

## **Knowing in a World of Broken Symmetries**

**A Symposium presented at  
The Society for Chaos Theory in Psychology and the Life Sciences  
17th ANNUAL CONFERENCE  
Friday - Sunday, 27-29 July 2007  
Chapman University  
Orange, California, USA**

### **Part 3 Sierpinski Gaskets: The Structure of Boolean Derivatives in TAO and Meta-TAO**

Joel M. Cooper, University of Utah  
Jonathan Butner, University of Utah  
Thomas E. Malloy, University of Utah  
Thomas Smith, University of Utah

*Abstract: Following the ecological epistemology of Gregory Bateson, we have modeled mental process as the flow of differences in a richly connected network. Bateson also suggested that taking differences in this flow of differences would produce higher order knowledge. Finally, Bateson considered symmetry to be the basis for the pattern which connects living forms and, while he never connected taking differences in differences with symmetry, we have done so here. We operationalize the taking of differences in the flow of differences through the XOR operator of symbolic logic. This makes the XOR operator key to our model of mental process. The Sierpinski gasket describes the pattern of the recursive application of the XOR operator in a Boolean flow of differences. Moreover, the recursive application of XOR interacts with the period (length) of attractor cycles in a way that under certain conditions it is self-canceling and under other circumstances wraps upon itself to create complex patterns. Thus Bateson's call to take differences in differences implies, within the dynamic system's perspective of Boolean simulations, an elegantly symmetrical Boolean process whose very symmetry is an ideal tool for discovering symmetry in a world of broken symmetries because the Sierpinski gasket has symmetry in terms of rotation, reflection, translation and magnification. Therefore processing any flow of differences through the gasket will detect and highlight how aspects of the flow of differences diverge from or conform to symmetry. This creates a perfect detection method for broken symmetry. The fact that taking multiple iterations of the difference taking process (TAO) can wrap upon itself now connects Bateson's ecological epistemology to topological insights of chaos theory.*

### Hidden Order

We are using an NK Boolean System (Kauffman, 1993) to formalize Bateson's (2000, 2002) ecological epistemology within a modern nonlinear dynamic systems approach. This formalization has three fundamental premises. First, the map (what humans and other sentient beings know) is not the territory (that which is known); moreover, what gets onto maps from territories are differences in the territories. Second, mental process can be abstracted as a flow of differences in a richly connected network; in Boolean terms, then, James' stream of consciousness becomes a stream of 0's and 1's. This stream of differences in dynamic systems terms can be construed to flow within the constraints of an adaptive landscape consisting of many basins of attraction, each with an attractor and, usually, many tributaries. Third, higher order knowledge emerges in the process of taking differences in the flow of differences (Bateson, 2000, p. 454ff). In the formal Boolean model this process of finding differences in the flow of differences begins with finding discrete derivatives (TAO's) and continues with finding differences among the derivatives themselves (Meta-TAO's). Considered meta-theoretically, this three premise perspective is extremely simple compared to the complexity of many cognitive approaches.

The third premise of the model, that mental phenomena of great interest will emerge from taking differences in the flow of differences (Bateson's thought experiment 2000, p. 463, 464) has led to the definition of the derivative taken on an attractor cycle in a discrete system in terms of the XOR operator and has produced hierarchical perceptual categories (Malloy, Bostic St Clair, & Grinder, 2005). Given our operationalization of Bateson's difference-taking process as XOR, and given Kauffman's arguments that the order which emerges is a function of the relations that define a system, we now turn our attention to the nature of the XOR operator. XOR is a key function in our simple model and as such we want to describe its operating characteristics and how those characteristics contribute the flow of mental process as we have modeled it. What is the hidden order implied by Bateson's call to take differences in the flow of differences? And what does this hidden order have to do with knowing in a world of symmetry and broken symmetry?

**Methodological comments.** The TAO-0 (attractor cycle) matrices used in Part 3 are arbitrary. By arbitrary we mean that they were not simulated Boolean systems generated by E42. We are concerned here in Part 3 with the properties of the XOR operator within the context of matrix algebra. Thus we start simply with a matrix that has convenient properties for our analyses and then use a spread sheet to produce the various transforms (TAO, Meta-TAO). When Boolean systems are simulated in E42 the emergent attractor cycles are described by  $N \times L$  matrices, where  $N$  is the number of nodes and  $L$  is the length of the attractor cycle. Nodes are the rows (vertical axis) of the matrix and iterations (time) are the columns. In Boolean math the cells of these matrices contain either a 0 or a 1; in our figures we typically have converted a 0 cell to a white cell and a 1 cell to a black cell so that the pattern of the attractor cycle is apparent for qualitative analysis. Once we have a  $N \times L$  attractor matrix, we use the XOR operator (see Part 1) to generate derivatives of the dynamics of the attractor cycles which we call the TAO functions; thus the derivatives are labeled as TAO-1 (first derivative), TAO-2 (second derivative), and so on. At times we refer to the original attractor cycle as the zero order derivative (TAO-0). All the TAO's are also  $N \times L$  matrices filled with 0's and 1's that for visual inspection we often convert to white and black cells. In short Boolean simulations produce  $N \times L$  matrices that describe the dynamic behavior of the system's attractor cycles. But the behavior and thus the matrices of a particular randomly generated or even engineered system is emergent and it is impossible to predict the details of the emergent landscape

### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

of simulated Boolean systems. Thus, in Part 3 because we want to examine matrices with certain properties and these properties may or may not show up in a simulation, we simply use arbitrary matrices that are convenient for exploring the nature of the XOR operator acting on matrices in various ways. Matrix algebra assures that the analyses transfer back to matrices generated by E42.

In Part 3 we will be particularly interested in row vectors selected from the  $N \times L$  matrices. For example, look at Figure 3.1 and at the  $L=4$  set of  $5 \times 4$  matrices running from the TAO-0 matrix to the TAO-6 matrix along the top of the figure. The vertical dimension of these matrices is individual nodes and the horizontal dimension is time (iterations). Thus each node has its own row vector,  $\underline{n}$ , which expresses the change in its states (0 vs 1) across time as the system cycles through its attractor. For the Node 1 in the TAO-0 matrix of the  $L=4$  system,  $\underline{n(0)} = \{1011\}$ . The same node has a corresponding vector in the TAO-1 matrix, which is  $\underline{n(1)} = \{1100\}$ , and so on across TAO levels up to TAO-6. In this paper we will focus on how a particular node's vectors change across TAO levels. [Technically, we could specify these node vectors to indicate  $\underline{n(1,3)} = \{1111\}$  means the vector for Node 1 in the TAO-3 matrix. We won't normally use this degree of specificity because we will focus on a single node in our discussions so that it will be obvious what  $\underline{n}$  refers to.]

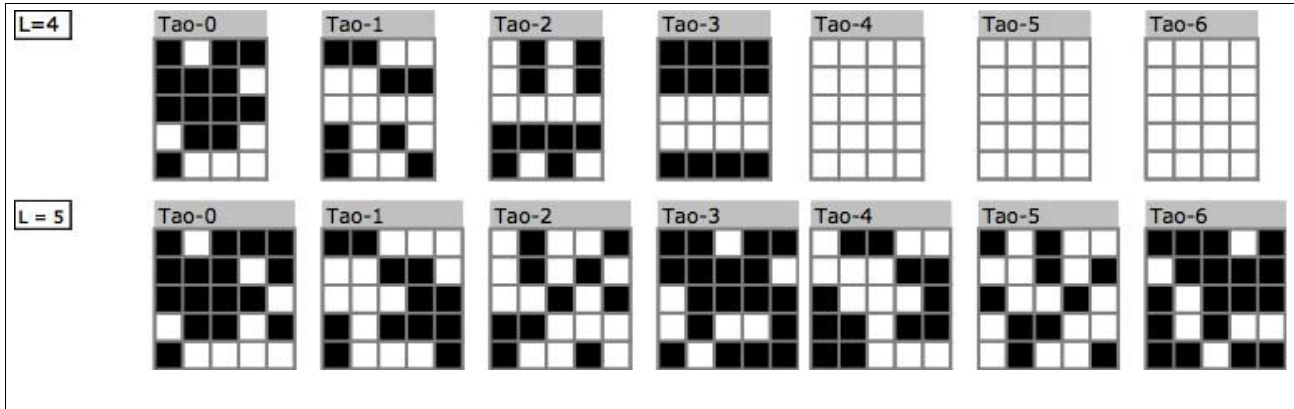
If it is not clear to the reader how, in Figure 3.1, the node vectors change from the TAO-0 matrix to the TAO-1 matrix to the TAO-2 matrix and so on, please review Part 1. In short, the TAO function in E42 can be used to find the difference in each node of a Boolean system generated in E42 as that node changes across iterations in an attractor cycle; it does so by applying the XOR function to that node's state across moments in time.

Recall that, as defined in Part 2, Meta-TAO is the XOR comparison, cell by cell, of any two matrices. Typically we use the Meta-TAO analysis to compare the original attractor cycle matrix (TAO-0) with one of its derivatives. So, when we compare an attractor matrix with one of its derivative matrices, Meta-TAO-1 is the cell by cell XOR comparison of TAO-0 and TAO-1 (that is, of an attractor and its first derivative), and Meta-TAO-2 is the XOR of an attractor matrix with its second derivative matrix. Meta-TAO in these cases indicates, for every cell, if the attractor matrix is the the same (0) or different (1) than a given derivative matrix. At the end of Part 3 we will use the Meta-TAO tool to compare matrices in a more general way. Finally, rather than solely using black and white squares to indicate node states, examples will use 1's and 0's or BLACK and WHITE interchangeably to represent the states of nodes.

**Patterns in the Recursive Application of TAO.** As previously demonstrated (Figure 1.6, Part 1), attractor matrices that are a length,  $L$ , that is a power of 2 resolve to a  $\mathbf{0}$  matrix after repeated applications of the TAO function. However attractor cycles whose length is not a power of 2 fail to resolve through recursive Tao application. Figure 3.1 demonstrates the effect of recursive TAO on both a power-of-two and a non-power-of-two attractor cycle.

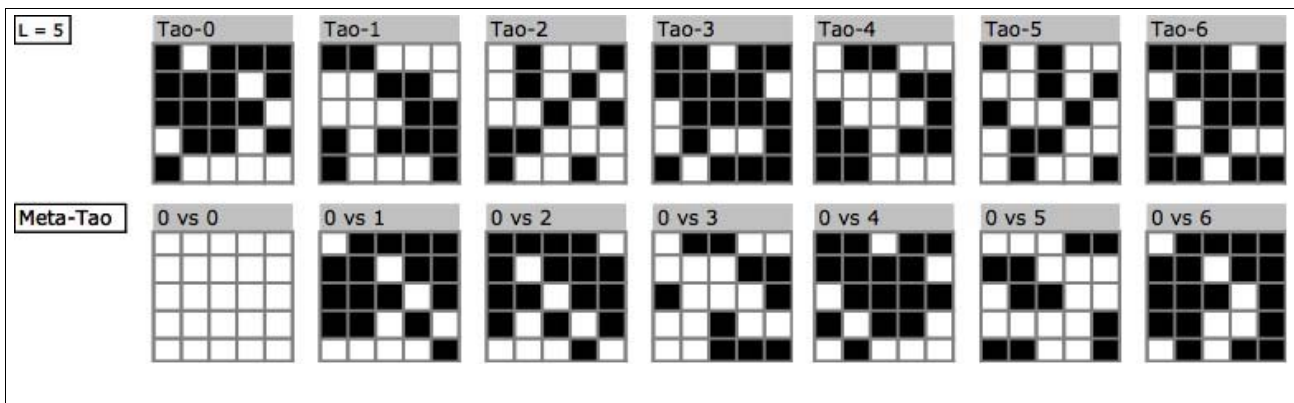
The bottom row of Figure 3.1 shows that, while not resolving to zero, the recursive application of TAO a to non-power-2 attractor cycle results in a cyclic repetition of TAO patterns. In this case, the attractor cycle of basin length 5 enters a TAO-loop at TAO-1 which is out of phase with but otherwise identical to TAO-4. Since the matrices in this figure have not been rotated, we recommend that the reader rotate the matrices mentally; this is most easily done by noticing, for example, that if you slide the top row of TAO-1 over one cell (iteration) to the right it will be identical to the top row of TAO-4, and so on down all the rows of TAO-1. This procedure will indicate that TAO-2 is identical (when shifted in phase to the right one iteration) to TAO-5 and that TAO-3 is identical to TAO-6. Note that TAO-7 (not shown) would be a repetition of TAO-4, and so on, ad infinitum.

Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives



**Figure 3.1.** In the matrices show in this paper, the white cells represent 0 in Boolean algebra and black cells represent 1. In this figure TAO matrices have not been rotated to begin with the lowest Boolean value. **Top Row:** Recursive application of the TAO function (discrete derivative) to an attractor cycle whose length (period) is a power of 2 diminishes to the  $\mathbf{0}$  matrix. **Bottom Row:** Recursive application of the TAO function does not diminish to a  $\mathbf{0}$  matrix but rather settles into a repeating set of TAO matrices (or TAO patterns). In this case the TAO patterns loop every third application of TAO (assuming you adjust them for phase).

We can start with the dynamics of an attractor cycle and take the first, second, third, fourth, etc. derivatives in a sequence for as long as we want. This is a recursive process in that the output of one derivative is the input for the next derivative. If the cycle length of the attractor is a power of 2 then the derivatives will diminish to  $\mathbf{0}$ . But for attractor cycle lengths that are not powers of 2, the patterns of 0's and 1's found in the derivative matrices will not diminish to  $\mathbf{0}$ . Indeed these patterns will repeat previous derivative patterns in a looping sequence (except that they are out of phase with the previous derivative). We refer to these descriptively as derivative loops and examine how they come about later in this paper. In Figure 3.1 (bottom row) the derivatives loop every third derivative (TAO-1 = TAO-4, TAO-2 = TAO-5, and so on).



**Figure 3.2.** TAO's and Meta-TAO's for an arbitrary attractor cycle (TAO-0) of length  $L=5$ .

Since Meta-TAO, as we are using it here, is a joint function of TAO-0 (original attractor cycle) and some other—higher order—derivative, we expect that there will be some repetitive

### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

pattern in the sequence of Meta-TAO's since there is a sequence in the TAO's. Notice in the bottom row of Figure 3.2 that Meta-TAO-1 is identical to TAO-0 but rotated one iteration to the right. Meta-TAO-2 is identical to TAO-0 but rotated two iterations to the right. (We skip Meta-TAO-3 for a moment.) Meta-TAO-4 is identical to TAO-0 except it is rotated four iterations to the right (or two to the left). In contrast, no matter how the matrices are rotated, Meta-TAO's 3 and 5 are not identical to the original attractor. This is a similar result to the one found with the pure XOR Ring in Part 2. For convenience we repeat a point made in Part 2 since it applies here. In our current example as well as in the pure XOR Ring used in Part 2, Meta-TAO's 1, 2, and 4 are identical to the original basin *when they are rotated*. In terms of matrix algebra these rotated Meta-TAO's can be generated by acting on the original attractor with the Identity operator. For example:

$$[\mathbf{TAO-0}] \times \mathbf{I} = [\mathbf{Meta-TAO-1}].$$

Thus we can call them *identity* Meta-TAO's.

But Meta-TAO's 3 and 5 are different than the original attractor; a simple descriptive term for these is non-identity Meta-TAO's because whatever transformation matrix  $\mathbf{T}$  is applied to TAO-0 is certainly not the identity matrix. In terms of matrix algebra and using Meta-TAO-3 as an example, we can express this non-identity as:

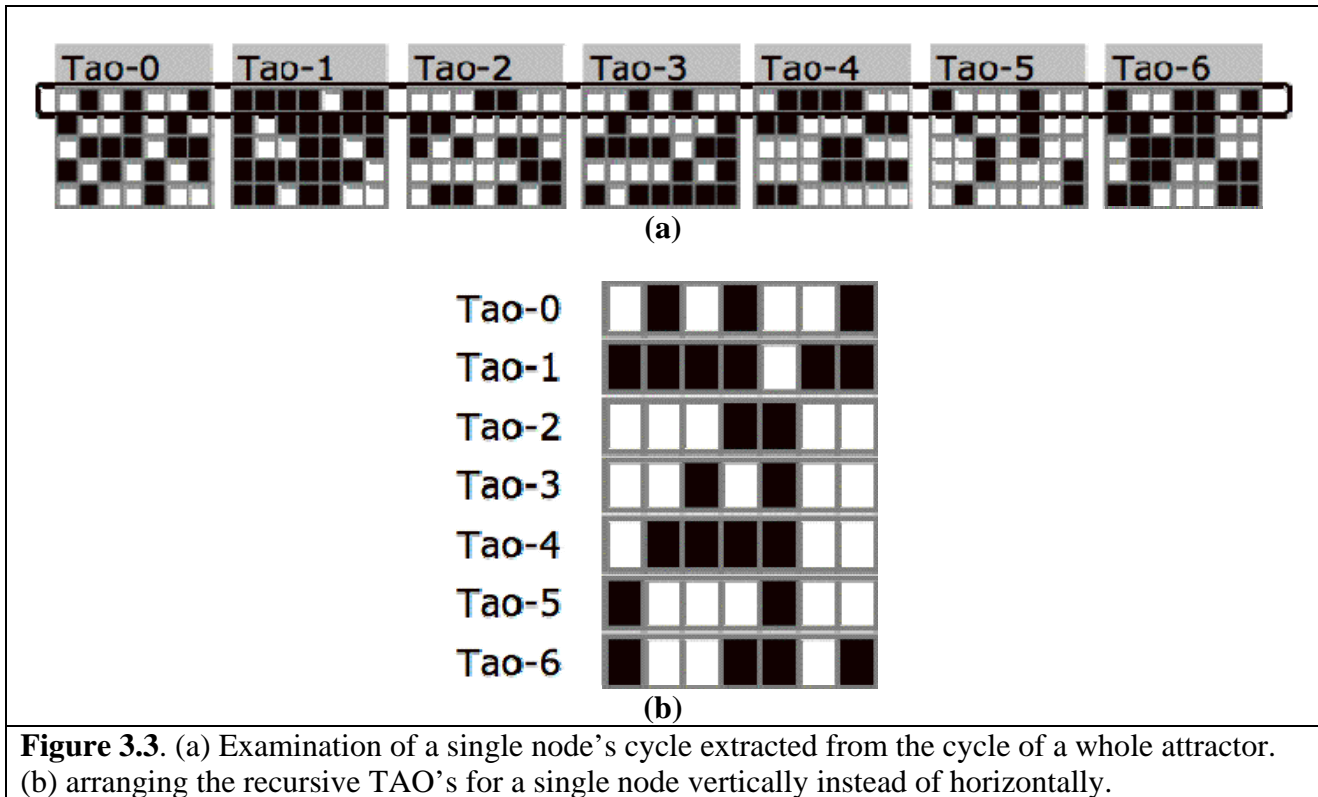
$$[\mathbf{TAO-0}] \times \mathbf{T} = [\mathbf{Meta-TAO-3}].$$

A primary focus of this symposium is on the nature of  $\mathbf{T}$ , on the relation of  $\mathbf{T}$  to symmetry, and on how  $\mathbf{T}$  underlies how knowledge begets knowledge.

As a summary up to this point, we have found a hidden order in the dynamics of attractor cycles. First, derivatives of attractors whose cycle length is a power of 2 diminish to  $\mathbf{0}$  but not so for attractors whose length is not a power of 2. Second, both TAO's and Meta-TAO's have repeating sequences. But the Meta-TAO sequence is not the same as the TAO sequence. In our example in Figure 3.1 the TAO sequence is that 1=4, 2=5, 3=6, and so on (assuming we rotate the matrices). But in Figure 3.2 for the identity Meta-TAO's in the lower row (when rotated) the sequence of equivalent Meta-TAO's is 1=2=4. Moreover, for non-identity Meta-TAO's, there is a repeating sequence which is not shown in Figure 3.2 because the figure does not show enough higher order Meta-TAO's.

Both the sequence of TAO's and the sequence of Meta-TAO's have interesting patterns which are puzzles calling for solutions. Our focus here is on “knowing begetting knowing” in symmetry groups so we will not fully solve these puzzles. But we note them and we turn our focus to the nature of the patterns generated by the XOR operator because it is crucial to our definition of symmetry groups (since XOR is the basis of both the TAO's and Meta-TAO's that define symmetry groups) and because the functional properties of the XOR operator is related to the interesting sequences of TAO's and Meta-TAO's we've just described.

Recall that XOR is not an arbitrary operator from an epistemological viewpoint; it describes formally Bateson's principle that knowledge is generated by finding differences in differences (2002, p. 454ff, particularly pp. 463, 464). So XOR is a theoretically driven operator; moreover, as we have examined recursive and repeated applications of XOR, we have found that it has interesting properties in and of itself. Indeed, work by Wolfram (2002) in cellular automata suggests that simple logical operators can result in complex and surprising patterns. Specifically Guy (1990), Schroeder (1991), and Wolfram (2002) all point out that recursive use of the XOR operator can be used to create the Sierpinski gasket/Pascal's triangle; this suggests that perhaps the key to the strange and perplexing behavior from both the TAO and Meta-TAO functions may be found in understanding the emergent structure of the XOR Boolean operator as it relates to the Sierpinski gasket.



**Figure 3.3.** (a) Examination of a single node's cycle extracted from the cycle of a whole attractor. (b) arranging the recursive TAO's for a single node vertically instead of horizontally.

### The Affine Boolean Sierpinski Gasket

In order to understand the driving structure of the TAO and Meta-TAO functions a number of simplifications will be made. Figure 3.3a shows the effect of applying TAO to an arbitrary (not generated by E42), 5 node, attractor cycle of length 7. Notice how recursive applications of TAO fail to resolve any of the row vectors,  $\underline{n}$ , to a  $\underline{0}$  vector. Note that Node 1 is outlined in Figure 3.3a and its pattern persists indefinitely but is shown only up through TAO 6.

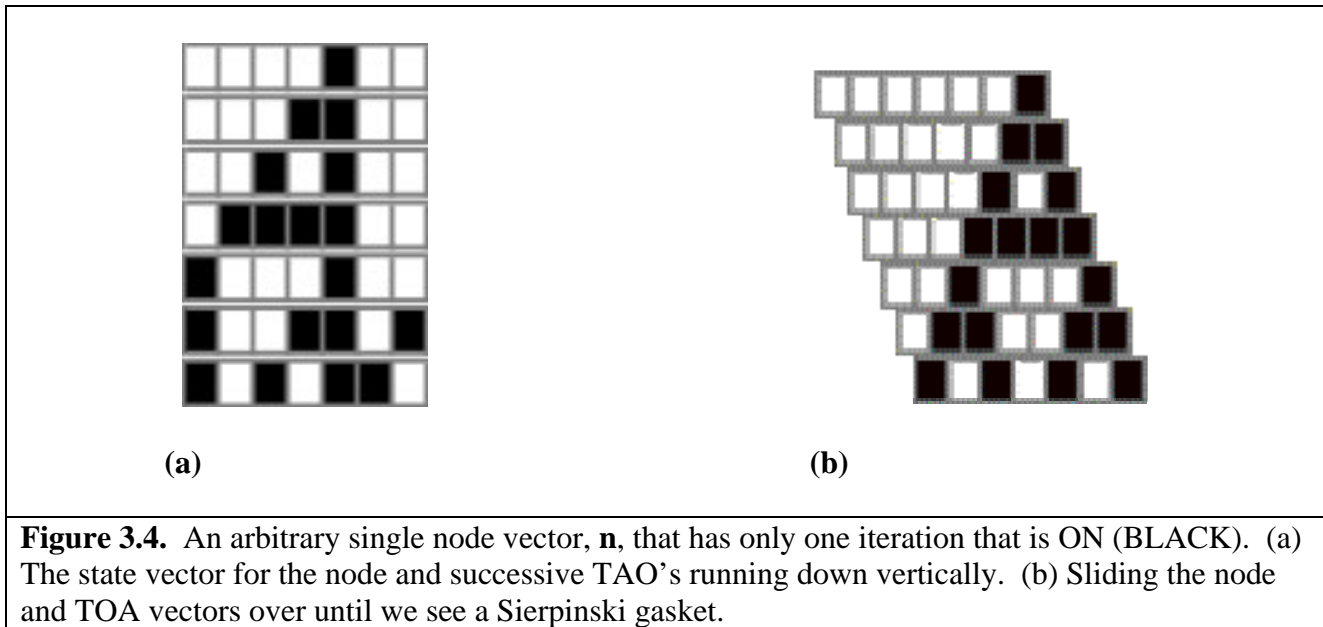
We now will look at this single node as it turns ON (BLACK) and OFF (WHITE) while cycling through an attractor. In order to visualize the effect of TAO on  $\underline{n}$  (a single node cycle) Figure 3.3b vertically stacks recursive applications of TAO onto a single Node (Node 1) from Figure 3.3a. This is a change of orientation from our previous representations; now TAO levels descend vertically and only a single node is examined. This change of orientation is done to visualize more easily the effect of TAO on a single node as it relates to the Sierpinski gasket.

Because Figure 3.3b remains visually complex we will use an even simpler single node example. Figure 3.4a is a single node that has a cycle length seven and that it is ON during iteration five and OFF during all other iterations. Now we recursively take TAO of the cycle for that node (See Part 1 for a detailed procedure of the functional analysis). The first row of Figure 3.4a is the original node cycle and subsequent rows represent the effect of recursively applying TAO to the row above. Figure 3.4b represents the same configuration as 3.4a but aligns the TAO applications to reveal an identifiable underlying structure generated by recursive TAO for a single BLACK state in a cycle of  $L=7$ . This realignment between Figures 3.4a and 3.4b may seem arbitrary but it is not. The nodes have been shifted from a rectangular array of TAO matrices to the triangular array of the Pascal triangle (see discussion below). As suggested by the work of Wolfram, the recursive use of

### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

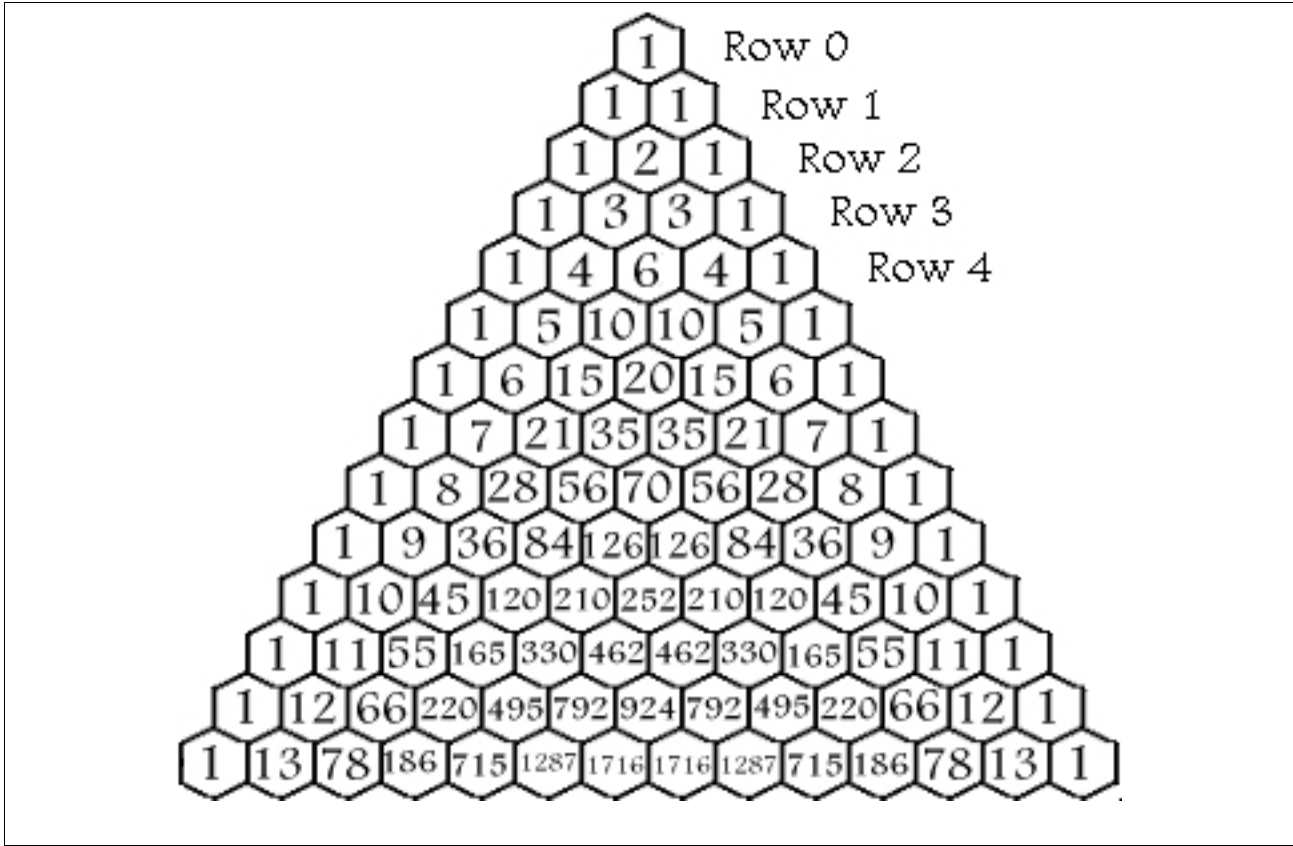
TAO imposes the structure of an affine Sierpinski gasket on individual node attractor cycles generated within E42. This is less evident when the TAO applications are aligned in the rectangle of the TAO matrix (as in Figures 3.3b and 3.4a) but clearly seen when slightly shifted to be aligned as a triangle (Figure 3.4b).

Both Figures 3.4.a and 3.4.b represent the simplest case: A single node that has only one iteration ON and therefore BLACK during the system's attractor cycle. To begin our analysis we will examine what happens to a node row vector ( $\underline{n}$ ) when one cell is BLACK and all other cells in the vector are WHITE. After that analysis, we will layout the logic of analyzing cases where a single node vector has multiple ON states.



Starting with a single ON (or BLACK) cell in a row vector and with these simplifications in mind, at this point it might be worthwhile to point out the similarity between the operations used to generate the triangular form shown in Figure 3.4b and those used to generate Pascal's triangle (see for example, Schroeder 1991, p. 387ff). At the tip of Pascal's Triangle is the number 1, which makes up the first row. The second row contains a 1 and a 1 to the right and left of the 1 on the first row. Each of these numbers are generated by adding the numbers above them to the right and to the left. In the case of the two ones on the second row, they are generated by adding 0 and 1 for the first, 1 and 0 for the second. This process is continued for all lower rows and can go on infinitely (See figure 3.5). As Schroeder (1991, p. 388) shows, Pascal's triangle becomes the Sierpinski gasket with all even numbers replaced by 0 (WHITE) and all odd numbers replaced by 1 (BLACK).

Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

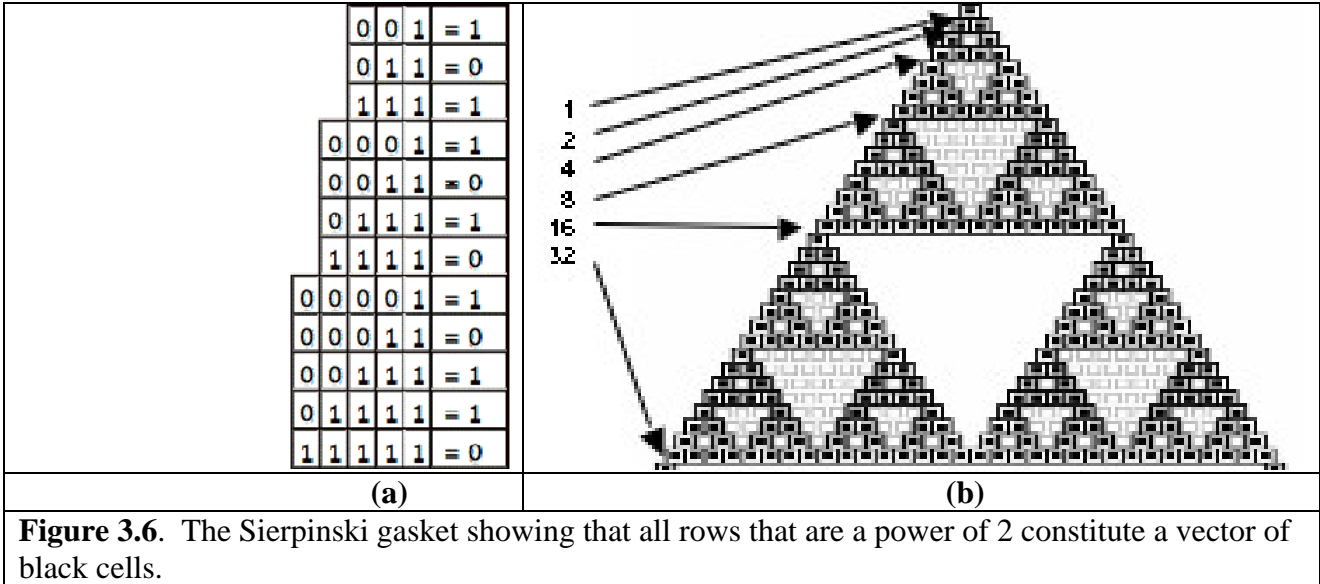


**Figure 3.5.** The Pascal Triangle.

As shown in Figure 3.6, the Sierpinski gasket follows a similar logic to the Pascal triangle, except that all Pascal summations that result in an even number are coded as a 0, and all odd summations are coded as a one. It is important to note that all base 10 sums that are even in Pascal’s triangle are replaced by white cells (0) and all base 10 sums that are odd are replaced by black cells (1) in the Sierpinski gasket. Finally, suppose we construct Pascal’s triangle in base 2; and suppose we replace the value of any Boolean sum with a 0 if the sum is odd in base 10 and with a 1 if the sum is odd in base 10. This also will generated the Sierpinski gasket (with 0’s and 1’s converted to white and black).

**Even Sums = 0; Odd Sums = 1.** Let us examine this odd/even summation logic more closely in terms of the XOR operator. Suppose we examine the second row (011) from the table in Figure 3.6a. We sequentially apply XOR to this string of binary digits; by this we mean we XOR one digit to the next and then XOR of the result with the next digit). In the specific case of {011} this means [(0 XOR 1) XOR 1]. In words, if we XOR the first two values (which are 0 and 1) we get a 1 because they are different. If take that result and XOR it with the last number (which is 1) we get 0 because they are the same (both 1’s). This sequential XOR will always generate a 0 if the sum (in base 10) of those digits is even and a 1 if the sum (in base 10) is odd, regardless of order of the 1’s in the sequence. Later, when talk about overlapping gaskets, the fundamental calculation to determine the resulting value of overlapping Sierpinski gaskets is this odd/even summation rule.

Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives



**Figure 3.6.** The Sierpinski gasket showing that all rows that are a power of 2 constitute a vector of black cells.

Finally, note in the Sierpinski gasket in Figure 3.6 that, counting down from the top, those rows that are powers of 2 are all black; this is related to the basis of the solution of the puzzle that the derivatives (which are based on XOR) that are powers of two resolve to the **0** matrix. But the solution, in fact, is more related to symmetry theory and the symmetrical characteristics of the Sierpinski gasket. Particularly notice that, after the apex row and looking only at black cells, the gasket is perfectly symmetrical around its medial vertical axis. The gasket has only white cells where this axis passes through it except for rows whose number is a power of two. For the power of 2 rows the length of the row is an even number and the central axis passes between two black cells. This symmetry will be important when, later, we wrap the gasket around a manifold.

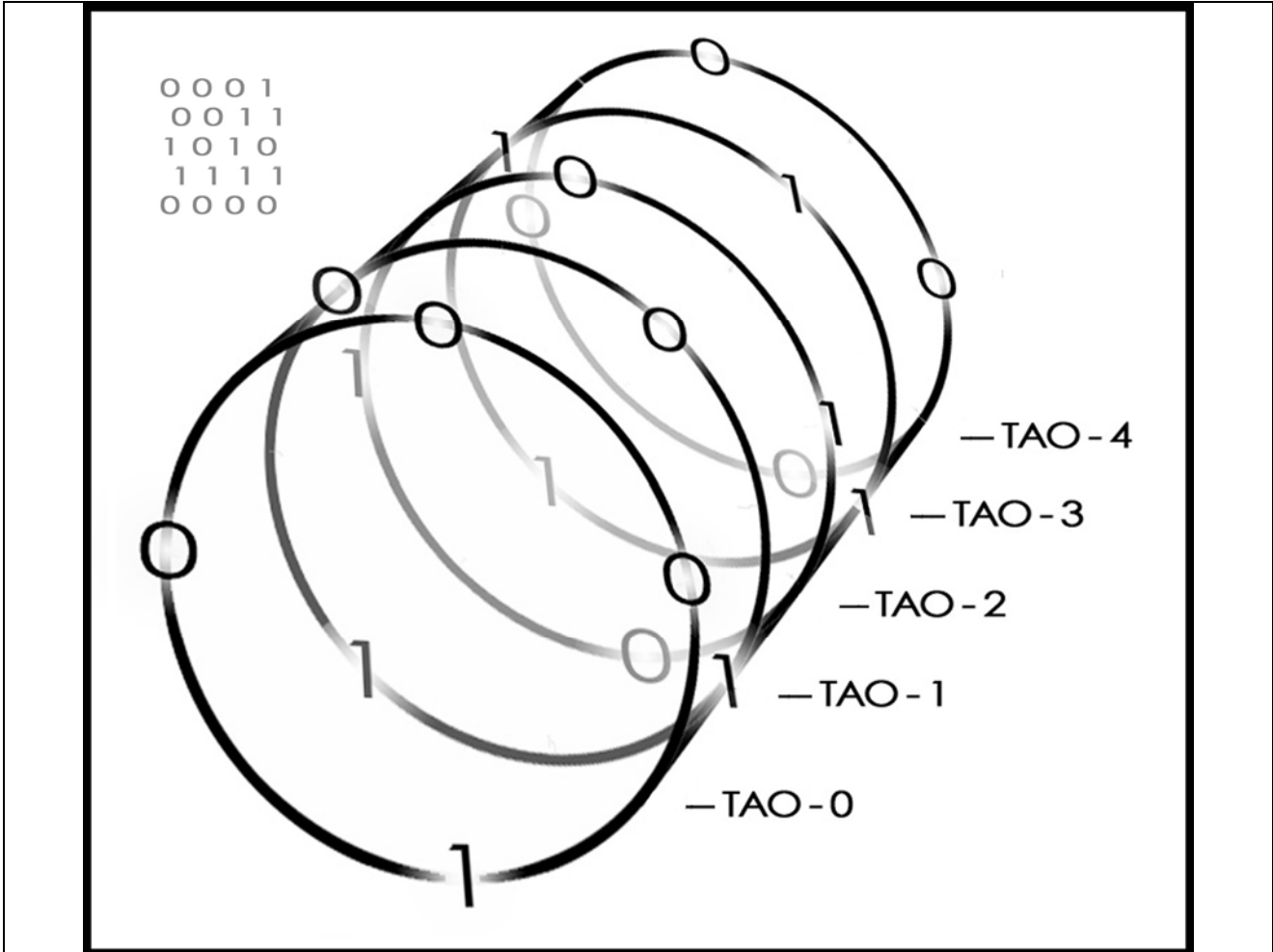
**Attractor Cycle Length Wraps a Gasket onto Itself**

For clarification we distinguish between L, which is the length (in iterations) of the attractor cycle of the whole complex of nodes that make up a system, and the sub-L, which is the length of the cycle of a particular node. Sub-L can and often does equal L but it need not. For example, a whole system may have L=12 but a particular node may alternate ON and OFF so that its sub-L=2. In this case Sub-L is a power of 2 but L is not a power of 2 (we consider sub-L=1 to be a power of the 2 since 2<sup>0</sup>=1). In this section we examine single nodes so we are always addressing sub-L. Figure 1.6, in Part 1, shows examples of a system in which the nodes have differing sub-L's and that some of these sub-L's are powers of 2 and resolve to **0**.

Suppose we have a node vector representing the flow nodes states for that particular node in a Boolean system. Suppose **n** = {00100010001000100010...}. If the system is running and there are no perturbations the vector is infinitely long and we notice it repeats the sequence {0010} over and over. Thus the node's states (0 or 1) are cycling with a period that is sub-L=4 iterations. Cycles, of course, can be represented as circles rather than repeating strings. Figure 3.8 shows the node's cycle, {0010}, indicated as TAO-0 and drawn as the circle on the front of the cylinder. It also shows the derivatives as circles; thus we can imagine the recursive TAO process a cylindrical manifold, except that in the discrete case it is a series of sections of a cylinder. To correspond to the geometry of Pascal's triangle as well as the Sierpinski gasket, the TAO vectors are not lined up but slightly offset (as in Figure 3.4b and as in the pale grey vectors in the upper left hand corner of

Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

Figure 3.8). Thus TAO-1 is rotated 45 degrees clockwise from TAO-0 and TAO-2 is rotated back 45 degrees counterclockwise from TAO1 (and so on in alternating fashion for higher-order TAO's). This makes the figure more perceptually confusing but easier to connect to analyses in other figures.



**Figure 3.8.** Recursive TAO for a sub-L=4 node represented as discrete sections from a cylindrical manifold. The staggered light grey vectors in the upper left remind the reader that the cylinder is visualized with the triangular logic of Pascal's triangle not the rectangular logic of a TAO matrix so TAO-1 is rotated 45 degrees clockwise from TAO-0 and TAO-2 is rotated back 45 degrees counterclockwise from TAO1 (and so on in alternating fashion).

Figure 3.9a shows the recursive TAO vectors up to TAO-18 for this sub-L=4 cycle in a different form than in Figure 3.8. The TAO's, of course, go to 0 vectors since sub-L is a power of 2. But how does this come about? Figure 3.9b shows the Sierpinski gasket flowing out from the single ON node. Because we start with TAO-0, the TAO numbers in 3.9a do not correspond to the row numbers in the Sierpinski gasket shown in 3.9b. TAO-0 corresponds to row 1 and this means that TAO-3 corresponds to row 4.

Notice that, due to the constraint of a cycle length of L=4, the rows of the gasket wrap around the sections of the manifold which were visualized in Figure 3.9. The vertical lines in Figure 3.9b are four columns apart (on the gasket) and represent where the constraint of sub-L=4





### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

**TAO Loops, Meta-TAO Loops, and the Sierpinski Gasket.** The symmetry of the Sierpinski gasket and the broken symmetry of the wrapped gasket when sub-L is not equal to a power of 2 can also help explain some of the hidden patterns generated using the TAO and Meta-TAO functions of E42. We will only sketch the outline of these patterns and how they are accounted for by the wrapping of the Sierpinski gasket. Notice that, with the sub-L=4 example in Figure 3.9a, no TAO repeats a previous TAO; rather, the TAO's resolve to 1 and then 0 without any repetitions. In contrast, in Figure 3.10a the TAO vectors do repeat. For the moment, notice that TAO-6 is a rotated repetition of TAO-3. Specifically, if the TAO-3 vector, {11110}, is rotated three iterations clockwise it becomes {11011}, which is the TAO-6 vector. And TAO-6, if rotated three iterations clockwise, is equal to TAO-9. The idea is that equivalent TAO vectors are repeating every third recursive application of TAO; thus the sequence TAO-3, TAO-6, TAO-9, TAO-12, TAO-15, TAO-18... are all either equal or rotated equivalents. Moreover because of the phase relations between the wrapping Sierpinski gasket and the sub-cycle length there are only two distinct vectors, either {1110} or {11011}, in the TAO 3, 6, 9, 12, ... sequence. A complete description of how these phase relations work is yet to be done and is only hinted at here.

Notice also, that there is a TAO-2, TAO-5, TAO-8,... sequence and a TAO-4, TAO-7, TAO-10... sequence. These three sequences form a plenum which covers all the integer values for TAO levels. This means, first, that there can be no other TAO's than those described here and, second, that TAO levels loop in some kind of meta-cycle that both describes and limits the higher-order differences that the original cycle is capable of generating. This suggests that there is some limitation on the degree of complexity that can be extracted from the dynamics of the original attractor cycle by taking differences in differences. We've made these arguments from a vector generated by a single node extracted from a system. But together, all the node vectors for a system comprise the TAO matrices for the system, and therefore the conclusions apply not just to the vectors for a single node but to the TAO matrices for the whole system. As a technical note, the rows (node vectors) of a TAO matrix are independent of each other in the following way. First, note that nodes, while the system is running, are dependent upon each other and the value of one in the general case influences the value of another. But once the system is in an attractor cycle we can describe the dynamics of the cycle with a TAO-0 matrix; at this point all recursive TAO applications are for individual row vectors and so the effects of the TAO operator generalize to from a single node (row) vector to all others.

There is one more notable feature from the simple example in Figure 3.10a. Since the wrapping of the gasket can take several XOR operations before generating an overlap, it is plausible for there to be transient TAO's early on in the recursive sequence of derivatives (TAO's) such that they are not a part of the TAO loops but precede them. Notice, to the right, in Figure 3.10b that the first wrap occurs at row 6 of the gasket (which corresponds to TAO-5); this is the point at which the full symmetry of the gasket is broken because the length of the gasket row wraps in a non-symmetric way around the circumference of the manifold. This is the point that defines the beginning of the looping structure of the TAO's in the following way. Notice that it is the wrapping of the gasket onto itself at row 6 that transforms TAO-5 into an equivalent to the TAO-2 vector. Thus the TAO loop must start at TAO-2; it cannot start earlier because the row lengths of the gasket are not long enough to wrap around the manifold further than a single circumference and so they cannot create composite sums in TAO. Consequently, TAO-1 will not be part of any loop. TAO-1 is, as it were, a transient TAO leading into the TAO loops described above. Our conjecture is that the structure of TAO loops and TAO transients will be describable by the topology of the wrapping of Sierpinski gaskets around discrete sections of a cylindrical manifold defined by sub-L. There is a

### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

caveat. We have not fully solved that topological puzzle. Across many examples with constraints of  $L=3$ ,  $L=5$ ,  $L=6$ ,  $L=7$ , and so on, we have found the repetitive looping patterns of TAO for  $L$  and sub- $L$  not equal to a power of 2 to be varied and complex. Describing these patterns in any way that has a hint of elegance is work still to be done.

**Meta-TAO.** The symmetry the Sierpinski gasket and the broken symmetry of the wrapped gasket can also help explain some of the hidden patterns generated using the Meta-TAO function of E42. As a reminder, whereas TAO is an XOR of a node (or set of nodes) with its own states through time, Meta-TAO is an XOR of any two matrices which takes the XOR between each corresponding cell of the two matrices. Recall that the dynamics of an attractor cycle are archived as the TAO-0 matrix. Most commonly Meta-TAO is a matrix operation applied to TAO-0 and a higher-order TAO's (e.g. 0 vs. 1, 0 vs. 2). Limiting ourselves to this case where Meta-TAO compares an attractor with its derivatives, we can examine what happens to the Meta-TAO in the single node examples shown in Figures 3.9 and 3.10.

Since Figures 3.9 and 3.10 are single node examples, for this discussion we reduce the Meta-TAO operation from a matrix operation to a vector operation. But, of course, a vector is a matrix and we can generalize back from the single node vector case to multiple-node Meta-TAO matrix operations when we are done. For example, in these single node cases, Meta-TAO 0 vs. 1 consists of an XOR of the TAO-0 vector with the TAO-1 vector and Meta-TAO-2 consists of an XOR of the TAO-0 vector with the TAO-2 vectors, and so forth. For the two examples above, these vectors are shown in Figures 3.9a or 3.10a. In the power of two case (Figure 3.9a), there are a limited number of unique TAO vectors before they become 0. Therefore there must also be a limited number of unique Meta-TAO's (possible unique XOR functions we could generate between TAO-0 and higher-order TAO's). In the non-power of two case (Figure 3.10a), there are also a limited number of unique TAO's derived from the original attractor; these form the TAO transients and the TAO loop. Therefore there must also be a limited number of unique Meta-TAO's. It is possible to show that this unique set is quite small, merely consisting of various rotations on a limited set of matrices and that these sets of Meta-TAO's also form loops. Moreover it should be clear from Part 2 of this symposium that the Meta-TAO loops are not the same as the TAO loops (see, for example, Figure 2.4 in Part 2). However, a full description of the Meta-TAO patterns is beyond the scope of this paper.

We have presented the effects of recursively applying TAO to the cyclic pattern of a single ON node; this recursive application of TAO turns out to have the form of the Sierpinski gasket. But, in general, a node vector in a Boolean system can have more than one iteration during which its state is ON. What we have shown up to this point (the wrapping of the Sierpinski gasket around the manifold sections defined by sub- $L$  and the summing rule for generating 0's and 1's for those cells where the gasket does wrap upon itself) can be generalized to the case of multiple ON states. However note that, with multiple ON states, there are two distinct sources of overlapping we can talk about. In the previous paragraphs we talked about how a single gasket created by a single ON state in a node vector wrapped around the manifold defined by the recursive TAO process; in wrapping around the manifold it overlapped itself. When there are two or more ON states, each ON state will generate its own gasket which will, even before its rows are long enough to wrap around the manifold, overlap the gaskets generated by the other ON states. The same summation rule will apply to these overlaps. But, at some level of TAO, all these gaskets will start wrapping around the manifold, possibly multiple times. So, first the gaskets will overlap their neighbors' gaskets. Then the gaskets will wrap around the manifold and create another level of overlap. Computationally these textured layers of overlaps, while tedious, are straightforward and worked examples have not

### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

disconfirmed the proposed topological wrapping conjecture presented here. Describing this process simply, as well as describing how this textured layering generates TAO and Meta-TAO loops is work yet to be done.

#### Summary

The XOR operator is the engine of Boolean discrete derivatives (TAO's). We, among others, have shown that the XOR operator, and therefore recursive TAO derivative operation, is imbued with the symmetry of the Sierpinski gasket. By extension this Sierpinski symmetry is related to Meta-TAO's and their ability to detect distant basins of attraction in a landscape solely from information found in a local basin of attraction (see Parts 2 and 4 of this symposium). When extent is unlimited (Wolfram's Rule 90) the Sierpinski gasket spreads forever. But the gasket, as expressed in the attractor cycles of a Boolean landscape and in the discrete derivatives of those attractors, is constrained within the period of an attractor cycle; thus the gasket cannot extend forever—it must encounter the constraints set by the period of the attractor cycle. How we can formalize how this constraint affects the patterns that emerge from the recursive application of TAO? The attractor cycle and the recursive application of TAO to that cycle can be described as sections from a cylindrical manifold. The constraints of the cycle come in the form of the discrete circumference of those manifold sections. Specifically that circumference is the period (or length) of the cycle in iterations. As the gaskets spreads out (over repeated applications of TAO) its row lengths increase. When gasket's row length exceeds the manifold circumference the gasket wraps over itself. Two cases result. When the attractor's period (in iterations) is a power of 2 the wrapping will be perfectly symmetrical and eventually produce a perfect cancellation of Boolean values and the gasket appears to disappear so that TAO becomes **0**. When the attractor's period is not a power of 2, the wrapping breaks the symmetry of the gasket and produces complex composite waves of pattern resulting from the overlap. These wave patterns repeat themselves in what we call TAO loops and Meta-TAO loops. In either case (power of 2 or not), there is a limited number of unique derivatives which implies that there is a limit on the degree of complexity that can be extracted from the dynamics of an attractor cycle by taking differences in differences.

Returning to epistemology, what we have shown is that Bateson's call to take differences in differences implies, within the dynamic system perspective of Boolean simulations, an elegantly symmetrical process whose very symmetry is an ideal tool for discovering symmetry in a world of broken symmetries. The Sierpinski gasket has symmetry in terms of rotation, reflection, translation and magnification. Moreover, the affine Boolean version of the gasket defined by the recursive application of the XOR operator is itself a structured flow of differences and so is a functional filtering process for differences. Therefore processing any flow of differences through the gasket, and therefore its symmetry, will detect and highlight how aspects of the flow of differences diverge from or conform to symmetry, in particular the symmetry of differences. The gasket creates a perfect detection method for broken symmetry in a flow of differences, providing a basis for understanding how the various uses of XOR in TAO and Meta-TAO in Part 2 are transforms that define symmetry groups in a Boolean landscape. The mapping of this Boolean landscape to a mental landscape fits exactly with Bateson's proposal that mental process could be considered abstractly as a flow of differences in a complex network. The fact that recursive applications of the difference-taking process (TAO) can wrap back onto itself now connects Bateson's version of ecological epistemology to the topological insights of chaos theory. More importantly, as noted in Part 2, Bateson considered symmetry to be the basis for the pattern which connects living forms

### Part 3: Sierpinski Gaskets: The Structure of Boolean Derivatives

and, while he never connected taking differences in differences with symmetry, we have done so here.