Learning the Structure of a Mathematical Group

Anna Jamrozik (anna.jamrozik@mail.mcgill.ca)
Department of Psychology, McGill University, 1205 Penfield Avenue
Montreal, QC H3A 1B1 Canada

Thomas R. Shultz (thomas.shultz@mcgill.ca)
Department of Psychology and School of Computer Science, McGill University, 1205 Penfield Avenue
Montreal, QC H3A 1B1 Canada

Abstract

A mathematical group is a set of operations that satisfies the properties of identity, inverse, associativity, and closure. Understanding of group structure can be assessed by changing elements and operations over different versions of the same underlying group. Participants learned this structure more quickly over four different, successive versions of a subset of the Klein 4-group, suggesting some understanding of the group structure (Halford, Bain, Mayberry, & Andrews, 1998). Because an artificial neural network learning the task failed to improve, it was argued that such models are incapable of learning abstract group structures (Phillips & Halford, 1997). Here we show that an improved neural model that adheres more closely to the task used with humans does speed its learning over changing versions of the task, showing that neural networks are capable of learning and generalizing abstract structure.

Keywords: generalization; mathematical groups; Klein 4-group; systematicity; neural networks; sibling-descendant cascade-correlation.

Introduction

Generalization allows application of what was learned in one task to new tasks, thus distinguishing understanding from mere memorization (Shultz, 2001). Phillips and Halford (1997) questioned the ability of current feedforward neural networks to learn abstract structures and to generalize to related problems. Mathematical groups offer a challenging context to test for such generalization. Generalization is achieved if the structure underlying a group can be extracted and applied to new instances.

Human participants demonstrated some ability to extrapolate the structure underlying a series of four different versions of what was described as a Klein 4-group task (Halford et al., 1998). Attempts to create a feedforward network able to do the same were unsuccessful (Phillips & Halford, 1997). The modeling failure was cited approvingly by Marcus (1998) as part of a general argument against the ability of neural networks to generalize outside of the training set as well as humans do. Here, we identify some serious shortcomings of that unsuccessful model and present a more appropriate neural-network model that exhibits generalization by learning the Klein 4-group structure more quickly with each new version of the task, as Halford et al.'s participants did.

The Klein 4-group

A mathematical group is a set of operations that satisfies four criteria: identity, inverse, associativity, and closure. The Klein 4-group, named after German mathematician Felix Klein, is one of only two possible four-element groups (the other is the cyclic 4-group). The four elements of the Klein 4-group are operations that, when combined, create other operations that are also members of the group. Table 1 shows how the Klein 4-group might be used to structure a task like that used by Halford et al. (1998), which can be visualized as moving from one position to another in a two-dimensional plane.

Table 1: Example of the Klein 4-group.

*	**Identity**	**Horizontal**	**Vertical**	**Diagonal**
Identity	Identity	Horizontal	Vertical	Diagonal
Horizontal	Horizontal	Identity	Diagonal	Vertical
Vertical	Vertical	Diagonal	Identity	Horizontal
Diagonal	Diagonal	Vertical	Horizontal	Identity

The identity criterion is satisfied by an operation that does not change the operation it acts on. Notice that the first row and first column in Table 1 each use an identity operation and thus preserve the effect of the other operation. For example, a horizontal operation followed by an identity operation is simply a horizontal operation. The inverse criterion is satisfied when every operation has an inverse that reverses the action of that operation. Inverse operations are designated in the diagonal of Table 1, marked by cells labeled *identity*. For example, a horizontal operation can be undone by a reverse horizontal operation, leaving the system in the same place it started.

The associative law of operations specifies that two operations applied in a certain order result in the same final state even if their order is reversed. For example, a horizontal operation followed by a vertical operation yields the same result as does a vertical operation followed by a horizontal operation. Finally, closure states that the result of any combination of operations can also be produced by a different, single operation. For example, combining a vertical operation with a horizontal operation is equivalent to using a diagonal operation.

The Klein 4-group, and foregoing examples, can be visualized as in Figure 1, with four states at the corners and the six edges representing the three non-identity operations.

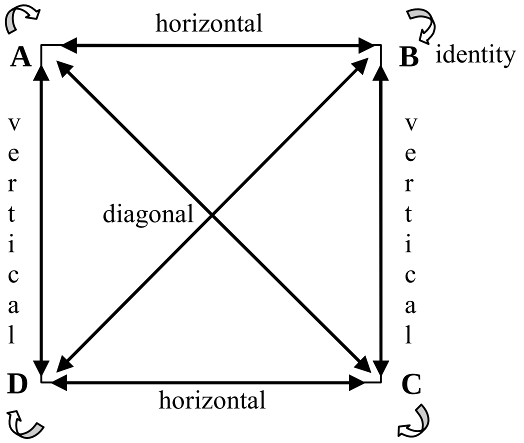

Figure 1: A graphical representation of the Klein 4-group**.**

Human Extrapolation of Group Structure

In a series of experiments, Halford et al. (1998) demonstrated people's ability to extrapolate an abstract structure underlying a series of four tasks involving a modified and simplified subset of the Klein 4-group. Curiously, the diagonal and identity operations, present in the full Klein 4-group, were missing from this subset. The Halford et al. subset is pictured in Figure 2. This structure allows for eight possible triads, or element-operation-element sequences (e.g., A horizontal D). Three-letter pronounceable but meaningless strings of letters (e.g., 'MIW') were substituted for each of the four elements in the group and simple shapes (e.g., Δ) were substituted for each of the two operations. An example of a triad could be 'MIW Δ VOL'.

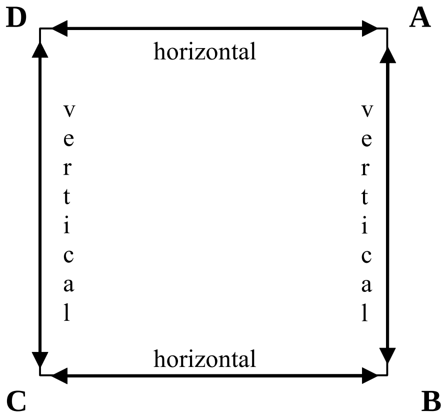

Figure 2: Halford et al.'s (1998) version of the Klein 4-group.

Participants completed four versions of the task, each one containing a maximum of six repetitions of the eight possible triads. All four versions of the task shared the same underlying structure but differed in the shapes and strings used in the triads. Each triad was composed of an initial pair formed of one element and operation. Participants were asked to predict the element resulting from the transformation by selecting from a list of four possible final elements. Following each identification attempt, participants were given feedback regarding the correct answer. After identifying every pair correctly within a trial or after a maximum of six trials, participants began the next version of the task, with a different set of shapes and strings. By the beginning of the fourth version of the task, participants' error had decreased, suggesting that participants had extracted something about the abstract relationship between the elements and operations.

Previous Klein 4-group Network Models

Phillips and Halford (1997) argued that neither a simple recurrent nor feedforward neural-network model could capture the same degree of generalization found in humans. They claimed that this was because the Klein 4-group problem exhibited systematicity, which they defined as the "property whereby cognitive capacities are grouped on the basis of common structure" (Phillips & Halford, 1997, p. 614). The basic idea is that the same structure underlies all four task versions and this structure can be learned and used to speed the learning of each successive version. They argued that connectionist networks are context-specific, not structure-specific. Because the context of the problem changes from task to task, they expected that networks would be unable to improve their performance across different versions of the Klein 4-group task.

Phillips and Halford (1997) created a fixed-architecture feedforward network to test this hypothesis: 6 input units leading to a layer of 3 hidden units, these leading to 2 hidden units, and finally to 4 output units. A localist coding scheme was used for both inputs and outputs. This is a binary scheme in which an input or output pattern contains one non-zero bit. For example, the four group elements could be represented as '1000', '0100', '0010', and '0001'. After the network successfully predicted each pattern in the first task, then one, two, or all three of the weights stemming from one input unit were reset, the network was retrained on all other patterns, and then tested on the pattern corresponding to the reset input unit. This was repeated for 10 trials for each weight reset (1, 2, or 3). With one weight reset, the network was able to generalize to 7 of the 10 test trials. With three reset weights, the network was unable to generalize to the test input patterns.

Shortcomings of Previous Models

Phillips and Halford (1997) did not test network generalization ability in the same way that humans were tested. Humans were tested for improvement in learning speed over four tasks all sharing an underlying subset of the

Klein 4-group structure. Phillips and Halford concluded that participants were generalizing the structure of the group and using it to solve later tasks because they became better at learning to predict the correct final elements by the fourth task. Their networks were tested in a different fashion. After resetting the weights between one input unit and the hidden units, the network's ability to find the solution to the element represented by that input unit was tested. No new versions of the task were used. The resetting of weights could be seen as erasing memory. The human experiment involved learning new tasks by generalizing the structure of earlier tasks, not as learning one task, having this knowledge erased, and trying to learn a similar task.

Using a local input/output representation is known to hamper generalization, because different elements are represented on different units and weights. By using a coding scheme known to limit generalization, feedforward networks' ability to generalize the Klein 4-group structure was compromised. In order to more fairly assess this ability, a coding scheme that facilitates generalization should be used. Analog representation facilitates generalization by representing different elements on the same units and weights and has allowed for generalization and other psychological effects to emerge naturally within neural networks (Shultz, 2003).

The feedforward network used in Phillips and Halford's (1997) simulation was designed by hand, such that the number of hidden units and their topology was fully determined before training began. This prevented the network from growing to meet the demands of the task. People were under no such restrictions. With the alternative of constructive learning, a learning algorithm can build its own network topology to suit the problem (Shultz, 2006). The topology can start with minimal structure and new hidden units can be recruited as needed and placed in a network topology that learns to minimize network error.

Finally, if the goal is to test human and network ability to learn the abstract structure of the Klein-4-group, then the full Klein 4-group should be used, not an arbitrary subset that is not technically a group.

SDCC

In order to allow the networks in our simulation to construct their own topologies, the Sibling-descendant Cascade-correlation (SDCC) algorithm (Baluja & Fahlman, 1994) was employed. As in ordinary cascade-correlation (CC), SDCC allows networks to recruit hidden units as they are needed in order to minimize error. But rather than installing each hidden unit on its own layer as in CC, a new unit is either installed on a new layer (descendant) or in the current highest layer (sibling), depending on which candidate's activation correlates best with network error. Networks created with SDCC are shallower than CC networks but have a similar ability to learn and generalize (Baluja & Fahlman, 1994). CC and more recently SDCC have been used to successfully simulate many phenomena in human learning and development (Shultz, 2003, 2006).

Method

Coding the Task

The task used in Phillips and Halford's (1997) simulation was modified to more fairly test network learning ability and to better correspond with the task used in the human experiments conducted by Halford et al. (1998).

Human participants in the Halford et al. (1998) experiment had to choose which of four elements best completed an element-operation pair. The difference between elements and operations was stressed by coding elements as three-letter strings and operations as shapes. In addition, participants were given cards depicting each of the four elements in order to facilitate solving the task (Halford et al., 1998). We implemented the difference between elements and operators by coding each element with a different integer between 5 and 16 and each operation with a different integer between 1 and 4.

Participants in the Halford et al. (1998) experiment completed four different versions of the Klein 4-group task, but networks in the Phillips and Halford (1997) simulation only completed two versions. In our simulation, networks completed four different versions of the task as humans did. Participants' performance in the Halford et al. (1998) task was measured by their error rate. Participants were found to have a lower error rate by the beginning of the fourth version of the task than they had at the beginning of the first version, suggesting that they were learning the task faster in later versions. To mimic this, we recorded networks' learning speed on each of the four tasks.

As noted, we used an analog coding scheme to facilitate generalization rather than the localist coding scheme used in Phillips and Halford's (1997) simulation, which is known to limit generalization.

Our networks were given an input of a triad, and asked to predict if the given triad was or was not a possible member of the Klein 4-group. There are 64 ways to combine one of four possible initial elements, one of four possible operations, and one of four possible final elements. Of these 64 possible combinations, 16 are valid members of the Klein 4-group and 48 are not. All 64 triads (e.g., element 1, horizontal, element 2; element 1, horizontal, element 3) were created and paired with the corresponding output (+0.5 if the triad was a valid example of the Klein 4-group; -0.5 if it was not). This approximated the limited choice given to participants in the Halford et al. (1998) experiment, who selected one of four possible elements to form a valid triad.

Each of the four elements was randomly designated with a different integer between 5 and 16 and each of the four operations was randomly designated with a different integer between 1 and 4. There are 285,120 ways in which to pair elements and operations with that coding. Operations and elements were chosen from different sets of numbers to mirror the distinction between operations and elements found in the Halford et al. (1998) study, in which elements were designated with letter strings and operations were designated with shapes.

By designating elements and operations with numbers, element-operation-element triads could be created and paired with the appropriate output for that triad. For example, if the triad 'element 1, horizontal, element 2', translated say to 12 2 7, is a possible member of the Klein 4-group, this input would be paired with the output +.5. By pairing triads and outputs, a training set of 64 input-output pairs, or training patterns, was created. Four sets of training patterns were formed. A training epoch consisted of the network processing each of the 64 training patterns.

Our network task thus approximated the task given to human participants by Halford et al. (1998). Elements and operations were differentiated and networks were given a choice of answer. Networks were required to learn and thus generalize over four different consecutive versions of the task, allowing us to measure learning speed. The fresh coding for each successive training set required as much extrapolation as did the human experiment. Networks were allowed to build their own topology and weights were not erased when moving to a new version of the task.

Testing for Generalization

Twelve SDCC networks, corresponding to the 12 participants in the Halford et al. (1998) study, were trained on four successive training sets, representing each of the four versions of the task. Networks varied from each other because of unique random initializations of their weights, and because of unique random codings of elements and operations.

Networks were trained on the first training set until reaching victory. Victory was declared if all of the output activations generated by the network for each training pattern were within a certain threshold of the correct output. In this case, this threshold was 0.4, the standard value for sigmoid output units (Shultz, 2003). All other network parameters were also left at their default values. Once the network completed this first training set, the network began training on the second set. At the start of this second phase, the weights created during the first phase remained intact. This allowed the network to preserve the knowledge used to reduce error during the first training set and then to continue learning from that point. The algorithm continued to recruit hidden units as needed. Once victory for this second training set was reached, a third training set was substituted, etc. Upon completing four training sets, the network's topology and knowledge representations were examined.

If the networks reached victory faster for later versions of the task than for earlier versions, then we would conclude that networks had abstracted something of the underlying structure of the task.

Results

The mean numbers of epochs needed to learn the successive training sets are pictured in Figure 3; and the mean numbers of hidden units recruited in successive training sets are pictured in Figure 4. Fewer epochs and hidden units were needed in later training sets. Hidden units were arranged on a mean of 11 layers by the end of the fourth training set.

A one-way repeated measures ANOVA was conducted to test whether a difference existed in the number of epochs needed to reach victory across the four successive training sets. This ANOVA revealed an overall difference between the means, *F(3, 33)* = 26, *p* < .001. There were both linear, *F(1, 11)* = 112, *p* < .001, and quadratic, *F(1, 11)* = 11, *p* < .01, trends in these data, reflecting increased learning speed across versions. Bonferroni–corrected pairwise comparisons revealed that the mean of first training set was greater than the mean of the third training set (*p* < .05), as for humans (Halford et al., 1998). Additionally, the mean of the first training set was also greater than the mean of the second training set (*p* < .05).

An analogous ANOVA was conducted to test whether a difference existed between the numbers of hidden units recruited in successive training sets. This ANOVA also revealed an overall difference between the means, *F(3, 33)* = 39, *p* < .001. Again, there were linear, *F(1, 11)* = 244, *p* < .001, and quadratic, *F(1, 11)* = 19, *p* < .002, trends in these data, indicative of increased learning speed over successive training sets.

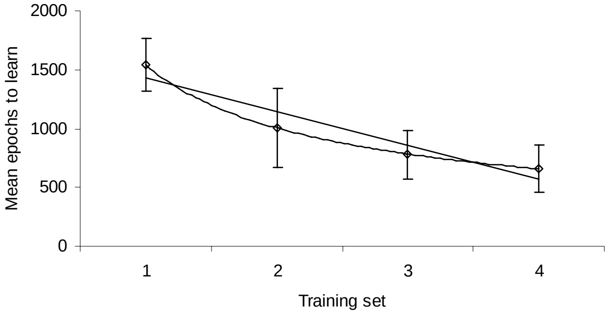

Figure 3: Mean number of epochs needed to reach victory during consecutive training sets, along with standard deviation bars and linear and quadratic trend lines.

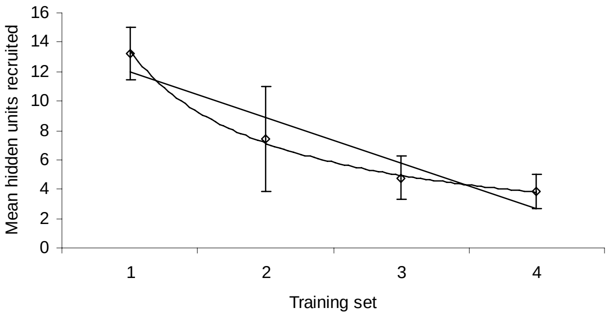

Figure 4: Mean number of hidden units added during consecutive training sets, along with standard deviation bars and linear and quadratic trend lines.

To understand the networks' knowledge representations, the contributions of input and hidden units were analyzed using Principal Component Analysis (PCA). Network contributions are the products of sending-unit activations and connection weights entering into output units (Shultz & Elman, 1994). Although the full results cannot be elaborated here because of space limitations, it was found that element input units loaded onto one principal component while the operation input unit loaded onto another principal component. Component scores suggested that networks distinguished the different element and operation values.

An example of the relationship between the four operations and the component the operation unit loaded onto is pictured in Figure 5 for one representative network. Mean scores on this component distinguished the four operations from each other. An example of the relation between the four elements and the component representing those elements is shown in Figure 6 for this same network. Both initial and final element values were distinguished from each other by these component scores. Such representations became somewhat more precise with each successive version of the task.

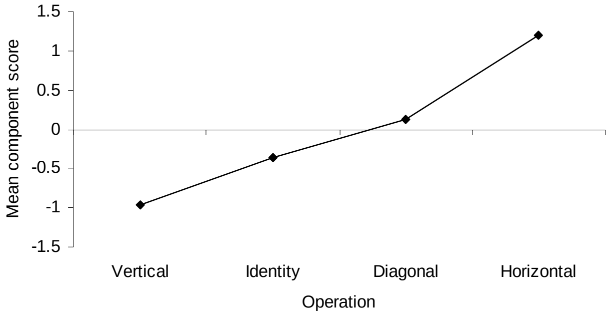

Figure 5: Mean component-2 scores for network 6 after the last training set. The operation input unit loaded heavily onto component 2, which distinguished the four different operations.

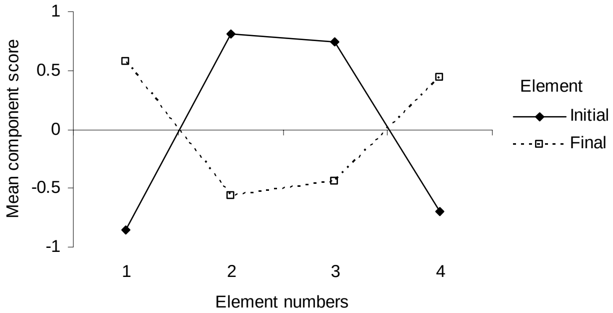

Figure 6: Mean component-1 scores for network 6 after the last training set. The initial and final element input units loaded heavily onto component 1, which distinguished the four different element values.

The performance of the 12 Klein 4-group networks training on the first training set was also compared to the performance of 12 networks that had been pre-trained on another moderately complex problem, continuous exclusive-or (Shultz & Elman, 1994) before learning one training set of the Klein 4-group task. This comparison was used to examine whether networks reached victory for later training sets because of the increased complexity of the networks learning later sets. The pre-trained networks had recruited an average of 6.17 hidden units during pre-training. When training on the Klein 4-group task, the pre-trained networks did not reach victory faster than networks without pre-training. These results suggest that mere network complexity cannot account for the increase in performance across successive training sets.

Discussion

It is evident that the networks abstracted something of the structure underlying all four of the training tasks because they reached victory faster for later versions of the task than for earlier versions. This simulates human performance in the Halford et al. (1998) experiment, in which people were faster in learning later than earlier tasks. The number of trials needed by human participants and the number of epochs needed by networks to reach victory are not directly comparable because there is no way of knowing how much processing humans are doing on each trial. What is important is that both humans and networks exhibited faster learning in successive versions of the task.

Knowledge-representation analyses consisting of PCAs of network contributions revealed that the element input units loaded onto one component and that the operation input unit loaded onto the other component. In addition, the specific elements and operations could be distinguished by variation in component scores. The fact that these representations became more precise with each new version of the task supports the idea that networks were gradually abstracting some of the structure of the Klein 4-group.

These learning speed and knowledge-representation results contradict the claims by Phillips and Halford (1997) and Marcus (1998) about the inability of neural networks to generalize abstract relations outside of the training set.

As demonstrated by Browne (2002), the coding scheme employed in training and testing networks plays a large role in a network's ability to generalize. The binary coding scheme used by Phillips and Halford (1997) prevented successful network generalization, as did their erasing of weights when moving from one phase to another. Our changing the identity of elements and operations in each new training phase, while preserving the weights, corresponded more closely to the method of Halford et al.'s human experiment. Plus, our use of analog representations allowed for generalization across the four different versions of the problem. Another factor in the current networks' ability to generalize may have been the use of SDCC, which allowed networks to configure their own architecture, but a direct test of this would require comparing the present results to static networks (Shultz, 2006).

The possibility that the networks were not abstracting the Klein 4-group structure but simply solving the task faster as their architectures became more complex was refuted by comparing the performance of networks only learning the Klein 4-group task and networks pre-trained on a continuous exclusive-or problem before learning the Klein 4-group task. Because the learning speed of the pre-trained networks did not differ from networks without pre-existing hidden units, more complex architectures could not explain increases in learning speed.

The use of PCA to analyze knowledge representations in our networks provided some understanding of how network knowledge was organized. The PCAs revealed that networks distinguished between operations and elements and between different values on each of these two dimensions. It would be interesting to determine if networks trained on other mathematical group tasks would develop a similar pattern of loadings, perhaps suggesting similar solutions to the tasks.

Other future work might use the present simulations as predictions for a psychology experiment in which participants would be asked to classify example triads of two elements and a transforming operation as instances or non-instances of the full Klein 4-group.

A better experimental control for network complexity than the continuous exclusive-or task used here might involve randomizing the weights of an SDCC network that was trained on a version of the Klein 4-group. Such a randomized network would be exactly as complex in topology as the network from which it was derived but lack any content knowledge of the Klein 4-group.

Although Phillips and Halford (1997) claimed that the Klein 4-group task is one of systematicity, this systematic structure may only appear at a level of abstraction achieved by rather skilled mathematicians. Shultz and Bale (2001, 2006) presented similar examples of CC networks exhibiting rule-like behavior without directly representing or manipulating symbolic rules. Symbolic rules can be seen as emerging from the subsymbolic processing of neural networks (Smolensky, 1988). In an analogous fashion, our networks learned something about the structure of the Klein 4-group without actually representing and processing the explicit rules of identity, inverse, associativity, and closure.

Acknowledgements

This research was supported by a grant from the Natural Sciences and Engineering Research Council of Canada to the second author. We are grateful to Dirk Schlimm for helpful comments on an earlier draft.

References

Baluja, S. & Fahlman, S. E. (1994). Reducing network depth in the Cascade-Correlation learning architecture. *Tech Report CMU*-CS-94-209, School of Computer Science, Carnegie Mellon University.

Browne, A. (2002). Representation and extrapolation in multilayer perceptrons. *Neural Computation, 14,* 1739-1754.

Halford, G. S., Bain, J. D., Maybery, M. T., & Andrews G. (1998). Induction of relational schemas: Common processes in reasoning and complex learning. *Cognitive Psychology, 35,* 201-245.

Marcus, G. F. (1998). Rethinking eliminative connectionism. *Cognitive Psychology, 37,* 243–282.

Phillips, S., & Halford, G. S. (1997). Systematicity: Psychological evidence with connectionist implications. In M. G. Shafto & P. Langley (Eds.), *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.

Shultz, T. R. (2001). Assessing generalization in connectionist and rule-based models under the learning constraint. In J. Moore & K. Stenning (Eds.), *Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.

Shultz, T. R. (2003). *Computational developmental psychology.* Cambridge, MA: MIT Press.

Shultz, T. R. (2006). Constructive learning in the modeling of psychological development. In Y. Munakata & M. H. Johnson (Eds.), *Processes of change in brain and cognitive development: Attention and performance XXI.* Oxford: Oxford University Press.

Shultz, T. R., & Bale, A. C. (2001). Neural network simulation of infant familiarization to artificial sentences: Rule-like behavior without explicit rules and variables. *Infancy, 2,* 501-536.

Shultz, T. R., & Bale, A. C. (2006). Neural networks discover a near-identity relation to distinguish simple syntactic forms. *Minds and Machines, 16*, 107-139.

Shultz, T. R., & Elman, J. L. (1994). Analyzing cross connected networks. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in Neural Information Processing Systems 6.* San Francisco, CA: Morgan Kaufmannn.

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