Connection Science

ISSN: 0954-0091 (Print) 1360-0494 (Online) Journal homepage: https://www.tandfonline.com/loi/ccos20

Are Feedforward and Recurrent Networks Systematic? Analysis and Implications for a Connectionist Cognitive Architecture

Steven Phillips

To cite this article: Steven Phillips (1998) Are Feedforward and Recurrent Networks Systematic? Analysis and Implications for a Connectionist Cognitive Architecture, Connection Science, 10:2, 137-160, DOI: 10.1080/095400998116549

To link to this article: https://doi.org/10.1080/095400998116549

Published online: 01 Jul 2010.

Submit your article to this journal

Article views: 29

Citing articles: 6 View citing articles

Connection Science, Vol. 10, No. 2, 1998, 137–160

Are Feedforward and Recurrent Networks Systematic? Analysis and Implications for a Connectionist Cognitive Architecture

STEVEN PHILLIPS

(Received for publication August 1997, revised paper accepted March 1998)

Human cognition is said to be systematic: cognitive ability generalizes to structurally related behaviours. The connectionist approach to cognitive theorizing has been strongly criticized for its failure to explain systematicity. Demonstrations of generalization notwithstanding, I show that two widely used networks (feedforward and recurrent) do not support systematicity under the condition of local input/output representations. For a connectionist explanation of systematicity, these results leave two choices: either (1) develop models capable of systematicity under local input/output representations or (2) justify the choice of similarity-based (non-local) component representations sufficient for systematicity.

KEYWORDS: Systematicity, architecture, feedforward, recurrent.

1. Introduction

Ten years (1988–1997) of the systematicity debate and its implications for cognitive architecture have had little if not no impact on the day-to-day practices of the connectionists' version of cognitive science. Despite the seminal papers of Fodor and Pylyshyn (1988) and Smolensky (1988), and numerous rebuttals from both sides (see Fodor, 1997; Fodor & McLaughlin, 1990; Smolensky, 1995a, among many others), architectures that predate the debate still appear to be the connectionists' first choice (e.g. Plunkett & Elman, 1997). In fact, enthusiasm for the recent 'Rethinking Innateness' series (see the cover note to Elman *et al.*, 1996) suggest that the debate has had no effect on the general programme of establishing principles for a connectionist cognitive architecture.

Toward this goal, the purpose of this paper is to analyze two widely used networks: the feedforward and recurrent networks, for their capacity to support systematicity. The main result is that neither network is systematic in the sense that learning some behaviours conveys 'appropriate' generalization to other structurally related behaviours. What it means to support appropriate generalization is defined in the next section, along with the term 'systematicity' and its significance to

S. Phillips, Information Science Division, Electrotechnical Laboratory, 1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan. Tel: 81 298 543315; Fax: 81 298 545857; E-mail: stevep@etl.go.jp.

cognitive architecture. Relative to that definition, simulations and analyses are then conducted on feedforward and recurrent networks for their capacity to demonstrate systematicity. Finally, the implications of these results are discussed in terms of principles for a connectionist cognitive architecture.

2. Systematicity and Generalization

One basic observation that can be made about human cognition is that cognitive ability is organized into groups, or collections of behaviours. Rather than being distributed arbitrarily over the space of possible behaviours, cognitive abilities are organized with particular regularity. Just as the firing of electrons into a sheet of material reveals a high degree of atomic structure in physics,[1] so too does the probing of mental capability in humans suggest some degree of structure in cognition. The structure-based grouping of cognitive ability is referred to s systematicity, and its implication for cognitive architecture is the systematicity debate.

The following examples (adapted from Fodor and Pylyshyn (1988)) illustrate the organized nature of cognitive ability. If one has the ability to infer that *John went to the store*, given that *John and Mary went to the store*, then one also has the ability to make the related inference *Mary went to the store*, given that *Mary and John went to the store*. Both these inferences share the same (logical) structure: $P \wedge Q \rightarrow P$. Rephrased, there are no cognitive states whereby one can make the first inference, but not the second. However, neither of these two inference capabilities says anything about the structurally unrelated inference *two plus three equals five*.

The classical solution to systematicity is to posit structured representations, and processes sensitive to the structure of those representations (Fodor & Pylyshyn, 1988). So, for example, having a representational state that identifies the components, P, $\wedge$ and Q, and a cognitive process that maps this state to a new state identifying P, this process extends to any situation in which the input (and output) can be mapped to (and from) those states (e.g. *John and Mary*, *Mary and John*, etc.). A parser is an example of this sort of architecture.

One half of Fodor and Pylyshyn's argument[2] against connectionism is that the architectural components of a network (e.g. activations, nodes and weighted connections) do not enforce systematicity. Very briefly, in the localist case, where one node represents some complex object, structure-sensitive processing cannot be supported by simply labelling the node with the corresponding representation. Since only node activity (e.g. 0's and 1's) represents the inputs to other network processes, labels have no causal consequences and so cannot be used to support systematicity. In the distributed case, where representations are the activations of collections of nodes, activity vectors alone will not suffice since, in general, decomposition of vectors into the structural components they supposedly represent is not uniquely determinable.

While one may accept that localistic associative networks are not sufficient to capture systematicity, demonstrations of generalization by more complex networks utilizing distributed representations and processes would seem to have answered Fodor and Pylyshyn's objections. If such generalization results are (or are not) demonstrations of systematicity, then we need to know why, so that the relevant architectural properties can be replicated (or avoided) in other connectionist systems designed to incorporate systematicity.

If learning is to be the additional connectionist property that enforces systematic behaviour, then a suitable criterion for systematicity as generalization must be specified. For the rest of the paper, criteria devised by Hadley (1993, 1994) are considered as the requirement for systematicity. There are difficulties with treating systematicity as a generalization problem. It is difficult to determine exactly what level of generalization is achieved by humans. This uncertainty is exaggerated by lack of control over a subject's background knowledge, and the strategies that they employ in any given task. Yet, one can take a bounds approach which can still yield interesting results. If, for example, one can safely identify a lower bound on the degree of generalization exhibited by humans, then one can immediately rule out a class of architectures that do not meet this lower bound. The results presented here are of this form.

2.1. *Strong Systematicity*

Based on linguistic evidence of generalization, Hadley formulated a criterion for systematicity. In Hadley's (1993) definition: "a model is regarded as strongly systematic if it demonstrates correct performance on components in novel (syntactic) positions".[3]

It is assumed that generalization performance must be above chance level. If there are only five possible output responses, and one takes the maximally activated output unit as the network's response, then there is a base level 20% chance of demonstrating generalization to the novel position. Yet, this would not qualify as a demonstration of generalization. I assume that human performance is significantly above chance level, and I refer to further experimental support for this assumption in Section 5. An example of strong systematicity would be to infer correctly that *John* is the lover of *Mary*, given the premise *John loves Mary*, having only been trained on examples where *John* appeared in the object position of the binary relation *loves*.

Hadley (1993, 1994) found that the models of McClelland and Kawamoto (1986), Chalmers (1990), Elman (1989, 1990, 1991), Smolensky (1990) and St John and McClelland (1990) did not demonstrate strong systematicity. Essentially, by a statistical analysis of training sets, in all likelihood, all components appeared in all of their allowable positions.

The result raises the question as to whether the lack of strong systematicity was due to a fundamental limitation of these connectionist models, or simply that the intention was not to demonstrate strong systematicity, but some other phenomenon. This section attempts to address this question through analysis and simulation of a number of connectionist architectures. However, before this analysis, a point must be made regarding component representations.

2.2. *Component Similarity*

Work by Niklasson (1993) claims to have demonstrated strong systematicity on a transformation of a logical expressions task. However, component representations presupposed a similarity based on their position, or role, within a complex expression. For example, the proposition symbols (p, q, r, s) all shared a common feature in their vector representation. Furthermore, the so-called novel component, s, on which the network was tested had as its representation the common feature with no other features (vector components) active. Consequently, the generalization demonstrated was due to the common feature on which the network had already been trained in the components of p, q, r, and so cannot be considered a demonstration of strong systematicity,[4] without an explanation as to how the propositional symbols obtained similar representations.

The use of surface similarity is a syntactically convenient way of denoting the role of a component, but it cannot be the sole basis for structure sensitivity, because identical component representations can fill different structural roles. Suppose a subject can infer *Mary*, given the statement *John loves Mary*, and the question *John loves who*? Then, by the property of strong systematicity, one can also infer *dogs* given the statement *John loves dogs*, and the same question. Although it may be reasonable to assume nouns have same similar (internal) representation, it cannot be surface (external) similarity that is the basis or the following inference. Suppose a subject can infer *loves* from the statement *John loves Mary* and the question *John does what to Mary*? Again, by the property of systematicity, one can also infer *dogs* from the statement *John dogs*[5] *Mary* and the same question. However, in the second case, the same surface representation (or physical stimulus) is supposed to identify a different structural role (i.e. verb). Of course, the reason such inferences are possible is because of the surrounding context. Novel input may drive similar 'internal' representations from context, but they cannot be relied upon to possess inherently similar 'external' physical characteristics. The implication for connectionist modelling is that if similar input/output representations are the basis for demonstrations of strong systematicity, one must also explain how such representations arise.

The use of similar component representations also occurs in Chalmers (1990), where words of a common class (e.g. verbs, nouns) share a common feature in their vector representations, and in Niklasson and van Gelder (1994a), where component representations of objects belonging to a common category (e.g. proposition, conjunctive) are constructed with vectors representing these categories. These two models assume a similarity that connectionist networks are expected to learn. The point is that systematicity is a property at the level of compositional objects, not at the level of component objects. In language, systematicity is a property at the level of sentences, not at the level of words (Fodor & Pylyshyn, 1988), that is, the ability to represent and process sentences of a particular structure relates to other sentences conforming to the same structure. So as to control the effect of component similarity on demonstrations of systematicity, orthogonal (actually local[6]) representations are used to represent components, in subsequent simulations and analyses.

3. Feedforward Network

The tasks considered here and in the following section (for the simple recurrent network) depend on processing two-tuples (i.e. ordered pairs of atomic objects). Two-tuples are the simplest complex object for demonstrating generalization across position. Although humans deal with more complex objects, the simplicity of two-tuples makes them attractive for analysis. The tasks involve auto-association of two-tuples as a test of the capacity to represent an object (task 1), and querying of the first (second) component of a two-tuple as a test of inference capacity (task 2). Generalization (lack of generalization) across position for the auto-association (inference) task is referred to as 'strong' ('weak') systematicity of representation (inference). Hadley's (1994) definition also includes generalization to recursively embedded structures. The lower bounds approach taken here means that failure to demonstrate systematicity for this task implies failure on more tasks involving more complex structures.

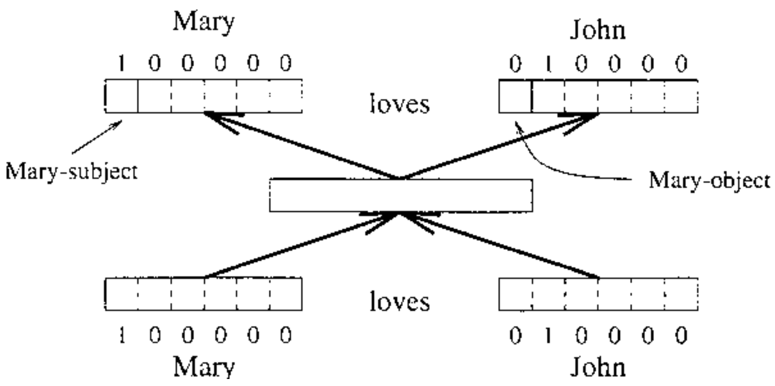

Figure 1. The feedforward network for auto-association of two-tuples.

3.1. Architecture

The feedforward network examined here consists of three layers: input, hidden and output, where there was one unit per component per position in both input and output layers (Figure 1).

3.2. Task 1: Auto-association of Two-tuples

To demonstrate systematicity of representation, the network must be capable of constructing representations of structurally related objects. In a connectionist framework, where representations are vectors in some activation space, construction of internal representations is simply allocating a vector in some internal (hidden-unit) activation space. However, to be a useful representation, representations of component objects must be subsequently accessible. Thus, auto-association tasks are useful in that they require networks to both construct and access representations of structured objects. One of the simplest structured objects in which to test this capacity are two-tuples (e.g. the binary relation: *loves*(*subject*, *object*)).

To demonstrate strong systematicity, the network must demonstrate correct processing of instances with components in positions that did not occur in the training set. In this task, for example, correctly auto-associating the two-tuple (*John*, *Mary*), having only experienced examples with *Mary* in the subject position.

3.3. Analysis: Weak Systematicity of Representation

Assuming a local representation (see Figure 1), each output unit is dedicated to detecting a particular component in a particular position from the network's hidden-unit (internal) representation of the complex object. For example, one output unit detects the presence or absence of *Mary* in the subject position. Correct performance of this task depends on orienting the *Mary–subject* hyperplane so as to position all points in the training set into two classes: *Mary* in the subject position and *Mary* not in the subject position. If there are enough training points, then the network will also correctly classify test cases.

However, for the output unit that detects *Mary* in the object position, there is only one class of points in the training set (i.e. *Mary* not in the object position), by the requirement of demonstrating strong systematicity. With only one class of points, the training set provides no information regarding the orientation of the hyperplane to classify correctly points with *Mary* in the object position, and so the network cannot be expected to generalize from components appearing in the subject position to components appearing in the object position. In this case, the network is said to be 'weakly' systematic, where generalization to novel combinations may be demonstrated, but not to novel positions (Hadley, 1993).

Of course, a fortuitous set of starting weights may permit the network to generalize to the new position; however, we also require the network to demonstrate generalization across position above chance level. Given the number of ways in which a hyperplane can be oriented, such fortuitous arrangements are very unlikely. Thus, it is concluded that the feedforward network does not demonstrate strong systematicity with respect to this task in the case of local input/output representations.

4. Simple Recurrent Network

4.1. *Architecture*

The lack of strong systematicity in the feedforward network is due to the independence of the weights accessing components from different positions. In the simple recurrent network (Elman, 1990), inputs and outputs are represented over the same set of units, and therefore transformed by the same set of weights. Thus, if the components of complex objects are presented temporally, rather than spatially, the simple recurrent network has the possibility of exhibiting strong systematicity, at least for the auto-association of a two-tuples task (Figure 2).

4.2. *Task 1: Auto-association of Two-tuples*

The task of the recurrent network is essentially the same as for the feedforward network described in Section 3 except that instead of presenting both components simultaneously, components are presented one per time step. After presenting both components, at which time the network should have constructed some internal representation of the ordered pair, the network is required to output the ordered pair in the order that it was presented. Figure 2 shows an example of the network and the input–output mapping it is required to perform.[7]

Each trial of the network consisted of the following:

- Random generation of training examples from a distribution where all five atomic objects can appear in the first position, but only four atomic objects can appear in the second position. Weights were updated at the end of each sequence (i.e. every four patterns). Context units were reset to zero at the beginning of each sequence.
- Random initialization of weights from a uniform distribution between -1 and 1.
- Training of the network using the standard error backpropagation algorithm with a 0.1 learning rate, no momentum term and the sum-of-squares error function (Rumelhart *et al.*, 1986) until performance on the training set was within 0.5 of the target for all output units for all patterns in the training set.
- Testing on all remaining combinations (i.e. every atomic object in the first position combined with the atomic object left out of the second position in the training set).

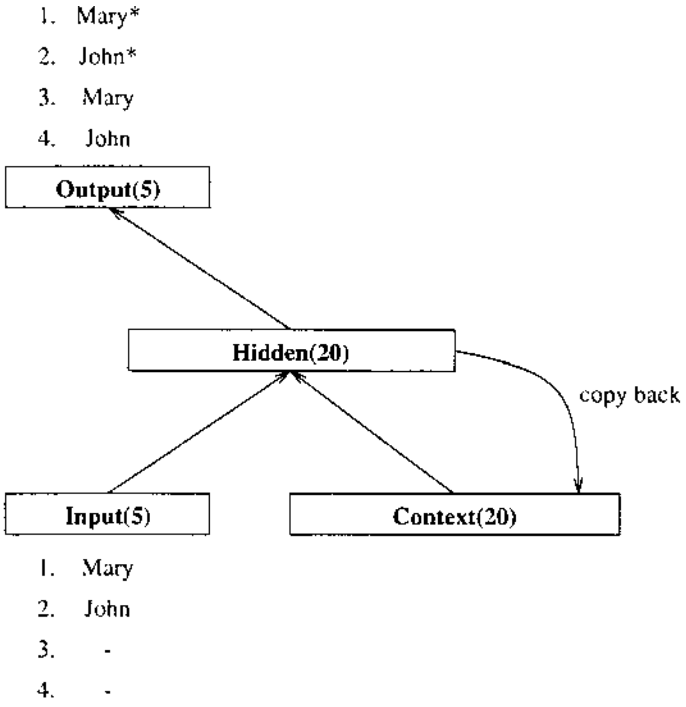

Figure 2. The simple recurrent network and the temporal version of the auto-association of a two-tuples task. Parenthesized values indicate number of units, dashes indicate zero input. Starred outputs were not considered for testing.

- Recording of network response to all output patterns in the test set. Two testing criteria were used. A network response was considered correct when (1) the maximally activated output unit had a target activation of one—maximum criterion or (2) all output units were within 0.5 of their target—0.5 criterion.

In addition, the number of training examples (ordered pairs) was varied from 10 to 200. Since each network was initialized from a random set of weights, each train-and-test trial was repeated five times for each training set size.

4.3. Result: Strong Systematicity of Representation

The recurrent network showed perfect performance (for both maximum and 0.5 testing criteria) on the unseen object in the second position in the test set when trained on 200 ordered pairs. When trained on 50 ordered pairs, performance dropped to a mean of 76% (maximum criterion) and 36% (0.5 criterion) over five trials (Figure 3).

4.4. Analysis

Due to the high dimensionality of the network's internal representational space (which is the number of hidden units) it is, in general, difficult to determine the nature of the internal representations constructed by a network to solve a particular task. However, examination of the relationship between hidden-unit activation

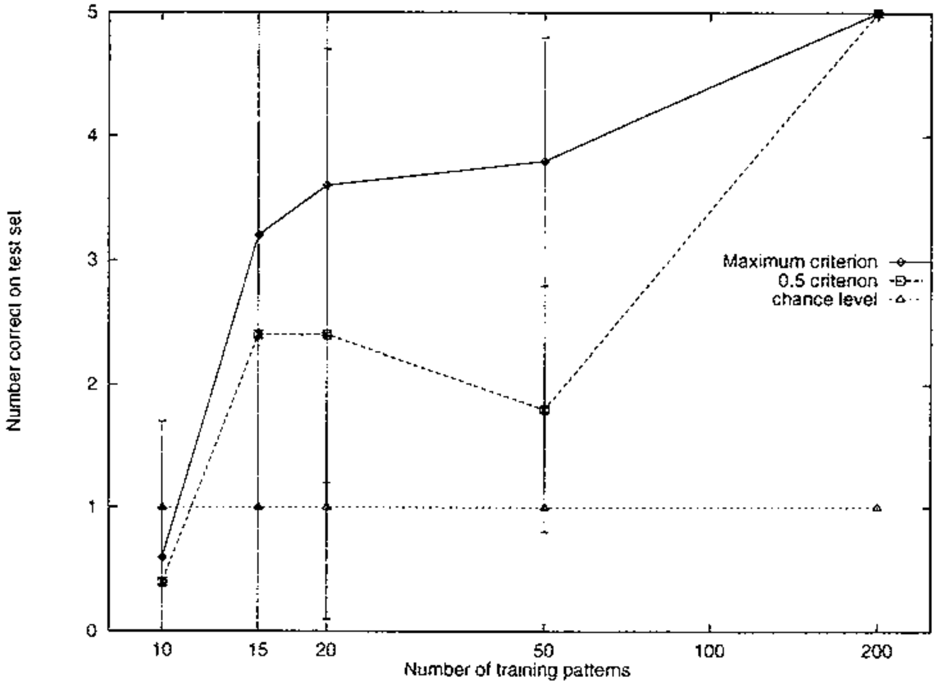

Figure 3. Generalization to second position as a function of the number of training patterns (log scale). The number of correct responses using maximum (solid line) and 0.5 (dashed line) test response criteria were averaged over five trials. Error bars indicate 95% confidence levels. The horizontal dotted line indicates chance level performance.

using principal components analysis (PCA) (Elman, 1989) and canonical discriminants analysis (CDA) (Wiles & Bloesch, 1992) can reveal the dimensions of greatest importance.

The internal representations learnt by the simple recurrent network were analyzed in the case where the network demonstrated generalization across position in all five trials (i.e. when the training set consisted of 200 training examples). After the network was trained to the 0.5 training criterion, all 200 training examples were presented to the network and the resulting 800 hidden-unit activation patterns were saved. Full details of this analysis are provided in Phillips (1995). Here, only two plots of the internal space are given as a guide to the nature of the internal representations suggested by these techniques.

At the second time step the network must construct a representation of each pair which maintains information regarding the first and second components of that pair. To help identify this information, a PCA of points at the second time step only was performed. A comparison of variance along hidden unit vs principal component dimensions (Figure 4) shows that the number of significant principal components was much less than the number of hidden units. The first five components account for 88% of the variance. Therefore, the activations of many of the hidden units were correlated, suggesting that multiple hidden units were coding for the same information.

The hidden-unit activations were then plotted on to the first and second

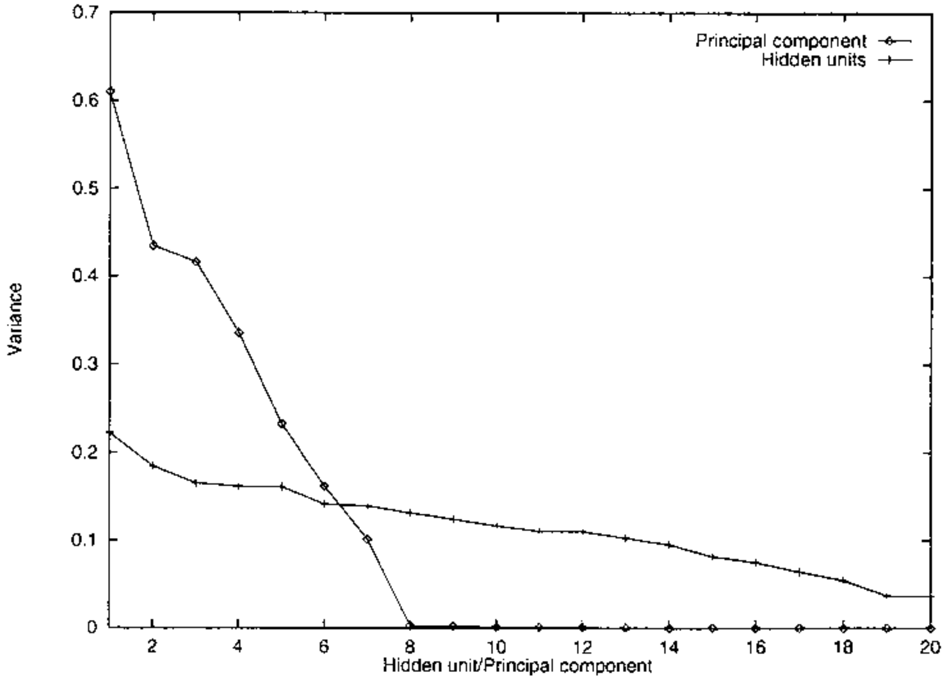

Figure 4. Variance in hidden-unit activation and principal components, ordered by magnitude.

principal components (Figure 5). The plot shows two levels of spatial organization. At one level, points are grouped into four clusters based on their output mapping.[8] At the other level, there appears to be the same within-cluster organization of points (labelled by their first object). This consistency is suggested by the same rotational ordering of points within each cluster.

The consistency of within-cluster organization suggests that first-object information is being encoded along other dimensions, independent (by virtue of the apparent regularity) of second-object information. If this suggestion is correct, then there may exist directions in hidden-unit activation space along which all four clusters can be superimposed. Or, in other words, there may be a projection from points in hidden-unit activation space such that, for example, the point representing the pair (*Karen*, *John*) and the point representing the pair (*Karen*, *Ann*) become the same.

If such a direction exists, it can be found using CDA by grouping all points on the basis of the first object. CDA will identify dimensions in hidden-unit activation space such that points in the same group (e.g. (*Karen*, *Ann*), (*Karen*, *Bill*), (*Karen*, *Karen*) and (*Karen*, *John*)) are close together, while points in different groups are far apart.

A CDA was performed on all hidden-unit activations at the second time step grouped by the object presented at the first time step. The points were then plotted on to the first and second canonical discriminants (Figure 6). The plot shows that the four clusters can be superimposed, and supports the suggestion that first-object information was encoded independent of second-object information. In each case, all points belonging to the same cluster differed in location by at most 10^{-6}.

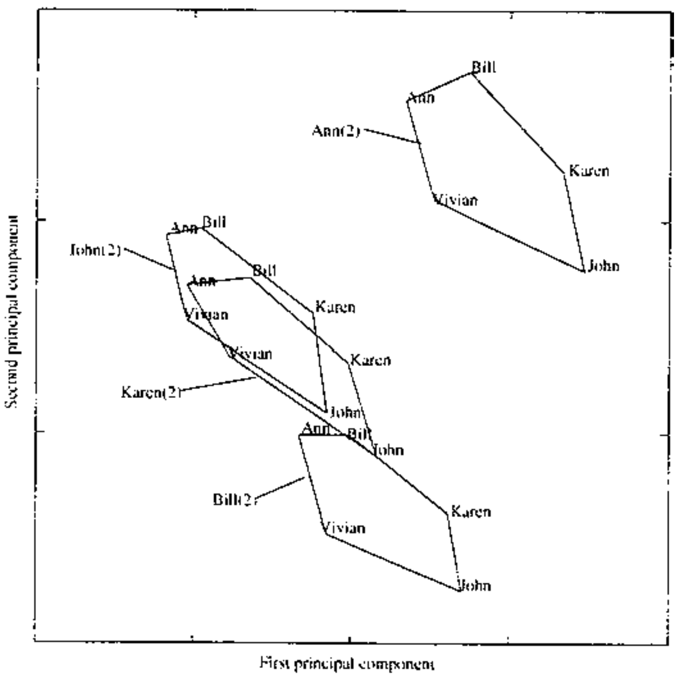

Figure 5. PCA of points in hidden-unit activation space generated from training sequences at the second time step.

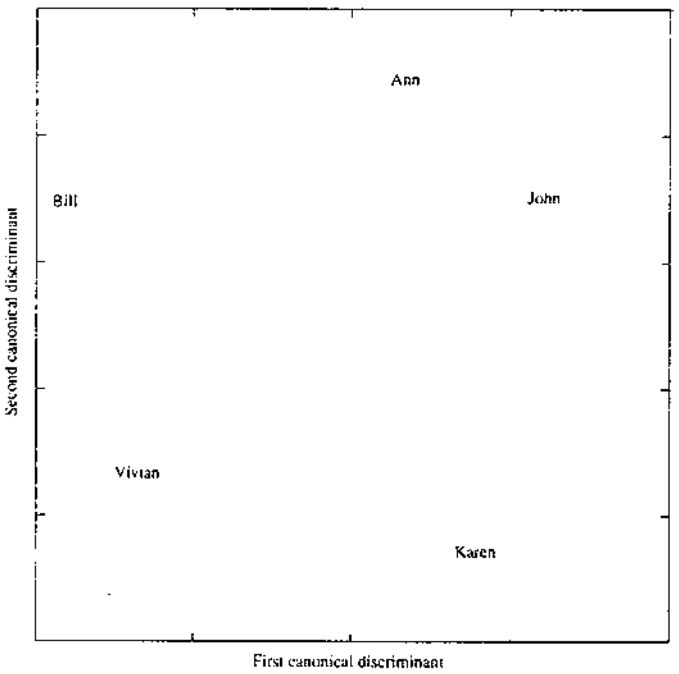

Figure 6. CDA of points in hidden-unit activation space generated from training sequences at the second time step grouped on the basis of the first input object.

A similar form of internal representation was found for the third and fourth time steps (Phillips, 1995), suggesting that the network also extracted components via the same set of dimensions. By encoding and decoding component representations via the same input and output sets of weights (respectively), the network only requires training on each unique object in one of the positions, but not necessarily both, as the simulations demonstrated.

4.5. Task 2: Querying of Two-tuples

In this task, the simple recurrent network is presented with an ordered pair and a query/question, which requests the first or second component of the ordered pair. For example, given the pair (*John*, *Bill*) and the question that requests the first component, the network should respond with *John*. As in the auto-association of a two-tuples task, there are five possible atomic objects that may appear in either the first or the second positions. In addition, there are two query objects that request either the first or second component of the ordered pair.

The simulation conditions for the querying of two-tuples task were as follows:

- Local encoding of input and output vectors. As there are seven possible input vectors in this task (i.e. five possible components plus two possible queries), there are seven input units to encode the input vectors and five output units to encode the five possible components.
- Generation of 40 training sequences (each sequence consisting of three patterns), consisting of all five components in the first position combined with four possible components in the second position combined with the two possible queries. The fifth component was used to test strong systematicity. Thus, the test set consisted of 10 sequences (i.e. all five components in the first position combined with the fifth component in the second position combined with the two queries).
- Random initialization of weights from a uniform distribution in the range -1 to 1.
- Training of the network using the standard error backpropagation algorithm with 0.1 learning rate, no momentum term and the sum-of-squares error function (Rumelhart *et al.*, 1986) until performance on the training set reached deferred criteria. Two training criteria were used: (1) all output units were within 0.5; (2) all outputs were within 0.4 of the target output for every unit on every training pattern. If the network did not meet these criteria by the 10 000th epoch (where one epoch is the presentation of every training pattern), then training was terminated. Weights were updated at the end of each sequence (i.e. every three patterns). Context units were reset to zero at the beginning of each sequence.
- Testing on all remaining tuples using two criteria for correctness: (1) the maximally activated output unit corresponds to the target activation of 1—maximum criterion; (2) all output units are with 0.5 of their target activation—0.5 criterion.
- Each train–test was repeated 10 times for each training criterion, with weights being randomly initialized at the beginning of each trial.

The simple recurrent network was examined with 20, 10 and eight hidden units. In the case of the eight-hidden-unit network, 0.5 was used as the training criterion.

Table I. Summary of the performance of the simple recurrent network on the systematicity of inference task for networks with 20, 10 and eight hidden units

Hidden units	20	10	8
Convergent trials (0.5/0.4)			
(10 trials)	10/10	10/7	1
Correct test cases (5)			
Train: 0.4; Test: (0.5/max)	0/0	0/0	—
Train: 0.5; Test: (0.5/max)	0/0	0/0	0/0 (8 trials) 0/1 (2 trials)

4.6. *Result: Weak Systematicity of Inference*

In the case of 20 hidden units, the simple recurrent network learnt perfectly on the training set for each of the 10 trials for both the 0.5 and 0.4 training criteria (i.e. all output units were within 0.4 of their target outputs for all patterns in the training set). However, in all trials, performance on the test set was zero for both the maximum and 0.5 testing criteria. The network did not correctly respond, in any trial, to any of the five sequences where the second component was the target.

In the case where 10 hidden units were used, training reached the 0.4 criterion in seven of the 10 trials. In all trials, regardless of whether the network reached the training criterion, the performance on the test set was zero, for both testing criteria, with regard to the extraction of the second component.

When only eight hidden units were used, only one of the 10 trials converged to the 0.5 training criterion. In all but two trials, performance was zero for the 0.5 and maximum testing criteria, on the extraction of the second component in the test set. For the other two trials, test set performance was one out of five for the maximum testing criterion and zero out of five for the 0.5 testing criterion. Since there are five possible components, this level of performance is no better than chance. Furthermore, in these two trials the network did not acquire perfect performance on the training set within 10 000 epochs. The results are summarized in Table I.

4.7. *Analysis*

In the querying of a two-tuples task, the network is presented with an ordered pair and a question requesting the first or second component of the pair. Now suppose *Mary* is the component on which the network is to demonstrate strong systematicity. Then, to solve the task, the network must implement the following function:

$$f(\mathbf{Q}_1, \mathbf{R}[(Mary, X)]) \rightarrow Mary \tag{1}$$

$$f(\mathbf{Q}_1, \mathbf{R}[(X, Mary)]) \rightarrow \neg Mary \tag{2}$$

$$f(\mathbf{Q}_2, \mathbf{R}[(Mary, X)]) \rightarrow \neg Mary \tag{3}$$

$$f(\mathbf{Q}_2, \mathbf{R}[(X, Mary)]) \rightarrow Mary \tag{4}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are the question vectors; $\mathbf{R}[(Mary, X)]$ is a representation of an ordered pair with *Mary* in the first position and some other object (X) in the second position; $\mathbf{R}[(X, Mary)]$ is a representation of an ordered pair with *Mary* in the second position and some other object (X) in the first position; and f is the function that implements the mapping. The $\mathbf{R}[.]$ notation is used to refer to the network's internal representation of an object which may be different from the object's external representation.

To discriminate correctly the *Mary* component, a hyperplane must be partitioned so as to satisfy the following inequalities:

$$\mathbf{W}_I\mathbf{Q}_1 + \mathbf{W}_C\mathbf{R}^{C}_{Mary,X} + \mathbf{B} > \theta \tag{5}$$

$$\mathbf{W}_I\mathbf{Q}_1 + \mathbf{W}_C\mathbf{R}^{C}_{X,Mary} + \mathbf{B} < \theta \tag{6}$$

$$\mathbf{W}_I\mathbf{Q}_2 + \mathbf{W}_C\mathbf{R}^{C}_{Mary,X} + \mathbf{B} < \theta \tag{7}$$

$$\mathbf{W}_I\mathbf{Q}_2 + \mathbf{W}_C\mathbf{R}^{C}_{X,Mary} + \mathbf{B} > \theta \tag{8}$$

where $\mathbf{W}_I$ and $\mathbf{W}_C$ are the input to, and context to, hidden weight vectors respectively; $\mathbf{B}$ is the bias (weight) to the unit and θ is the threshold above which the component *Mary* is considered to be the correct response. Subtracting equation (7) from equation (5), and equation (8) from equation (6) leaves

$$\mathbf{W}_I(\mathbf{Q}_1 - \mathbf{Q}_2) > 0 \tag{9}$$

$$\mathbf{W}_I(\mathbf{Q}_2 - \mathbf{Q}_1) > 0 \tag{10}$$

Since there does not exist a weight ($\mathbf{W}_I$) that satisfies both inequalities (9) and (10), the network cannot represent the solution with a single hyperplane for each component. Consequently, correct extraction of the *Mary* component requires at least two hyperplanes positioned in the input space (by two hidden units at the hidden layer) and a hyperplane positioned in the hidden space (by the output unit detecting the *Mary* component).[9] Correct orientation of hyperplanes at the hidden layer is depicted in Figure 7. Circles indicate *Mary* is the target output and squares indicate *Mary* is not the target output. Numerals indicate a request for the first or second component, X indicates some other component object and $\mathbf{C}$ is just the difference vector.

Assuming the two hidden-unit hyperplanes are in correct position, the resulting

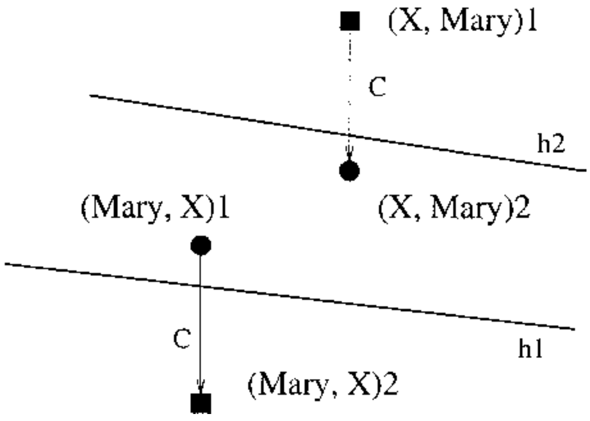

Figure 7. Orientation of the hidden-unit hyperplanes to extract the *Mary* component.

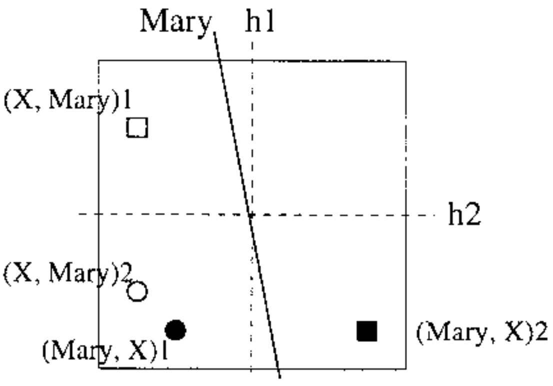

Figure 8. Orientation of the output-unit hyperplane in the hidden-unit activation space.

representation at the hidden layer for these four types of points, dependent on whether they are on the high or low side (relative to some threshold) of these two hyperplanes, is shown in Figure 8. In the hidden-unit activation space, these four point types must be partitioned by a single hyperplane into two groups: *Mary* (circles) and *not-Mary* (squares). Clearly, such a partitioning is possible. However, in addition, the network is required to demonstrate strong systematicity on the *Mary* component. Therefore, in the training set, one of the positions (e.g. *Mary* in the second position) does not occur (empty circle and square). Consequently, the training set does not provide any information regarding the partitioning of points along the vertical dimension and, therefore, the network cannot be expected to generalize to these cases.

The critical difference with this inference task is that, at the third time step, the network may be required to extract a component from either the first or second position. In the case of auto-association, the task only demanded that the first-position component be extracted at the third time step and the second-position component be extracted at the fourth time step. Therefore, in that case, components can be extracted via the same set of weights (but at different times). So in the inference case, two different sets of weights must be trained to perform the extraction. Hence, the network cannot demonstrate strong systematicity with respect to this task.

5. Discussion

The simulations and analyses of the previous sections demonstrated that the general architectures of and recurrent networks using local input/output representations do not have the necessary properties to support systematicity. Fodor and Pylyshyn (1988) argued that local internal representations cannot support systematicity. For the technical reasons presented here, this limitation also applies to distributed representations, given local input/outputs for these two networks. These results are general in that they do not depend on the size of the training set or the number of internal units and connections. The lack of generalization is because the connections that map component representations from different positions are learned independently. In one special case, auto-association of two-tuples by a recurrent network, some degree of generalization across position was demonstrated. The apparent solution was to utilize the same group of weights to retrieve each component in what resembled a first-in–first-out buffer. This solution did not suffice for task 2. Hence, the only positive example cannot be said to be sufficient support for systematicity.

One way of attempting to defer the implication of these results is to argue that human cognition is not as systematic as the classicists (e.g. Fodor and Pylyshyn) claim, but that human cognition is mostly context sensitive and so properties such as systematicity are only ideals at best. Therefore, (connectionist) models that are not systematic may still provide a reasonable approximation to human cognition. Van Gelder and Niklasson (1994) argued this position from experimental results on logical inferences. Subjects are significantly better at inference when thematic[10] rather than abstract examples are used. However, this lack of systematicity may only reflect familiarity within a domain. Other psychological experiments (Halford *et al.*, forthcoming) in other domains (schema induction) clearly demonstrated situations of perfect (symbol-like) generalization by all subjects.

In the experiments of Halford *et al.* (forthcoming), subjects are given a series of four tasks, where each task consists of learning a mapping from (*string*, *shape*) pairs to strings. In each task there are four (three-letter) strings and two shapes (e.g. *KEL*, *triangle* predicts *GOH*; *GOH*, *square* predicts *DUS*, etc.). Each string is paired with each shape to generate eight mappings. The strings and shapes are meaningless, and chosen randomly for each new task. The relations between strings and shapes within a task conform to a specific structure, and that structure is the same for each task, although each new task uses a new set of strings and shapes. The structure of each task instance can be visualized as a square, where each string lies on its own vertex and each shape is consistently either a horizontal or a vertical transition from one vertex to another, identifying the predicted string.

The significance of these experiments is that, by the fourth task, all subjects made perfect predictions (six out of eight) after seeing only two of the eight possible mappings. In fact, this degree of generalization goes beyond strong systematicity (generalization to novel position) in that one of the four strings does not appear anywhere in the first two mappings.[11] Thus, the implications of systematicity for a (connectionist) cognitive architecture cannot be so easily dismissed on the grounds that human cognition is, at best, quasi-systematic (Phillips & Halford, 1997). Although subjects are highly unlikely to have experience at this particular problem, knowledge of spatial analogues would greatly facilitate performance. However, that subject can call upon such knowledge acquired through the course of development does not alter the point. Rather, it reiterates the need to develop techniques that encode and make accessible such information. It is worth noting that one possible technique, weight sharing,[12] although sufficient to demonstrate some degree of learning transfer (see, for example, Hinton, 1990) does not support the same degree of transfer in these experiments (Phillips & Halford, forthcoming).

5.1. Other Connectionist Networks Lacking Strong Systematicity

The weak systematicity result of the simple recurrent network does not make any assumptions about the nature of the internal representations of the ordered pairs. The assumptions made were that the external representations of components are encoded locally, requiring an intermediate layer of first-order units between the

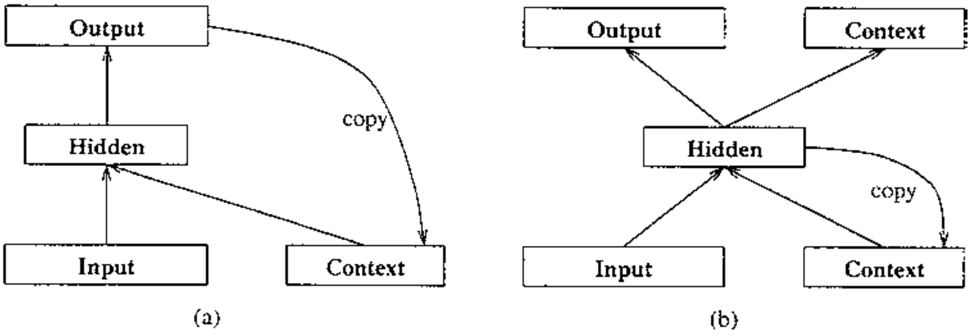

Figure 9. Two alternative recurrent network architectures: (a) Jordan's recurrent network and (b) Pollack's RAAM.

inputs and the outputs (which are also first-order units). Therefore, the result can also be applied to other architectures conforming to these restrictions.

5.1.1. Feedforward network. The context units in the simple recurrent network can be regarded as another group of input units holding a representation of the ordered pair. The two sets of input units map representations to hidden units which in turn map representations to output units. Since no specific assumptions are made regarding the nature of the order pair representation, three-layered feedforward networks are weakly systematic of inference.

5.1.2. Jordan's recurrent network. Jordan's (1990) recurrent network (see Figure 9(a)) differs from the simple recurrent network in that context is copied from the output-unit activations rather than the hidden-unit activations of the previous time step. Like the simple recurrent network, this network requires two hidden units to discriminate each component, but the training set provides no information about retrieving the component in the missing position. Therefore, this network is also weakly systematic of inference.

5.1.3. Pollack's recursive auto-associative memory. Pollack's (1990) recursive auto-associative memory (RAAM; see Figure 9(b)) is essentially the same as the simple recurrent network except there is an additional set of output units whose targets are the current context values (i.e. the network not only auto-associates the current input, but also the current context). However, these units (and their associated weights) play no part in the extraction of components. Their purpose is to encourage the formation of useful representations at the context layer. Since generalization is crucially dependent on training weights into the units representing the retrieved components, this architecture cannot demonstrate strong systematicity of inference on this task.

5.1.4. Forced simple recurrent network. The forced simple recurrent network (Maskara & Noetzel, 1992) is a variation of the RAAM in that the targets for the output units are the previous context, the current input and the next input. Additional information (hints) regarding the target function can improve learnability in networks (see, for example, Abu-Mostafa, 1990). However, as with the RAAM, the additional output units play no part in the extraction of components in the forced simple recurrent network. Therefore, the network cannot demonstrate strong systematicity of inference on this task.

5.1.5. Recurrent networks with higher-order connections. Some research on generalization has been conducted on networks of higher-order-weighted connections, where weights multiply, rather than add, the incoming activations (see, for example, Omlin & Giles, 1992). While none of this work has specifically addressed systematicity, one can make predictions by comparing the number of free parameters in higher-order networks to the first-order networks analyzed here. Since the number of higher-order connections between two layers of units is at least as great as the number of first-order connections (and, in general, many times more), the additional number of parameters make generalization across position even less likely for completely connected networks, given the same number of training patterns.

From the analysis of the simple recurrent network, at most a network of $2k$ hidden units is sufficient to represent the task, assuming k possible components in each position. With $2+k$ input units[13] and k output units, the total number of weights is $(2+k+2k)2k+2k^2$, which is $O(k^2)$. A second-order network does not require a hidden layer, since the decision boundaries are curved surfaces, not hyperplanes, in which case $2+k$ input and k context units are connected to k output units. The total number of second-order connections is $(2+k)(1+k)k/2$ (i.e. there are $(2+k)(1+k)$ unique pairs of units in the first layer connected to each of k units in the second layer), which is $O(k^3)$.[14] Since the number of free parameters is greater, generalization across position is less likely given the same training set. So, a fully connected higher-order network is unlikely to support strong systematicity.

5.1.6. Simple recurrent network on word prediction. Christiansen and Chater (1994) test for strong systematicity in the simple recurrent network on a word prediction task. They claim some limited support for strong systematicity in one of two experiments, but analysis here shows no support for strong systematicity.

A simple recurrent network was trained to predict words in sentences conforming to a particular phrase-structured grammar. In their second experiment, supposedly demonstrating strong systematicity, the network was given the sentence fragment *Mary says that John and boy* ..., at which point plural verbs and prepositions were correctly predicted. Since the word *boy* did not appear in the same position in the training set, the authors claim a demonstration of strong systematicity. Yet, in this case, the prediction of a plural verb or preposition does not required 'any' knowledge of the word *boy*. In fact, it only requires knowledge of 'some' noun. When the network received as input the word *and*, it correctly predicted that a singular or plural noun should follow. However, "... the activations for both 'boy' and 'boys' [were] *minimal* compared to the other nouns" (p. 283, emphasis added) suggests that they played no role in prediction. Since the hidden units contained information about nouns (e.g. *John*, *Mary*, but not necessarily *boy*), this information was copied to the context units for the next time step (i.e. at which time *boy* was presented to the input units). The information at the context units can be used to predict (conjointly[15]) the next word, which can be learned from many other examples in the training set that do not include *boy*. Christiansen and Chater remarked, following a suggestion by Hadley, that the network also achieved correct prediction when the novel word *boy* appeared on both sides of the conjunction. However, even this demonstration does not change the explanation, since the intervening word *and* always predicts a noun phase according to their grammar. At best, this experiment is ambiguous on the issue of strong systematicity. A better test is that prediction of novel words, which was examined in their first experiment.

In their first experiment, the network received the word *Mary's* from which is correctly predicted either a singular or plural noun to follow; but as Christiansen and Chater point out, none of the predicted nouns included the two novel nouns *girl* and *girls*. In light of these results, their second experiment does not provide any evidence for strong systematicity.

5.2. Two Networks Demonstrating Strong Systematicity

In Hadley (1994), the definition of strong systematicity was extended to include recursively embedded structures. Relative to this definition, a model is said to be strongly systematic if, for example, having only been trained on instances where *Mary* appears in the subject position, it generalizes to instances such as *John loves Mary* and *Tom knows that John loves Mary*. Two more complex networks satisfying these definitions are: the 'transformation network' of Niklasson and van Gelder (1994b);[16] and the 'associative semantic network' of Hadley and Hayward (1995).[17] The definition of systematicity was again revised to include the capacity to assign appropriate meanings to words in test sentences (Hadley & Hayward, 1995). This degree of systematicity, called strong semantic systematicity, is exhibited by the associative semantic network.

5.2.1. Transformation network. The transformation network (Niklasson & van Gelder, 1994b) consists of two RAAM networks (Pollack, 1990) and a feedforward network. The task of the transformation network was to learn to make transformations ($\Leftrightarrow$) between logical expressions (e.g. $p \rightarrow q \Leftrightarrow \neg p \vee q$). One RAAM was trained to encode a vector representation of logical expressions from their atomic constituents (e.g. p, q and $\rightarrow$), and the other RAAM was trained to decode representations of expressions back to their constituents. The feedforward network was trained to map between those representations for the logical expressions. The network demonstrated strong systematicity: having been trained on expressions containing propositions p, q and r, the network generalized to expressions containing s.

A plot of hidden-unit activations from a trained network (Niklasson & van Gelder, 1994b, Figure 4) showed a similar form of two-level clustering as was found with the simple recurrent network. The first graph clearly shows clusters based on the second proposition, and the same within-cluster positioning based on the first proposition. This organization suggests that the component objects were decoded from within their complex representations along the same set of dimensions (weights). Interestingly, the transformation task is a variation of the auto-association task (task 1) on which the simple recurrent network was examined. That is, the propositions were encoded and retrieved in the same first-in–first-out manner, except with the addition of intervening connectives (e.g. $\rightarrow$, $\vee$, etc.). So, in light of the analysis conducted here on the simple recurrent network, such a solution makes sense.

The network also demonstrated generalization when the novel constituent s did not appear anywhere in the training set. At first, this result seems surprising given that the weights from the input unit representing s are never updated during training (since its activation was always zero for all other patterns). However, an examination of the encoding of constituents suggests how such generalization was achieved. All propositions have one unit in their input vector set to 1, to distinguish them from all other symbols. Thus, even the novel constituent *s* received training to distinguish them from the connectives. In addition, the use of the minimum Euclidean distance to determine the correct constituent works well with locally encoded constituents. Regardless of the input to hidden weights, no component of the novel proposition *s* will be projected in the direction of other propositions, since the activations to the inputs representing those constituents will be zero (under a local representation). Thus, the closest point is likely to be the novel constituent. An interesting extension to these simulations is an inference task, like task 2, where the network is required to extract a specific value for a specific constituent within a complex representation. In light of the results on the simple recurrent network, I expect that the transformation network would not demonstrate strong systematicity on such tasks.

5.2.2. Semantic network. The associative semantic network (Hadley & Hayward, 1995) is quite different in structure and functionality to any of the previously discussed networks. There is not sufficient space for a full explanation of this network. However, one observation is made that suggests how this network achieves generalization across position.

The task consists of representing the semantic content of a given sentence. Semantic concepts such as principal proposition, agent, action, patient and individual words are represented as individual nodes. The network learns by association between input nodes (representing words) and their target semantic representations. The point to note about this architecture, with respect to generalization across position, is that there are no direct connections between the input nodes and the position/role nodes (e.g. agent, patient). Information between the input and position nodes is mitigated by a single proposition node. Therefore, weight vectors from the input layer to each position node are effectively related by a scalar factor. Linking input to position nodes via a single intermediate node introduces a weight dependency across position. It is suggested that this property facilitates generalization across position for unembedded sentences in the associative semantic network.

5.3. *Implications for a Connectionist Cognitive Architecture*

The main implication of these results is that the architecture of higher cognition is not that of completely connected networks coupled to a learning function, as exemplified in the feedforward and recurrent networks. Such networks permit states that have no counterpart in higher cognition (specifically, states of weak systematicity).

Yet, the negative results do not imply that all of connectionism is unsuitable for cognitive modelling.[18] What it does tell us, to echo Fodor and McLaughlin (1990), is that there must be additional principles above those of units and weights that organize network connectivity in such a way as to enforce systematicity. One possibility is tensor networks (Smolensky, 1990),[19] whose connectivity is arranged in such a way as to implement the inner ($\odot$) and outer ($\otimes$) product operators.

Tensor networks were introduced by Smolensky (1990) as a connectionist method for representing symbol structures. An important property of tensors is that construction and recovery of component representations occurs over the same set of units (via the inner product operator), independent of the component's value and position within the complex object. Hence, learning to extract a component from one position automatically extends to all other positions. The following example illustrates why.

Suppose *John loves Mary* is an instance of the binary relation *loves*. Following Smolensky (1990), these relational instances can be represented by the tensors $T_{JM} = J \otimes r_1 + M \otimes r_2$, where J and M are vector representations for components *John* and *Mary* and r_1 and r_2 are their respective roles. Now, assuming that r_1 and r_2 are orthonormal, the person being loved is determined by $T_{JM} \odot r_2 = M$. This case automatically extends to the related instance *Mary loves John*, by the properties of inner and outer products $T_{MJ} \odot r_2 = (M \otimes r_1 + J \otimes r_2) \odot r_2 = J$. In fact, it extends to all instances of the relation, so that $T_{XY} \odot r_1 = X$ and $T_{YX} \odot r_1 = Y$, and similarly for the second component accessed by role r_2.

Although tensors are a representation scheme, they can be included in a larger learning network where appropriate role vectors are learned to demonstrate generalization across position (Phillips, 1994, 1995). Whether such a network can address all of Hadley's strong systematicity criteria (including generalization to novel levels of embedding) is the subject of further study. The important point is that additional principles must be built into these connectionist networks so as to demonstrate systematicity.

5.3.1. Component similarity revisited. At the beginning of the paper, I argued that one cannot rely on a priori similarity between component input/output representations as the basis for systematicity. However, this restriction does not preclude a network from constructing similar 'internal' component representations in previous learning as the basis for systematicity in subsequent tasks.

Suppose, through prior learning (e.g. categorization), previously dissimilar input stimuli are assigned similar internal representations. In an extreme case, all components in the two-tuples task could be evenly distributed along a single internal unit dimension (e.g. *Ann*—0.1; *Bill*—0.3; *John*—0.5; etc.). This new internal unit could serve as an input to a subnetwork for the purpose of the auto-association task. In this case, the auto-association task only requires learning a two-element vector mapping: auto-association of (*John*, *Bill*) is the mapping $(0.5, 0.3) \rightarrow (0.5, 0.3)$. Since there is a linear relationship between the new similarity-based input and output representations, only three two-tuples are necessary for generalization to the remaining pairs in a 2–2 feedforward network (i.e. two input units completely connected to two output units). At least two of the five components would not have appeared in both positions in the training set. Even though all components must have appeared in some training condition for a previous (categorization) task, with respect to auto-association the network can be said to have demonstrated strong sytematicity.

Clearly, choice of component representation has a significant impact on generalization, but permitting arbitrary input/output representations potentially trivializes the issue. As the previous example demonstrated, generalization may be ensured by making novel components sufficiently 'close' to trained components, independent of whatever structure they belong to. If one insists on component similarity as the basis for generalization, then the crucial question becomes 'What justifies a particular choice of component representation?'

6. Conclusions

In hindsight, it may see obvious that there should be no generalization given that the weights that extract components from different positions are trained independently, but this observation does not come directly from inspection of network connectivity alone. It comes from an analysis of connectivity in conjunction with learning and task requirements: for example, while the simple recurrent network exhibited generalization across position in task 1, there was no generalization for the same network in task 2.

Feedforward and recurrent networks are not, in themselves, systematic. In the case of local component representation, the two networks require additional properties (constraints) to enforce generalization across position. I have focused on one property: dependency between the weights associated with the representation of components in different positions. Dependency is enforced by using the same group of weights for each position, as in the tensor and semantic networks. It may also be enforced by the learning method, as appeared to be the case for the transformation network, although further analysis is needed in this area.

A cognitive architecture based on a network of units coupled with a learning algorithm for configuring the pattern of connectivity is attractive. It makes fewer commitments to the design of specific mechanisms that realize cognitive behaviours, and those commitments can be grounded, partly, by observation (learning) of external environmental events. Nevertheless, if one accepts the requirements of systematicity, then those requirements are not met by just this type of architecture. Either additional properties are necessary to explain why networks are configured in a particular way so as to exhibit systematicity or additional subnetworks are required to preprocess potential components into similarity-based representations, for which it may be possible to demonstrate strong systematicity. Either way, the standard approach will not suffice.

In the 1980s, computer science saw a revolution in software design from the so-called procedural (third-generation) languages such as C to relational (fourth-generation) languages of which SQL is the standard for the purposes of modelling information systems. Analogously, answers to the 'why' question with respect to structure-related properties such as systematicity should refocus attention towards other approaches in connectionist modelling of higher cognition.

Acknowledgements

I thank Janet Wiles and Simon Dennis for many helpful discussions on an earlier version of this paper; Jeff Elman, Mike Humphreys and Michael Johnson for their critical comments; members of the Information Science Division for discussion on analysis methods; and two anonymous reviewers for their critical comments and suggested improvements. This work began at the Department of Computer Science, The University of Queensland, with the support of a UQPRA scholarship, and was completed at the Information Science Division of the Electrotechnical Laboratory, with support from an STA fellowship.

Notes

1. The angle at which electrons are scattered is not random, but quantized and dependent on the structure of the material at which they were fired.

2. The other half is that connectionist demonstrations of systematicity implement classical architectures. This paper is only concerned with identifying connectionist properties that do (do not) enforce systematicity. See Fodor (1997) and Smolensky (1995b) for continued and apposing arguments on the implementation issue.
3. Subsequently, Hadley's (1994) definition of strong systematicity also includes correct performance on components in novel positions at novel levels of embedding (e.g. *Mary knew Tom hit John*). Although the simulations and analyses presented here focus on the first definition, the negative results clearly apply to the extended definition. See also Niklasson and van Gelder (1994a) for other generalization criteria for systematicity.
4. See also Hadley (1994) for a similar comment on McClelland and Kawamoto (1986).
5. *Dogs* here means chases relentlessly.
6. One vector component ON, the rest OFF (e.g. 01000).
7. The simple recurrent network in Elman (1990) was trained to perform word prediction from a set of sentences generated from a grammar.
8. There are only four clusters, since one object (*Vivian*) was left out of the second position in order to test strong systematicity.
9. This analysis essentially parallels the explanation as to why perceptrons cannot solve the 'exclusive-or' task (Hertz *et al.*, 1991; Minsky & Papert, 1990).
10. For example: *If it is raining then there are clouds. There are no clouds. Therefore, it is not raining.*, which is an instance of the inference *modus tollens*: $P \rightarrow Q \wedge \neg Q \simeq \neg P$.
11. To see why, suppose strings are labelled A, B, C and D; and shapes *triangle* and *square*. In the first two presentations, subjects see that (A, *triangle*) predicts B; and (B, *square*) predicts C. At the third presentation, given (D, *triangle*) subjects correctly predict C, yet D did not appear anywhere in the previous mappings, nor in any of the previous tasks.
12. Weight sharing means weights shared by several subnetworks to encode knowledge common to the various tasks to which each subnetwork is dedicated.
13. The additional two units represent the question inputs.
14. This means complexity is no better than k^3, whereas $O(k^2)$ is no worse than k^2.
15. This may explain why prediction of plural verbs was much less in the test case (see Figure 6(g), p. 283, in Christiansen and Chater (1994)).
16. This network actually satisfied a more stringent criterion in demonstrating generalization to components that did not appear at all in the training set.
17. This network also satisfies strong semantic systematicity (Hadley & Hayward, 1995) which requires identifying the roles (e.g. agent, patient, etc.) that components play in novel positions at novel levels of embedding. This definition is not considered here.
18. There may even be domains where humans are weakly systematic.
19. Fodor and McLaughlin (1990) and Fodor (1997) rejected this solution on the implementation issue, not the issue of realizing systematicity.

References

Abu-Mostafa, Y.S. (1990) Learning from hints in neural networks. *Journal of Complexity*, **6**, 192–198.

Chalmers, D.J. (1990) Syntactic transformations on distributed representations. *Connection Science*, **2**, 53–62.

Christiansen, M.H. & Chater, N. (1994) Generalization and connectionist language learning. *Mind and Language*, **9**, 273–287.

Elman, J.L. (1989) *Representation and Structure in Connectionist Models*. Technical Report 8903, University of California, San Diego, CA.

Elman, J.L. (1990) Finding structure in time. *Cognitive Science*, **14**, 179–211.

Elman, J.L. (1991) *Incremental Learning, or the Importance of Starting Small*. Technical Report 9101, University of California, San Diego, CA.

Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D. & Plunkett, K. (1996) *Rethinking Innateness: A Connectionist Perspective on Development. Neural Network Modeling and Connectionism*. Cambridge, MA: MIT.

Fodor, J.A. (1997) Connectionism and the problem of systematicity (continued): why Smolensky's solution still doesn't work. *Cognition*, **63**, 109–119.

Fodor, J.A. & McLaughlin, B.P. (1990) Connectionism and the problem of systematicity: why Smolensky's solution doesn't work. *Cognition*, **35**, 183–204.

Fodor, J.A. & Pylyshyn, Z.W. (1988) Connectionism and cognitive architecture: a critical analysis. *Cognition*, **28**, 3–71.

Hadley, R.F. (1993) *Compositionality and Systematicity in Connectionist Language Learning*. Technical Report CSS-IS TR 93-01, Simon Fraser University, Burnaby, BC.

Hadley, R.F. (1994) Systematicity in connectionist language learning. *Mind and Language*, **9**, 247–272.

Hadley, R.F. & Hayward, M. (1995) Strong semantic systematicity from unsupervised connectionist learning. In J.D. Moore & J.F. Lehman (Eds), *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pp. 358–363. Hillsdale, NJ: Lawrence Erlbaum.

Halford, G.S., Bain, J.D., Maybery, M.T. & Andrews, G. (forthcoming) Induction of relational schemas: common processes in reasoning and complex learning. *Cognitive Psychology.*

Hertz, J., Krogh, A. & Palmer, R.G. (1991) *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison Wesley.

Hinton, G.E. (1990) Mapping part-whole hierarchies in connectionist networks. *Artificial Intelligence*, **46**, 47–76.

Jordan, M.I. (1990) *Serial Order: A Parallel Distributed Processing Approach*. Technical Report, MIT, Cambridge, MA.

Maskara, A. & Noetzel, A. (1992) *Forcing Simple Recurrent Networks to Encode Context*. Technical Report, New Jersey Institute of Technology, NJ. Also in *Proceedings of the 1992 Long Island Conference on Artificial Intelligence and Computer Graphics*.

McClelland, J.L. & Kawamoto, A.H. (1986) *Computational Models of Cognition and Perception, Vol. 2, Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences*, Chapter 19. Cambridge, MA: MIT.

Minsky, M.L. & Papert, S.A. (1990) *Perceptrons*, 4th Edn. Cambridge, MA: MIT.

Niklasson, L.F. (1993) Structure sensitivity in connectionist models. *Proceedings of the 1993 Connectionist Summer School*, pp. 162–169. Hillsdale, NJ: Lawrence Erlbaum.

Niklasson, L.F. & van Gelder, T. (1994a) Can connectionist models exhibit non-classical structure sensitivity? In A. Ram & K. Eiselt (Eds), *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pp. 644–669. Hillsdale, NJ: Lawrence Erlbaum.

Niklasson, L.F. & van Gelder, T. (1994b) Systematicity and connectionist language learning. *Mind and Language*, **9**, 28–302.

Omlin, C.W. & Giles, C.L. (1992) Training second-order recurrent neural networks using hints. In D. Sleeman & P. Edwards (Eds), *Machine Learning: Proceedings of the 9th International Conference*. San Mateo, CA: Morgan Kaufmann.

Phillips, S. (1994) Strong systematicity within a connectionist framework: the tensor-recurrent network. In A. Ram & K. Eiselt (Eds), *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pp. 723–727. Hillsdale, NJ: Lawrence Erlbaum.

Phillips, S. (1995) *Connectionism and the Problem of Systematicity*. PhD thesis, Department of Computer Science, University of Queensland, Brisbane, Australia.

Phillips, S. & Halford, G.S. (1997) Systematicity: psychological evidence with connectionist implications. In M.G. Shafto & P. Langley (Eds), *Proceedings of the 19th Annual Conference of the Cognitive Science Society*, pp. 614–619. Hillsdale, NJ: Lawrence Erlbaum.

Phillips, S. & Halford, G.S. (forthcoming) An analysis of learning transfer in neural networks with relevance to connectionism.

Plunkett, K. & Elman, J.L. (1997) *Exercises in Rethinking Innateness: A Handbook for Connectionist Simulations. Neural Network Modeling and Connectionism*. Cambridge, MA: MIT.

Pollack, J.B. (1990) Recursive distributed representations. *Artificial Intelligence*, **46**, 77–105.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986) *Computational Models of Cognition and Perception, Vol. 1, Learning Internal Representations by Error Propagation*, Chapter 8. Cambridge, MA: MIT.

Smolensky, P. (1988) On the proper treatment of connectionism. *Behavioral and Brain Sciences*, **11**, 1–74.

Smolensky, P. (1990) Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, **46**, 159–216.

Smolensky, P. (1995a) Reply: Constituent structure and explanation in an integrated connectionist/symbolist cognitive architecture. In C. Macdonald & G. Macdonald (Eds), *Connectionism: Debates on Psychological Explanation*, Vol. 2, Chapter 6, pp. 223–290. Cambridge, MA: Blackwell.

Smolensky, P. (1995b) Connectionism, constituency and the language of thought. In C. Macdonald & G. Macdonald (Eds), *Connectionism: Debates on Psychological Explanation*, Vol. 2, Chapter 4, pp. 164–198. Cambridge, MA: Blackwell.

St John, M.F. & McClelland, J.L. (1990) Learning and applying contextual constraints in sentence comprehension. *Artificial Intelligence*, **46**, 217–257.

van Gelder, T. & Niklasson, L. (1994) Classicism and cognitive architecture. In A. Ram & K. Eiselt (Eds), *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, pp. 905–909. Hillsdale, NJ: Lawrence Erlbaum.

Wiles, J. & Bloesch, A. (1992) Operators and curried functions: Training and analysis of simple recurrent networks. In S.J. Hanson & R.P. Lippman (Eds), *Advances in Neural Information Processing Systems 4*. San Mateo, CA: Morgan Kaufmann.