PAUL ST. DENIS and PATRICK GRIM

FRACTAL IMAGES OF FORMAL SYSTEMS

ABSTRACT. Formal systems are standardly envisaged in terms of a grammar specifying well-formed formulae together with a set of axioms and rules. Derivations are ordered lists of formulae each of which is either an axiom or is generated from earlier items on the list by means of the rules of the system; the theorems of a formal system are simply those formulae for which there are derivations. Here we outline a set of alternative and explicitly visual ways of envisaging and analyzing at least simple formal systems using fractal patterns of infinite depth. Progressively deeper dimensions of such a fractal can be used to map increasingly complex wffs or increasingly complex 'value spaces', with tautologies, contradictions, and various forms of contingency coded in terms of color. This and related approaches, it turns out, offer not only visually immediate and geometrically intriguing representations of formal systems as a whole but also promising formal links (1) between standard systems and classical patterns in fractal geometry, (2) between quite different kinds of value spaces in classical and infinite-valued logics, and (3) between cellular automata and logic. It is hoped that pattern analysis of this kind may open possibilities for a geometrical approach to further questions within logic and metalogic.

1. INTRODUCTION

Familiar formal systems include propositional calculus, predicate calculus, higher-order logic and systems of number theory and arithmetic. As standardly envisaged, these consist of a grammar specifying well-formed formulae together with a set of axioms and rules. Derivations are ordered lists or series of formulae each of which is either an axiom or is generated from earlier items by means of the rules of the system, and the theorems of a formal system are simply those formulae for which there are derivations.

Given this standard approach to formal systems, however, attempts to envisage formal systems as a whole seem of necessity remotely abstract and incomplete. As a psychological matter, if one is asked to envisage the theorems of predicate calculus as a whole, one seems at best able to conjure up an image of the axioms and an empty category of 'all that follows from them'. The incompleteness of such a psychological picture accords perfectly with constructivist approaches to formal systems, and may even seem to confirm them.

In what follows we want to outline some importantly different and immediately visual ways of envisaging formal systems, including a modelling of systems in terms of fractals. The progressively deeper dimensions of fractal images can be used to map increasingly complex wffs or what we will term ‘value spaces’, which correspond quite directly to columns of traditional truth tables. Within such an image, tautologies, contradictions, and various forms of contingency can be coded in terms of color or shading, resulting in a visually immediate and geometrically suggestive representation of systems as an infinite whole. One promise of such an approach, it is hoped, is the possibility of asking and answering questions about formal systems in terms of fractal geometry. As a psychological matter, it is interesting to note, complete fractal images of formal systems seem to correspond to a realist and non-constructivist approach to formal systems.

In what follows we begin with the example of tic-tac-toe, a simple game rather than a simple formal system, in order to make clear both the general approach and a number of the tools used at later stages. In Sections 3 and 4 we offer different geometrical patterns for mapping aspects of formal systems, starting with ‘rug’ images for fragments of predicate calculus and moving on to more complex systems and more complete forms of mapping. An alternative portrayal of formal systems in terms of ‘value spaces’ and ‘value solids’ offers a number of surprises, three of which are emphasized in Sections 5 through 7: the appearance of the Sierpinski gasket, a familiar fractal, as the pattern of tautologies in standard value spaces; an intriguing correspondence between value solids for classical logic and rival connectives for infinite-valued logics; and the possibility of generating the value spaces of standard logics using elementary cellular automata.

2. The example of tic-tac-toe

Although our primary concern is with fractal images for formal theories, rather than for games, many of the techniques to be used can be made clear using a fractal image for the simple game of tic-tac-toe.

The first player in tic-tac-toe, conventionally labelled X, has a choice of one of nine squares in which to place his marker. The opposing player O then has a choice of one of the remaining eight squares. On X’s next turn again he has a choice of seven squares, and so forth. There are thus a total of 9! possible series of moves (9 factorial: $9 \times 8 \times 7 \times \cdots \times 1$), giving us 9! possible tic-tac-toe games. Some of these are wins for X, some for O, and some draws (wins for neither player). The fractal image shown in Figure 1, sections of which are progressively enlarged in Figure 2, offers an analytic presentation of all possible tic-tac-toe games.

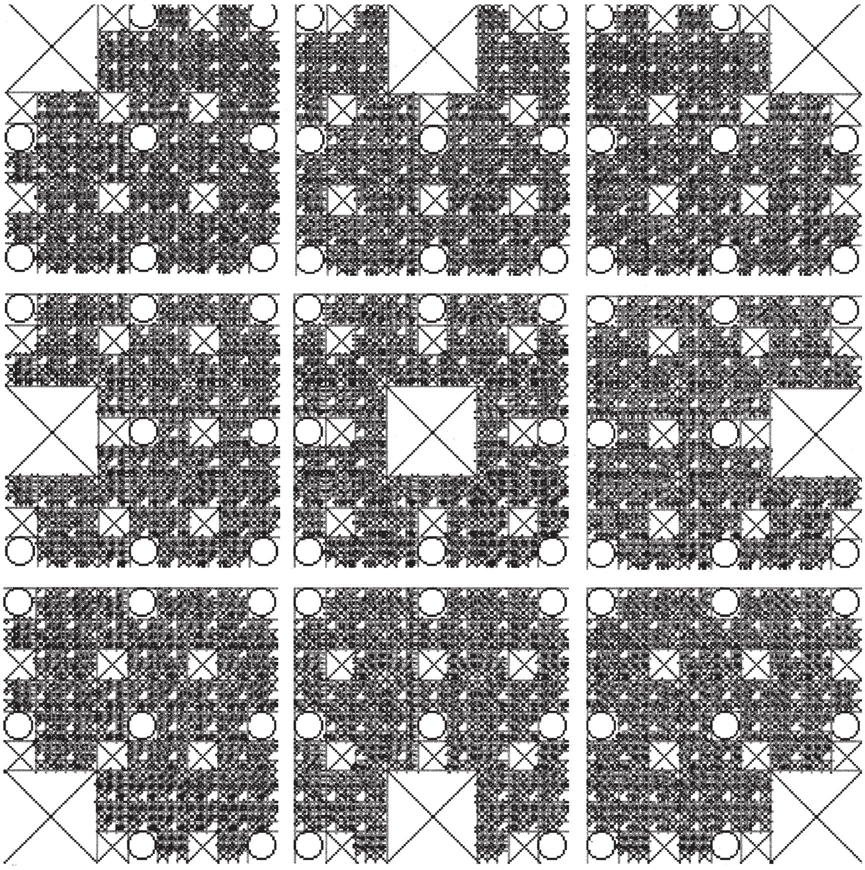


Figure 1.

In Figure 1 we've emphasized the divisions corresponding to the 9 basic squares of the tic-tac-toe grid. Let us now suppose that X's first move is to the upper left hand corner. The progress of possible games from that point is contained in the contents of the upper left hand square, enlarged in Figure 2A. The upper left square of the enlarged portion is now occupied, having already been played by X, but O can choose any of the remaining eight squares for the second move of the game.

Suppose O chooses the upper right hand corner. This is represented by a move to the upper right square of the 9 squares in 2A. That square is enlarged in Figure 2B, showing the two moves by X and O in place but seven further options for X.

Figure 2 continues by enlarging the squares chosen by particular players in the series we've chosen as an example: in this case the series

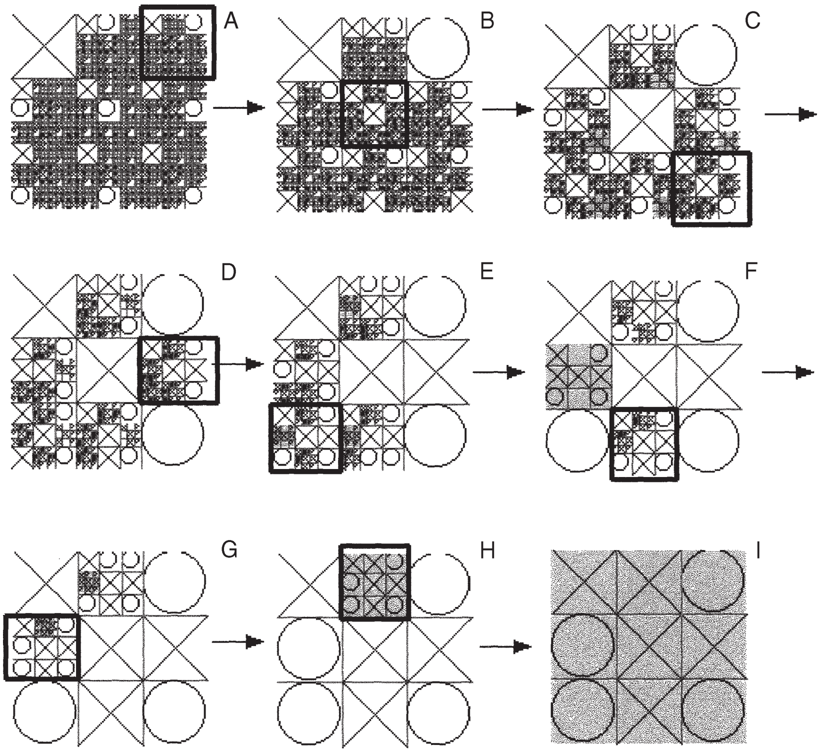


Figure 2.

continues with X to the center on the third move, then O to the lower right, X to the center right, O to lower left, X to center bottom, O to center left, and X to center top. The result is a win for X, indicated in color by a blue shading.

In the largest view of the fractal, shown in Figure 1, patterns of yellow and blue can be used in a color version to indicate wins for O and X, respectively, though the yellow wins in particular are small enough – meaning deep enough in the game – so as to be practically invisible. Were the resolution of our illustration great enough, however, and our eyes sharp enough, we would be able to see all such wins embedded in the image. Winning strategies, as a matter of fact, can be thought of as spatial movements through the fractal toward those winning games. (An interactive program of Figure 1, which allows progressive navigation into deeper levels, is available from the authors on request.)

Tic-tac-toe is convenient as an illustration because we have only two players, only three final outcomes of concern (a win for X, a win for O, or a draw), and because the game has a clear terminus after 9 plays. The basic idea of a game fractal could in principle be extended to checkers and even chess, though it's also clear that these would become explosively complex in short order.[1]

In what follows we begin by applying much the same fractal techniques to simple formal systems. In this application wffs or equivalence classes of wffs will replace moves or series of moves in the example of tic-tac-toe, colors for wins and draws will be replaced with colors coded to tautologies, contradictions, and various contingencies, and the fractal images used will often be infinitely rather than merely finitely deep.

3. 'RUG' ENUMERATION IMAGES

We begin with an extremely simple formal system for which we will construct several different forms of images. The system at issue is propositional logic, made even simpler by restricting it to a single sentence letter p. In order to make things simpler still, we use a single connective: either the Sheffer stroke $|$, which can be read as NAND, or the dagger $\downarrow$, which can be read as NOR. As is well known, either NAND or NOR suffices as a complete base for all Boolean connectives.

Our goal, then, is to construct an image of truth-values for all formulae expressible in terms only of p and $|$ or $\downarrow$. The values at issue are merely four, equivalent to $p, \sim p\,(p \mid p$ or $p \downarrow p)$, tautology $\top$ or contradiction $\perp$. In Figures 4 through 8 we use the following colors and grey shades for these values: green (dark grey) for p, blue (light grey) for $\sim p$, red (white) for tautologies and dark blue (black) for contradictions.

Let us start with a simple 'rug' pattern with an enumeration of all formulae expressible in terms of our single sentence-letter and single connective. For a first enumeration of wffs the plan of the rug is laid out schematically as in Figure 3.

At the upper left-hand corner of the array, in position 1, we construct a formula $(p \mid p)$. To the left of the slash is the formula at the top of the column for this position in the array – a simple p, in this case. To the right of the slash is the formula at the left of its row – p again, in this case. For simplicity we refer to formula $(p \mid p)$ as formula 1, and position it as the second formulae on each of our axes. The formula in position 2 is now formed in a similar manner, putting the formula at the head of its column to the left of a slash and the formula at the left of its row to the right of the slash. Formula 2 is thus $(p \mid 1)$ or $(p \mid (p \mid p))$. It is then added as the third formula on each axis. Following the pattern indicated, we continue to construct new formulae from old and continue to place them on the axes as formulae from which later formulae will be formed.

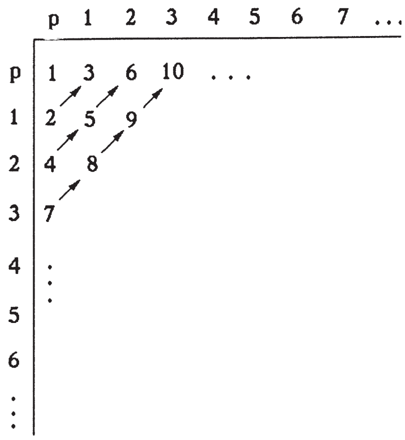


Figure 3.

Formula 3 is thus $((p \mid p) \mid p)$, formula 4 is $(p \mid (p \mid (p \mid p)))$, formula 5 is $((p \mid p) \mid (p \mid p))$, formula 6 is $((p \mid (p \mid p)) \mid p)$, and so forth. The pattern generates more complex formulae as it proceeds, constituting in the abstract an infinite partial plane extending to the bottom and right and containing all formulae of our simple single-sentence-letter form of propositional calculus.

In Figure 4 the schema is shown in shades of color. Squares correspond directly to the formulae indicated in the schematic sketch above, including formulae along the axes, and are colored in terms of their values: as noted, green (dark grey) $= p$ or equivalent formulae, light blue (light grey) $= p \mid p$ or $\sim p$, red (white) represents tautologies $\top$ and dark blue (black) represents contradictions $\perp$. The first graph in Figure 4 is a smaller fragment of the upper left corner of the rug, with the values of formulae indicated on axes as well. The second image in Figure 4 shows a larger section, incorporating the first.

Here a number of systematic features are immediately evident. The first is that the images in Figure 4 are symmetric, reflecting the fact that $x \mid y$ has the same value as $y \mid x$ for any formulae x and y. The 'stripes' in the rug are also obvious, reflecting the fact that both $x \mid \perp$ and $\perp \mid x$ will be tautologous for any x: once any formula on either axis has the value $\perp$, any formula composed from it with a single stroke will have the value $\top$. Closer attention shows that rows in which the value of the formula at the top is $\top$ will simply reflect the value of the formulae on the column axis, with the same true for columns with the value $\top$ and the formulae listed along the top.

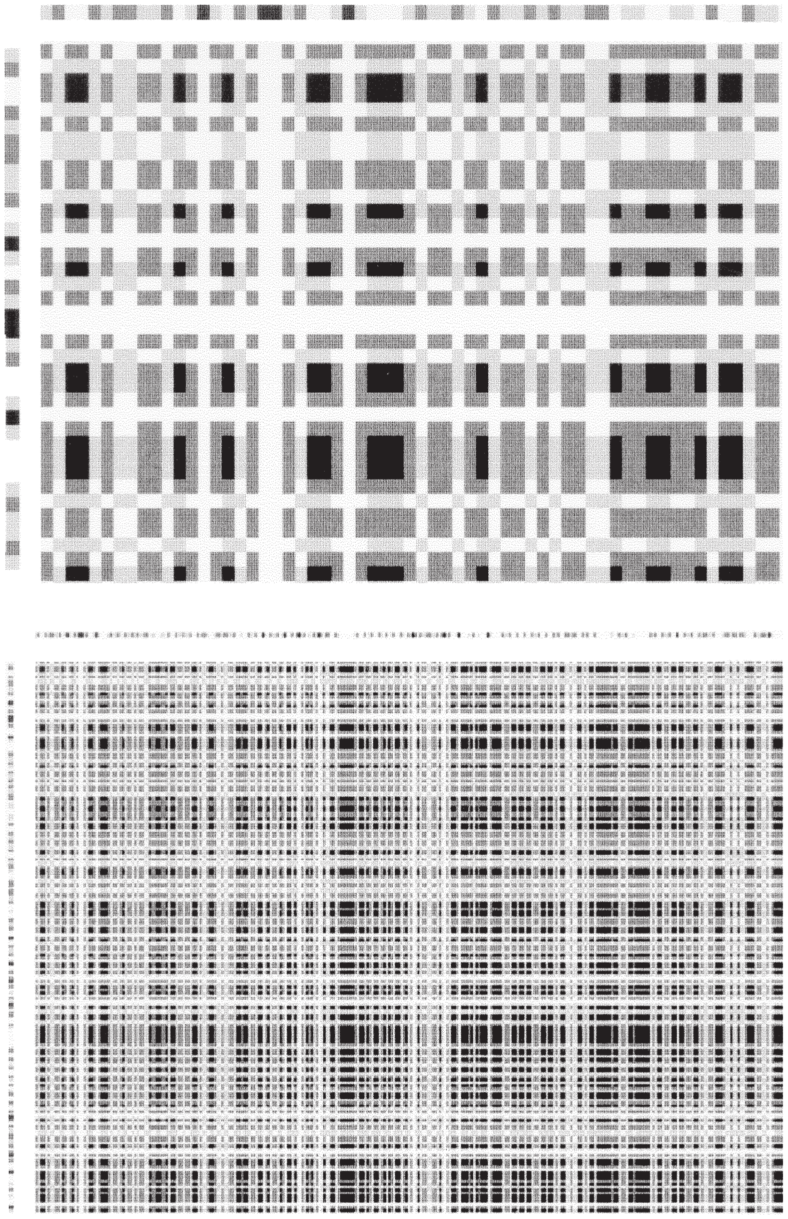


Figure 4.

Figure 5 shows the rug pattern created from the same enumeration of formulae but in which the Sheffer stroke | is replaced with the NOR connective ↓. Side by side, Figures 4 and 5 also serve to make obvious certain relationships between these two connectives: the fact that a contradiction on either side of the stroke gives us a tautology, for example, whereas a tautology on either side of the dagger gives us a contradiction. It is clear that dagger tautologies mirror Sheffer stroke contradictions, with dagger contradictions corresponding to Sheffer tautologies. For simple systems with a single sentence letter, moreover, the values of all contingent formulae in each system are the same. In none of these cases do our images offer genuinely new information regarding the stroke and dagger, of course – all the facts indicated are well known – though these patterns do make such features vividly evident.

Figures 7 and 8 show a rug pattern using a different enumeration of formulae, following the alternative schematic in Figure 6.

Nothing, it should be noted, dictates any particular form for enumeration in such a display; nothing dictates the diagonal enumeration of Figure 3 over the square enumeration of Figure 6, for example, nor either of these over any of the infinite alternatives. There is therefore an ineliminable arbitrariness to the choice of any particular rug pattern for a formal system. It is also clear, however, that certain properties of patterns – including those noted above – will appear regardless of the pattern of enumeration chosen. Pattern-properties invariant under enumeration can be expected to correspond to deep or basic properties of the system.

The rug patterns sketched above are for an extremely simple form of propositional calculus, explicitly restricted to just one sentence letter. Can such an approach be extended to include systems with additional sentence letters as well?

Of course. One way of extending the enumeration schemata above so as to include two sentence letters rather than one is simply to begin with the two sentence letters on each axis, as shown in Figure 9. In all other regards enumeration can proceed as before.

With two sentence letters, of course, four colors will no longer suffice for values of tautologies, contradictions, and all possible shades of contingency. For a system with both p and q we will require 16 colors in all, corresponding to the sixteen possible truth tables composed of four lines, or equivalently the sixteen binaries composed of four digits.

Complete color shade patterns – employing a complete palette of contingencies – are shown for propositional calculus with p and q in Figure 10. These represent NAND and NOR with our intial diagonal enumeration scheme. Corresponding illustrations in terms of our second mode of enumeration appear as Figure 11. It is interesting to note that although a number of the characteristics marked with respect to propositional calculus with a single sentence letter above still hold, one does not:

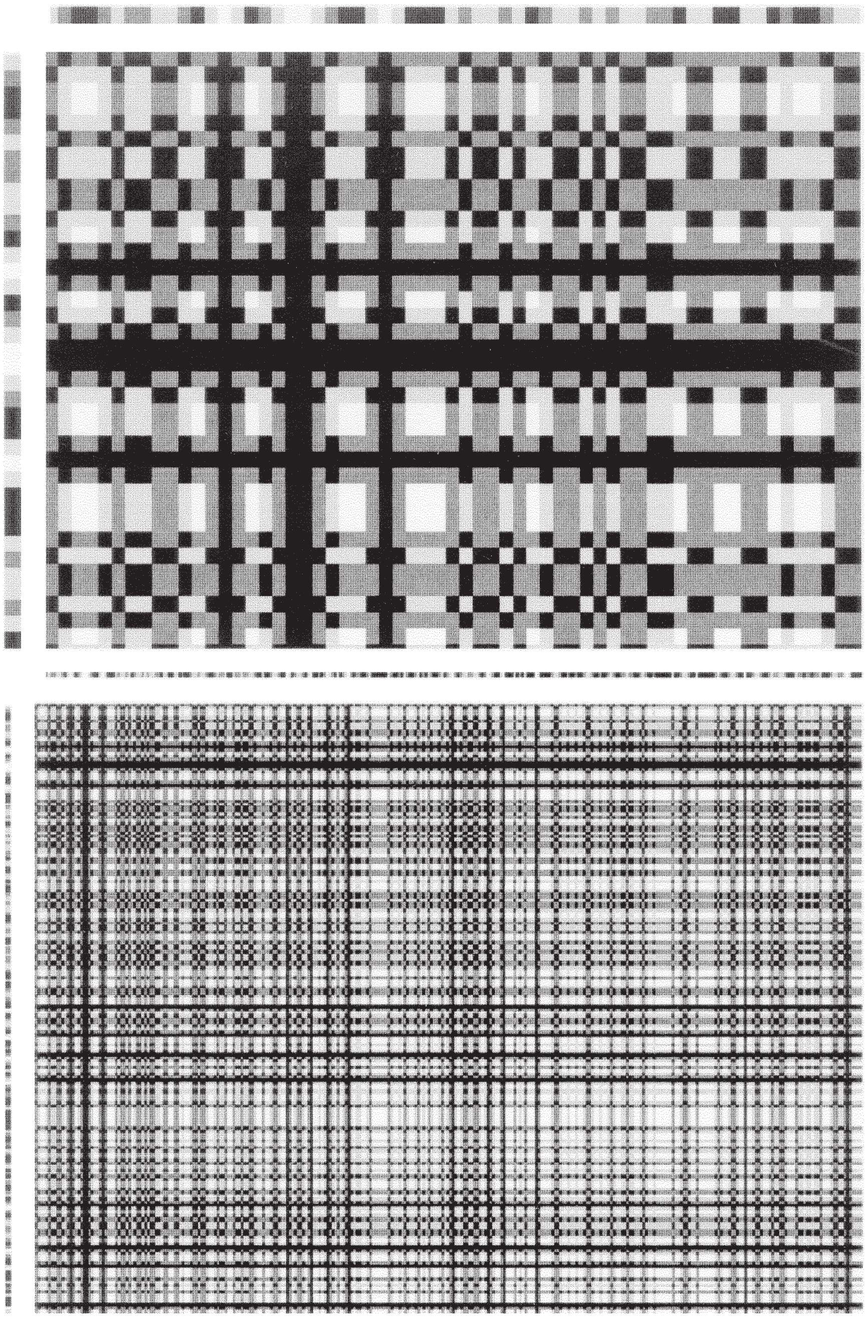


Figure 5.

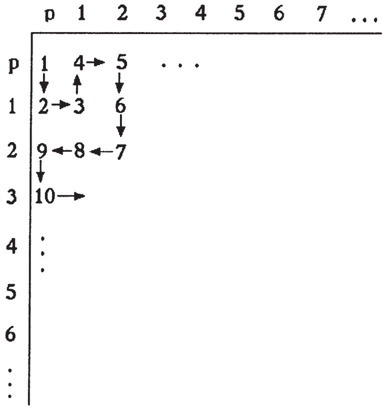


Figure 6.

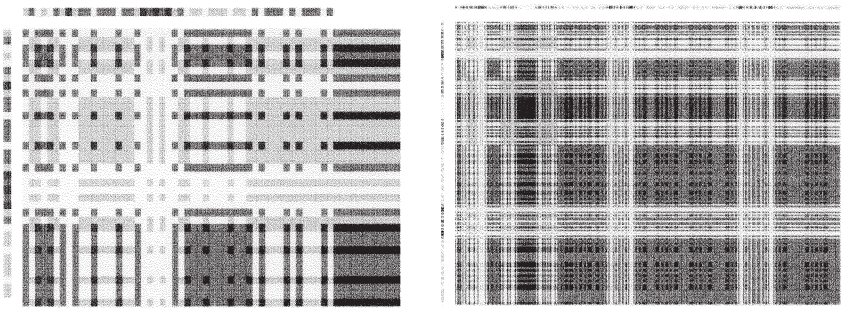


Figure 7. NAND using square enumeration.

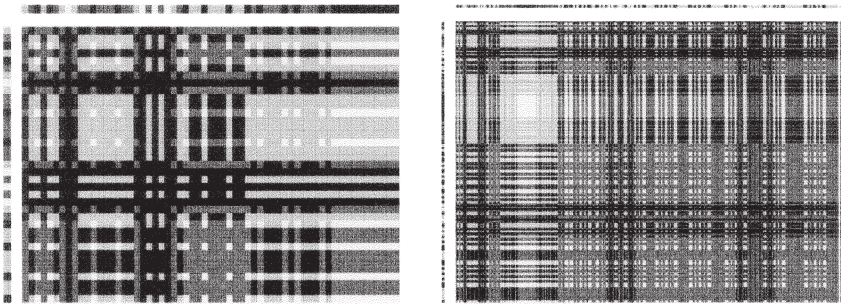


Figure 8. NOR using square enumeration.

here it is no longer true that contingent values match between NAND and NOR versions. That property, though provable for propositional calculus with a single sentence letter, disappears in richer systems.

In both of these illustrations the number of colors at issue becomes even more bewildering in larger sections of the display. In Figures 12 and

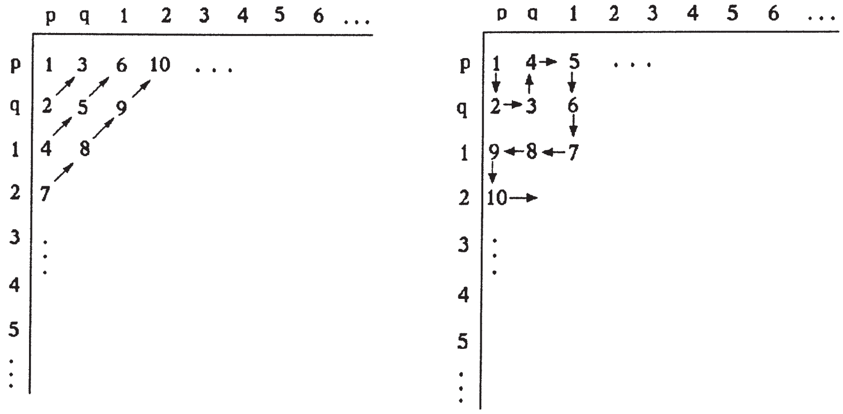
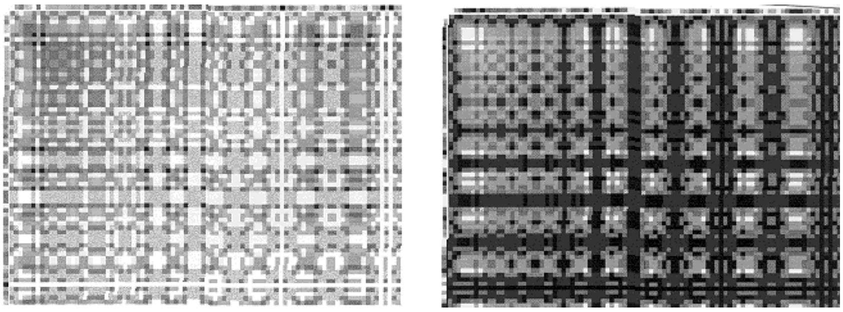


Figure 9.



NAND

NOR

Figure 10.

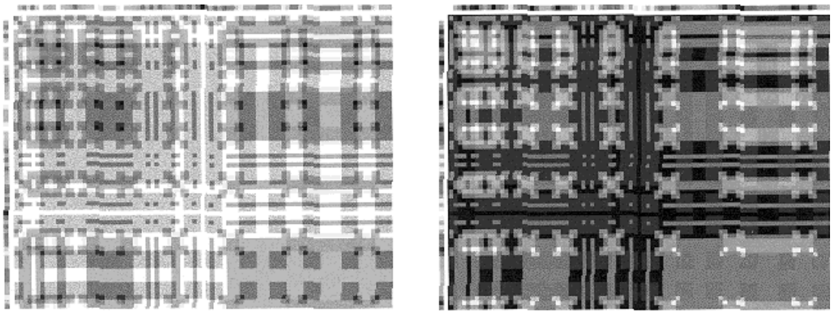


Figure 11.

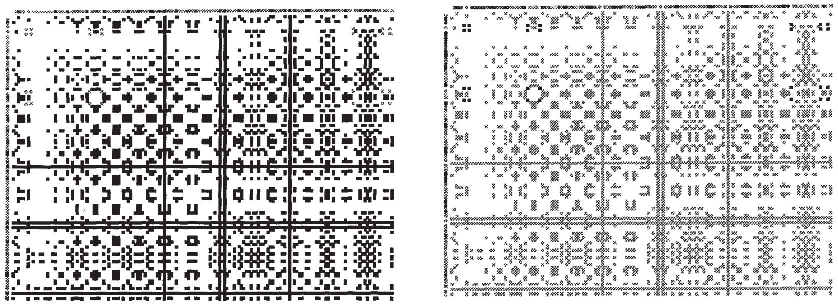


Figure 12.

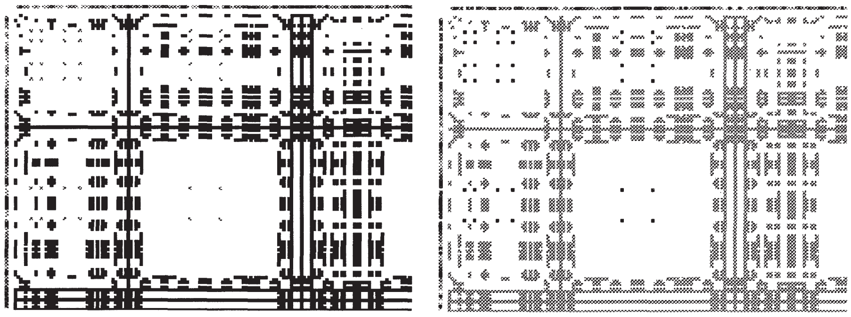


Figure 13.

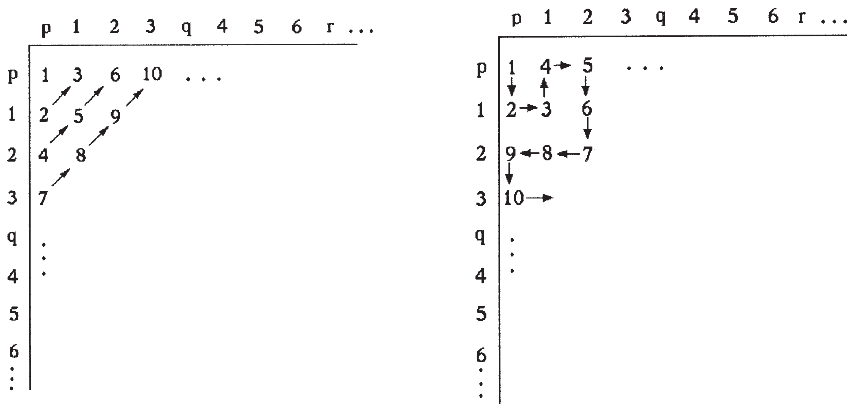


Figure 14.

13 we have compensated for this difficulty by eliminating all colors for various contingencies in a larger array, leaving only black for tautologies and grey for contradictions.[2] Figure 12 shows NAND and NOR for our

first pattern of enumeration; Figure 13 shows NAND and NOR for the second.

In theory any finite number of sentence letters can be added at the beginning of an array in the manner of the enumerations in Figure 9. For n sentence letters, however, the number of colors required to cover all contingencies is 2 raised to 2^n colors. At three variables, therefore, we have already hit 2^8 or 256 contingency colors. At four variables we hit 65,536.

In theory the full countable set of sentence letters required for standard propositional calculus might also be introduced along the axes, simply by adding an additional sentence letter at some regular interval (Figure 14). Because the standard propositional calculus is limited to finite connectives, we would here require countably many contingency colors as well.

Similar representations of formal systems are possible, beyond propositional calculus, for predicate calculus as well. One way of mapping a form of predicate calculus with multiple quantifiers but limited to monadic predicates, for example, would be the following. In a first grid we enumerate all combinations of one-place predicates and variables, giving us Fx, Fy, Fz, …, Gx, Gy, Gz, …. These we can think of as a series of propositional functions P1, P2, P3, …, which can be introduced into a grid for full propositional logic simply by placing them between our progressively introduced sentence letters – p, P1, q, P2, r, P3, … – in an expansion of an enumeration pattern such as those outlined in Figure 14. Quantification over formulae in variables $x, y, z, \ldots$ might then be introduced by adding spaced occurrences for $\forall x, \forall y, \forall z$ along just one axis. Here the application of a lone quantifier to formulae in its row could be interpreted as a universal quantification in that variable over that formula. All other intersections would be interpreted as before, in terms, for example, of the Sheffer stroke (Figure 15). Since existential quantification can be expressed in terms of universal quantification and negation, and the latter can be expressed by the Sheffer stroke in familiar ways, such a schema will include all wffs of predicate calculus involving only monadic predicate letters, together with all corresponding propositional formulae. In assigning values to such a grid, a special 'formula value' would have to be reserved for mere propositional formulae, representing the fact that they fall short of full formal sentences capable of truth values.

This illustration has been limited to monadic predicates simply because the scheme becomes so complicated so quickly even in that case. A representation of the full propositional calculus would demand only the

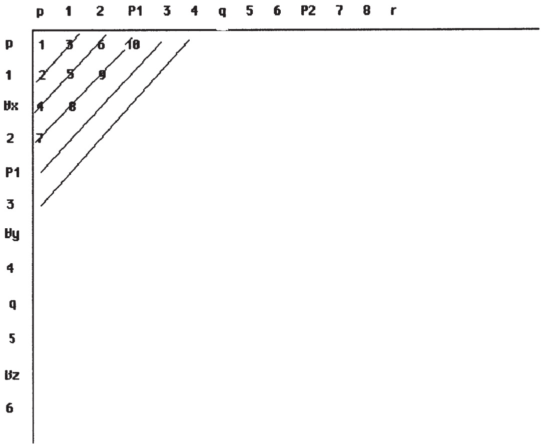


Figure 15.

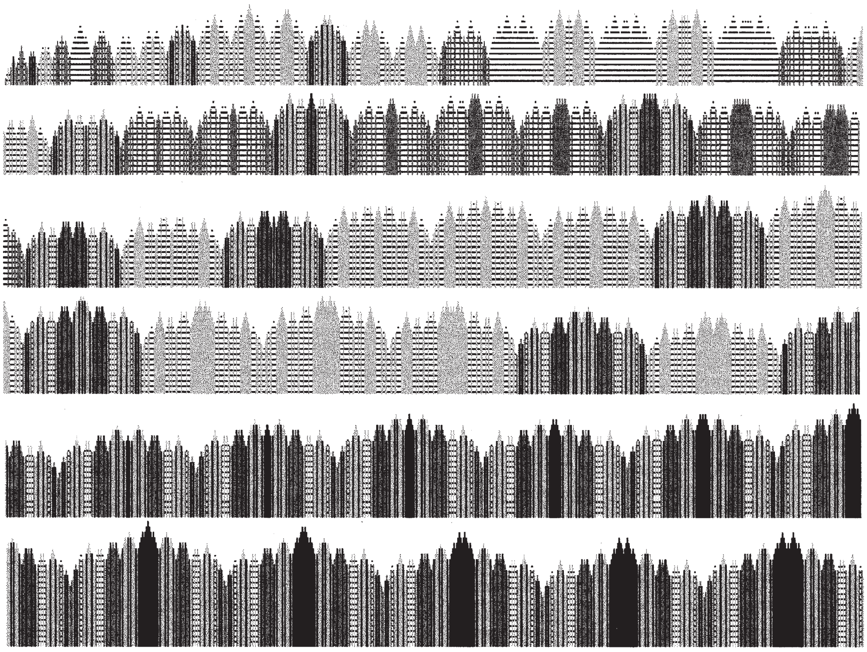


Figure 16.

further complication that we include all n-valued Predicate letters paired with n-tuplets of our variables. These can be generated in separate grids first so as to form a single enumeration, then introduced into the main grid in the position of P1, P2, P3, ... (Figure 15).

On seeing a dog walk on two legs, Abraham Lincoln is reputed to have said, “It’s not that he does it well, but that the thing can be done at all.” In something of the same spirit, the purpose of outlining the schema above is simply to shown that the thing can be done at all: that the full predicate calculus *can* be envisaged in the form of a grid of logical values of this kind. In even the simple case of propositional logic with a single sentence letter, however, we remarked on the artificiality introduced by arbitrary choices of enumeration for wffs. In the schema outlined for propositional calculus this artificiality is magnified many times over – by repeated arbitrary choices regarding forms of enumeration within a grid, by choices of how to incorporate different infinite classes of formulae on the axes, and by choices of how to incorporate quantification into the grid. The end product does succeed in showing that the thing can be done. But it should not be expected, we think, to give any particularly perspicuous view of the theorems of the calculus.

If we return for a moment to the simple form of propositional calculus with which we began, restricted to a single sentence letter, it should be clear that either of the enumerations offered above will generate progressively longer wffs. It is not true, of course, that the length of wffs within such an enumeration increases monotonically; formula 10 in our original enumeration is shorter than formula 9, for example. Along the diagonal of either schema, however, formulae do increase in size with each step.

How does such an enumeration look if we graph our formulae sequentially in terms of length with colors assigned for value? The beginning of such a result, using NAND and our first enumeration, is shown in the progressive panels of Figure 16. Colors used are the same as those in the rug patterns above except that tautologies are indicated in white with horizontal cross-hatching so that height will be visible. In the program used for generating this image, one can continue to flip through progressively longer wffs, with no apparent repeat of color patterns (program available on request).

4. Tautology fractals

The rug enumeration patterns offered above are perhaps the most direct way to attempt to model a complete formal system in terms of the values of its wffs. There is one major respect in which these patterns do *not* correspond to the fractal outlined for tic-tac-toe in introduction, however. That fractal is deep: all tic-tac-toe games are contained within the large initial square, though at decreasing scale. The rug patterns offered above,

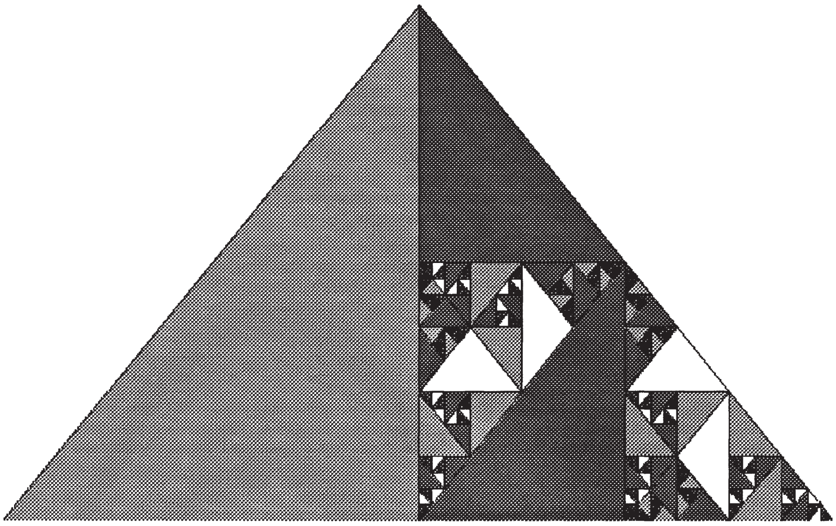


Figure 17.

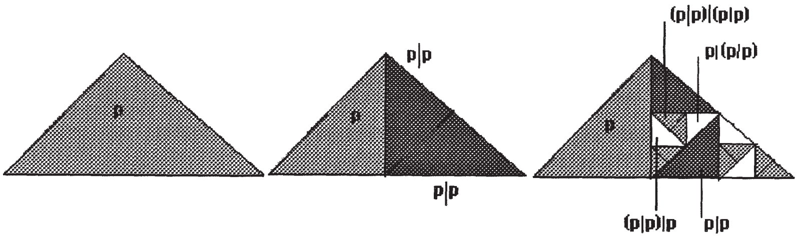


Figure 18.

on the other hand, are not in principle exhibitable in a finite space: all occupy an infinite plane extending without limit to the right and bottom.

It is also possible, however, to outline fractals for at least simple systems of propositional calculus which *are* deep in the sense of the tic-tac-toe game. In the case of systems with infinite wffs, of course, the corresponding fractal must be infinitely deep. For a simple form of propositional calculus with one sentence letter and a single connective $|$ such a fractal is shown in Figure 17.

The form of this fractal can most easily be outlined developmentally (Figure 18). We start from a single triangle occupying the whole space, representing the formula p and assigned light grey as the contingent value of p. We then take half of this space and divide it into two smaller triangles. One of these triangles is to represent Sheffer stroke formulae $(a \mid b)$ for the formula a of the divided triangle over all previous formulae

b. The other small triangle is to represent the symmetrical Sheffer stroke formulae $(b \mid a)$ for all previous formulae b over the present formula a. At the first step both of these amount to simply $(p \mid p)$, colored dark grey as a representation of the contingent value $\sim p$.

At the next step we take each of the new triangles thus created, divide half of their space into two, and embed in each of these smaller triangles an appropriately colored image of the whole – representing Sheffer stroke formulae $(a \mid b)$ of the present formulae over all previous formulae and Sheffer stroke formulae $(b \mid a)$ of all previous formulae over the present.

At each step a new set of more complex formulae are created, and at each step all Sheffer stroke combinations of elements of this new set with all previous formulae are embedded in the total image. Tautologies are colored white, as before, and contradictions black. All formulae of our simple formal system are thus represented with their value colors somewhere in the infinite depths of the fractal. Indeed many are represented redundantly – $(p \mid p)$ appears twice at the first step, for example (representing the present formula p over the previous formula p and vice versa), and later complexes with $(p \mid p)$ on either side will carry the redundancy further.

The complete fractal represents propositional calculus formulated in terms of the Sheffer stroke for a single sentence letter as a whole, infinitely embedded on the model of the tic-tac-toe fractal with which we began.

Modelling in terms of theorem fractals can be extended to more than a single sentence letter by starting with a larger number of initial areas: an initial triangle with two major divisions for p and q, for example, with three for p, q, and r, and so forth. Any of these could then be subdivided precisely as before, once again embedding the whole image into each subdivision. If we wish, we can even envisage an initial triangle with room for infinitely many sentence letters arranged Zeno-style in infinitely smaller areas. The embedding procedure would proceed as before, though of course each embedding would involve the mirroring of infinitely many areas into infinitely many. We haven't yet tried to extend such a pattern to quantification.

5. THE SIERPINSKI TAUTOLOGY MAP

In this section we outline another way of visualizing simple formal systems, which to our surprise turned out to offer intriguing links to classic patterns in fractal geometry.

In the rug patterns above we graphed an enumeration of formulae for simple forms of propositional calculi, coloring the grid locations of formulae in terms of their values. For forms of propositional calculus with n sentence letters, we noted, there will be 2 raised to 2^n such colors or values – essentially, a color for each possible truth table of length 2^n. Here we consider a different type of display for such systems, constructed using those values themselves on the axes rather than enumerated wffs. This frees us from particular enumerations of formulae because it frees us from the formulae themselves; the value space is constructed not in terms of particular formulae but in terms of the values of equivalence classes of formulae.

Consider two sentence letters p and q in standard truth-table form,

p	q		p	q
T	T		1	1
T	F		1	0
F	T		0	1
F	F		0	0

and sentences containing only the sentence letters p and q. Each such sentence will have a four-line truth-table, one of sixteen possible combinations of T's and F's, or 1's and 0's. These include solid 0's, corresponding to a contradiction or necessary falsehood; solid 1's, corresponding to a tautology; the pattern 1100, corresponding to the value of p; and the pattern 1010, corresponding to q. The sixteen values for two sentence letters can be thought of simply as all four-digit binaries.

These can be arranged in ascending order along the two axes of a two-dimensional display. Following the approach above we can think of these values as distinguished by color as well (Figure 19). In the illustrations that follow we will use only this color code on axes.

Combinatorial values for any chosen binary connective can now be mapped in the interior value space. If our value map is that of the Sheffer stroke, for example, the value of $(\bot \mid \bot)$ will appear at the intersection of 0000 and 0000; the value of $(\top \mid p)$ at the intersection of 1111 and 1100, etc. In terms of the colors on our axes the complete graph for the Sheffer stroke appears in Figure 20.

It is clear that a Sheffer stroke between $\bot$ and 0000 or any other value amounts to a tautology. In Figure 20, 0000 is represented using the darkest grey, tautologies are shown in black, and the fact just noted is represented by black values representing tautology in all cases along the left column and along the top row – all cases in which a value of 0000 appears on either side of the stroke. A Sheffer stroke between two tautologies, on the other hand, amounts to a contradiction, indicated by the dark grey square at the intersection of two black axis values in the

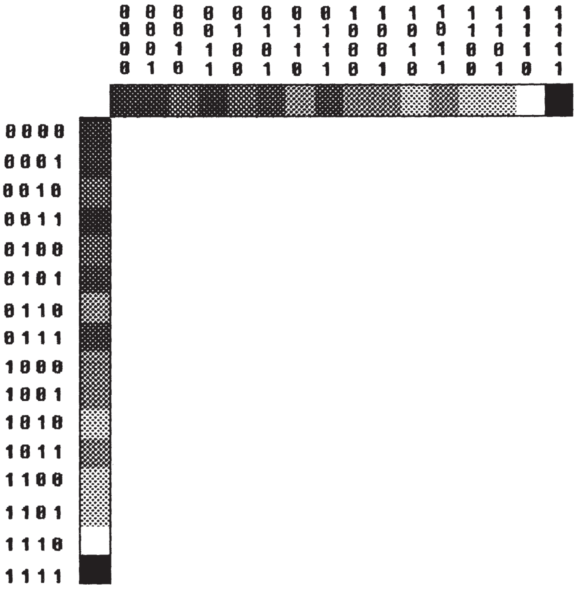


Figure 19.

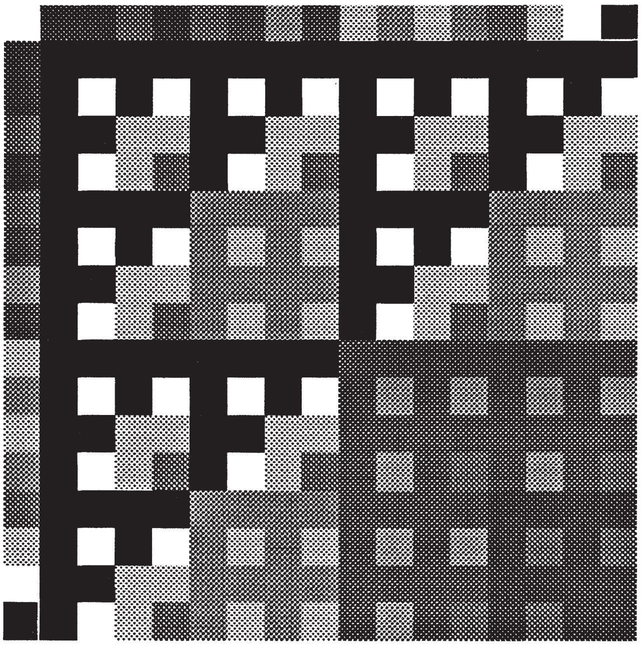


Figure 20.

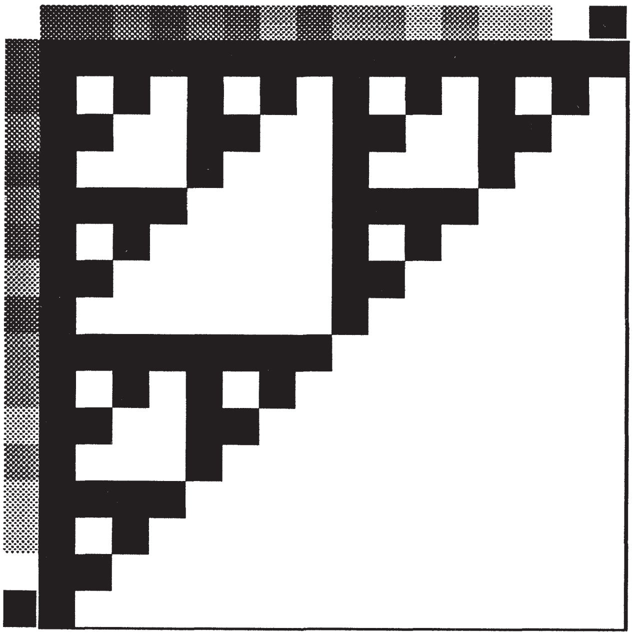


Figure 21.

lower right corner. As a whole the graph represents the value space for all Sheffer stroke combinations of our sixteen values.

A particularly intriguing feature of the value space appears more dramatically if we emphasize tautologies in particular by whiting out all other values (Figure 21). The fractal pattern formed here in black is that of the Sierpinski gasket, which has long been a primary exhibit in fractal geometry.[3]

If we expand our value space to that of three variables, with 256 values corresponding to all eight-digit binaries, an even finer representation of the Sierpinski gasket appears (Figure 22).

At any number of variables, given a standard listing of binaries corresponding to truth-table values, the tautologous Sheffer combinations will form a Sierpinski gasket. As indicated below, we can in fact think of diagrams with increasing numbers of sentence letters as increasingly finer approximations to a full system, with infinitely many sentence letters and infinitely many values. For that diagram, the tautologies of the system would form an infinitely fine Sierpinski dust.

The main connective of Figures 20 through 22 is NAND or the Sheffer stroke. A similar display for NOR, or the dagger, appears in Figure 23. Here there is only one tautology, at the intersection of 0000 and 0000.

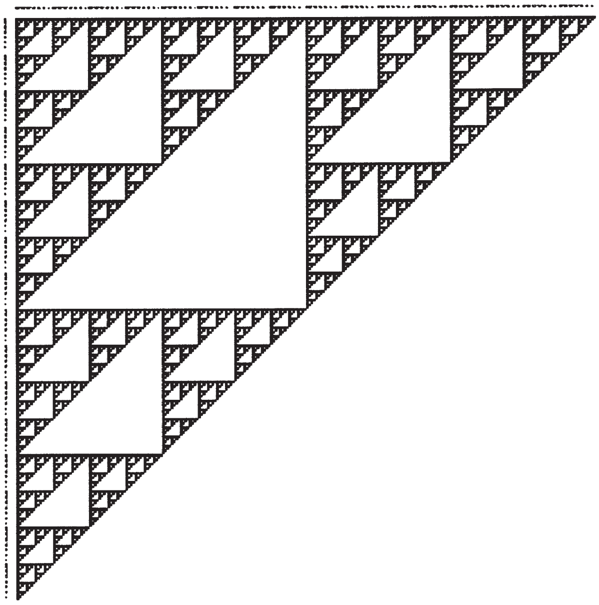


Figure 22.

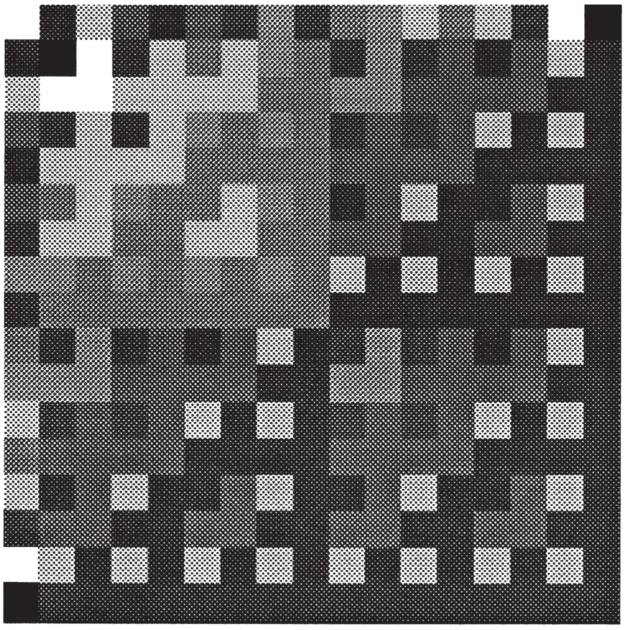
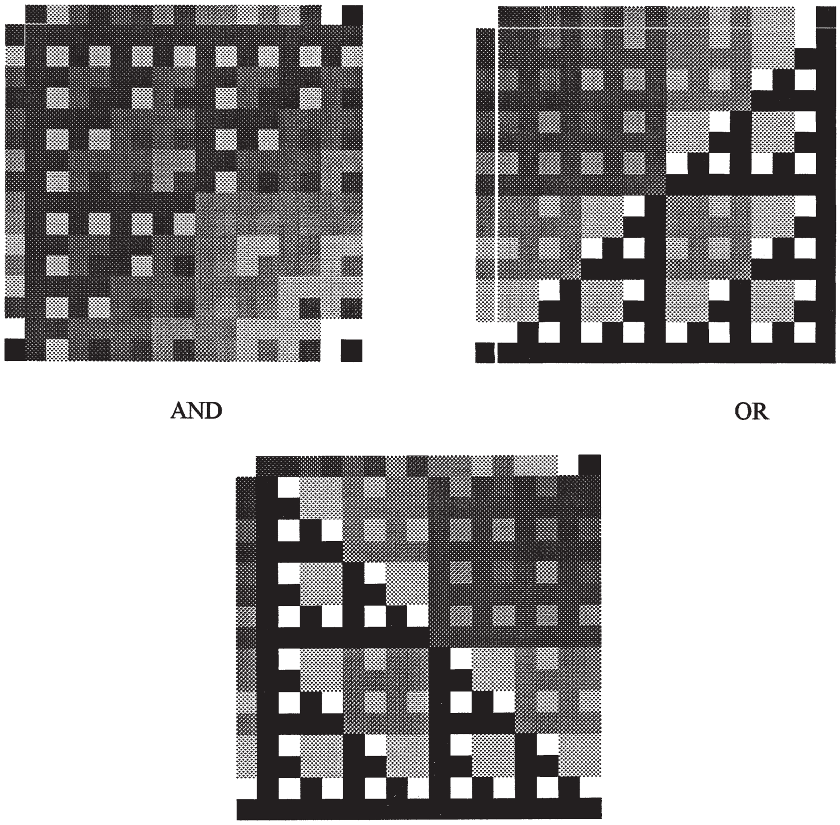


Figure 23.



AND

OR

MATERIAL IMPLICATION

Figure 24.

The Sierpinski gasket does show up again, however, as a graph of contradictions in the lower right-hand corner.

Other connectives generate other patterns in value space. A standard ‘and’ and ‘or’, for example, are shown in Figure 24. In the case of ‘and’ the persistent image of the Sierpinski gasket appears in the upper left as a value pattern for contradiction; in the case of ‘or’ it appears in the lower right as a value pattern for tautology. In material implication the Sierpinski triangle shifts to the lower left as a value pattern for tautology.

In the course of our research the appearance of the Sierpinski gasket within the value space of propositional logic came as a surprise. But its appearance can easily be understood after the fact.

As indicated above, we can think of value space displays for forms of propositional calculus with increasing numbers of sentence letters as approximations to a fuller system. As long as we have some finite number of sentence letters n we will have finitely many value spaces, corresponding to all possible truth tables of length 2^n. But the full propositional calculus has a countably infinite number of sentence letters. Because standard propositional calculus is limited to wffs of finite length, it turns out, we can construct a value space for it using something like truth-tables of infinite length.

This is less difficult than it may at first appear. In constructing finite truth-tables for n variables the standard procedure is to start with a sentence letter represented as 0101… to length 2^n, to represent the next sentence letter with 00110011… to that length, the third with 00001111…, and so forth. For infinite truth-tables adequate to finite complexes of countably many sentence letters our first sentence letter p can be thought of as having an infinite truth-table that starts 01010101… . Our second sentence letter q can be thought of as having the infinite truth-table that starts 00110011… , our third sentence letter r as having the infinite truth-table 000011111… , and so forth. Each of our sentence letters, in other words, can be thought of as having infinitely periodic truth-tables which otherwise follow the standard scheme used for constructing truth-tables of finite length. There will always be room for 'one more' sentence letter because it will always be possible to introduce a larger period of 0's and 1's for the next sentence-letter needed. Sentence-letters of a full form of propositional calculus can thus be thought of as corresponding to a subset of the periodic binary decimals: those which alternate series of 0's and 1's of length 2^n for some n.

Any set of values for any finite set of sentence letters will have an appropriate line in this infinite extension of truth-tables, and in fact will have a line that will itself reappear an infinite number of times. Because the infinite truth-tables for our sentence letters are periodic in this way, complex sentences formed of finitely many connectives between finitely many sentence letters will be periodic as well. The largest period possible for a complex sentence of this sort will in fact be the longest period of its sentence-letter components. All values for the full propositional calculus will thus be represented by periodic binary decimals. It is important to note, however, that not all periodic binary decimals will have a corresponding formula; those periodic in multiples of 3, for example, will not be producible by finite combination from sentence-letters periodic in powers of 2.[4]

The important point here is simply that any value space for finitely many sentence letters can be thought of as an approximation to this richer system, adequate to propositional calculus as a whole. In the rich-

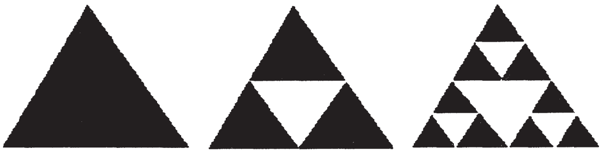


Figure 25.

er system, of course, the squares of the value spaces illustrated above shrink to mere points in value space, just as values on the axes shrink to mere points on the continuum. Although these points do not by any means exhaust the full [0,1] interval – they constitute merely a subset of the periodic decimals – they can be envisaged as embedded within it. The argument below regarding the appearance of the Sierpinski gasket applies to a full continuum as well as the envisaged subsets, and will be valid both for the full propositional calculus and for the envisaged approximations to it.

In terms of NAND or the Sheffer stroke the appearance of the Sierpinski gasket can be outlined as follows. Similar explanations will apply for the other connectives.

Let us emphasize that the binary representations of values on each axis of our value spaces, whether finite or infinite, correspond to columns of a truth-table. The value assigned to any value space or point **v** is a function of the truth-table values from which it is perpendicular on each axis. In asking whether a point in the value space represents a tautology in a graph for NAND, for example, what we're really asking is whether the truth-tables of these two axis values share any line in which both show a '1'. If there is such a line, their combination by way of NAND is not a tautology. The value point **v** will have the value of a tautology if and only if its axis values at no point show a '1' on the same line.

Consider now one standard route to the Sierpinski gasket, which generates the gasket from a triangle in terms of a rule for doubling distance from the nearest vertex.[5] For any given triangle, there is a set of points which, when distance is doubled from the nearest vertex, will be 'thrown' outside of the triangle itself – more precisely, which will map under doubling from the nearest vertex to points outside the triangle itself. These points in fact form an inverted triangle in the center (Figure 25). There are a further set of points which, when distance is doubled from the nearest vertex, will be thrown into this central region – and thus which will be thrown out of the triangle upon *two* iterations of the 'doubling from nearest vertex' procedure. The Sierpinski gasket is composed of

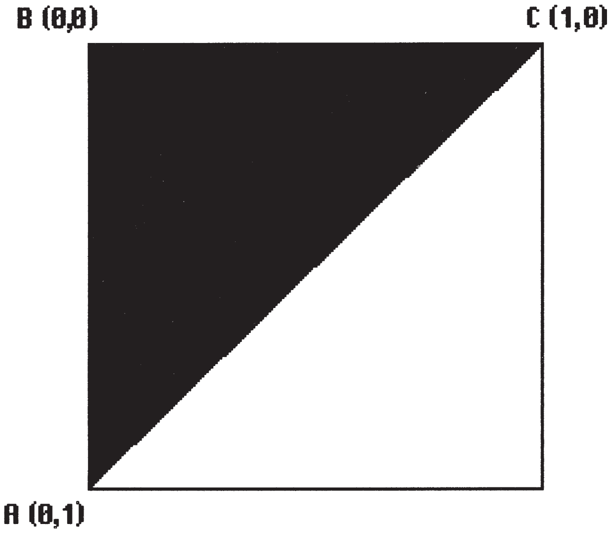


Figure 26.

all those points which will remain within the triangle despite unlimited iteration of such a procedure.

It turns out that this route to the Sierpinski gasket corresponds quite neatly to its appearance as a map of theorems in the value space for NAND.

Consider the diagram of a unit square in Figure 26, and the upper triangle marked between A (0,1), B (0,0), and C (1,0). This 'inverted' form of the unit square corresponds to our axes for value spaces above. Were we to characterize the rule of doubling the distance from the closest vertex in terms of x and y values for particular points within this triangle, our rules might be rendered as follows:

$$\begin{aligned}
&\text{If } x \leqslant 1/2,\ y \leqslant 1/2,\ (x_n, y_n) = (2x, 2y)\\
&\text{If } x \leqslant 1/2,\ y > 1/2,\ (x_n, y_n) = (2x, 1 - 2(1 - y))\\
&\text{If } x > 1/2,\ y > 1/2,\ (x_n, y_n) = (1 - 2(1 - x), 2y).
\end{aligned}$$

These will give us the Sierpinski gasket by the standard route of doubling the distance from the nearest vertex.

Here it's clear that 'doubling the distance' is in all cases a matter of either multiplying an axis value by 2 or substracting 1 from a multiplication by 2. But now let us envisage the axes of our unit square as marked in terms of binary decimals. For binary decimals multiplication by 2 simply involves moving a decimal point one place to the right. $1 - 2(1 - x)$ equals $2x - 1$, which moves the decimal point one place to the right and ‘lops off’ any ones that thereby migrate to the left of the decimal point. The crucial point is that for binary decimal expression of axis values, both forms of transform preserve the order of digits which remain beyond the decimal point. Iterated application of such transforms to pairs of values (x, y) thus effectively moves down each series of binary digits one at a time, checking for whether a 1 occurs in both places. If it does, our iterated transforms have resulted in two values both of which are greater than 1/2 as expressed in binary, and the point will therefore have migrated under iteration outside the region of the dark triangle.

The points of the triangle ABC which will not migrate out under an iterated procedure of doubling the distance from the nearest vertex – the points of a Sierpinski triangle in that upper region – are therefore those points (x, y) such that the binary representation of x and y do not both have a 1 in the same decimal place. Given our representation of values in terms of binary decimals, those points which generate tautologies under NAND will be precisely those same points: points with axis values with no 1’s in corresponding truth-table lines, or equivalently with no 1’s in corresponding decimal places of their binary representation. The initially startling appearance of the Sierpinski gasket as a map of tautologies under the Sheffer stroke can thus be understood in terms of (i) what a binary representation of values means and (ii) a corresponding rendering of a familiar ‘doubling the distance from the nearest vertex’ route to the Sierpinski in terms of binaries. An outline for the appearance of the Sierpinski in the value space of other connectives can be drawn along the same lines.

The standard ‘distance doubling’ route to the Sierpinski does involve a full-continuum unit square. As indicated above, even the full propositional calculus has a value space short of that full continuum; although each sentence letter and each connective corresponds to an infinite decimal, these form a subset of even the merely periodic decimals. None of that, however, affects the basic mechanism of the argument above, which turns merely on the question of whether two decimals, finite or infinite, share a particular value at any place. Multiplication by 2 from the closest vertex simply ‘checks’ them place by place. Thus the fact that our value space for the forms of logic at issue constitute mere subsets of the full unit square gives us simply the result that tautologies in the case of NAND, for example, will constitute an infinitely fine Sierpinski dust within that grainy unit square.

One of the promises of a graphic approach to formal systems of this sort is that there may be results of fractal geometry that can be read off as facts about the logical systems at issue. Here the appearance of the

Sierpinski gasket as a map of theorems in value space offers a few minor but tantalizing examples.

It is well known that the points constitutive of the Sierpinski gasket within a continuous unit square are uncountably many, but nonetheless 'very few' in the sense that points chosen at random within the unit square have a probability approaching zero of being in the Sierpinski set. Much the same will be true within the full propositional calculus; there will be infinitely many complexes with the value of tautology in such value space, but the probability of hitting a tautology by a Sheffer combination of random axis values will approach zero.

A similar point can be expressed in terms of area. Within any finite approximation to an infinitely fine-grained unit square, the Sierpinski gasket will retain a definite area. Within any value space limited to n sentence-letters, tautologies will retain a similar area of values space. In the case of an infinitely-grained unit square, on the other hand – whether fully continuous or not – the Sierpinski gasket has an area approaching 0. Within the full propositional calculus the relative area of tautologies will similarly amount to zero. In terms of the Sheffer stroke, tautologies end up distributed as unconnected points within value space on the model of three-dimensional Cantor dust.[6]

For smooth curves, an approximate length $\mathrm{L}(r)$ can be given as the product of the number N of straight-line segments of length r required to 'cover' the curve from one end to the other. As r goes to zero, $\mathrm{L}(r)$ approaches the length of the curve as a finite limit. For fractal curves, on the other hand, it is standard for $\mathrm{L}(r)$ to go to infinity as r goes to zero, since what are being 'covered' are increasingly fine parts of the curve. A standard measure d of the intricacy of fractal curves, known as the Hausdorff dimension, is that exponent d such that the product N* r^d stays finite. The Hausdorff dimension of the Sierpinski gasket is known to be $\log 3/\log 2 \approx 1.58$. The work above suggests a related value for tautologies within the value space of the Sheffer stroke.

6. Value solids and multi-valued logics

A slight variation in the representation of the value spaces outlined above offers an intriguing comparison with a way of envisaging connectives in multi-valued logics, including infinite-valued or fuzzy logics.

Rather than graphing values in our value space in terms of color, the use of binary decimals makes it easy to graph them in terms of height

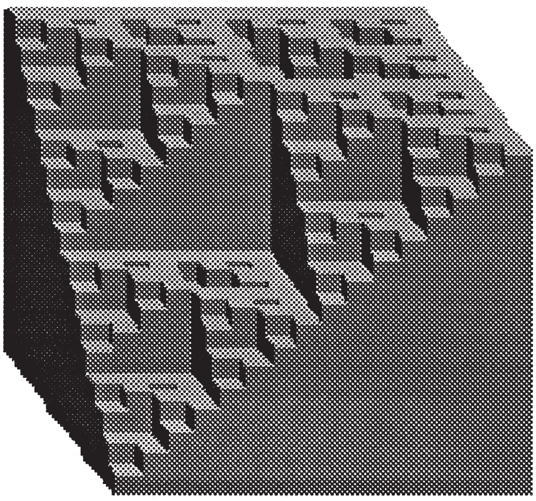


Figure 27.

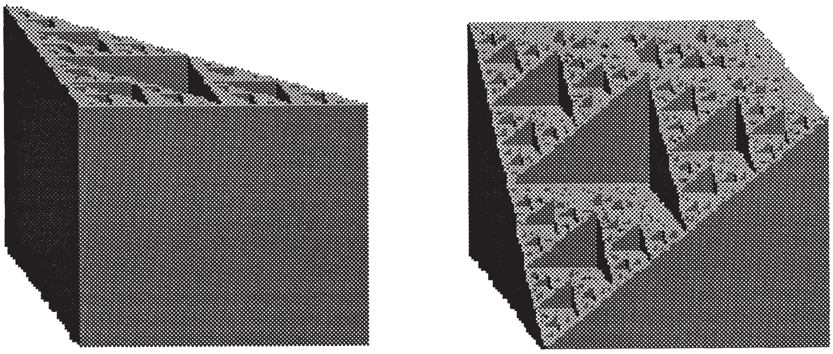
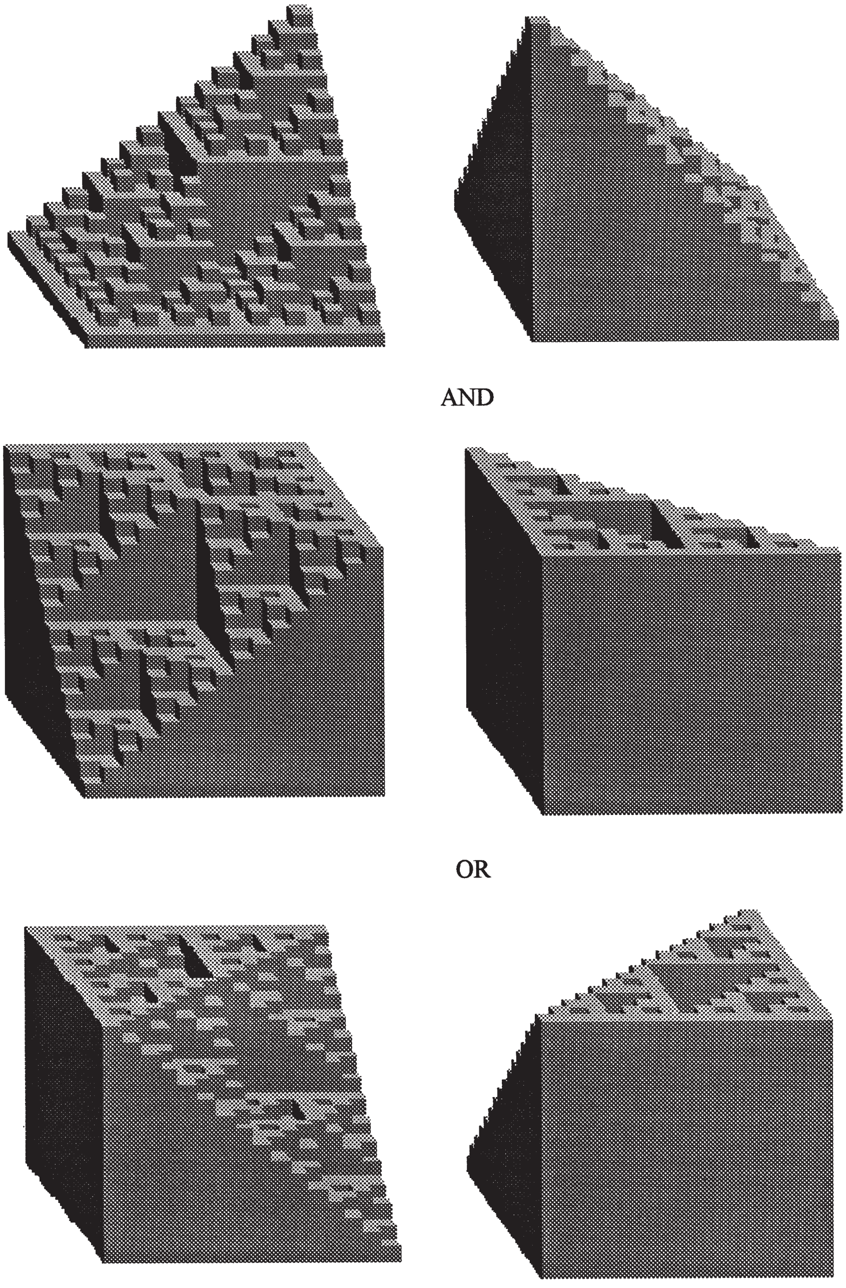


Figure 28.

in a third dimension. A value of .0000 will graph as 0, a value of .1000 as the decimal equivalent .5, .1100 as .75, and so forth.

A fairly rough graph of this sort for NAND, seen from a particular angle, appears in Figure 27. This corresponds directly to Figure 20, though here the origin is in the right rear corner. Smoother forms of the value solid for NAND, from two angles, appear in Figure 28. Because the rough solids are often more revealing of basic structure, however, we will continue with these throughout.

Value solids for conjunction, disjunction, and material implication appear in Figure 29. In each case the origin is shown in the left figure at the front left, and in the right figure at the rear right.



AND

OR

IMPLICATION

Figure 29.

These value solids make obvious the relationships between NAND and OR, the dual character of conjunction and disjunction, and the rotation properties of negation. Of perhaps deeper significance, however, these value solids for simple classical systems also show a striking resemblance to a very different type of solid that can be drawn for connectives within multi-valued or infinite-valued logics.

In this second type of solid, values on the axes represent not truth-table columns but degrees of truth. Height at a certain point represents the degree of truth of a complex of two sentences with the axis values of that point. In one standard treatment of infinite-valued connectives, for example, the value of a conjunction of sentences p and q is the minimum value of the two, represented as:

$$/p \,\&\, q/ = \text{min}(p, q).$$

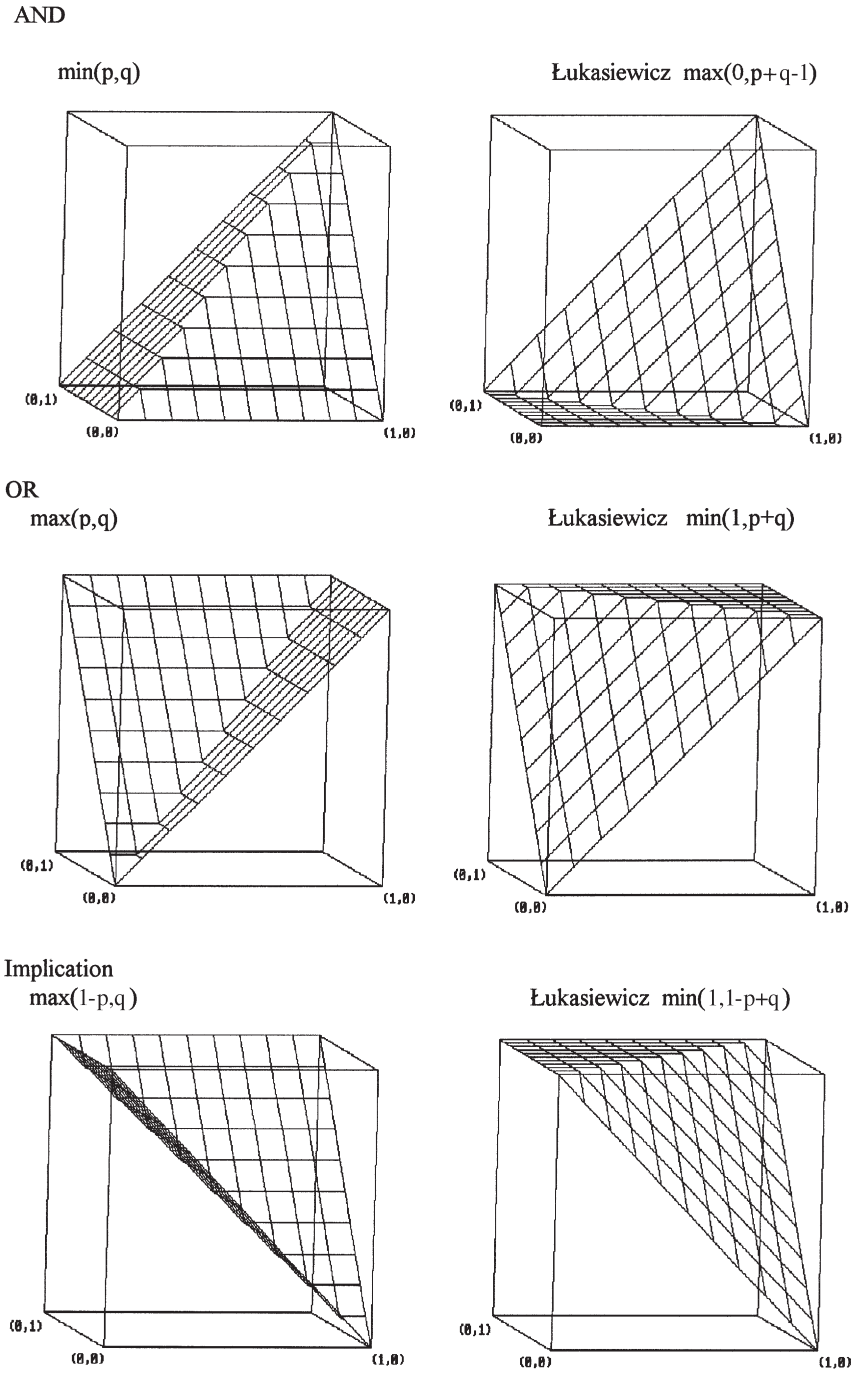
The value solid of this type for conjunction will thus at each point have a height corresponding to the minimum of its axis values.[7]

There are, however, rival sets of connectives that have been proposed for multi-valued and infinite-valued or fuzzy logics. One such set, perhaps most common within multi-valued and fuzzy logics, is shown in the left column of Figure 30. Another set, grounded more directly in the original multi-valued logic of Łukasiewicz,[8] is shown in the right column.

It should be emphasized that the value solids appropriate to connectives in infinite-valued logic are radically different from the value-solids for systems outlined above. In system value solids, for example, .1000 might represent a truth-table in which the first line has a ‘T’ and the others do not. In that regard it is perfectly symmetrical to .0001, which simply has a ‘T’ on a different line. Using similar binary decimals for the values of sentences in an infinite-valued logic, on the other hand, a statement with the value .1000 would be half true. One with a value of .0001 would be almost completely false.

Given that radical difference, the value solids outlined here for classical systems and those sketched in Figure 30 for infinite-valued logics seem much more alike than they have any right to be. Intriguingly, the system-solid for each connective seems to be embody a compromise between the corresponding infinite-valued connective solids. The system-solid for ‘or’, for example, amounts neither to ‘max’ nor to the Łukasiewicz ‘or’. It rather appears to be a compromise, in which some values correspond to one treatment of the infinite-valued connective and one to another.

Indeed this is precisely what is happening. How it occurs – and why there is such a resemblance between these two radically different kinds of



AND

min(p,q)

Łukasiewicz max(0,p+q-1)

OR

max(p,q)

Łukasiewicz min(1,p+q)

Implication

max(1-p,q)

Łukasiewicz min(1,1-p+q)

Figure 30.

value solid – becomes clear if we return to two dimensions and consider a simple form of our basic value grid.

In a system grid for ‘or’, in which we are calculating the truth-table values for an ‘or’ between truth-table values on the axes, the value assigned to any intersection point is what might be called a ‘bitwise or’ of the values on the corresponding axes. A ‘1’ occurs in any row in the value of that intersection point just in case a ‘1’ occurs in that row in one or the other of the corresponding axis values. In bitwise ‘or’ the 1’s cannot of course add together and carry to another row:

$$\begin{array}{cclcc} 0 & 0 & & & 0 \\ 1 & 1 & & & 1 \\ 1 & 0 & \text{bitwise or} & = & 1 \\ 0 & 1 & & & 1 \end{array}$$

The values assigned in a system grid for ‘or’, then, correspond to a bitwise ‘or’. The values assigned to intersection points in an infinite-valued grid will be more complicated, amounting to either the maximum of the axis values p and q or, in the case of the Łukasiewicz ‘or’, to $\min(1, p + q)$. Nonetheless these three values for intersection points will occasionally overlap.

In the simple case of three-digit binary decimals, in fact, where we take 111 as the closest approximation to 1 in the Łukasiewicz formula, every bitwise ‘or’ is equal to either max, the Łukasiewicz ‘or’, or both. This is reflected in the 8 by 8 grids shown in Figure 31. On the left are mapped those intersection points in which a bitwise ‘or’ corresponds to ‘max’. On the right are mapped those intersection points in the grid in which bitwise ‘or’ corresponds to the Łukasiewicz ‘or’. Here it is clear (a) that the middle areas, exclusive of the edges, are the negatives of each other, (b) that together these two graphs will therefore cover the entire area of the grid, and (c) that each grid is composed of simple Sierpinski gaskets facing each other and rotated 90 degrees from the other graph. A value solid for bitwise ‘or’ geared to just three-digit binaries, then, would at each intersection point correspond precisely to one or the other of the two infinite-valued connectives outlined above: the 8-valued system solid constitutes a perfect Sierpinski compromise between the two infinite-valued solids.

The result does not generalize in this pure form to system- and infinite-valued solids of more than eight units on a side, however. In more complex cases the Sierpinski patterns persist, but their overlap fails to cover the entire area. For a grid of 256 values on each side, Figure 32 shows in black those intersection points in which bitwise ‘or’ will equal one or the other of our two infinite-valued connectives. The holes left are

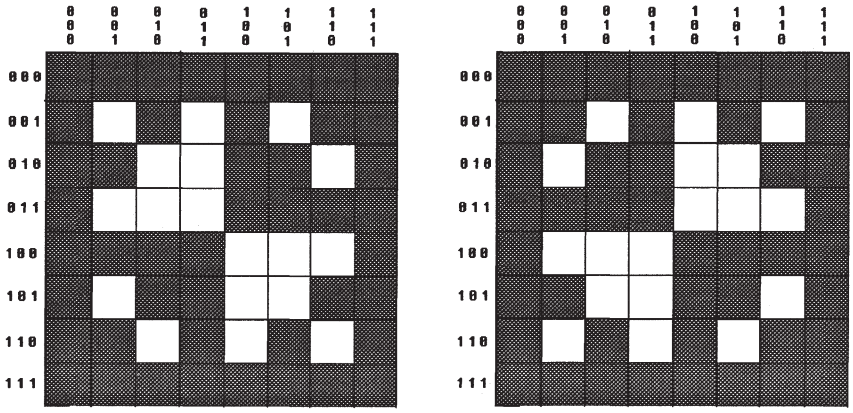


Figure 31.

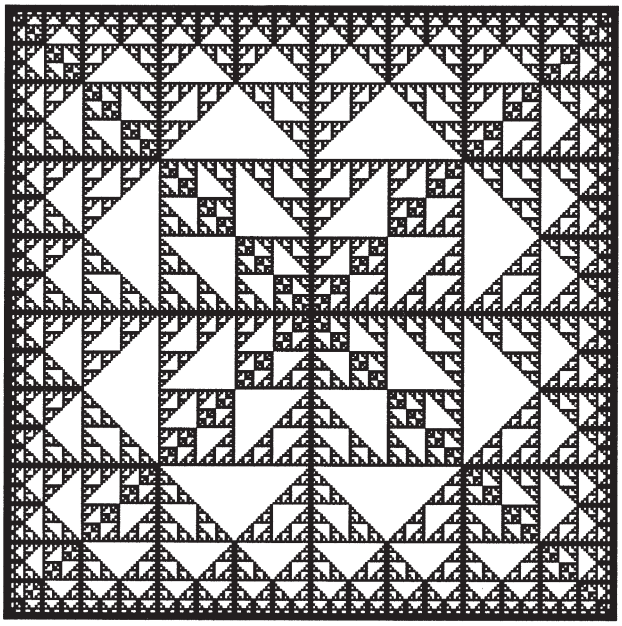


Figure 32.

the holes formed by facing pairs of Sierpinski gaskets overlying another pair and rotated at 90 degrees. Even in more complex systems a type of compromise remains, however. For in all cases the bitwise ‘or’ for an intersection point will equal either one of the two infinite-valued ‘or’s above or will have a value between them, less than the Łukasiewicz but greater than simply ‘max’. Similar compromises hold in the case of the other connectives. Thus in an intriguing way value-solids for simple systems do map a compromise between the quite different value-solids appropriate to rival connectives within infinite-valued or fuzzy logic.

7. Cellular automata in value space

Cellular automata consist of a lattice of discrete sites, each of which may take on values from a finite set. In classical (synchronous) automata the values of sites evolve in discrete time steps from an initial configuration s_0 in accordance with deterministic rules that specify the value of each site in terms of the values of sites in a chosen neighborhood n.

The two-dimensional value graphs outlined for systems above might also be thought of on the model of two-dimensional automata arrays of this type. Much to our surprise, we found that the distribution of values under particular connectives within such arrays can also be generated by simple automata rules.

Consider, for example, an array corresponding to a system value-space with 16 units along each axis, such as that shown in Figure 21. Here, however, we will be concerned only with the lattice of spaces itself. Each cell in such a lattice, with the exception of those at the edges, is surrounded by eight neighbors. We will be concerned in particular with just three of these neighbors, which we will term ‘southeastern’ neighbors and which are marked with x’s in the sketch below.

□ □ □
□   ☒
□ ☒ ☒

Let us start with a ‘seed’ in the lower right-hand corner of our sixteen-by-sixteen grid, consisting of one darkened square. Consider now the following cellular automata rule:

> A square will be darkened on the next round if and only if exactly one square in its southeast corner is. (Edge squares will be treated as having non-darkened neighbors at the edge.)

The series of steps in the evolution of a sixteen-sided array under this simple rule is shown in Figure 33. The surprising fact is that the squares occupied by black in each step in this evolution correspond precisely and in order to the values occupied by 0000, 0001, 0010, ... in our original value space for the Sheffer stroke (Figure 20). This simple cellular automaton, in other words, is ‘ticking off’ progressive values for NAND or the Sheffer stroke, plotted in value space. By the sixteenth

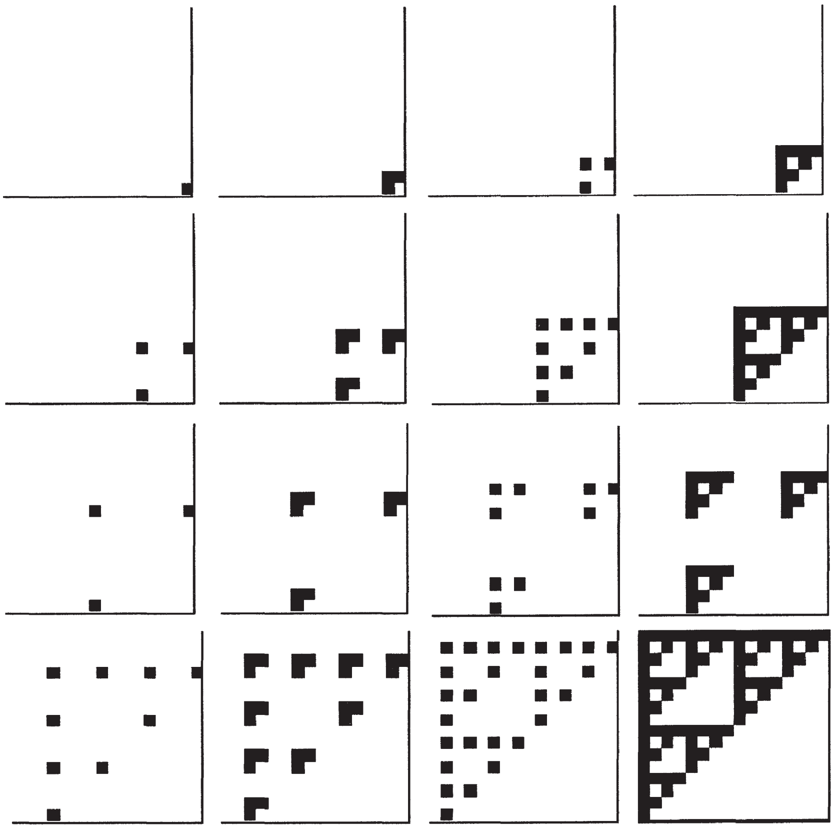


Figure 33.

step – for the value 1111 – the array evolves into the Sierpinski pattern for tautologies noted in Section 5.

An exactly similar progression through all values represented appears if we begin with 256 values on each side instead of 16. This same simple automata rule, in fact, generates progressive values in the proper places for a value space corresponding to NAND regardless of the number of cells in our space: for any finite approximation such an automaton is in effect constructing a value space for a limited form of propositional calculus.

Other equally simple automata will generate value spaces for the other connectives outlined above. With precisely the same rule and starting point, but thinking of our values in reverse – from 1111 to 0000 in the case of a 16-sided value space, for example – the value space generated step by step is that of conjunction. The value space for disjunction is generated by beginning in the upper left hand corner with the value 0000 following a second rule, symmetrical to that above:

$$\begin{array}{ccc} \boxtimes & \boxtimes & \square \\ \boxtimes & & \square \\ \square & \square & \square \end{array}$$

A square will be darkened on the next round if and only if exactly one square in its northwest corner is.

This second rule and starting place, thought of as enumerating values in order from 1111 through 0000, generates the value space for NOR or the dagger. A further change in rule and beginning position give us a cellular automaton adequate to implication.

A bit of thought shows that indeed these rules must generate the progressive values noted within the lattice of any value space. Consider as a single example the case of ‘or’, beginning from the upper left corner with the second rule above. The ‘or’ of the system-value grid, it will be remembered, is what we have termed a ‘bitwise or’, giving a ‘1’ in any row just in case at least one of its disjuncts has a 1 in that row. Regardless of the number of binary digits in our value representation, it should also be noted, each step along the axis amounts to addition by 1: our values are listed in binary sequence . . . 000, . . . 001, . . . 010, and so forth. What we want to show for the general case, therefore, given axes numbered in binaries of any given number of digits, is that the central cell marked D below will take a binary value of $n + 1$ if and only if precisely one of the cells marked with an $\times$ takes a value of n.

$$\begin{array}{ccc} \boxtimes & \boxtimes & \square \\ \boxtimes & \mathrm{D} & \square \\ \square & \square & \square \end{array}$$

We first show, left to right, that if just one of the squares marked x has a value n, y must have the value $n + 1$. Consider to begin with the case in which it is $\mathbf{A}$ that is the square with value n, using x and y to represent the axis values which combine in a bitwise ‘or’ to give us $\mathbf{A}$. Axis values for D are then of course $x + 1$ and $y + 1$.

$$\begin{array}{cccccc} & & & x & & \\ & & & \cdot & & \\ & & & \vdots & & \\ y & \cdot & \cdots & \mathbf{A} & \mathrm{B} & \square \\ & & & \mathrm{C} & \mathrm{D} & \square \\ & & & \square & \square & \square \end{array}$$

In this case, since B does not have the value n, the bitwise compound 'y or $x+1$' must have a different value from bitwise 'y or x'. (Unless specified otherwise, we will use simply 'or' for bitwise 'or' throughout.) Since C has a value other than n, 'x or $y+1$' must similarly differ from 'y or x'. If either x or y ends in 0, then, both must end in 0: were only one to end in 0, addition to that one would not change the value of their bitwise 'or', and thus either B or C would equal **A**, contrary to hypothesis. The same argument applies not only to a 0 in the last digit position but in any first position counting from the right: x has a first 0 in a given position counting from the right if and only if y also has a first 0 in that position. Otherwise either B or C would equal **A**, contrary to hypothesis.

Either x and y will contain no zeros, therefore, or will share a 0 in the same first position from the right. If neither contains zeros, **A** occupies the lower right-hand corner of the lattice and there is no place for D; the position exhibited does not form a part of our lattice. In all other cases x and y share a 0 in the same first position from the right. Adding 1 to each of x and y – moving along the axes from x $x+1$ and from y to $y+1$ – will close that 0 with a 1, changing all 1's to its right to 0's in each case. The series of digits represented by x and y will stay the same in all other regards. A bitwise 'or' between $x+1$ and $y+1$ will therefore give us an increase of precisely 1 over the value of the bitwise or between x and y: given a value of n for **A**, D will take a value of $n+1$.

Consider secondly the case in which it is **B** that carries the value n, once again using x and y to represent A's axis values:

$$\begin{array}{cccccc}
 & & & x & & \\
 & & & . & & \\
 & & & \vdots & & \\
y & . & \ldots & \mathbf{A} & \mathrm{B} & \square \\
 & & & \mathrm{C} & \mathrm{D} & \square \\
 & & & \square & \square & \square
\end{array}$$

Since $\mathbf{B} \neq \mathrm{C}$, x and y cannot share either a final 0 or a rightmost 0 in the same place. If they did, addition of 1 to either would produce the same change from A in a bitwise 'or', giving us $\mathbf{B} = \mathrm{C}$, contrary to hypothesis. One of x and y, then, has a rightmost 0 farther to the right than the other. Since $\mathbf{B} \neq \mathrm{A}$, it must be y that has a 0 furthest to the right: otherwise x's furthest right 0 would be 'masked' by 1's in y, and thus the bitwise ($x+1$ or y) would equal that of (x or y), contrary to our hypothesis that $\mathbf{B} \neq \mathrm{A}$.

In this case x and y therefore have the form:

$$\begin{array}{ll} x: & \ldots\ 111 \\ y: & \ldots\ 011 \end{array}$$

for some number of 1's (perhaps none) to the right of y's 0. It is clear, therefore, that $x + 1$ and y side by side will have the form:

$$\begin{array}{rl} x+1: & \ldots\ 000 \\ y: & \ldots\ 011 \end{array}$$

since addition of 1 to x will have changed some zero to the left of y's to a 1 with all 1's to its right changed to 0's. $\mathbf{B}$'s value is that of a bitwise 'or' between these two. But then it is clear that adding 1 to y will result in an increase of precisely 1 for the bitwise compound ($x + 1$ or $y + 1$). Thus if $\mathbf{B}$ is the cell with a value of n, D must again take a value of $n + 1$. A symmetrical argument shows that if it is C that is the single northwest cell with value n, D must again take a value of $n + 1$.

For the case of 'or' we have shown that if precisely one of the cells northwest of any D has a value of n, D must itself take a value of $n+1$. It suffices for the rest of our justification to show that if a cell D has a value $n+1$, one and only one of its northwest cells must have a value n.

$$\begin{array}{cccccc} & & & x & & \\ & & & \cdot & & \\ & & & \vdots & & \\ & & & \mathrm{A} & \mathrm{B} & \square \\ y & \cdot & \ldots & \mathrm{C} & \mathrm{D} & \square \\ & & & \square & \square & \square \end{array}$$

We specify that D has a value $n+1$, generated as the bitwise compound (x or y). Subtraction of 1 from either x or y amounts to changing its rightmost 1 to a 0 and all 0's from there to the right to 1's.

Suppose now that x and y have a rightmost 1 in the same position. In that case substracting 1 from each will result in a substraction of 1 from bitwise (x or y), and thus A – representing ($x - 1$ or $y - 1$) – will have the value n. Substraction of 1 from just one of these, however, cannot result in n. In that case a rightmost 1 in either x or y will change to a 0, but the other will have a rightmost 1 which masks that change in terms of the bitwise 'or'. What will change in the bitwise 'or' is that all digits to the right of that place (if any) will change from 0 to 1. Since this can only represent a figure equal to or greater than $n + 1$, however, it cannot equal n.

Suppose secondly that x has the furthest 1 to the right: that y has a 0 in that position and at all places to the right. Substracting 1 from x will then change its rightmost 1 to 0 and all 0's to its right to 1's. Because y has only 0's from that position to the right, the change from bitwise (x or y) to ($x - 1$ or y) will be precisely the same, representing a substraction of 1, and thus it will be C that has a value of n.

In this case substracting 1 from only y or from both x and y could not result in n. Substraction of 1 from $n + 1$ demands that the rightmost 1 in $n + 1$ be changed to a 0, with all 0's to its right (if any) changed to 1's. Given our hypothesis, however, the rightmost 1 in $n + 1$ must correspond to x's rightmost 1. Because y has 0's from that point to the right, substraction of 1 from y must result in 1's from that point to the right, which will of course also appear in those positions in any bitwise 'or' involving $y - 1$. Thus neither (x or $y - 1$) nor ($x - 1$ or $y - 1$) will have a 0 in the position of x's rightmost 1; $y - 1$ will mask anything in that position and to the right with 1's. Since a 0 in that position is what a value of n would demand, neither A nor B can have a value of n.

A similar argument can be constructed for the case in which it is y that is assumed to have the furthest 1 to the right.

To sum up: if a single northwest unit has a value of n, a cell will take a value of $n + 1$, and if a cell has a value of $n + 1$ one and only one of its northwest units will have a value of n. Thus a cell will take a value of $n + 1$ if and only if precisely one of its northwest neighbors carries a value of n.

Similar arguments can clearly be constructed in the case of other connectives. What they demonstrate is in the inevitability of the cellular automata rules outlined for value spaces of any chosen dimension. It must be confessed, however, that despite such an explanation we continue to find something magical in the fact that such simple automata rules can generate a value space appropriate to propositional calculus for any chosen approximation.

In all seriousness we offer this cellular automata generation of value spaces simply as a phenomenon worthy of further study. In passing and in a spirit of wild speculation, on the other hand, we might also note a link to the fictional substance 'computronium', introduced in a review of Margolus and Toffoli's CAM-8 by Ivan Amato.[9] As envisaged by Amato, computronium would be a 'computing crystal' – a natural substance capable of functioning as a ready-made CPU. The speculation which the work above invites is that there may be natural processes which follow something akin to the simple cellular automata rules above and which thereby 'grow' units instantiating value spaces appropriate to forms of propositional calculus. If so, there may be natural processes capable of ‘growing’ something like Amato’s computronium. The lattice positions of computronium might be occuped by particular molecules or by molecules in particular states, for example, with the directionality of our rules above corresponding perhaps to magnetic orientation.

8. Conclusion

Our attempt here has been to open for consideration some new ways of envisaging and analyzing simple formal systems. What these approaches have in common is a clear emphasis on visual and spatial instantiations of systems, with perhaps an inevitable affinity to fractal images. Our hope, however, is that in the long run such approaches can offer more than a visual glimpse of systems as infinite wholes; that new perspectives of this type might suggest genuinely new results. In the manner of the three simple examples offered in our final sections – the Sierpinski map of tautologies in value space, formal parallels between value solids for systems of propositional logic and the quite different value solids appropriate to infinite-valued connectives, and an approach to the values of propositional calculus in terms of cellular automata – our hope is that visual and spatial approaches to formal systems may introduce the possibility of approaching some logical and meta-logical questions in terms of geometry. Number-theoretical analysis of logical systems forms a familiar and powerful part of the work of Gödel and others. Analysis in terms of geometry and fractal geometry, we want to suggest, may be a promising further step.

Acknowledgments

We are grateful to Gary Mar for discussion throughout the project and to Robert Rothenberg for diligent and creative programming.

NOTES

[1] The game fractal outlined here can be thought of as a fractally embedded form of the familiar game tree. See, for example, A. K. Dewdney, *The New Turing Omnibus*, New York: Computer Science Press, 1993, esp. Chapter 6, and A. L. Samuel, “Some studies in machine learning using the game of checkers,” in *Computers and Thought*, ed. E. A. Feigenbaum and J. Feldman, New York: McGraw-Hill, 1968, pp. 71–108.

[2] Because of color manipulation, the shades on the axes in these illustrations are no longer reliable.

[3] See, for example, Robert L. Devaney, *Chaos, Fractals, and Dynamics*, Menlo Park: Addison-Wesley, 1990; Heinz-Otto Peitgen, Hartman Jürgens, and Dietmar Saupe, *Fractals for the Classroom*, New York: Springer-Verlag, 1992; and A. J. Crilly, R. A. Earnshaw, and H. Jones, eds., *Fractals and Chaos*, New York: Springer-Verlag, 1991.

[4] It is tempting – but would be mistaken – to try to use this schema as a representation not only for full propositional calculus, but for a full infinitary propositional calculus, allowing for infinite formulae involving infinite connectives by way of conjunction, disjunction, or Sheffer strokes. (Infinitary systems of this type appear in Leon Henkin, "Some Remarks on Infinitely Long Formulas," in International Mathematical Union and Mathematical Institute of the Polish Academy of Sciences, eds., *Infinitistic Methods*, New York: Pergamon Press, pp. 167–183 and in Carol Karp, *Languages with Expressions of Infinite Length*, Amsterdam: North-Holland, 1964.)

This is tempting for one reason because infinite disjunctions of sentence letters represented in this way might seem to offer *non*periodic binary decimals. A simple example would consist of the disjunction of all our atomic sentence letters, giving us the truth table 01111. . ., with no repetition of its initial zero. For a more interesting example, consider an infinite disjunction which leaves out some of the set of sentence letters. Leave out only the second sentence letter, as outlined above, and you would appear to get the disjunctive value 01011111. . . . Leave out only the third and you would appear to get the pattern 01110111. . . . In general, leaving out the nth sentence letter from an infinite disjunction of all sentence letters would appears to introduce a zero in the $(2^{n-1} + 1)$th place. If every even sentence letter of the set were left out, so the reasoning goes, the result would be a classic non-periodic decimal in which 0's are separated by ever-increasing expanses of 1's.

An interpretation of infinitely-extended truth-tables is also tempting because universal quantification can be thought of as an infinite conjunction, existential quantification as an infinite disjunction. Were this scheme interpretable in such a way, then, it would offer a model not only for propositional but predicate calculus. Restricted to finite connectives it can at best correspond only to arbitrarily large finite models for propositional calculus.

The difficulty which blocks both of these tempting moves, however, is that the infinite extension of truth-tables outlined, although adequate for arbitrarily large finite complexes, cannot be thought of as adequate for genuinely infinite complexes. This becomes evident if one asks at what point in the table we will find a row which represents a '1' value for all of our sentence letters; it is clear that such a row can have no (finite) place in the scheme. A standard diagonal argument gives the same result: that there will be an infinite complex of our sentence letters which has no corresponding row in the table, and thus that the table will not be adequate for representation of all values in a genuinely infinitary system. For that we would require truth-tables somehow not merely of countably infinite but of uncountable length.

[5] See Manfred Schroeder, *Fractals, Chaos, and Power Laws*, New York: W. H. Freeman and Co., 1991, esp. pp. 20–25.

[6] See Gerald A. Edgar, *Measure, Topology, and Fractal Geometry*, New York: Springer-Verlag, 1990.

[7] This second type of value solid first appears in Gary Mar and Paul St. Denis, "Chaos in Cooperation: Continuous-Valued Prisoner's Dilemmas in Infinite-Valued Logic," *International Journal of Bifurcation and Chaos*, 4 (1994), 943–958.

[8] Łukasiewicz himself outlined his system in terms of implication and negation. Here we take as a Łukasiewicz 'or' the classical transform from implication: $/p \vee q/ = / \sim p \rightarrow q/$, with 'and' by a similar transformation. See Nicholas Rescher, *Many-valued Logic*, New York: McGraw-Hill, 1969.

[9] Ivan Amato, "Speculating in Precious Computronium," *Science* 253, August 1991, 856–857.

Group for Logic and Formal Semantics,
Dept. of Philosophy,
SUNY at Stony Brook,
Stony Brook, NY 11794,
U.S.A.