and s either B, A, D, C; A, $\sim D$, B, C; $\sim D$, B, $\sim A$, C; or B, $\sim D$, $\sim A$, $\sim C$.

There were a total of 252 inferential steps in these derivations, of which the network was correct on 198 (78.6% correct). An examination of the errors is also revealing as they were not equally distributed. Half

Table 3. Six new derivation schemas used to test generalization of the natural deduction network.

XV		XVI	
1. $p \vee {\sim}q$	:pr	1. $p \supset {\sim}q$	:pr
2. $q \vee r$	:pr	2. $q \vee r$	:pr
3. ${\sim}p$	:pr	3. p	:pr
4. ${\sim}q$	:1,2 v-elim	4. ${\sim}q$	:1,3 ⊃-elim
5. r	:2,4 v-elim	5. r	:2,4 v-elim
6. $r \vee s$	:5 v-intro	6. $r \mathbin{\&} {\sim}q$	:5,4 &-intro
XVII		XVIII	
1. $p \supset {\sim}q$	:pr	1. $p \mathbin{\&} q$	:pr
2. $q \vee r$	:pr	2. $p \supset r$	:pr
3. $p \mathbin{\&} q$	:pr	3. $r \supset s$	:pr
4. p	:3, &-elim	4. p	:1, &-elim
5. ${\sim}q$	:1,4 ⊃-elim	5. r	:2,4 ⊃-elim
6. r	:2,5 v-elim	6. s	:3,5 ⊃-intro
		7. q	:1 &-elim
		8. $s \mathbin{\&} q$	:6,7 &-elim
XIX		XX	
1. $p \vee {\sim}q$	:pr	1. $p \supset q$	:pr
2. $q \vee r$	:pr	2. $q \supset {\sim}r$	:pr
3. $r \supset s$	:pr	3. $r \vee s$	:pr
4. ${\sim}p$	:3, &-elim	4. p	:pr
5. ${\sim}q$	:1,4 v-elim	5. q	:1,4 ⊃-elim
6. r	:2,5 v-intro	6. ${\sim}r$	:2,5 ⊃-elim
7. s	:3,6 ⊃-elim	7. s	:3,6 v-elim
8. $s \mathbin{\&} {\sim}q$	:7,5 &-intro		

the errors occurred on problems from the fourth substitution set. If this set is not considered, the network's score rises to 85.7% correct. Moreover, the problems were not equally difficult: 17 errors were made over the 48 inferences required on problem XIX, while only 2 errors were made on the 36 inferences required on problem XVI. Once again, there was considerable systematicity in the errors the network made. If the network had a problem with one step in a derivation in one substitution set, then it frequently experienced the same trouble when the order of the premises was reversed or when different problems required similar inferences. An unusual feature of this generalization test is that 34 of the errors involved incorrect use of the negation sign on inferences on which the network supplied the correct sentential constant. Whereas before the network seemed to have little difficulty with proper placement of negation signs, such errors now constituted the majority. Overall, though, the network performed well on the novel derivations.

3. DISCUSSION OF THE NATURAL DEDUCTION NETWORKS

The networks I have described in the previous section were able to perform the circumscribed problems of natural deduction which were posed to them. This demonstrates that a feedforward network that is set up to provide an external memory for keeping track of a stepwise proof can acquire sufficient sensitivity to the structure of derivations to generate each successive line of simple derivations of the types on which it has been trained. We saw that such networks can generalize to derivation instances constructed by making different substitutions for the atomic sentences in the derivation patterns beyond those on which they have been trained. Moreover, they can generalize to new derivation patterns which use individual inferences like those used in the schemas on which they have been trained, but put together in a new pattern. While generalization rates in the 78–85 % range are less than we might desire, they are comparable to what many students are able to achieve and also to the results of many other network simulations. Analysis of the errors indicated that many of the failures were due to the networks' having learned incorrect generalizations for particular patterns of inference, which further suggests that the networks are in fact sensitive to the logical structure of these derivations.

At the same time, it is important to note that these networks have not demonstrated the full range of capacities required for natural deduction. The inference rules used by these networks are not complete. For a natural deduction system to be complete, one must make use of indirect proofs such as found in *reductio* arguments. Moreover, each of the lines employed in these derivations involved at most two atomic propositions and a logical connective. Real natural deduction systems allow for embedding of compound propositions within the components of a proposition, as in $((A \& B) \vee C)$. Further, only four sentential constants were used. As a result, the network may not have learned the general principle that whenever a sentence of the form $p \supset q$ appeared as a previous line and the sentential constant that substitutes for p appears on another line, then the sentential constant that substitutes for q could be derived as a new line; thus, even if the encoding system allowed for encoding an new sentential constant *E*, it is doubtful that the network could now generalize to a new derivation using it. Whether it is possible to overcome these limitations with a simple feedforward network such as I have been using is unclear; sorting out claims concerning competing architectures will presumably form an important part of the future research agenda for network models of performance in logic. Finally, an inherent limitation of these networks is that, insofar as they do not know a natural language, they do not really understand what a sentence or a connective is. They have only made procedural distinctions between them based on their patterns of occurrence.

One quite legitimate worry about these simulations is that the network might be responding to simple cues rather than the logical properties of the problems. This objection was in fact raised by John Nolt in response to an early simulation with this network. In that simulation, only the first six argument forms from Table 2 were employed, and the premises were only presented in the order given. Nolt constructed an alternative set of three principles that could account for success on these problems, relying not on principles of logic, but on the position of particular statements in the premises. One way to determine which principles the network is adhering to is to analyze the weights in the network. Given the number of weights involved, however, such an investigation would be quite difficult. Another strategy is to present the problems in a way that would defeat such an alternative strategy. This was the reason for presenting the premises in three different orders: there were no longer cues in the presentation of the premises so as to permit Nolt's principles to solve the problem. As well, by adding additional types of problems, eventually the simplest strategy for solving the problems is to learn to obey the principles of logic. While it is possible that the network is adhering to principles other than those of logic, this seems unlikely given its success is this range of problems and its ability once trained to generalize to the arguments in Table 3.

To the extent that the network is sensitive to the logical properties of the problems, the internal processing part of the simulation has been successful. Recall, however, that this simulation was designed to explore a division of labor between (a) an internal cognitive system which relies primarily on pattern recognition capacities and (b) external symbols which themselves employ a compositional syntax, such as that of propositional logic. My suggestion has been that by using external symbolic representations a network can accomplish tasks that require a compositional syntax, but that representations employing this compositional syntax need not be employed in the internal processing of the cognitive system itself.

When Fodor and Pylyshyn present their objections to connectionism, they do admit that there is a way in which connectionists might overcome their objections. Connectionists might do this by implementing a symbolic system. If connectionists did this, however, they contend that it is the classical symbolic system, not the connectionist implementation, that is responsible for the cognitive performance. One might worry that this is precisely what I am doing here: the network is performing operations upon syntactically structured symbols, successively adding new symbols to the symbol string until a deduction is complete. However, only the input and output layers are explicitly symbolic. The key issue is the nature of the operations used to traverse the internal part of the network. Explicitly, the internal operations are those suggested by a neural metaphor: they are statistical operations on large numbers of weights and activations that produce distributed patterns of activation on hidden layers. If these explicit operations are actually just a means of implementing symbolic processing, then there is parallel activity at a higher (but implicit) level of description. The implicit operations would be those suggested by a linguistic metaphor: the application of rules so as to obtain and act on compositional representations. It is reasonable to discuss whether some aspects of network processing can fruitfully be summarized by higher-level descriptions. However, it would be an exercise in futility to try to prove that all of the explicit activity generated by one of my networks in solving a problem corresponds exactly to a series of implicit symbolic operations. It is neither necessary nor desirable that this be the case. Instead, I would point again to the advantages of a division of labor between language-like external representations and neural-like internal processing. To do its job, the internal processing system need not replicate the structure of an external representation that is made available on its input units; its knowledge of language and logic is procedural, cooperative, and sensitive to composition without being compositional. We would all like to attain a better understanding of the internal operations of networks, but focusing our search on functional equivalents to symbolic operations could keep us from noticing what is most worth seeing.

The difference between the approach I have pursued and that advanced by Fodor can be further clarified by returning to the approach to explaining human deductive abilities advanced by Braine et al. (1984), Osherson (1975) and Rips (1983). These theorists propose that rules of deduction are explicitly stored within the cognitive system and applied to mental representations of the premises of the argument, perhaps in successive steps, to arrive at the conclusion. Smith, Langston and Nisbett (1992), in making their case for the role of mental stored rules in reasoning, offer this sketch of what is going on when a rule is applied:

> When a test item or problem is presented, it is coded in a form that is *sufficiently abstract* to lead to access of an abstract rule: Once accessed, if need be, the rule can be used for further abstract coding of the test item. The next stage is to instantiate, or bind, the variables in the rule with entities from the input. Finally, the rule is applied to yield the desired answer; that is, inspection of the instantiated representation reveals that the antecedent of the rule has been satisfied, thereby licensing the conclusion. There are therefore four stages: coding, access, instantiation, (variable binding), and application. (*p*. 8)

These are, very clearly, not the sorts of activity going on in this network. Nonetheless, one might characterize the network as applying rules (what I have referred to as *principles*) to the inputs to arrive at outputs. This indicates how a connectionist might answer an objection Smith et al. advance against connectionism. Smith et al. present evidence that there is rule following in logic, and then note that their account of such rule following employs "notions of *explicit data structures* and *inspection of explicit structures* [which] simply lie outside the ontology of connectionism" (p. 35). There is a sense in which the logic network is engaged in rule following: one interpretation of its success is to say that the network has mastered a number of principles and applies them to the different symbolic structures presented as inputs. In this respect the natural deduction network is closer in spirit to the approach to deduction of Braine et al. (1984), Osherson (1975) and Rips (1983), than it is to the alternative approach of reasoning through mental models advanced by Johnson-Laird and Byrne (1991). Johnson-Laird and Byrne reject the proposal that reasoners master principles (rules) of natural deduction and propose that instead they learn to mentally manipulate semantic models generated for the premises. When one characterizes the natural deduction network as employing principles of natural deduction, however, it is important to note that it is the whole network which is applying the rule and that what it is applied to is an external representation, not a mental representation.

The aspect of my approach that distinguishes it from the classical symbolic approach is that I limit compositional syntax and semantics to what I am calling *external symbols*, not to mental representations that utilize a language of thought and then appeal to the compositional syntax of these external symbols to explain systematic and productive features of cognition. This poses a question: how could a compositional language have been created in the first place?[7] This is not the place to develop a theory of language origins, but answering this question would require appealing to both the phylogeny and ontogeny of language. Very briefly, once organisms learn to use single symbols referentially (Bechtel 1993), it is a natural further development to join such symbols together in utterances. Joining symbols, however, creates great potential for ambiguity as to the intended relations between the symbols. But the medium of the external symbols provides means of solving this problem. When we produce symbols, we do so in a serial manner. Thus, one potential tool for resolving ambiguity is word order. It is also possible to mark the external symbols (e.g., with syntactic case markings) so as to specify the relationships between symbols. Thus, external symbols afford ways of constructing compositional syntax that can be exploited by a tool-creating organism. Presumably social units select amongst these ways one that is sufficient for their communicative functions. The cognitive capacities required for developing compositional syntax in external symbols are the same ones I have been emphasizing in this paper: the capacity to generate external symbols and respond to them. As social units begin to select specific compositional principles from those that are tried out, the cognitive system must be one that can extract information from such structured symbols and construct symbols in accord with that structure.

4. CONCLUSIONS

The simulations presented in this paper offer support for the conception of logic that I developed in the first part of this paper. We do not need to view capacities in logic, including capacities for constructing natural deductions, as the product of a system which performs internal logical manipulations of symbols. Rather, the ability can be developed by a system capable only of pattern recognition in conjunction with an external representational system in which structured symbols can be stored. A cognitive processing system with capacities limited to those of pattern recognition, such as a simple connectionist network, can recognize patterns in these symbols, and as a product of recognizing these patterns, generate additional symbols which provide inputs for additional steps of processing. When we study the sequences of symbols produced in this manner we may discover that they are highly structured and that we can specify principles governing the permissible sequence of symbols and the strategies for constructing new sets of symbols. In order to adhere to these principles, the cognitive system requires extensive training. But a dynamical system that does not use syntactically encoded internal representations and syntactic rules to operate on them may nonetheless be able to conform in its construction of sequences of symbols to the principles of a highly syntactically structured system such as natural deduction.

ACKNOWLEDGEMENTS

Research for this paper was supported by a Chancellor's Initiative Grant from Georgia State University, which is gratefully acknowledged. I also thank Adele Abrahamsen for detailed comments and discussion of this paper as well as two anonymous referees for this journal. An earlier version of this paper was presented at the *Southern Society for Philosophy and Psychology* in March 1993. I thank John Nolt, who was the official commentator, and many members of the audience for their useful comments and suggestions.

NOTES

[1] Vygotsky (1962) argued that the capacity to produce inner speech is a further development of the ability to engage in public speech. Producing and responding to inner speech may well involve many of the same activities as for public speech, except that vocal tract activities are suppressed. I am construing such inner speech as a product of the cognitive system and hence external to it. For a related view, see Rumelhart (1989).

[2] This and the other simulations described below were all conducted using software for implementing backpropagation provided by McClelland and Rumelhart (1987).

[3] These principles are often referred to as *rules*. However, to make it clear that I am not referring to the internal processing of the system, I will refer to them as *principles*.

[4] Moreover, it seems likely that artificial intelligence architectures such as Newell's (1989) SOAR could use procedures such as chunking in building up a viable set of strategies and employ them as rules in constructing derivations.

[5] This was done to insure that the network was not relying on the order of the premises and any local cues that might provide. Otherwise, rather than adhering to a principle of logic, the network might simply learn that in certain circumstances it should copy over what appears on the final five units of the third line of the derivation.

[6] In fact in this and the other tests reported below, using this relatively lax criterion for correctness only marginally improved the percentage correct. Most of the units that should have been on had activations well above 0.9 and those that should have been off had activations well below 0.1.

[7] This important question was raised by an anonymous reviewer for this journal.

REFERENCES

Bechtel, W.: 1993, 'Decomposing Intentionality: Perspectives on Intentionality Drawn from Language Research with Two Species of Chimpanzees', *Biology and Philosophy* **8**, 1–32.

Bechtel, W.: in press, 'What Knowledge Must Be in the Head that We Might Acquire Language?', in B. Velichkovsky and D. M. Rumbaugh (eds.), *Naturally Human: Origins and Destiny of Language*.

Bechtel, W. and Abrahamsen, A.: 1991, *Connectionism and the Mind. An Introduction to Parallel Processing in Networks*, Basil Blackwell, Oxford.

Benner, P.: 1984, *From Novice to Expert: Excellence and Power in Clinical Nursing Practice*, Addison-Wesley, Boston.

Black, F.: 1968, 'A Deductive Question Answering System', in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, Massachusetts, pp. 354–402.

Braine, M. D. S., Reiser, B. J. and Rumain, B.: 1984, 'Some Empirical Justification for a Theory of Natural Propositional Logic', in G. H. Bower (ed.), *The Psychology of Learning and Motivation*, Vol. 18, Academic Press, Orlando, pp. 313–71.

Clark, A.: 1989, *Microcognition: Philosophy, Cognitive Science and Parallel Distributed Processing*, MIT Press, Cambridge, Massachusetts.

Dreyfus, H. L. and Dreyfus, S. E.: 1986, *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*, Free Press, New York.

Elman, J. L.: 1992, 'Grammatical Structure and Distributed Representations', in S. Davis (ed.), *Connectionism; Theory and Practice*, Oxford University Press, New York.

Fodor, J. A:. 1975, *The Language of Thought*, Crowell, New York.

Fodor, J. A 1987, *Psychosemantics: The Problem of Meaning in the Philosophy of Mind*, MIT Press, Cambridge, Massachusetts.

Fodor, J. A. and Pylyshyn, Z. A.: 1988, 'Connectionism and Cognitive Architecture: A Critical Analysis', *Cognition* **28**, 3–71.

Johnson-Laird, P. N. and Byrne, R. M. J.: 1991, *Deduction*, Erlbaum, Hillsdale, New Jersey.

Margolis, H.: 1987, *Patterns, Thinking. and Cognition: A Theory of Judgment*, University of Chicago Press, Chicago.

Newell, A.: 1989, *Unified Theories of Cognition*, Harvard University Press, Cambridge, Massachusetts.

Osherson, D. N.: 1975, *Logical Abilities in Children*, Vol. 3, Erlbaum, Hillsdale, New Jersey.

Osherson, D. N.: 1976, *Logical Abilities in Children*, Vol. 4, Erlbaum, Hillsdale, New Jersey.

Pollack, J.: 1990, 'Recursive Distributed Representations', *Artificial Intelligence* **46**, 77–105.

Rips, L. J.: 1983, 'Cognitive Processes in Propositional Reasoning', *Psychological Review* **90**, 38–71.

Rips, L. J.: 1990, 'Reasoning', *Annual Review of Psychology* **41**, 321–53.

Rumelhart, D. E.: 1989, 'Towards a Microstructural Account of Human Reasoning', in S. Vosniadou and A. Ortony (eds.), *Similarity and Analogical Reasoning*, Cambridge University Press, New York.

Rumelhart, D. E., McClelland, J. L. and the PDP Research Group: 1986, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, *Foundations*, MIT Press, Cambridge, Massachusetts.

Rumelhart, D. E., Smolensky, P., McClelland, J. L. and Hinton, G.E.: 1986, 'Schemas and Sequential Thought Processes in PDP Models', in J. L. McClelland, D. E. Rumelhart and the PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 2, *Psychological and Biological Models*, MIT Press, Cambridge, Massachusetts.

Smith, E. E., Langston, D. and Nisbett, R. E.: 1992. 'The Case for Rules in Reasoning', *Cognitive Science* **16**, 1–40.

Smolensky, P.: 1990, 'Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems', *Artificial Intelligence* **46** 159–216.

van Gelder, T.: 1990, 'Compositionality: A Connectionist Variation on a Classical Theme', *Cognitive Science* **14**, 355–84.

Vygotsky L.: 1962, *Thought and Language*, MIT Press, Cambridge, Massachusetts.

Wason, P. and Johnson-Laird, P. N.: 1972, *Psychology of Reasoning: Structure and Content*, Harvard University Press, Cambridge, Massachusetts.

Philosophy–Neuroscience–Psychology Program
Department of Philosophy
Washington University in St. Louis
St. Louis, MO 63124
U.S.A.