

Meta-TAO: A Phase Portrait for Boolean Systems

Abstract

Phase portraits have been fundamental to analyzing and describing systems in the continuous case. This paper develops a methodology that is an analogue to the phase portrait for the case of discrete Boolean systems. Attractor cycles and their derivatives for NK Boolean landscapes can be coded as  $N \times L$  matrices, where  $N$  is the number of nodes in the system and  $L$  is the number of iterations in the attractor cycle. In classic oscillator theory the phase portrait results from putting the zero-order derivative (position) in relation to the first-order derivative (velocity). In this paper Meta-TAO is a matrix operation that we use to place the zero-order derivative (attractor matrix) in relation to its various derivative matrices. Meta-TAO, acting like the continuous phase portrait, reveals surprising relations among the attractors on a Boolean landscape.

## A Phase Portrait for Boolean Systems

Systems theory has long been synonymous with a velocity flow-field in the form of a phase portrait, phase space, or state space. Going all the way back to Poincaré's solution to the three body problem, these topographical representations of change give insight into the simplicity of some patterns and the complexity of others. Specifically, they generate visually identifiable and measurable properties that define many of the emergent processes implicit in the behavior of systems. Attraction, repulsion, attractor cycles, separatrices, basins, and even Lyapunov exponents all have visual definitions in the flow-field. In the world of continuous systems, where outcomes function on a metric continuum, new tools based on these original ideas continue to be developed (e.g., recurrent quantification analysis is less than ten years old – see Riley & Van Orden, 2005 for a review). For discrete systems where outcomes are metric categories, little that we are aware of has been done to establish any parallels to the velocity flow-field in general and the phase portrait in particular. Our goal is to do exactly this: develop a parallel for the phase portrait for Boolean systems.

Boolean systems where many interacting parts each have only two states (e.g. neural networks and cellular automata) are perhaps the most explored form of discrete system. Interestingly, many, if not all, of the concepts found in continuous systems exist discrete systems. Point attractors, attractor cycles and landscapes with basins attraction emerge from lower order generating processes in Boolean systems (Malloy, Jensen, & Song, 2005). It is based on these parallels that we contrive our conceptual approach to the Boolean domain.

The application of a phase portrait to Boolean systems has an added bonus in that many simulation-based explorations into evolution and landscape searches are conducted

## A Phase Portrait for Boolean Systems

in this domain (e.g., Choi, Jung, & Kim, 2005; Skellett, Cairns, Geard, Tonkes, & Wiles, 2005). Models of genetic algorithms, for instance, systematically perturb parts of the system and recombine them to identify new, potentially improved patterns (Watson & Jansen, 2007). Yet Kauffman has argued that, beyond the search strategy, the nature of the landscape upon which adaptation occurs must also matter and that as of yet we do not have sufficient methodology for knowing the nature of landscapes (1995, p.188). We propose that both the nature of Boolean landscapes and how long leap adaptation take place on landscapes might be deeply informed by a method that yields a phase portrait on Boolean landscapes. Flow fields and phase portraits are closely linked to basins of attraction on adaptive landscapes. Furthermore, flow fields have provided a clarity to systems so far unsurpassed by alternative approaches. We consider the ability to identify emergent properties of the landscape via our discrete parallel to phase portraits as an important methodological step toward finding central characteristics of Boolean landscapes, particularly a step toward the description of new adaptive search strategies.

### **Boolean Derivatives and the Relations Between Derivatives**

At the heart of the velocity flow-field is, of course, velocity which is defined in continuous systems as the first order derivative. Further derivatives can also be defined: acceleration as the second order derivative, jerk as the third, and so on, each being merely the rate of change in the previous derivative. Under this logic position can be defined as the zero order derivative. In contrast to the continuous case, NK Boolean networks (Kauffman, 1993) have N nodes where each node can have only two states, 0 or 1. Malloy, Jensen, and Song (2005) defined an operator that is parallel to continuous derivatives for Boolean networks; this operator is called TAO (change over time) to

## A Phase Portrait for Boolean Systems

distinguish it from the continuous derivative. As we will detail below, TAO simplifies the concept of change to the difference (or lack of difference) between two temporally consecutive binary values. In the same way that continuous derivatives are expandable into higher orders, there are higher order TAOs, making TAO an optimal Boolean parallel for velocity, acceleration, jerk and so on. TAO, like derivatives, is the basis for constructing phase portraits.

In terms of identifying the organizing principles of dynamics found in phase portraits, it is the relationships amongst derivatives that are key (e.g., position versus velocity, or, put another way, zero order derivative versus first order derivative). This is directly represented in a velocity flow-field for a single outcome in continuous systems by plotting one form of derivative against another. In fact, based on Taken's theorem, one need only consider the relationship amongst higher order derivatives to also capture relationships between other variables (Takens, 1981). It is this same principle upon which we define our version of velocity flow-field for Boolean systems; that is, we ask how one can identify the relationship between TAOs of different orders. The simple operator that outputs the relationships between different TAOs is what we call Meta-TAO. We believe the Meta-TAO comparisons between TAOs of different orders will result in a parallel that does not necessarily look like a flow-field but in fact is one under a Boolean logic. Moreover we will show that when the Meta-TAOs are combined together across the stable attractors found within a landscape, they map out the entire landscape, just like a phase portrait does.

### **Developing a Phase Portrait Methodology for NK Boolean Systems**

NK Boolean networks (described in detail by Kauffman 1993, p. 188ff;

## A Phase Portrait for Boolean Systems

Kauffman, 1995, p. 75ff; and Malloy, Jensen, and Song, 2005) consist of an arbitrary number  $N$  of abstract entities called nodes. Since the model is Boolean, the nodes have two states: ON (state = 1) and OFF (state = 0). Thus by state we mean Boolean value. Each node takes input (0 or 1) from  $K$  nodes. Time flows in discrete iterations. On iteration  $T$ , each node, takes the input (0 or 1) it has received from its  $K$  input nodes and uses a logical truth table to determine what its own value will be (either 0 or 1) on the next iteration,  $T+1$ . That is, the nodes are coupled; the output of one is input to others and visa versa.  $NK$  Boolean networks can thus be thought of as the general case for cellular automata (where the network structure follows a direct spatial representation) and some forms of neural networks (excluding those that allow for continuous underlying values for nodes). Table 1 fully specifies a randomly generated  $N$  (nodes) =4,  $K$  (inputs per node) =2 Boolean system which we will call “System 1.” The four nodes are arbitrarily labeled **A** through **D**. Under each of the Nodes, **A** to **D**, in Table 1 is a sub-table specifying the truth table that the particular node uses to generate its future output from current input. For example, in Table 1 note that Node **A** takes input from Nodes **B** and **C** at time  $T$ ; then, based on the states of **B** and **C** at time  $T$  (leftmost two columns under Node **A**), **A** will either generate a 0 value or a 1 value at time  $T+1$  (third column under Node **A**). To continue the example, if **B**=0 and **C**=1 at time  $T$ , **A**=1 at time  $T+1$ . As explained in greater detail in Malloy, Jensen and Song (2005) the node sub-tables in Table 1 each specify a logical operator that relates a node’s two inputs at time  $T$  and calculates that node’s value at time  $T+1$ . In the case of Node **A** in Table 1 the logical relation is the XOR operator that detects difference. Notice in Table 1 if the two inputs to **A** are the same (both 0 or both 1) at  $T$ , **A** will be 0 at  $T+1$ ; in contrast if the two inputs are

different at  $T$ , **A** will be 1 at  $T+1$ . Node **C** also uses XOR as a relation between inputs at  $T$  to determine its state at  $T+1$ . In contrast Node **B** uses an operator that ignores its input from **A** and takes on the same value at  $T+1$  as **D** had at  $T$ . Similarly Node **D** ignores its input from **B** and mimics its input from **C**. There are sixteen possible logical operators for  $K=2$  inputs and the possibilities increase exponentially as  $K$  increases.

Malloy, Jensen and Song (2005) have built an open code NK Boolean simulation program written in Java and named E42. System 1 was generated pseudo-randomly by E42. That is, the two inputs to each node and logical operators for each node were determined by a randomization algorithm. The network size, density, inputs and operators can also be manually specified and the resulting networks can be examined with a series of traditional and specialized visual interfaces. All of the methods discussed here are currently accessible in E42 publicly.

-----

Insert Table 1 about here

-----

**State Vectors and Basins of Attraction.** The on/off state (Boolean values) of all the nodes can be described at any moment in time (i.e., on any iteration) by a *state vector*. The state vector is ordered from the first to the last node (in this case from Node **A** to Node **D**). Because Malloy, Jensen and Song (2005) describe in detail the derivation of a Boolean landscape, we will skip some of the steps for the formal derivations of a landscape for System 1. But for the reader's convenience Table 2 specifies all of the state vectors of System 1 (left hand four columns) as well as the deterministic transition of each state vector at  $T$  to a state vector at  $T+1$ . The logical relations among nodes in

Table 1 determine the transitions from  $T$  to  $T+1$  in Table 2 (see Malloy, Jensen, and Song 2005). This table defines the state space of this network in its entirety. Unfortunately, this representation of state space fails to show the inherent order that emerges from the Boolean network (this will become clearer as we show that this example generates a landscape of attractor cycles). Also, it is reasonable to create this state space for small systems such as this one, but loses its inherent value when  $N$  is large (too many possibilities). Even though we do not derive, step by step, the state transitions in Table 2, Table 1 gives enough information to do so.

-----

Insert Table 2 about here

-----

### **The Emergence of a Boolean Landscape**

The emergence of a landscape is important for our arguments and we will detail this process by using the transitions between state vectors in Table 2 to find basins and attractors that comprise a landscape. Suppose on iteration  $T=1$  the ON-OFF pattern for the four nodes from **A** to **D** is OFF, OFF, ON, OFF. This can be written as a state vector:  $S(1) = \{0010\}$ . Using Table 2 we can look up the state vector  $\{0010\}$  at time  $T$  and see that it becomes  $\{1001\}$  at time  $T+1$ . Following that procedure we derive the deterministic sequence from one state vector to another:  $S(1) = \{0010\} \rightarrow S(2) = \{1001\} \rightarrow S(3) = \{0100\} \rightarrow S(4) = \{1000\} \rightarrow S(5) = \{0010\} \rightarrow S(6) = \{1001\} \rightarrow S(7) = \{0100\} \rightarrow$  etc... This sequence therefore describes the flow of Boolean values (or, alternatively, the behavior) of the system across time. We can do this for any initial state vector. For written text it is convenient to write state vectors as row vectors as we have just done but in the figures and tables below we will transpose them so they are column

## A Phase Portrait for Boolean Systems

vectors; this places nodes on the vertical axis and time on its traditional, horizontal, axis.

Because the system is deterministic, a given state vector  $\mathbf{S}(T)$  will always be followed by the same state vector  $\mathbf{S}(T+1)$ . Therefore the fact that in the above example  $\mathbf{S}(5)$  is identical to  $\mathbf{S}(1)$  means that  $\mathbf{S}(6)$  must be identical  $\mathbf{S}(2)$  and  $\mathbf{S}(7)$  must be identical to  $\mathbf{S}(3)$ , and so on. This indicates that the system is in an attractor cycle of four distinct state vectors,  $\mathbf{S}(1)$  through  $\mathbf{S}(4)$ , repeating endlessly. This is the attractor cycle (which we will label A3 on the landscape shown in Figure 1, below. The length of an attractor cycle, measured in the number of state vectors (or iterations) before it repeats, is the fundamental frequency or length,  $L$ , of the cycle. Due to complete determinism in the system, it cannot escape this attractor cycle unless the system is perturbed. (Perturbation amounts to changing the state of one or more nodes.) The important points here are that the nodes of NK Boolean dynamic systems flow from state to state by a deterministic relational logic, that at any moment the entire system can be characterized by a state vector, and that the flow from state vector to state vector across time can fall into cyclic attractor cycles within basins of attraction. All of these basins from the same underlying network comprise a landscape.

**Attractor Matrix.** An attractor can be coded as a matrix of Boolean values (0's and 1's); the  $N$  nodes of the system are arrayed along the vertical dimension of the matrix (we place the first node,  $\mathbf{A}$  in System 1, at the top of the vertical dimension of the matrix and the last node,  $\mathbf{D}$ , at the bottom) while the horizontal dimension of the matrix counts the discrete units of time (iterations) required by the length,  $L$ , of the attractor. The attractor matrix is therefore  $N \times L$  (e.g., the attractor we are discussing is a  $4 \times 4$  matrix – four nodes that have four unique state vectors across time). Since the attractor cycle will

repeat without limit, this matrix fully describes the possible state vectors seen at all future iterations. Our convention in writing an attractor matrix is to begin with the state vector (column vector) that has the lowest binary value (defined by starting with the value of the first node in the system which is at the top of the columnar state vectors and reading down to the value of the last node). Starting an attractor matrix with the lowest valued vector has many virtues including the ease of finding and identifying a specific attractor in a database of attractor matrices.

**Full Landscape.** If we follow the deterministic path for vectors in the state space of Table 2 we will eventually derive the landscape of System 1 shown in Figure 1. That is, Table 2 provides the reader with the full information for deriving the landscape in Figure 1. In Holland's (1998) terms, the node truth tables in Table 1 are the constrained generating procedures from which this landscape emerges. Notice that the dynamics of this small Boolean system fall into six basins of attraction; each basin has an attractor cycle - a series of state vectors that loop (as they did in the example above where  $S(5)$  was identical to  $S(1)$ ). State vectors can also flow into attractors but not be a part of the looping vectors that define an attractor, these are known as *tributaries* (see Malloy, Jensen, and Song, 2005 for an example with tributaries). Tributaries are highly related to transience within continuous systems in that they are temporary as a function of the initial state vector or as a function of some perturbation to the system. System 1 is unusual in that, in the general case, a Boolean landscape is comprised of basins that have both tributaries and attractors. In the case of System 1, the Attractors, A1 through A6, are isomorphic with the basins because there are no tributary state vectors. While unusual in not having tributaries, System 1 is an excellent example for the conceptual points we

## A Phase Portrait for Boolean Systems

want to develop here. We will use  $A_1, A_2, A_3, \dots$  as symbols for the attractors in Basin 1, Basin 2, Basin 3, ... The basin numbering system in E42 is rational in that E42 archives basins first rotating each attractor matrix so that its first column has the lowest binary value among the state vectors in the attractor and then, using the values of the first column vector, naming them with the counting numbers ( $A_1, A_2, \dots$ ) based on the binary value of each attractors first column. Thus notice that the Attractor ( $A_1$ ) for Basin 1 in Figure 1 begins with the lowest Boolean value of vectors in the state space,  $A_2$  begins with the second lowest Boolean value, etc.

Basins 3, 4 and 6 have attractor cycles ( $A_3, A_4, A_6$ ) that repeat every four iterations, that is the cycle length is  $L = 4$ . Basins 1 and 5 have cycle lengths of  $L = 1$ . Basin 2 has an attractor,  $A_2$ , with  $L=2$ . In which basin does a system currently reside? That depends on the initial state vector in which the system starts and on perturbations to the system. We will return to the issue of how to traverse a landscape within the NK Boolean system in the discussion.

-----

Insert Figure 1 about here

-----

### **Visualizing and Simplifying Boolean System Dynamics**

We now switch focus from the methodology of how NK Boolean systems generate an emergent landscape to the ways under which we can visualize the dynamics of the system. The advantage to the methodology laid out so far is that we can fully determine the landscape of any system we generate – though they quickly become quite large and complex (the example shown here is intentionally small and simple, though the

principles hold for the more complex case). The goal of visualizing the underlying dynamics is to generate a simplification. We will first generate a visual representation of the state vectors in time, called a historical trace. This will then be combined with the discrete form of derivatives, seeking a simplification of the pattern. Finally, we will seek a visual representation of the relationship between the discrete derivatives, yielding the phase portrait equivalent.

**Historical Trace.** As noted earlier, the columns of 0's and 1's in Figure 1 can be construed as  $N \times L$  attractor matrices. To visualize these state vectors in a perceptually useful way we place these matrices on a grid with 0's represented as white cells and 1's represented as black cells. The vertical axis will then represent individual nodes from 1 (on top) to  $N$  (on the bottom) and the horizontal axis will represent iterations from 1 to  $L$ . Figure 2 shows the  $A_3$ ,  $A_4$ ,  $A_6$  attractor matrices from Figure 1 represented in this way. We call this a historical trace because it shows the pattern of states vectors of the system over some period of time (four iterations in Figure 2). Different attractor cycles generate different visual patterns; Figure 2a shows the form generated by the  $A_3$  attractor cycle, Figure 2b shows  $A_4$ , and Figure 2c shows  $A_6$ . A fuller discussion of the historical trace can be found in Malloy, Jensen, and Song (2005). For the methodology we are presenting it is important that we be able to see, with immediacy, the patterns found both in the system's dynamics and in the analyses of those dynamics we are about to develop.

-----

Insert Figure 2 about here

-----

**Derivative Simplification: TAO.** As with continuous systems, we embrace

derivatives as a way of simplifying the description of a system. We therefore now turn to the discrete derivative and its visual representation developed by Malloy, Jensen and Song (2005) and Malloy and Jensen (2008) known as the TAO function. As noted in Table 1, XOR compares two inputs and if the states of the inputs are the same it returns a 0 and if the states of the inputs are different it returns a 1. Therefore we say that the XOR operator detects difference. Since the definition of a derivative is change in time, we apply the XOR operator to compare two successive state vectors to determine if there was change or no change.

To understand TAO, consider two state vectors,  $\mathbf{S}(1)$  and  $\mathbf{S}(2)$ , from the discussion of A3 above (see text and Figure 1) and which are shown as column vectors in Table 3; these column vectors represent the states of the four nodes at  $T = 1$  and at  $T = 2$ . The top value in a vector represents the state of Node **A**, the second value represents the state of Node **B**, and so on. The reader merely has to compare the Boolean values, node by node, moving down  $\mathbf{S}(1)$  and  $\mathbf{S}(2)$  confirming that a 0 appears in the TAO column when the values for a node are the same across time and 1 when the values for a node are different across time. Because it tracks the absolute change (or lack of change) in the flow of Boolean values across time TAO is the basis for the discrete analogue to the first derivative (The difference between TAO and XOR is that TAO is a matrix operator that compares, node by node, one column vector in the matrix with a succeeding vector in time). Since all of the TAO values are also merely 0 (same) or 1 (different), it can also be visualized as white and black cells in place of zeros and ones, having no effect on the logic.

-----

Insert Table 3 about here

-----

We find it useful to label the attractor matrix TAO-0; this indicates a null application of the TAO operator to the attractor's dynamics. Since TAO-0 is a matrix representation of the state vectors, in temporal order, the first derivative can also be expressed in matrix form. To complete the logic of TAO, examine the relation between the TAO-0 (attractor matrix) pattern and the TAO-1 pattern in Figure 3. Begin with Node **A** in the TAO-0 matrix; notice that Node **A** changes from white to black between iteration 1 and iteration 2. When a node changes its value between two successive iterations the XOR operator indicates this difference by returning a 1 (black cell). Now notice in the top (Node **A**) row of the TAO-1 matrix that the first column is black; this black cell corresponds to the change in the value of Node **A** between iteration 1 and 2. In fact, notice in the TAO-0 attractor matrix that Node **A** changes from black to white between each successive pair of iterations; therefore the first row of the TAO-1 matrix is all black indicating that Node **A** changed on each iteration. To what do we compare the last column in the attractor matrix? To state the obvious, the attractor is a cycle that repeats its first value after its last value no matter where we start in the cycle; therefore the fourth column in the TAO-0 matrix is compared with the first column.

Now examine Node **B**, which changes twice during the attractor cycle (from the second iteration to the third and from the third iteration to the fourth). Thus the Node **B** row in the TAO-1 matrix is white, black, black, white or {0110}. Using this logic, the reader can also examine and confirm the TAO-1 values for Nodes **C** and **D**. Recall that the attractor cycle is a flow of changes in Boolean values. Therefore the full TAO-1

matrix codes all the moment to moment changes in the flow of changes in the attractor cycle. Like velocity codes the momentary changes in position, TAO-1 codes the momentary changes in Boolean values in an attractor's cyclic dynamics.

**Recursive TAO.** The TAO function can be applied (like the derivative function) to its own output. TAO-2 simply takes the first column vector of TAO-1 and compares it, node by node, to the second column vector of TAO-1 yielding 1 (black) for a change and a 0 (white) for no change. TAO-3 applies TAO to TAO-2 and TAO-4 applies TAO to TAO-3. This analysis should be easy to follow in Figure 3.

In Figure 3 the derivative diminishes to the  $\mathbf{0}$  matrix (all cells white). As reported in the Malloy & Jensen (2008) this always happens when the attractor cycle length,  $L$ , is a power of 2. In the case of  $A_3$  ( $L=4$ ), since 4 is a power of 2 the derivative function diminishes to no change in change by the fourth derivative. As a side note, when  $L$  is not equal to a power of 2, rather than the higher order derivative matrices resolving to  $\mathbf{0}$ , the derivatives begin repeating themselves in long, complex patterns. Indeed the Boolean phase portrait methodology we are about to describe is aimed at providing a theoretical description for both the power of two (simple) and non-power of two (more complex) patterns.

### **Meta-TAO: The Boolean Phase Portrait**

The TAO matrices, in historical trace form, visualize the derivatives of a given attractor cycle from the Boolean landscape. As noted in the introduction, it is the relationship amongst these derivatives that is represented in the traditional continuous phase portrait. To begin our construction of an operation that describes the relationship among Boolean derivatives, we define a matrix operation whereby every corresponding

cell in two conformable matrices (containing binary values) is compared (by the XOR operator); if the two cells have the same value then XOR returns a 0 (or, visually, a white cell) and if the two corresponding cells are different XOR returns a 1 (black cell). To construct a Boolean phase portrait the two matrices so compared are always two TAO matrices (usually including TAO-0 as one of the pair); thus we call the operation Meta-TAO. In terms of the Boolean logic of the systems we are analyzing, Meta-TAO can also be thought of as a way of assessing the Boolean relationship between two TAO matrices akin to a phase space.

Figure 4 has three panels. The upper row of Panel (a) shows the A3 matrix along with its four derivatives, TAO-1 through TAO-4. Note that the A3 matrix is also labeled TAO-0. The lower row of Panel (a) shows Meta-TAO matrices visualizing the Boolean relationship between the attractor matrix (TAO-0) for A3 and each of its derivatives. Panels (b) and (c) show the same information for attractors A4 and A6. The 0 v 0 Meta-TAO comparison in all three panels is an all white matrix because it is comparing TAO-0 with itself and so there are no differences.

### **Features of Meta-TAO**

**Identity versus Non-identity Meta-TAOs.** We distinguish two different types of meta-TAOs, both of which are represented in Figure 4a. The first type we call *identity* Meta-TAOs. Place your attention on Meta-TAO (0 v 1) for A3 in Panel (a) of Figure 4. Notice how similar its pattern is to the pattern of the A3 attractor. In fact, if we rotate the columns of the Meta-TAO (0v1) matrix so that it begins with the column with the lowest Boolean value (as was actually done for the A3 attractor matrix, see landscape methodology, above), the Meta-TAO (0v1) matrix *is* identical to the A3

matrix. We therefore call this type of meta-TAO an identity meta-TAO because it amounts to multiplying the original attractor matrix times an identity matrix. Under this rotation of column vectors to-lowest-binary-value logic, Meta-TAO (0 v 2) and (0 v 4) are also examples of identity Meta-TAO; they are each identical to the A3 matrix. In contrast Meta-TAO (0 v 3) is not an identity Meta-TAO; it cannot be rotated to become identical to the A3 matrix. To be theoretically neutral, we therefore label this a *non-identity* Meta-TAO. But Meta-TAO (0v3) in Panel (a) of Figure 4 has important properties as we will now discuss.

-----

Insert Figure 4 about here

-----

**Meta-TAO and Phase Portraits.** A system's phase portrait is "the key to the geometric theory of dynamical systems," (Abraham & Shaw, 1984, p. 11) and is typically plotted in classical oscillator theory (Abraham & Shaw, 1984, p. 64) as a first derivative (velocity) against zero-order derivative (position) and portrays the trajectories of a system. It then maps the relationships amongst basins: where they are in relation to one another. As mentioned already, the basins themselves are easily identifiable as the stable attractor cycles and their constituent tributaries. However, Meta-TAO provides the relations amongst the basins, just like the phase portrait. To illustrate this point we wish to draw your attention to the non-identity Meta-TAO (0 v 3) matrix for A3 in Figure 4a; to review, this matrix codes the Boolean XOR relation between A3's TAO-0 matrix and its TAO-3 matrix, that is, between the zero-order derivative and the third-order derivative. What is notable about the Meta-TAO (0v3) matrix for A3 is that (when

rotated to begin with the lowest binary column vector value) it is exactly the A4 attractor matrix (see Figure 2 or Figure 4b). In other words, the Meta-TAO-3 of A3 transforms A3 into A4. This establishes a deep relation between the attractors of two different basins (A3, A4) from the same landscape.

The ability for Meta-TAO to map the relations amongst the basins and thus establish the phase portrait requires expanding the approach to all observed basins. Examination of Figure 4a and Figure 4b indicates A3 and A4 code each other through the non-identity Meta-TAO (0 v 3) analysis. That is, Meta-TAO (0 v 3) for each yields the attractor of the other. In contrast, Figure 4c shows that the Meta-TAO (0 v 3) matrix for A6 yields A6 itself; that is, Meta-TAO (0v3) for A6 is an identity Meta-TAO. Thus A3 and A4 are related to each other by a Meta-TAO analysis while A6 is related to itself. Taken together, these relations form a map of how each basin is related to one another, spatially depicting the stable patterns of change.

**Landscape Searching and Meta-TAO.** Though the goal here was to develop a methodology for a complete phase portrait, the true utility of this approach becomes clear when only partial information about a system is known. Since Tao and Meta-TAO can be applied to a single (observed) stable basin, it can be used to create a localized phase portrait. The Meta-TAOs can then identify other potential basins that are spatially adjacent to the current basin. In essence, this establishes an alternate route to landscape searches. The utility of this approach (when it works and why it works) are left to future expansions due to page constraints.

### **Summary**

The primary contribution of this paper is methodological. It develops the Meta-

## A Phase Portrait for Boolean Systems

TAO procedure for using Boolean discrete derivatives to create a discrete analogue to a phase portrait. We believe this direction will be particularly fruitful in revealing both intuitive and counter-intuitive aspects of systems given the well documented utility of phase portraits in continuous dynamical systems (Baker & Gollub,1996) .

The nature of fitness landscapes and the relation between fitness landscapes and search strategies (adaptive walks) is not well understood. Kauffman (1993, 1995), among others, has detailed numerous poorly understood issues that could be productively informed by what can be learned from a discrete analogue to a phase portrait for Boolean landscapes. Indeed, the methodology in this paper is the basis of our future work (previewed in a conference paper: Malloy, Butner, Cooper, Smith, & Dickerson, 2007) integrating the Meta-TAO phase portrait procedure with symmetry theory, as a means for describing the relations among all the basins on an NK Boolean landscape. Given the extreme utility of phase portraits in the history of dynamics, the development of a procedure that acts like a phase portrait for Boolean landscapes offers an avenue toward a fuller understanding of such landscapes. For example, the ability to derive, from local information alone (e.g., A3) the existence of and the characteristics of another attractor (e.g., A4) potentially has deep implications for adaptive navigation on Boolean landscape and therefore for evolutionary theory, particularly for how long leap adaptation could achieve a high degree of success.

**Tables**

**Table 1.** Logical relations among the nodes of **System 1**, a randomly-generated N=4, K=2 Boolean system. Each of the four nodes takes input from two other nodes.

<b>Node A</b>			<b>Node B</b>			<b>Node C</b>			<b>Node D</b>		
B at T	C at T	A at T+1	D at T	A at T	B at T+1	D at T	A at T	C at T+1	B at T	C at T	D at T+1
0	0	<b>0</b>	0	0	<b>0</b>	0	0	<b>0</b>	0	0	<b>0</b>
0	1	<b>1</b>	0	1	<b>0</b>	0	1	<b>1</b>	0	1	<b>1</b>
1	0	<b>1</b>	1	0	<b>1</b>	1	0	<b>1</b>	1	0	<b>0</b>
1	1	<b>0</b>	1	1	<b>1</b>	1	1	<b>0</b>	1	1	<b>1</b>

**Table 2.** State Transition Table. All possible 16 state row vectors are shown in Time T column; this constitutes the state space. The vectors shown under T+1 are derived from the logic of Table 1 and show the transition of every vector in the state space into some vector in the state space.

<b>Time T:</b>	<b>Time T+1:</b>
0000	>> 0000
0001	>> 0110
0010	>> 1001
0011	>> 1111
0100	>> 1000
0101	>> 1110
0110	>> 0001
0111	>> 0111
1000	>> 0010
1001	>> 0100
1010	>> 1011
1011	>> 1101
1100	>> 1010
1101	>> 1100
1110	>> 0011
1111	>> 0101

**Table 3.** Two state column vectors from the A3 attractor cycle with the resulting TAO column vector indicating change versus no-change for each node from T1 to T2.

Node	S(1)	S(2)	TAO
A	0	1	1
B	0	0	0
C	1	0	1
D	0	1	1

**Figures Captions**

**Figure 1.** The basin landscape of a small 4 node system named System 1. The landscape includes six basins, each with an attractor cycle. Each attractor is coded as a matrix of zeros and ones; the vertical dimension of the matrix contains the nodes of a system and the horizontal axis is discrete units of time (iterations). In general, a landscape has tributaries (state vectors) that run into attractors but are not part of the attractor loop; System 1 is unusual in that all of its state vectors exist in attractors and none of them are tributaries.

**Figure 2.** Visual patterns formed by the state column vectors of attractors A1, A2, and A3 from System 1. Nodes are on the horizontal axis and time (iterations) is on the horizontal axis.

**Figure 3.** The attractor matrix (TAO-0) and first four derivatives (TAO-1 through TAO=4) of A3. A white cell = 0 and a black cell =1 in Boolean values. In TAO-0 the vertical matrix dimension indexes the four nodes (A at the top, D at the bottom) and the horizontal dimension indexes time (iterations). In the TAO-1 through 4 matrices, the horizontal dimension indicates the results of an XOR comparison between two columns (e.g., 1 v 2 or 2 v 3) in the previous (to the left) matrix.

**Figure 4.** Complete TAO and Meta-TAO analyses for attractor cycles A(3), A(4) and A(6)

Figures

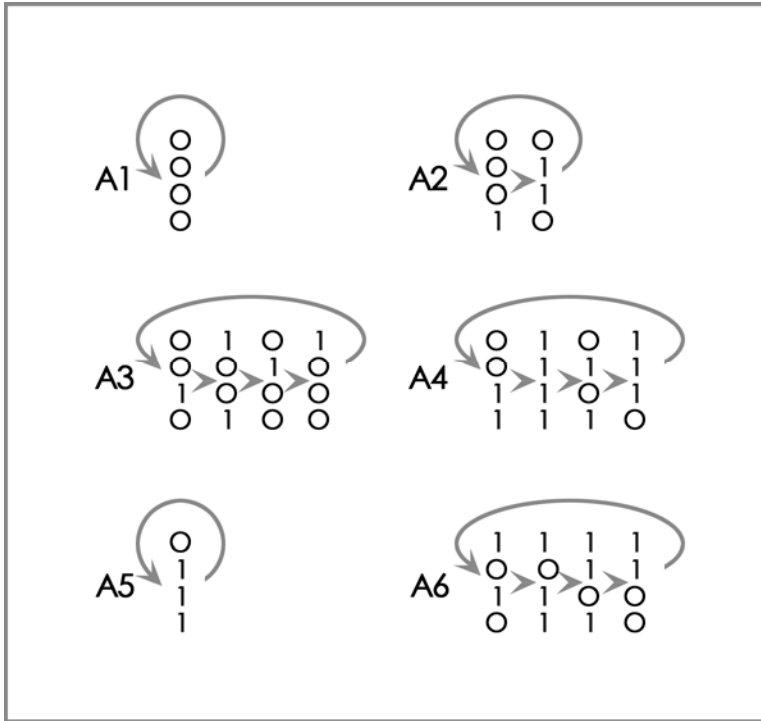


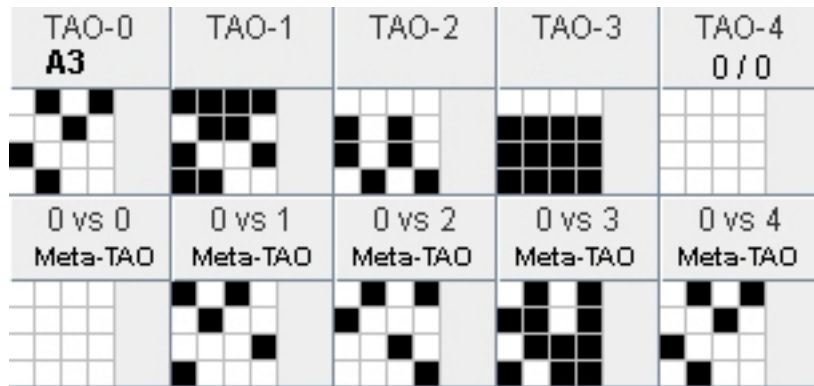
Figure 1

A3	A4	A6

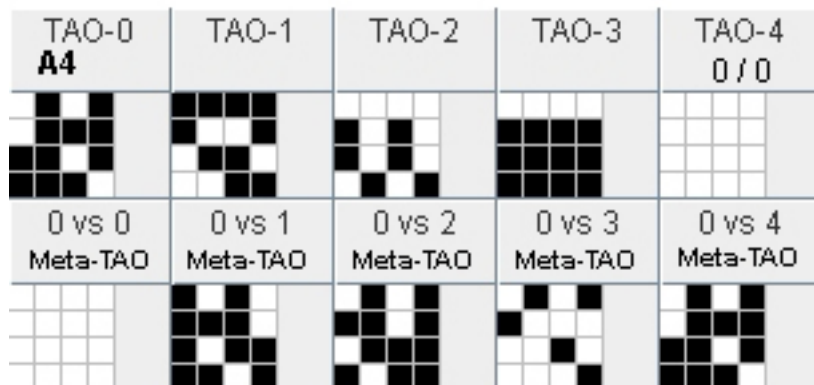
Figure 2

	TAO-0	TAO-1	TAO-2	TAO-3	TAO-4
N O D E S	<b>A3 MATRIX</b>				
	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4
	Iterations	v v v v	v v v v	v v v v	v v v v
		2 3 4 1	2 3 4 1	2 3 4 1	2 3 4 1
Comparisons of Column Vectors in previous matrix					

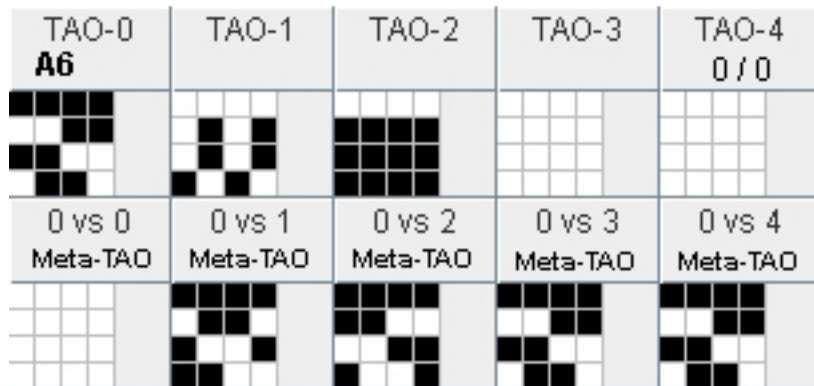
Figure 3



(a) TAO and Meta-TAO (lower row) matrices for A3



(b) TAO and Meta-TAO (lower row) matrices of A4



(c) TAO and Meta-TAO (lower row) matrices for A6

Figure 4

## References

Abraham, R. H., & Shaw, C. D. (1984). *Dynamics--the geometry of behavior part one: Periodic behavior*. Santa Cruz, CA: Ariel Press.

Baker, G. L. & Gollub, J. P. (1996). *Chaotic dynamics* (2<sup>nd</sup> ed.). Cambridge, UK: Cambridge University Press.

Choi, S., Jung, K., & Kim, J. H. (2005). Phase transition in a random NK landscape model. *GEOCC, 05*, 1241-1248.

Holland, J. H. (1998). *Emergence: From chaos to order*. Cambridge, MA: Perseus Books.

Kauffman, S. A. (1993). *The origins of order: Self-organization and selection in evolution*. Oxford: Oxford University Press.

Kauffman, S. A. (1995). *At home in the universe: The search for the laws of self-organization and complexity*. Oxford: Oxford University Press.

Malloy, T. E., Butner, J., Cooper, J., Smith, T., & Dickerson, C. (2007, July). Knowing begets knowing: Derivatives and meta-derivatives reveal the topology of basin landscapes in Boolean XOR rings. The Society for Chaos Theory in Psychology and the Life Sciences Annual International Conference, Orange, CA.

Malloy, T. E., Jensen, G. C., & Song, T. (2005). Mapping knowledge to Boolean dynamic systems in Bateson's epistemology. *Nonlinear Dynamics, Psychology, and Life Sciences, 9*, 37-60.

Malloy, T. E. & Jensen, G. C. (2008). Dynamic constancy as a basis for perceptual hierarchies. *Nonlinear Dynamics, Psychology, and Life Sciences, 12*, 191-203.

Riley, M. A., & Van Orden, G. C. (2005). *Tutorials in contemporary nonlinear methods for the behavioral sciences*. Retrieved March 1, 2005, from <http://www.nsf.gov/sbe/bcs/pac/nmbs/nmbs.jsp>

Skellet, B., Cairns, B., Geard, N., Tonkes, B., & Wiles, J. (2005). Maximally rugged NK landscapes contain the highest peaks. *GEOCC, 05*, 579-584.

[Takens](#), F (1981). "Detecting strange attractors in turbulence". [D. A. Rand](#) and [L.-S. Young](#). *Dynamical Systems and Turbulence, Lecture Notes in Mathematics*, vol. 898: 366–381, Springer-Verlag.

Watson, R. A., & Jansen, T. (2007). A building-block royal road where crossover is provably essential. *GEOCC, 07*, 1452-1459.