10 Conversational Networks

Joseph Woelfel
Department of Communication
State University of New York at Buffalo

William Richards
Department of Communciation
Simon Fraser University

N. J. Stoyanoff
Buffalo, New York

In Chapter 2 of this volume, Woelfel presented a general theory of networks as a mechanism for the generation of cognitive processes. In this chapter, we focus on a specific cognitive process, *conversation*. Conversations, as we consider them here, represent sequences of conceptual patterns, whether spoken or written, whether uttered by individuals, machines or organizations. Within this definition, conversations can include simple exchanges of greetings among passersby, theoretical or political disputes, or multimilleneal dialectic processes among opposing philosophical opinions.

The underlying philosophy of the approach is Durkheimian, in the sense that each linguistic pattern is considered a "social fact" independent of its substrate, and we consider the ways in which these social facts are associated with each other. While Durkheim was hopeful of the possibility that the regular association and succession of social facts might be understood without attention to the specific mechanisms which underlay these associations, recent developments in the study of neural networks, both real and artificial, have shown great promise of providing a plausible mechanism for cognitive processes both on an individual and a cultural level. In this chapter we explore the extent to which simple artificial neural network models can model individual and collective conversations.

We consider communication networks as the substrate and mechanism through which the social facts become associated, and present computer models of such conversations. Two kinds of conversational networks are considered here: *unidimensional networks*, in which each node is unidimensional, that is capable of taking on only a single value at a time, regardless of whether numerical or symbolic, two-valued or real; and *multidimensional networks*, in which each node is capable of taking on several values at once. Unidimensional networks are typical of neural networks and human individuals; multidimensional networks are typical of social networks in which nodes are either individual persons or organizations.

I. INTRODUCTION

The main characteristic of communication networks as discussed in Woelfel (this volume) is their ability to represent patterns and to associate one pattern with another[1]. In the most general sense, conversations may be construed as sequences of patterns, with each utterance considered a pattern of sounds, words, or even letters. With this in mind, it is possible to construct communication networks whose structures are optimized for the recognition and association of linguistic patterns. The process of constructing a communication network consists essentially of defining the pattern of communication links which are allowed among the nodes.

Although it is possible in principle to conceive of every node in a network communicating with every other node, in concrete situations such networks seldom occur. In the human brain, for example, conventional estimates set the number of neurons at about 10^{11}, but each neuron is generally thought to connect to perhaps between 10^3 and 10^4 other neurons. Similarly, although there are about $5X10^9$ people on the earth, each person communicates with only a very small subset of them, and communicates substantially with only a few.

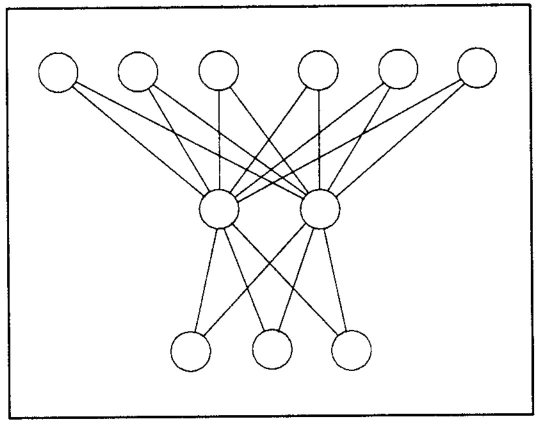

Figure 1. A Three Layer Feedforward Network

In fact, constraints on which communication links may not be made -- that is, which weights must always be set to zero -- determine the overall structure or architecture of a network. Figure 1 shows a simple yet interesting information processing network: a three layer feed-forward network. The row of nodes at the top of the figure represent input nodes; they are connected to a row of hidden nodes, which in turn are connected to a row of output nodes. Nodes within a row are not connected to each other, nor are any of the input nodes connected to any of the output nodes except through the hidden layer, and these connections are themselves only one way paths. Input nodes may communicate to the hidden nodes, and hidden nodes to the output nodes, but the reverse processes -- hidden to input and output to hidden -- are prohibited.

A network of this configuration can receive inputs from its environment, form an

[1] Patterns, like networks, may be arbitrarily partitioned. It may be convenient for some purposes, for example, to consider the phrase "How are you?" to be a single pattern, and to consider the phrase "I'm well, thank you" to be another. Or it may, for other purposes, be useful to consider both phrases part of a single pattern. Depending on the arbitrary terminology employed, a network might be considered a "heteroassociator", which associates one pattern with another or one part of a pattern with another part, or an "autoassociator", which associates any part of a pattern with the entire pattern.

internal representation of the input patterns in the hidden layer, and output a pattern corresponding to the input pattern. The pattern of weights between the input and hidden layer and the hidden and output layer will determine the relationship of the output pattern to the input pattern.

A network of this general configuration is implemented in the FORTRAN program SPOT. Its the input and output layers are each in the form of the 50X7 matrix, with a 1 X 115 vector of hidden nodes between them. Each of the 7 nodes in each row of the input and output matrices are required for the distributed encoding of each character in the ASCII set, and each row of the input and output layer is thus able to represent one such character. This network can learn to associate a number of utterances or "strings" of up to 50 characters with any other arbitrary set of strings of up to 50 characters.[2] The "memory" of what output strings "go with" what input strings is contained entirely in the pattern of weights among the layers.

The network works as follows: input strings of letters are encoded into their ASCII representations which consists of a seven digit array of 1's and 0's; and each node corresponding to a 1 in the appropriate row of the input matrix is turned "on", (that is,its activation value is set to one), and all others are turned "off", or set to zero. Each letter in the string of letters thus corresponds to one row of the 50X7 matrix of input nodes.

The activation values of the input nodes (either 1 or 0) are multiplied by the weights which represent the communication strengths between the input nodes and the hidden nodes. The activation values of the hidden nodes are then calculated as the logistic of the sum of the activations of the input nodes multiplied by the weights from input to hidden nodes (see Woelfel, this volume). The pattern of activations of the hidden nodes represents an internal symbolic representation of the input pattern, and serve an intermediary role between the input pattern and the output pattern. The mapping of the input pattern as it is encoded on the input neurons onto the hidden neurons represents an internal restructuring, and, (in the case of the SPOT and ROVER algorithms presented here) more parsimonious form of the input pattern, but it is fair to say that a complete understanding of the way in which hidden nodes functions awaits further research. It is known, however, that the hidden layer makes possible solutions to problems that cannot be solved by linear models (Minsky & Papert, 1969; Rumelhart, et. al., 1986, pp. 318-362, McClelland & Rumelhart, 1988, Chapter 2). The level of complexity of the internal symbolic structures that can be formed, and the level of complexity of the functional relations between input patterns and output patterns that can be learned by a network are related to the number of hidden neurons[3].

[2] How many such patterns the network could learn would be determined by the number of connections possible between the layers, which in turn is determined by the number of nodes in the layers. For input and output layers of fixed size, as in the present example, increasing the number of nodes in the hidden layer will increase the number of patterns the network can learn.

The activations of the hidden nodes are then propagated to the output nodes by exactly the same process. The values of the weights determines completely what output nodes will be activated for any given pattern of input node activations, and thus determines uniquely what the network will output or "say" for any given input. How the weights are actually set will be discussed below. These output nodes are then "thresholded"; that is, if their calculated values exceed an arbitrary threshold value, the are set to one, otherwise they are set to zero. The resulting 50X7 binary matrix of output nodes can then be "decoded" into the appropriate ASCII characters.

A network like SPOT can be taught to associate any input phrase with any output phrase by setting the communication weights appropriately. Thus, for example, it is possible to choose a set of weights such that the pattern of activations of input nodes corresponding to a phrase such as "How are you, SPOT?" turns on the set of output nodes which correspond to the phrase "I'm fine, thank you," while the pattern of input activations corresponding to another phrase will activate a pattern of output nodes corresponding to still another phrase. How many such pairs of input and output patterns the network can learn is a function of the number of nodes and the degree of similarity among the patterns.

"Learning" to associate input patterns with output patterns consists of modifying the weights among layers until the error (i.e., the difference between the actual activation values of the output nodes and the activation values demanded by the correct pattern) is minimized. Both the SPOT and ROVER programs implement the "back-propagation" method (see Woelfel, this volume), which modifies the weights in a semi steepest descent algorithm.

Figure 2 shows SPOT learning the "correct" responses for two phrases: to the phrase "Hello, Spot!", it is expected to say "Hello.", and to the phrase "How are you, Spot," it is expected to reply "I'm well, thank you." As implemented in SPOT, the network requires 8 tries to get both responses correct, although in the example presented here, the network "overlearns" for two more iterations (not shown) until the error is within the prespecified tolerance.

[3] Increasing the number of hidden nodes in an otherwise unchanged network increases not only the number and complexity of associations the network can learn, but also increases the speed with which it can learn them. This fact is somewhat confounded when the parallel architecture is simulated on a Von Neumann machine, since the number of operations the algorithm must perform increases as a function of the number of connections in the network. The SPOT algorithm,for example, learned a simple training set in 1,005 seconds with 55 hidden nodes, but took 27 trials to do so. Increasing the number of hidden nodes to 75 cut the time to 757 seconds and reduced the number of trials to 15. Increasing the number of hidden nodes to 100 reduced the number of trials to 14, but *increased* the time to 960 seconds. Increasing the number of hidden nodes to 115 reduced the number of trials to 12 and the time to 935 seconds. If the network were able to carry out its activities in a fully parallel fashion, increasing the number of nodes would result in a monotonic reduction in both number of trials and overall time.

1. Hello, Spot! ^_^_!/ ^_^_5!A^_!/' ^_/'^_ ?^_	9. Hello, Spot! Hello.
2. How are you, Spot? Hello.	10. How are you, Spot? Igm wed(, tha$d you.
3. Hello, Spot! Hmmlo. ,(0' ' _ p(	11. Hello, Spot! Hello.
4. How are you, Spot? Jello. 0 (	12. How are you, Spot? I'm well, thafk you.
5. Hello, Spot! Hmmlw.$,(0' (''t,	13. Hello, Spot! Hello.
6. How are you, Spot? Jen w. (p' /_d,	14. How are you, Spot? I'm well, thafk you.
7. Hello, Spot! Hmllo.	15. Hello, Spot! Hello.
8. How are you, Spot? J_m w&'($ t'b$' wof,	16. How are you, Spot? I'm well, thank you

Figure 2. Three Layer Feedforward Network SPOT Learning Two Phrases (Dialogue)

Total Squared Error by Elapsed Time forLearning Two Phrases.*

CumElap Time	Tot ErrSqrd	Elapsed Time
16.92000	22.85055	16.92000
29.55000	27.98030	8.40000
42.85000	24.09617	8.68000
55.20000	16.42570	8.18000
67.24000	12.22997	7.86000
78.66000	6.93973	7.52000
89.37999	2.93760	7.09000
100.03000	1.62046	7.03000
110.63000	1.67598	7.03000
120.73000	.76299	6.59000

* Threshold: .50; Momentum: .30; Local Tolerance: .30; Heat: 0.00; Rate: 1.00; Nodes: 210 Input; 210 Output; 115 Hidden; Toshiba 30386 @20mhz with 30387 math chip

Figure 3. Three Layer Feedforward Network SPOT Learning Two Phrases

The network begins by producing a random response to the initial inputs, but quickly learns the correct response through a series of successive approximations, each time closer to the pattern than the last. Figure 3 shows errors for each iteration, along with the elapsed time for each trial.

The values of the weights represent the learned pathways of communication between the input nodes and the output nodes. They represent the network's "memory" of what string of ASCII characters it should output for any given input string of ASCII characters.

Such a network can be taught to carry on a rudimentary conversation. For an input phrase such as "How are you, Spot?", it might be taught to respond "I'm fine, thank you." What's more, as shown earlier, because the activation values of the output nodes are determined by the entire pattern of activations in the layers preceding them and the entire pattern of connections among those layers, the complete output pattern can be activated by an input of only part of the input pattern; minor misspellings of the input string, or even leaving parts of the input blank will still result in the output of the entire pattern.

Thus, for the input string "How are you, Sotp?", or just "How are you?", the network would respond "I'm fine, thank you." As the input pattern deviated further from the pattern the network had learned to associate with the output, the output pattern would degrade fairly gracefully, but would retain the main features of the correct output pattern even with considerable distortion or deletion from the input pattern.

Figure 4 shows a brief conversation with a three layer feed forward neural network (SPOT) in which the input pattern (marked "O:" in the figure) is gradually changed from the pattern the network has learned to recognize. The network is able to output the correct response even after substantial changes in the input pattern, but eventually degrades as the input pattern departs further from the learned pattern.

Conversation with Galileo*SPOT O: = Other; S: = SPOT	
O: How are you, Spot? S: I'm well, thank you.	O: Hew are you, Spit? S: I'm well, thank you.
O: How are you, Sopt? S: I'm well, thank you.	O: How are S: A'm well, thank you.
O: How are you? S: I'm well, thank you.	O: How you doin', Buddy? S: Agm weld, t _nk)>v,
O: Hou art yee, Spet? S: I'm well, thank you.	O: Howdy do? S: A'm"weld, phanc yov(

Figure 4. Conversation with Three Layer Feedforward Neural Network SPOT

A. Innovative Language:

As the previous example shows, the thresholding function guarantees that a network can continue to produce without error an output pattern it has learned to associate with a given input pattern, even when the input pattern differs to some extent from the pattern originally learned. One way, then, that a network deals with an innovative input pattern, that is, one it has not previously encountered, is to output the pattern that corresponds to an input pattern which is similar to the novel input pattern. As the example also shows, however, as the input pattern deviates still further from the original form, the network outputs a pattern which also differs from the one learned. When these deviations are arbitrary, as they are in the example, the network can produce an output pattern that is itself arbitrarily degraded or noisy. The network will, in other words, produce a novel output when receiving a novel input, although in this situation the output will typically be meaningless, as in Figure 4. Such productions of novel but meaningless output patterns in response to novel input patterns does not meet Chomsky's criterion that the novel output be meaningful.

It is possible, however, for novel inputs to a network to be related in systematic ways to input patterns the network has already learned. When this happens, it is possible for the network to output a novel pattern which it has not previously encountered, but which is still a meaningful pattern.

A network identical in structure to the network shown in the previous example, although employing local rather than distributed encoding[4], was taught to associate the input pattern "GREET" with the output pattern "HELLO", and also to associate the input pattern "MY FRIEND" with the output pattern "BOB". When the pattern "GREET MY FRIEND" is input to the network, it responds "HELLO BOB"[5]. Neither the input pattern "GREET MY FRIEND" nor the output pattern "HELLO BOB" has ever been encountered by the network before, but the network is able nonetheless to generate an appropriate English sentence which is a "correct" novel response to the novel input pattern.

To be sure, this is a very limited example of innovation, but, in principle, it responds

[4] When a network of this type encounters two or more previously learned input patterns simultaneously, it activates the combination of communication channels or connections appropriate to that combined input set. This produces an output pattern which is a combination of the output patterns associated with each of the input patterns separately. In the particular distributed encoding scheme employed by the SPOT algorithm, this results in an output pattern which represents a set of ASCII characters which, while a proper combination and in fact a valid inference, nevertheless requires additional interpretation to be understood. In the locally encoded network used in this example, each node represents a single letter; thus when the network outputs a combination of these letters, they can be understood easily.

[5] Whenever SPOT sees an input pattern that differs from one it has learned, its outputs are somewhat attenuated; although activation values of output nodes may still be high enough to exceed their threshold values and thus be turned on, the pre-thresholding levels are often somewhat lower. This results in a somewhat more "tentative" response; in fact, SPOT is somewhat unsure about the first letter in the word "BOB", and guesses it might be either a "B" or a "G".

to Chomsky's (1972) argument that the number of possible English sentences is simply too large to have been learned and remembered, but must instead be generated from a set of internal rules. (Chomsky, 1972, pp 11-12). The sentence "HELLO BOB" was "generated" by the network in response to a novel input not previously encountered, but the network was not following any rules in so doing. Nor was the novel response in any meaningful sense programmed into the network, but rather was exclusively the result of its training.

B. Self Referencing Networks:

The network shown in Figure 3 has several interesting conversational properties: it can associate appropriate linguistic outputs with arbitrary language inputs, it can recognize a known input pattern even if it differs fairly substantially from the exact form in which it was learned, and it can produce meaningful and novel output utterances in response to novel inputs. It will, however, always respond in exactly the same way to the same input pattern regardless of the context in which it occurred. The network illustrated in Figure 5, on the other hand, is somewhat more sophisticated. This network resembles the previous network except for feedback loops from the output nodes to half of the input nodes. This means that the input pattern which is associated with a given output pattern includes not only the pattern from the environment, but also the pattern previously output by the network. This network need not respond in exactly the same way twice to any given input pattern from the environment. The network in Figure 5 is *self referential* in that it takes its immediate past behavior as part of the pattern to which it must respond.

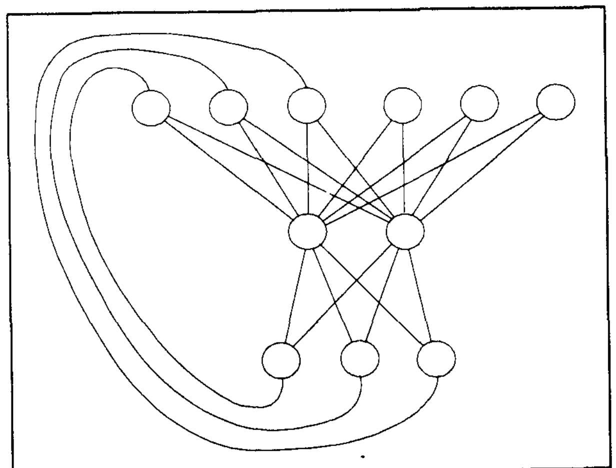

Figure 5. Three Layer Feedforward Network with Feedback

Figure 6 shows a conversation between a network of this type and a person (other). Note that the network responds differently to exactly the same input string depending on what it has previously said. The network has taken its past behavior into account in determining its response to the input from its conversation partner.[6]

Conversation with Three Layer Network with Feedback
O: = Other; S; = Spot
O: How are you, Spot? S: I'm fine, thank you.
O: How are you, Spot? S: Still fine, Thanks.
O: Oh, I'm sorry. S: That's O.K.

Figure 6. Conversation with a Three Layer Network with Output Feedback

The feedback loops from output to input add another important property to the network. Since it can take its own output as input, its future behavior can be cued by its past behavior. This means that such a network can be taught to emit a sequence of outputs of indefinite length. The network implemented in ROVER, for example, was able to learn to recite the text of "A Bicycle Built for Two" in about 20 minutes. Each line of the text serves as an input pattern which the network learns to associate with the following line. Once output, a given line then becomes the input, which ROVER in turn learns to associate with the next line, and so on. When the network receives no new input from its environment at a given stage, the only input available to it is its previous output, and so it responds by producing the next line, and so on until it has recited the entire lyric.

[6] While the network implemented in ROVER takes into account only the last utterance the network has made along with the new input from its conversation partner, there is no reason in principle, nor any particular technical difficulty in extending the model back for as many stages as desired; a network can easily be programmed which will take into account the last two or four or eight or any number of previous exchanges in determining what it should output. It is also easily possible to weight earlier episodes differentially, giving them successively less weight as they recede into the past. How many stages (or how long in real time) a network ought to take into account in determining its response in order to be an interesting conversation partner remains an unanswered empirical question, but presents no special programming difficulties. Such networks could not be accused of "linear, sequential" thinking, since they might well revise their understanding of a previous utterance given a later one.

While not implemented here, it is clear from these examples that the stacking of inputs is a simple variation, which would allow the network to cue its next response from any *n* of its previous responses, so that what it said at any given juncture could be a function of its last *n* utterances.

In ROVER, the computer program which implements this design, the feedback from output nodes to input nodes is done after thresholding the output units. It is more appropriate to think of this network as monitoring its *behavior* rather than its *"thinking"*. If the feedback loop were implemented before thresholding, the input nodes would be aware of what the network was *thinking* just before it "spoke", but would not be aware of what it actually said. A more sophisticated network could, of course, be aware of both by taking feedback from both places. This can be accomplished quite simply, as Figure 5 shows.

While the network shown in Figure 5 is self-referential in an important sense, the network shown in Figure 7 is even more so.

The networks described so far associate input patterns with output patterns through weighted communication connections from input nodes to output nodes through hidden layers of nodes. When the network has learned an association, the activation of the nodes associated with the input pattern will be channelled through the weighted communication channels to the nodes associated with the proper output pattern. It is also possible, as shown earlier, for a novel input pattern to be related in a systematic way to patterns which a network has previously learned, so that the network "knows" a correct response for even these novel input patterns. But when an input pattern that the network has not learned to associate with any particular output pattern is input to the network, it will output an arbitrary nonsensical pattern. The network does not know whether it "knows" what it is about to say, and will produce babbling for unlearned input patterns.

It is quite important that a self-referential network like ROVER not babble, since such a network will necessarily take into account the immediate history of a conversation as the pattern to which it must respond. If that history contains a sequence of random or arbitrary utterances, there will likely never be a consistent pattern for the network to learn, which would seem to present a formidable barrier to developing conversational competence.

The network in Figure 7 has an additional node which monitors the other output nodes to determine whether they are patterned or not. In order to understand how this monitor node operates, it is useful to recall that the network represents a pattern by turning some of its output nodes "on" and turning the rest "off". When the network is representing a pattern it has learned, therefore, its output values all be either nearly 1.0 or 0.0. (Since the activation function for this network is the logistic, actual values range closer to .9 and .1.) When the network is representing arbitrary or random nonsense, on the other hand, the values of the output nodes will take on the full range of values between 0.0 and 1.0, with a mean value of about .5. Thus a network which is representing a learned pattern will have output activation values that are maximally different from the mean activation level.

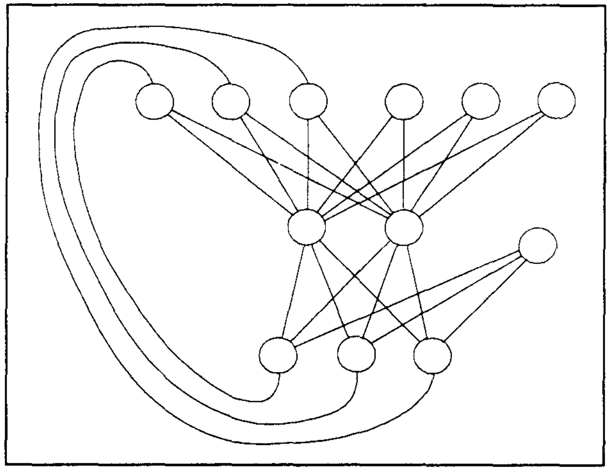

Figure 7. Three Layer Network with Feedback and Output Monitoring

Input to the monitor node, then, consists of the (squared) differences between the actual values of each output node and .5, the mean value expected for an arbitrary nonsense output. Once appropriately normalized, these values are summed and entered into the activation function for the monitor node; if its actual activation exceeds a preset[7] threshold, the monitor node "senses" a learned, patterned output, and activates the network's output. If, on the other hand, the activation value of the monitor node falls below its critical threshold, it is quite likely that the pattern represented by the output nodes is simply an arbitrary, unlearned nonsense pattern. In this case, the network's output is set to "blank". It is important to note that this node does not determine whether the output pattern is "correct" or "sensible", but simply can detect the difference (in most cases) between a systematic, patterned output and gibberish.

While it would be wrong to attribute too much sophistication to the model implemented in ROVER II, the monitor neuron goes beyond simple self reference, and adds a minor but nonetheless important self *evaluative* dimension to the network. While the model implemented in ROVER is "aware" of its past behavior and takes it into account in determining its subsequent behavior, the model in ROVER II is aware of both

[7] While in the ROVER II implementation this threshold is hardwired, it would be a straightforward modification to make its value depend on inputs to the network, so that, for some kinds of input, the network would be very careful not to babble; that is, to make very sure it "knew" what it was about to say before responding, while, in response to other inputs, it might be more willing to guess at a response even though there was a high likelihood it was nonsensical.

its past behavior and certain characteristics of its present "mental state" or "thinking", and it "evaluates" that state before implementing the action implied therein.

While the networks implemented in the SPOT and ROVER algorithms show in principle that one may construct conversational, self referencing systems of communication networks, they are in fact very simple, small and limited networks. The largest (ROVER II) consists of only 601 neurons, and 39,725 possible communication links[8]. Compared to a single human brain, with perhaps 10^{11} neurons, these networks are minuscule. Further, while the most sophisticated of the networks described here, the architecture of ROVER II is severely limited compared to that of a single human individual. ROVER II has only one input "sense": its input is restricted to 50 ASCII characters from a file or keyboard, while a human individual can receive information from multiple senses. The simultaneous activation of nodes connected to visual, auditory, taste, olfactory and tactile senses, coupled with a simple Hebbian learning rule which enhanced the connection among those nodes simultaneously activated, make possible the formation of complex internal patterns which can be activated by partial inputs, so that a picture of food, for example, could produce the same pattern as the taste or smell of the same food. This is in principle possible for an artificial network like ROVER II, although the technical difficulties of simulating such massive parallelism on Von Neumann architecture machines are for the moment quite formidable. (Although the ROVER II architecture is completely parallel, its implementation is simulated on a Von Neumann serial machine. This means that it cannot actually do any two things simultaneously, and must take in information in "batches" and operate in discrete "jumps" or "cycles".)

ROVER II is thus substantially handicapped when taking in information needed to define its social situation; it may well be more appropriate to compare it to a person who received all his or her information about the world from a teletype which could deliver only 50 ASCII characters at a time. In spite of these limitations, however, ROVER II provides a useful basis for understanding the way in which the basic structure of a network functions in the processing of information which can be useful to an analysis of social networks and their information processing capabilities.

C. Self Referencing Social Networks

SPOT and ROVER give some indication of how communication networks can represent, learn and associate complex patterns. In this section we turn our attention to a higher level of network, in which each node itself represents a network. Because each node in these higher level networks is itself a network, the activation values of the nodes

[8] There are 350 input neurons, 75 hidden neurons, 175 output neurons and one monitor neuron. There are thus 350 X 75 = 26,250 possible pathways from the input layer to the hidden layer, another 13,125 pathways from the hidden layer to the output layer, 175 pathways from the output layer to the monitor neuron, and 25 feedback pathways from output to input.

are multidimensional, with each node itself capable of representing a pattern of arbitrary complexity. In a previous chapter (Woelfel, this volume) such a network consisting of individual people was taught to recall the pattern of distance relations among the objects in a room; each node's output in that example was itself a matrix of 105 real valued numbers, each of which represented a distance between a pair of objects in the room. Although each of the node's representations was highly erroneous, nonetheless the average values from all the nodes gave a good approximation of the actual distances.

While the capacity of the social network to store complicated patterns of information quickly and to recover them even from subsets of the original network is very substantial, the sophistication of the network illustrated in the previous example is, in certain ways, less than that illustrated in ROVER II. Like ROVER, this social network can record a pattern and output that pattern or a pattern related to it, but, unlike ROVER II, it, as a single entity independent of its constituent members, is itself unaware that it knows the pattern or not. In fact, if exactly the same questionnaire is administered to a collection of people who have not heard the paragraph read, they will for the most part output arbitrary numbers and produce a "room" which bears only accidental relations to the room described in the text.[9] Moreover, as a direct consequence of this lack of self awareness, it is not possible for this social network to correct whatever errors might exist in its representation.

The inability of the social network in the example to be self aware and self correcting is a direct consequence of the simple architecture of the network, which consists of (in the present case) about 40 multidimensional real-valued nodes, each one of which serves both as an input node and an output node. There are no hidden nodes, nor do any of the nodes communicate with any other. There is no "monitor node", which functions to compile the activation values of the other nodes, and so the network, regardless of its information storage capacity, has essentially no cognitive capacity whatever. It is essentially nothing more than an elaborate memory. The very simple network embodied in ROVER II, therefore, while quite limited compared to a human individual, can exhibit more complicated cognitive activity within the limits of its very limited memory than the set of 40 human individuals with approximately $4X10^{12}$ neurons, *as long as no internal communication structure is allowed to develop among those individuals.*

[9] Of course, each individual person in the sample will believe that he or she does not have a pattern in mind which he/she is being asked to describe, but that belief also characterizes those who did hear one reading of the description. In fact, no individual but the most exceptional does have an internal representation of the room after a single hearing. The issue here is the larger sociological or cultural issue: in the case where no one has heard the reading, neither the group as a whole nor any individual member of the group has a sense of the overall pattern of the room. But in the case where the group has heard a reading, it is still the case that no single individual has a clear grasp of the room as a pattern, but the group as a whole does have such a pattern embedded in it. Like ROVER, however, the collective group has no idea that it "knows" the pattern, even though it does know it.

The reason for this is that the size and complexity of any single pattern which can be encoded by a network is a direct function of the number of nodes, the dimensionality of the nodes, and the number of values each node may take on for each dimension, but the cognitive capacity of a network -- that is, the number of such patterns the network might store, its capacity to associate them with other patterns, and its capacity to monitor its own activities -- is a function of the pattern of communication among the nodes. *It is possible, therefore, to construct an intelligent, self referencing network from a set of nodes which are themselves not intelligent, and it is possible to construct a network of intelligent, self referencing nodes which collectively is neither intelligent nor self-referencing.*

II. SOCIAL STRUCTURE AS NETWORKS OF NETWORKS

Figure 8 shows one way in which several networks like ROVER can be connected to form a larger, higher-order network. Figure 8 pictures nine ROVER-type networks. Light gauge lines show the connections within each of the smaller networks, while darker lines depict the connections among the networks. Using ordinary matrix notation to identify each subnetwork in the larger network, we can see that three "cliques" are present in the larger network: networks (1,1), (1,2), (2,1) and (2,2) are fairly tightly connected together to form one clique, while networks (3,1) AND (3,2) form a second clique and networks (1,3), (2,3), and (3,3) form a third.. Within the first clique, the networks (1,1), (1,2) and (2,1) are very tightly connected, with all the outputs of each connected to the inputs of the other, while (2,2) is only weakly connected directly to (1,2).

Input nodes in each network which are not connected to any other networks in the larger network are considered to be connected to the environment of the larger network, and thus receive inputs from the ambient environment. The input pattern for node (1,1), therefore, consists of outputs from (1,2) and (2,1) as well as inputs from the environment. Its output patterns directly affect its own inputs through its direct feedback loop, directly affect the inputs of (1,2) and (2,1) through the (darkened) connections, and indirectly affect the inputs to (2,2) through their effects on the outputs of (1,2).

Assuming each of the subnetworks is identical to all the others, there ore nearly $2X10^{15}$ ways the 27 output nodes in Figure 8 can be connected to the 54 input nodes. Moreover, due to the highly non-recursive nature of the connections possible, the complexity of the potential behavior of even such a limited network is quite formidable, although still minuscule compared to the complexity possible in a similar size network of human individuals.

A very simple system which includes communication links among multidimensional nodes is simulated in the FORTRAN program GALIAC. In GALIAC, a network consists of a set of *n* nodes, each of which, when activated, may emit a multidimensional pattern of arbitrary complexity.[10] These nodes are interconnected by a set of weights w_{ij}. The weights may take on any real value, and express the degree to which the activation value of any node j is communicated to any other node i.

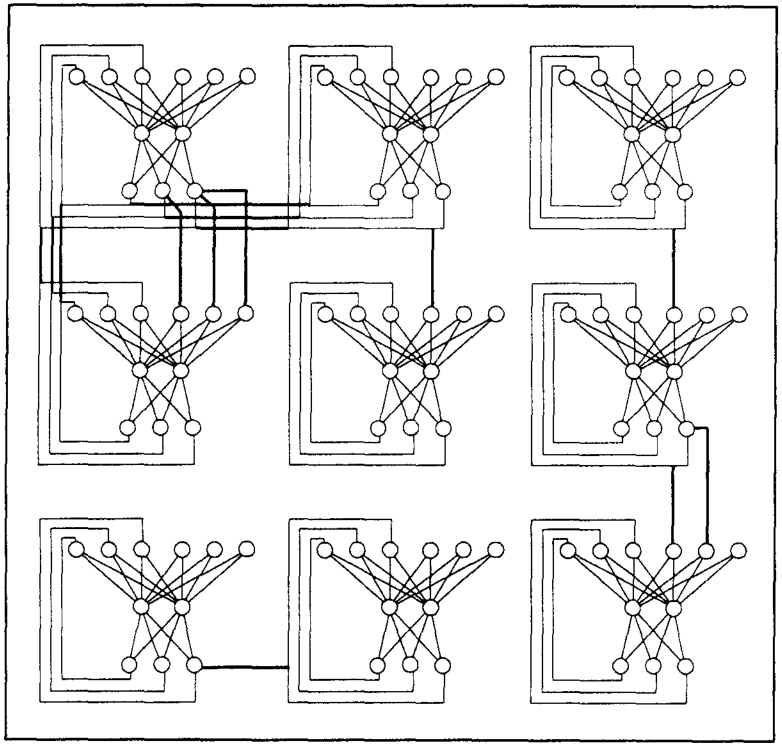

Figure 8. A network of self-referencing networks

GALIAC allows the choice of the two most common activation rules; the linear model and the logistic model. In the linear model, the activation of any node or set of nodes is communicated to other nodes by a simple algorithm: for any given node i, the activations of each of the remaining nodes are multiplied by the weight expressing the degree of communication between those nodes and the target node. These products are summed to provide the total net input to each node.

[10] In the present implementation, nodes are restricted arbitrarily to a pattern of 20 ASCII characters, but it should be understood that these characters may refer to a pattern of arbitrary length and complexity.

$$net_i = \sum_{1}^{n} a_j w_{ij} \tag{1}$$

GALIAC allows the choice of either digital or analog mode. In analog mode, these sums provide the activation value of the node; in digital mode, if they exceed an arbitrary threshold value, the target (i_{th}) node is activated; otherwise it remains off, or

$$\textit{if } net_i \geq thresh \; a_i=1; \; \textit{else } a_i=0 \tag{2}$$

where:

n = total number of nodes in the network
$thresh$ = an arbitrary threshold value
a_i = the activation value of the i_{th} node
w_{ij} = weight representing the amount of communication from the j_{th} node to the i_{th} node.

The GALIAC model operates in a cyclic fashion. During the initial cycle, some subset of the nodes may be activated by information from the environment. During the next cycle, the activation values of the nodes activated on the first cycle are communicated to the other nodes, which are then thresholded to determine their activation values. These values, in turn, are then communicated to the other nodes on the next cycle, and so forth.

The GALIAC model, however, does not assume that a node will remain active indefinitely; rather it assumes that each node will lose a proportion of its activation value on each cycle (in a typical default mode, this value is set at .90.) This is equivalent to assuming that the equilibrium value of any node is to be set to "off."

Finally, in order to dampen potential oscillations in the system, a "momentum" term is included in the activation function. The basic GALIAC model is a mechanical one, in which mass-points are deflected from an equilibrium position by an external force, and are pulled back to that equilibrium position by a restoring force. Such systems can be made to oscillate easily unless some random losses such as friction or viscosity are included in the model. In the present case, these nonconservative forces are simulated by a "momentum" term, which adds to the net activation value of each node (prior to thresholding) a proportion of its previous value. The activation function, then, for each cycle is:

$$net_{i,t} = \sum_{j=1}^{n} a_j t\, w_i - k\, a_{(i,t-1)} + C a_{(i,t-1)} \tag{3}$$

where:

k = proportion by which the activation value is reduced for each cycle, and
C = the proportion of the previous activation value which is added to the current value.

In the logistic mode, the operation is fundamentally similar, except that the initial term of the activation function is no longer linear, but rather logistic:

$$net_{i,t}=[1/(1+e^{(\sum_{j=1}^{n} a_{jt}w_{ij})}]-ka_{(i,t-1)}+Ca_{(i,t-1)} \tag{4}$$

The logistic mode of GALIAC may also be thresholded at the user's option, following equation (2).

III. GALIAC AS A CONVERSATIONAL NETWORK

Although GALIAC is about as simple as a network of linked multidimensional nodes can be, and employs substantial simplifications over linked networks which occur in nature which will be discussed later, it nevertheless bears some resemblance to organizational structures common in everyday life. The multidimensional nodes which emit unchanging multidimensional patterns when sufficiently activated are not unlike single interest groups in society. Each of these organizations can emit a complex, multidimensional pattern of information, which is usually unchanging over short time periods, when sufficiently activated by external stimulation. Table A of Appendix 1 shows the behavior of the GALIAC model given different patterns of initial activation and for various values of its restoring force and momentum parameters for both the linear and logistic models in both digital and analog form. A detailed examination of the many possible simulations GALIAC can provide is beyond the scope of this chapter, but a careful examination of these data reveal several interesting findings:

First, given the pattern of positive and negative connections among nodes explicit in the input weight matrix, certain patterns -- that is, sets of nodes which may be simultaneously activated -- are unstable. (If two nodes are strongly and negatively connected, for example, it is impossible for both to be activated simultaneously unless the stimulation from the environment is sufficiently strong to overcome the negative connection; similarly, if two or more nodes are strongly positively connected, it is difficult for one or more to be active while others are not.) Put another way, certain patterns of activation of the nodes are "unlikely" states corresponding to a high level of energy. As the nodes communicate with each other they tend to settle into a more probable, lower energy state. This is true regardless of the choice of linear or logistic activation function, and across a wide range of values for threshold, restoring force, and momentum constants. To be sure, the exact trajectory the network will follow in finding the minimum energy pattern will depend on the shape of the activation function and the values of the threshold, restoring force and momentum coefficients, but the final state remains the same over a wide variety of such choices.

A given network may encode many such stable patterns, and, depending on the initial pattern of activation, will tend to fall into the stable pattern which is "nearest" to the initial pattern of activation. Thus, stimulation of the system by activating some subset of the nodes at the initial time period results in a pattern of activation which spreads through the communication system given by the weights until a stable output pattern is achieved. *This means that the patterns which the network will ultimately display when activated are completely given by the matrix of connection weights, that is, by the underlying pattern of communication among the nodes. The successive sets of patterns of activation which represent the "path" along which the network settles into its minimum energy state represents the "conversation" through which it reaches that final conclusion.* In social networks, these conversations might be thought of as a dialogue, dialectic, political process or other dynamic process through which collective decisions are made, while, for the individual, this conversation might be considered the internal "conversation of gestures" or thought process by which an individual person "makes up" his or her mind.

Secondly, in the absence of sufficient damping (momentum) the system can show interesting patterns of oscillations among possible patterns. This pattern of oscillation can itself be stable, and looks remarkably like a dialectic process in which the opinion of the networks shifts alternately among two or more patterns. When sufficient damping is present, the network can settle down to a stable, non-oscillating solution quite rapidly and one of the possible stable patterns prevails over the other possibilities.

IV. LEARNING NETWORKS

Although the processes described in Table A of Appendix I are intriguing, certain important aspects of communication networks are not modelled in these examples. The first of these is that communication between the network and its environment is limited to impulsive inputs at the first time period, but after this the network is allowed to function without further information from outside. Secondly, the network exhibits no learning; that is, the weights of the communication connections among nodes remain invariant across each cycle. GALIAC is able to model these more advanced processes, however, quite simply.

The first issue can be addressed by "clamping" the values of the input nodes, so that some nodes can be activated and remain activated throughout the process. The second issue can be addressed in several ways, but the simplest and most obvious way is to implement a Hebb or "passive learning" rule for changing the network weights. In its simplest form, the Hebb rule (Hebb, 1949) simply requires that the weight of the connections among nodes that are simultaneously activated should be increased by some amount. In the variation implemented in GALIAC, weights among nodes that are simultaneously activated in any period are strengthened, weights between nodes whose values are opposite are decremented[11], and weights among the set of nodes which are simultaneously inactive are left unchanged, or

$$w_{ij(t+1)} = w_{ij(t)} + a_{i(t)} a_{j(t)} h \tag{5}$$

where:

$w_{ij(t)}$ = the weight representing the proportion of the activation of the j_{th} node which is communicated to the i_{th} node at time t,

$a_{i(t)}, a_{j(t)}$ = the activation values of the i_{th} and j_{th} nodes at t,

h = a constant.

Within this enhanced model, nodes which are clamped on might represent either constant inputs from the environment, and/or constant inputs from nodes within the network which emit the same message regardless of other activities in the network. These latter nodes might represent particularly energetic interest groups or vested interests. The Hebb learning rule assures that nodes who participate in the conversation together will become more closely associated. If they agree in their participation, they will be come increasingly positively linked, while if they disagree in their participation, they will become increasingly antagonistic or negatively associated, and will exercise an increasingly negative effect on each other's participation. The direct relationships among those who do not participate in the conversation remain unchanged, although their overall relationships might be affected due to indirect links to participants who are changing.

V. SUMMARY

It is important to emphasize that neither the specific parameters by which the learning rule is implemented nor the specific parameters utilized to implement the activation functions of the nodes are at issue here.Of central concern are the global characteristics of the network. A wide variety of alternative rules, functions and parameters have been investigated in the literature of artificial neural networks, and different choices will have effects on the performance of the network overall, some of which can be substantial. Nonetheless, the overall character of the conversational network remains fundamentally the same regardless of these choices, as does the basic way it functions. Communication networks all share the same fundamental features: they store information in the form of patterns of communication linkages among the nodes, they cycle through sequences of

[11] In the linear mode, activation values may take on both positive and negative values, so that activations opposite in sign are possible, in that case, equation 5 results in a direct reduction in the communication weight between the nodes whose signs differ. In the logistic model, however, all activations are positive, with a mean value of .5. Thus, in the logistic model, .5 is subtracted from the activation values before entry into equation 5.

patterns of activations that are determined by underlying communication links as well as by the shape of the individual activation functions and their parameters, and they modify themselves by changing their underlying communication patterns in response to patterned information imposed on the network. Communication networks thus form and modify internal representations of themselves and their environment, and these representations can be retrieved by activation of appropriate subsets of nodes in the network. The process by which the network moves from the initial pattern of activation to the stored underlying pattern represents a conversation which bears substantial similarities to human individual and social interaction and decision making processes.

References

Hebb, D. O. (1949). *The organization of behavior; a neuropsychological theory.* New York: Wiley.

McClelland, J. L., and D. E. Rumelhart. (1988). Explorations in Parallel Distributed Processing: A handbook of models, programs and exercises, Cambridge, MA, The MIT Press.

Minsky, M., and S. Papert. (1969). *Perceptron*, Cambridge, MA, MIT Press.

Rumelhart, D. E., and J. L. McClelland, (Eds.). (1986). *Parallel Distributed Processing: Explorations in the microstructure of cognition*, Cambridge, MA, The MIT Press.

Appendix

This appendix presents outupts of GALIAC for various values of parameters. All models described here are based on the linear model; all but one are thresholded. Each column of the output indicates which of the nodes are activated with a 1.0; inactive nodes are indicated with .0. Nodes which are activated in the first column were set to their indicated activation values by the experimenter at the outset of the run. In all but the last table, each column represents one cycle of the system; in the last table each column represents the state of the system at ten-cycle intervals.

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	1.0	1.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
MAN	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
WOMAN	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
WOMAN	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
WOMAN	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
SERPENT	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
MAN	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
WOMAN	.0	1.0	.0	.0	1.0	.0	1.0	.0	1.0	.0
SERPENT	1.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
MAN	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
WOMAN	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
EVIL	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	1.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	1.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept Cycles X 1

Concept	1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MAN	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	1.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .60

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	-.7	-1.0	-2.4	-4.1	-8.5	-15.9	-31.5	-60.5
MAN	.0	.5	-.1	.1	.3	.5	1.0	2.0	3.9	7.6
WOMAN	1.0	-.1	.5	.1	1.2	1.2	3.5	5.6	12.1	22.2
SERPENT	.0	.4	.4	1.1	2.0	4.0	7.7	14.9	29.0	56.3
GOOD	.0	-.6	-.3	-1.1	-1.3	-3.7	-6.0	-12.9	-23.8	-47.5
EVIL	.0	.1	.1	1.1	1.2	3.2	5.5	11.4	21.4	42.3

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
MAN	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
WOMAN	.0	1.0	.0	.0	1.0	.0	1.0	.0	1.0	.0
SERPENT	1.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = 1.00 Damping Factor = .00

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0
MAN	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
WOMAN	.0	1.0	.0	.0	1.0	.0	1.0	.0	1.0	.0
SERPENT	1.0	.0	1.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	1.0	.0	1.0	.0	1.0	.0	1.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .00 Damping Factor = .00

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	1.0	1.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .20 Damping Factor = .00

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshhold = .00 Restoring Force = .50 Damping Factor = .00

Concept	Cycles X 1: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	.0	1.0	.0	.0	1.0	.0	.0	1.0	.0
WOMAN	.0	1.0	1.0	.0	1.0	1.0	.0	1.0	1.0	.0
SERPENT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EVIL	.0	1.0	.0	.0	1.0	.0	.0	1.0	.0	.0

Threshhold = .00 Restoring Force = .90 Damping Factor = .00

Concept	Cycles X 10: 1	2	3	4	5	6	7	8	9	10
GOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WOMAN	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SERPENT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
EVIL	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0