TAKING NOTHING SERIOUSLY: A FOUNDATIONAL DIAGRAMMATIC FORMALISM

William Bricken

December 2005

ABSTRACT: We explore the consequences of constructing a diagrammatic formalism from scratch. Mathematical interpretation and application of this construction is carefully avoided, in favor of simply pointing toward potential mathematical anchors for formal diagrammatic structure.

Diagrams are drawn in a two-dimensional space, while conventional mathematics emphasizes the one-dimensional linguistic space of strings. A theme is that the type of REPRESENTATIONAL SPACE interacts with the expressive power of a formal system.

Starting at the simplest beginning, we examine only two aspects of diagrammatic representation: forms that SHARE the representational space, and forms that enclose, or BOUND, portions of that space. The formalization of SHARING and BOUNDING is called boundary mathematics.

A pure boundary mathematics is a set of formal transformation rules on configurations of spatial boundaries. Due to the desirability of implementing spatial transformation rules by pattern-matching, we introduce boundary algebra, that is, boundary mathematics with equality defined by valid substitutions. This step connects us firmly to the known mathematical structure of partial orderings, algebraic monoids, Boolean algebras, and Peirce's Alpha graphs.

SHARING and BOUNDING do not form an algebraic group, leading to the surprising conclusion that boundary algebra is not isomorphic to Boolean algebra although it is equally expressive. This result is a formal consequence of Shin's observation that Alpha graphs have multiple readings for logic, providing formal evidence that, at least for propositional calculus, diagrammatic formalism is more powerful than linguistic formalism.

CONTENTS

1 INTRODUCTION

We examine the formal foundations of a simple class of diagrams consisting only of enclosures of space. The intent is to construct a mathematics of non-intersecting closed planar curves, and to compare it to known foundational mathematical systems such as set theory, partial orderings, propositional logic and numerics. A primary motivation is to elucidate a different way of thinking about mathematical symbolism.

In order to discover the inherent formal properties of configurations of closed boundaries, we develop a pure boundary mathematics, without anchors to conventional mathematical interpretation. We share Kempe's goal: "...to separate the necessary matter of exact or mathematical thought from the accidental clothing -- geometrical, algebraical, logical, etc." ref[Kempe] Since the diagrammatic approach incorporates types of structure that are simply not available within a string-based notation, it is critical to examine the construction of diagrams anew. Along the way, we indicate the potential connections between string-based and boundary-based systems.

Central to this innovative approach is to consider seriously the semantics of representational space, the space upon which we construct symbolic forms. In particular, void space has no attributes. By rigorously enforcing the emptiness of blank space, pure diagrammatic structures and operations can later be given different conventional interpretations. This permits various branches of foundational mathematics to be unified under a single diagrammatic formalism.

1.1 MAKING MEANING

Each step in the development of a system of mathematical notation and transformation incorporates decisions that constrain the intended meanings of the notational forms. The mere choice of a string of tokens to represent the concepts of a particular system limits the conceptual range available to the intended system. For example, the assignment of tokens to meanings incorporates a conventional assumption that the domain of discourse consists of unique atomic objects.

The question to date has been whether or not diagrams can convey the same formal information as strings. Venn developed his diagrams with set theory in mind; Peirce developed his existential graphs with logic in mind; Hasse developed his diagrams with partial orderings in mind. But there has been a history of exclusion of diagrams from the formal foundations of mathematics [refs], and a general disregard for the attempts by Venn, Peirce, Frege, and others to incorporate the expressive power of spatial forms into formal mathematics.

Showing isomorphism between string-based and diagrammatic formalism is useful, but fails to tap into the strengths of spatial representation. Isomorphic systems are structurally identical but for relabeling. A demonstration of isomorphism might show that diagrammatic forms can be mathematically equivalent to linguistic forms, but it does not show the ways in which diagrams can be formally superior to token strings. The intent here is certainly not to criticize the rich and powerful edifice of conventional mathematics; it is simply to explore new notational and conceptual structures that may possibly contribute to our foundational knowledge of mathematics.

One aspect of the agenda for diagrammatic mathematics is to develop formal spatial structures that provide different capabilities than those provided by one-dimensional string-based expressions. Not only can intuition be tapped through visual channels, but some realms of linguistic mathematics can be made more elegant, more efficient, more expressive, and wider in scope through the incorporation of diagrammatic techniques. Mathematics grows through innovation; string-based formalism is not the sine qua non of mathematical form.

1.2 FORMALISM

Metamathematics is the study the nature of mathematics itself. Without becoming mired in the philosophy of mathematics, we seek here to expand metamathematics by expanding the basis of what is considered to be a formal system. We embrace the notions of rigor, proof, and structured transformation, and continue to expect the benefits of the axiomatic method, including consistency, completeness, and independence of basis theorems. However, spatial formalism rests on an entirely different theory of representation, one that places semantics in non-representation, that abandons uniqueness of interpretation, and that freely confounds syntax and semantics.

Conventional mathematics is largely blind to relationships between representation and meaning, both out of choice to separate syntax from semantics, and out of myopia to forms of representation not based on strings. One-dimensional token systems rigorously maintain the independence of syntax and semantics by rendering tokens meaningless but for pure linear structure. However, structure itself is firmly a spatial and dimensional concept.

A central observation is that the dimension of representation interacts with that which can be conveyed. Strings of tokens embody a linear linguistic structure constrained by a temporal unfolding of a sequence of unit forms. Two-dimensional iconic and pictorial forms facilitate parallel access to visual information, while also allowing structure to connect more closely to the intended meaning of the representation. Three dimensional forms accommodate manipulability, allowing touch and kinesthesia to contribute to formal understanding.

Here we study the formal structure of a simple two-dimensional system of non-overlapping enclosures. We initially bar interpretation of these spatial forms in order to expose the unique characteristics of spatial formalism itself. Some of these characteristics map to conventional mathematical structures while others do not, requiring instead new perspectives on mathematical thought. This approach represents a significant variance, we consider the pure mathematics of a structurally limited class of diagrams, using only the overt and obvious characteristics of the diagrams themselves.

1.3 REPRESENTATIONAL SPACE

The space in which forms are recorded is called the representational space. The assumptions embedded in a representational space influence the expressive capability of the tokens lying in that space. By shifting our analytic attention from the symbolic figures to the representational ground, we can clarify the fundament upon which syntactic forms are constructed.

The most common representational space is the page supporting written language. We string words out on an invisible line and read these words by scanning the line from left to right. Implicit conventions permit our focus of attention to roll off the end of one line and start again at the beginning of the next line without interruption. In this way, we organize a two-dimensional space so that it accommodates a long one-dimensional sequence of words. We similarly structure stacks of pages into three dimensional books that yield a yet longer string of words. Words themselves are encoded sequences of tokens; the unit tokens are meaningless and unique, replicates are differentiated only by their position relative to other unit tokens.

An important deviation from the linear regime is when the text includes a picture, illustration, table or diagram. These forms are constructed in two-dimensional space. Such non-linear representational spaces are set aside in a frame, highlighting an entirely different form of information that is contained therein. This provides an initial structural distinction between strings and diagrams: diagrams are set aside by a two-dimensional frame, while strings are set aside by an implicit one-dimensional frame. The frame that exists prior to both tokens and drawings is the frame of the space itself.

1.4 STRING NOTATION

Strings of tokens define a total order with regard to concatenation. Some mathematical concepts, such as symmetry, require a greater structural flexibility than provided by a total order. This flexibility is obtained by various permissions for rearrangement of tokens. Importantly, syntactic transformation (i.e. computation) is implemented as substitution of patterns.

Consider simple algebra as an example. Linear structure is maintained through binary operators with two sides; through argument precedence to disambiguate asymmetry from symmetry; and through operator precedence to disambiguate sequences from trees of evaluation. These notational conventions of string-based mathematics come under scrutiny in the context of two-dimensional, spatial, diagrammatic forms that do not necessarily impose total ordering on the tokens for objects and operators.

1.4.1 PROPERTIES

Mathematical structures rely on ordered pairs. Lining up tokens in a linear representational space permits us to associate by proximity two object-tokens with each operator-token. Multiple arguments however require multiple operators, while associative rules specify the ordering of operations. In linear notations, we nominate a first and a second object for each binary operation. To accommodate symmetry, we then invoke rules that permit order not to matter. A generally unused alternative is to invoke rules only when order does matter.

Recording binary operators in a line creates a problem of precedence. In order to remove the inherent ambiguity of concatenation of operations, operators and arguments are explicitly grouped. We add parentheses so that nested operations take sequential precedence. This solution for disambiguation calls upon the additional dimension of depth added by delimiting tokens.

1.4.2 UNIT BLOCKS

To one unfamiliar with the rich and elaborated tools of algebra, properties such as binary scope, associativity, commutativity and precedence might appear to be artifacts of the linear notation. Some mathematical operations, such as subtraction, rely on ordered pairs. Other operations, such as addition, do not. The traditional form of addition, however, leads us to potentially confuse the meaning of addition with the machinery that achieves the meaning. Place notation dictates addition algorithms by building multiplication into the location of digits in a string, thus enforcing a particular mechanism for sequential numerical manipulation.

Imagine instead a unit-block notation for integers. Cardinality is represented by a counted pile of small unitary blocks that can be physically manipulated. Unit-blocks can be added simply by pushing piles together and counting the resulting collection. The complexity of the addition operation itself is exchanged for a complexity of reading the result of the operation. Unit-block addition is symmetrical (no pile is considered as primary, or is "pushed into").

However, any number of piles can be pushed together concurrently; unit-block addition is not necessarily binary. When several piles are pushed together in concert, there is no concept of associativity (which pairs of piles are pushed together first).

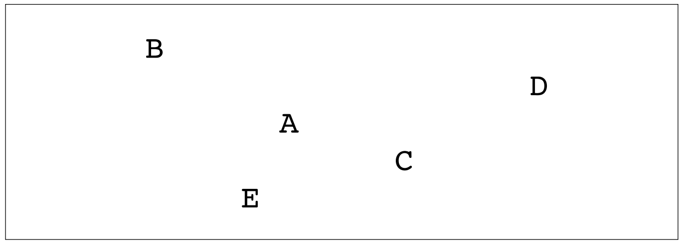
Associativity and binary scope are neither "implicitly" nor "tacitly" included in unit-block addition; these particular concepts are simply not relevant. They apply to a specific sequential mechanism (how piles are pushed together), not to mathematical structure (the result of pushing piles together). Commutativity is implicit (it does not have to be implemented), however there are no contrasting properties such non-commutativity. Thus, the meaning and meaningfulness of abstract mathematical properties of relations vary across the form of a representation, since different representations incur different mechanisms. Although mappings and functional forms remain isomorphic, the relational properties of forms vary according the structure of their representation.

This is widely acknowledged in the data structures of software programming languages. Some representations are more efficient for particular hardware architectures. What is data and what is engine varies widely between declarative and procedural languages. The set of transformation axioms used by a pattern-matcher or theorem prover is specific to the form of representation of the problem. These variations occur within string-based systems; when diagrammatic systems are considered, it should not be surprising that the group theoretic properties and classifications of conventional formalisms may be inappropriate.

1.5 A SIMPLE SPACE

Let us generalize the nature of unit-block addition on the surface of a table to an abstract mathematical structure. We first construct a simple representational space that can contain an unspecified multitude of physical unitary forms. This space is variary rather than binary, any number of piles are accommodated concurrently, including none. This space does not support commutativity, not because unit-block representational space is non-commutative but because the symmetry of the space dominates all transformations. Piles are simply pushed together, there are no rules that specify which hand does the pushing.

As formulated, the unit-block representational space supports grouping but not for the purposes of operator precedence. Addition, in fact, is specifically the concurrent removal of all grouping structure. We can further simplify by calling each group, independent of cardinality, by a different name-token, say from the set of tokens {A, B, C, D,...}. The image that emerges is of a representational space in which unique tokens float, unconnected and unstrung. The only communal property they share is that they jointly occupy the same space, as illustrated below.



The mathematical concept of a set achieves these abstract objectives. Sets can contain an arbitrary number of unique members while not supporting duplicity or ordering. To represent a set, standard notation places a boundary around the tokens that represent the set members: {a, b, c, d}. An awkward convention of set notation is the prevalence of the comma-token, presumably intended as a reminder that the members are not really in order, they are just written that way. The need for comma-tokens can be alleviated by using both dimensions of the page. As in the above illustration, tokens can float in this non-linear space, not grouped or related in any way other than by their common presence in the same space.

Still, set theory is too structured since it enforces us to view the members of a set as specific objects. Set equality is defined by a one-to-one correspondence of unique object-tokens within each set.   The simple space does not yet require a token for an empty space, nor does it need axioms (such as pairing and union) that permit construction of new spaces.

2 CONSTRUCTING A DIAGRAMMATIC FORMALISM

The construction of a diagrammatic formalism begins in the same way that the construction of a linguistic formalism begins: an empty space is distinguished by a frame. The frame may be represented explicitly or assumed implicitly by convention. Formal systems then usually launch into the complex system of encodements, encryptions and understandings that is known as symbol processing.

What happens between nothing and so much? What is the simplest diagrammatic construction that can convey significant formal meaning? C. S. Peirce pioneered this unfamiliar territory in the development of existential graphs.

The strategy here is to begin as simple as possible (with nothing) and then to add diagrammatic structure that is also as simple as possible (a closed frame). We show that replication of non-intersecting closed frames recorded on a surface of representation is sufficient to generate a language with the expressibility of propositional calculus and integer arithmetic. We show that operations in this language are algebraic and void-based.

At first, simplicity includes abstinence from mathematical interpretation. The goal is to uncover the formal structure of very simple diagrams from the ground up. Concepts that evolve from this endeavor include representational space, semantic void, SHARING and BOUNDING space, spatial patterns, pattern equations, and void-equivalence. The computational mechanism for boundary forms includes void-substitution, absence of parsing, and semi-permeable boundaries.

2.1 LITERALLY NOTHING

The simplest beginning is a mathematical intention and a blank surface; that is, we begin with an empty space. It is an unexpectedly bold innovation to accept empty space for what it is, featureless and without properties. To be serious about nothing, it is necessary to avoid encumbering the emptiness of empty space by filling it with points, positions, metrics, attributes, or structure of any type. The concept is that of void, the absence of properties, the absence of markings, and absence of meanings.

Unlike the empty set, which has an explicit representation, { }, that interacts with set operators, empty space has no representation. Token-based formal systems invariably use labels to identify nothing. In contrast, placing any label on or in empty diagrammatic space negates rather than identifies the emptiness.

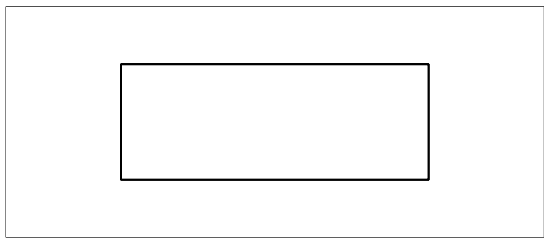
A space devoid of markings is located by a frame, a boundary that contains and delimits the space. Indirectly, nothingness is defined by its container. The empty page is defined by its edges. What we see as empty space depends on our point of view. If we focus on the outside, we see the boundary that frames nothing. If we focus on the inside, we see nothing, the absence of markings.

The bounded two-dimensional representational space is purely diagrammatic, partaking of no linguistic structure or interpretation. Most importantly, the empty page directly embodies its semantics (that of emptiness), rather than indirectly representing its semantics through assignment of meaning to an abstract token. Empty space is the foundation upon which a deeper understanding of the process of creating diagrammatic formalism can be built. In particular, the boundary of an empty space provides a model for the first conceivable marking.

2.2 BOUNDING NOTHING

Our mathematical intent is to construct the simplest formal diagrammatic system, to construct notation and structure that introduces nothing more than what is overt and unavoidable. An advantage of spatial formalism is that it can directly represent its intended meaning. Thus far, we have used emptiness to convey the meaning of absence of marking.

For the first marking, the only structure that is explicitly available is the frame that delineates empty representational space from the rest of physical



reality. We take the mathematical step of abstraction by drawing a picture of this frame within the empty representational space, and call this drawing a mark.

The drawing is of a closed curve, aligning the diagrammatic syntax with the closure of the frame it represents. The concept represented by the mark is that of BOUNDING, the identification of a space by an enclosure. The representation of the boundary has only one property, that of spatial closure, since the emptiness of the underlying substrate does not support metric concepts such as size, shape, location, origin, and other points of reference, and does not support topological concepts such as proximity, connectedness and continuity.

The accidental properties ref[shin] of a mark include all structure that may be associated with a metric. The accidental properties of a representational space include all structure that may be associated with a topology. The simple non-mathematical semantics of a mark is that it represents the boundary of an intentional space of representation. The mathematically abstract mark, divorced from its visceral semantics, simply distinguishes an interior. Importantly, as a diagrammatic form, a mark looks like what it means.

It is important to realize that a boundary in a space does not necessarily separate that space into two independent portions. The outermost space can also be seen to pervade all bounded spaces within it, just as physical space pervades all objects within it. The underlying page is not fragmented, it continues to support each bounded space, inside and outside of each boundary. Here, we wish to place no interpretation on the meaning of a boundary in a space, particularly the interpretation that the boundary divides a space into two separate spaces, since such an interpretation introduces duplicity and cardinality (rather than introducing only a new enclosed space).

2.3 DISTINCTION

There is a subtle bias in beginning with the frame of an empty page: we read from the outside. The simplest diagram would not incorporate information about the location of the reader. In order to fit a closed curve within our field of view, we place our perspective on the outside of the enclosure. Consider, as a contrast, the surface of the Earth itself. It is a closed curve that we view from the inside rather than from the outside. Only mathematical abstraction permits us to consider the simplest frame from either side, without perceptual bias. BOUNDING makes a distinction, it marks a difference in space.

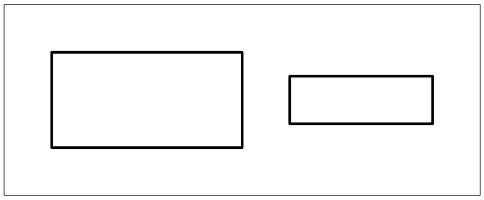
By placing the reader in a specific location on the outside of the frame, we immediately create an ordering: the exterior space of the reader is "greater-than" the interior space of the frame. In the case of the surface of the Earth, the space that frames our everyday activity is "less-than" the space containing the Earth. Only recently have we learned to see the Earth from the outside.

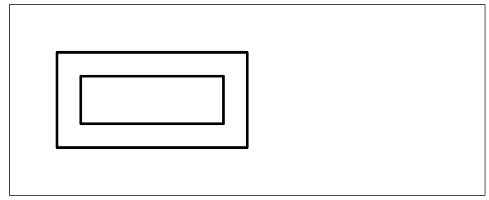
As our first concession to the abandonment of simplicity, let us assume that the position of reading is from the outside of the closed boundary. This concession aligns nicely with the ideas that nested boundaries construct an ordering, that void is the bottom of boundary forms, and that what is outside has full access to what is inside.

2.4 REPLICATION AND STRUCTURE

We are considering only two primitive diagrammatic structures, blank space and the boundary that defines the extent of the enclosed blank space. In order to construct an expressive diagrammatic language, we introduce structure through the idea of replication of boundaries. We permit more than one copy of the mark, and we permit the underlying space to support these replicates. Replicates of closed curves must not touch or intersect, since this undermines their closure. Notice that semantically the semantic frame is necessarily singular and unique. Replicates are purely syntactic, they cannot carry new semantic intentions, although their enclosure relation can.

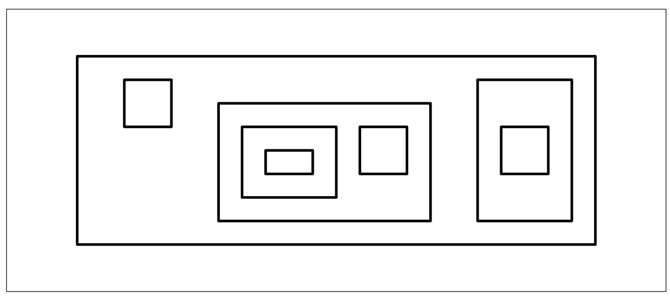
In constructing the first abstract boundary, we have no choice but to draw it inside the empty space of the page. Once drawn, however, the original abstract boundary distinguishes a new space, the (sub)space on its inside. Drawing a mark identifies two spaces, the bounded interior, and the space outside the mark, that original, formerly blank, space that holds both the boundary and the space enclosed by the boundary. A replicate of the first drawn mark can be constructed in either of these two distinguished spaces, either outside of the first diagrammatic boundary, or inside that boundary:





Boundaries thus distinguish spaces that support the construction of complex structures. A spatial arrangement of boundaries is called a form.

Both empty space and space containing existent structure can be bounded. Any space, whether contained or not contained by various boundaries, supports any number of bounded forms within it. As replicates of boundaries proliferate, they can be nested to any depth, and they can populate any space with multiple copies. An example of a more complex form consisting solely of boundaries:



2.5   PARENS NOTATION

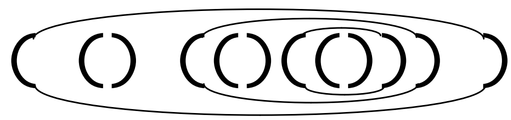
We will immediately introduce a more economical spatial notation, parens.

Parentheses, and other delimiting tokens such as brackets, braces, and quotation marks, are by construction unusual tokens. In a string language, it takes two different tokens, right and left parentheses for example, to express the elementary idea of containment, or bounding. Parentheses put non-linear boundaries on strings of linear tokens. There are two of them because the parenthesis actually exists in two dimensions, around token strings. Parentheses permit the construction of trees of token strings.

Herein, parens notation is used solely as a shorthand for the spatial structure of boundaries. A mark is represented in parens notation as empty parens, ( ).

The set of well-formed parenthesis structures is a well-known mathematical structure, since operator precedence is conventionally expressed using parentheses. Well-formed parenthesis configurations are unfortunately defined as balanced parenthesis pairs within strings of tokens. This leads to analyses that violate the closure of a boundary by addressing the "left" or the "right" portion of the intended enclosure independently, an excellent example of how string formalisms can modify the semantics of diagrammatic forms. A parens, in contrast, is an abbreviation for a unitary closed curve.

(() (()(())))



Well-formed parens can be defined recursively.
The conventional technique of definition focuses on objects with structure.
These and no other structures are parens forms:

void is a parens form	BASE
If **a** is a form, so is (**a**).	BOUNDING
If **a** and **b** are forms, so is **a b**.	SHARING

This definition of structure embodies two apparently constructive operations: BOUNDING and SHARING. A bound can be placed in any space, regardless of which contents, if any, are on the interior of the bound. Multiple forms can be constructed in any particular space, in which case we say that the forms SHARE that space. But for diagrammatics, there are significant problems with this conventional definition:

Non-existent void: Basing the definition of well-formed parens in void places a property (existence) on that which is featureless. The ground must be an empty parens BOUNDING void space. To respect the semantic intention of void, and to avoid regressing into string-based thinking, the non-existent void may not be used at all within the object language. Rigorously, void has no properties, and thus is accessible only through meta-language.

Non-structural void: Above, SHARING is defined between objects within a space that supports multiple forms. That is, SHARING is a relation between two bounded forms. However, from a diagrammatic approach, forms in the same space do not possess relations to each other, because an inert void does support spatial relations. The space co-occupied by bounded forms is defined solely by the outer boundary, or frame, of the shared space. Conventional SHARING is a boundary operation of identifying an equivalence class of forms that are in a diagrammatic SHARING relation with the same outer boundary.

Variary scope: The above definition also incorporates unnecessary linear structure. SHARING is not a binary composition of forms; the number of forms within a given space is unlimited. Similarly, any number of forms can be bounded during the construction of a single bound.

To express these freedoms, we introduce a specialized type of label. In diagrammatic style, we then use these labels to identify patterns of space rather than structures of objects.

2.6 LABELS

Boundary forms require careful labeling to maintain their essential spatial semantics. We adopt the following conventions:

Small letters label single bounded forms.
Capital letters label the contents contained by a boundary.
A boundary is labeled by a small letter on the outside,
and by the same letter capitalized on the inside: a(A)

Each small letter identifies a single boundary and its contents; small letter forms consist of one outer boundary and the contents of that boundary. Each unique label is assumed to identify a unique bounded structure.

Each capital letter identifies the forms within the space contained by a given single boundary; capital letter forms consist of the contents of a boundary, which may range from zero to many bounded forms sharing the same space.

Each space is unique by construction. In terms of Peirce's theory of signs, the empty boundary is an icon (representing what it means), compound forms are symbols (having a potential assigned meaning) and labels are indices (unique names associated with a boundary). Space itself is unmentionable in syntactic forms.

We associate labels with forms using an equal sign:

c = (( ) (( )( )( )))

The count of forms within a boundary is the count of small letter bounded forms directly enclosed within that boundary. In contrast, a boundary contains only one space, the entire contents of a boundary are identified by one capital letter:

C = ( ) (( )( )( ))

Both capital and small letters identify the same boundary from two different perspectives: c from the outside, and C from the inside: c = (C). Illustrations of various valid notational varieties of forms follow.

Let a = ( ), b = (( )( )( )), c = (( ) (( )( )( ))). We can say:

```
b = (a a a) = (B)
B =  a a a
c = (a b) = (a (B)) = (a (a a a)) = (C)
C =  a b  =  a (B)  =  a (a a a)
```

A third type of identifier would be needed to identify partial contents of a space, since small and capital letters are one-to-one. These are not needed for the objectives of this paper.

The small and capital letter distinctions are particularly important when identifying patterns of boundaries for transformation. They are also important because the shift from object-orientation (small letters) to spatial-orientation (capital letters) is a primary difference between linguistic and diagrammatic formalisms.

2.7 OPERATIONS AND RELATIONS

A space is the substrate under all forms within that space; the semantic endeavor is to refrain from attributions of any kind to space. An initial question then arises as to what "within a space" means. The answer relies upon our labeling conventions. Forms within a space are in relation to the outer boundary that defines that space, not to space itself, and therefore not to each other.

Capital letters identify a collection of forms within a specific space. More properly, capital letters implicitly define a collecting operation, in essence they identify an arbitrary collection of bounded forms. Rather than postulating that a space itself is defined by its membership, as in set theory, we instead make collections into an explicit property of a relation. Extensionality in set theory (that two sets are equal iff they have the same members) does not mention how to compare members of sets, or even how to determine identicality of members. Set equality assumes that the elaborate infrastructure of predicate calculus is already in place.

In the diagrammatic formalism, BOUNDING is the only operator, it constructs both existent forms and depths of nesting. SHARING is a descriptive consequence of forms enclosed by a given boundary. Later, we will introduce pattern equations that restrict the construction of boundaries to forms with an interpretable semantics. In specific, all validly constructed forms must be void-equivalent.

When a boundary is added to a form, it bounds a new space on its inside, as well as everything within that space. BOUNDING is an operation that explicitly adds a new form (via containment of existent forms), the new form consists of the new boundary and the entire space inside. There is a defined one-to-one mapping between semantically explicit boundaries and semantically inert spaces.

SHARING is a relation between a bounded form and the immediate outer boundary that encloses that form. It is not a relation between two or more forms in a space, since we are disallowing all varieties of structural support or connectivity that may be associated with space. Since spaces are coextensive with their outer boundaries, SHARING is a strictly binary relation between an outer boundary and a directly inner boundary. The conventional concept of sharing as a relationship between members of a collection becomes an explicit process of collecting relations having a specific structure, that of being SHARING related to the same outer boundary. Conventional membership-like sharing is a property of relations rather than a particular relation.

SHARING is weaker than cardinality, since it involves no concepts of ordering or counting. It is weaker than membership, since replicates can share the same space. Given that we permit ourselves opaque boundaries, so that no space accesses what is on its inside, SHARING is a universal equivalence class that does not partition the domain contained by the outer space. SHARING indexes an inner boundary to the boundary that is its unique outer container. In that way, SHARING is a property of each contained boundary, the value of the property is the name of the outer boundary. We do not yet need to provide a formal mechanism to "collect members" of a particular space. We do honor the diagrammatic style: SHARING is visually obvious when we place our perspective outside the outermost boundary and all it contains.

2.8 STRUCTURAL EQUIVALENCE OF FORMS

Two forms with the same bounding structure are equal. The intuition is similar to the equality of strings and sets. Structurally equal forms can be superimposed exactly.

The idea of extensionality in set theory codefines a set with its members. Boundary forms, in contrast, are minimalist, there is no distinction between a form and its constituent parts; each constituent is simply another boundary form. A recursive definition of well-formed boundaries defines valid syntax, valid constructions, and valid deconstructions. Two forms are equal if their construction or deconstruction is identical.

The definition of form equality is broader than string and set theoretic equality, since it integrates structural equivalence over both length (SHARING) and depth (BOUNDING). We now modify the conventional definition of well-formed parentheses to recursively specify well-formed boundary forms. We use iff rather than implication to maintain a algebraic style.

Let **A** stand in place of the contents of any boundary **a**

```
( ) is a form                                        BASE
A is a form iff   (A)  is a form                     BOUNDING
A is a form iff  A ( ) is a form                     MARKING
```

Two forms have the same structure:

```
( ) = ( )                                            BASE
 a  =  b   iff  delete[( ),a] = delete[( ),b]        RECUR
```

Two spaces are equal iff their bounded spaces are equal. Bounded spaces are equal iff their contents are equal. That is, spaces with the same contents are equal.

The ground for this recursive definition cannot rely on the use of void, since void cannot be distinguished sufficiently to lay on both sides of the equality. In a diagrammatic and quite visual sense, the presence of equality itself places an improper attribute on nothing. This somewhat unusual leakage of definition into the meta-language is an expedient to suppress potentially dysfunctional attributions to nothing. Thus the ground is that of the empty boundary, the enclosure of nothing. Empty boundaries are equivalent.

With a diagrammatic formalism in mind, this definition acknowledges how we use spatial enclosures that have yet to be refined by an intended representational utility. As a concrete example, we seamlessly join together empty sheets of paper, and number them as distinct only when they acquire differential content. In a formal diagrammatic system, objects that are acted upon by functions give way to purely relational structures over spaces.

2.8 PATTERNS AND EQUATIONS

The equal sign can be used to compare two forms with different spatial structure, and thus determine an equivalence relation between forms.

Within a featureless space, all structure is directly visible as configurations of boundaries. By construction, boundary forms have no implicit structure that requires a transformation rule to express. Forms consisting of boundaries, with or without spatial variables, are patterns. In a diagrammatic formalism, spatial patterns take the place of algebraic properties such as transitivity and symmetry. Patterns that can be validly substituted for one another are expressed as pattern equations; equivalent patterns are equal. Labels can be used as variables within patterns. Small letters identify bounded forms, capital letters identify the entire content of spaces.

Pattern substitution maintains the intention, or meaning, of a form; exchangeable patterns can be considered to be equal with regard to their intention. For instance, permission to construct and to eliminate replicate bounded forms sharing a space might be written as the substitution pattern

$$b\ b = b$$

where b stands for any bounded configuration.

An algebra of spatial forms is defined by diagrammatic patterns, pattern variables (small and capital letters), the assertion of equality, and the mechanism of substitution. Operators in this algebra are the aforementioned BOUNDING and SHARING. Pattern equations define pre- and post-conditions of transformations. Equations can be used to define the result of an operation, the constraints of a particular interpretation, and the presence of void-equivalent forms.

2.10 VOID-EQUIVALENCE

Computation over spatial forms involves the use of void in fundamental ways. For most substitution patterns, pattern equations identify forms with structure that can be deleted. A structure that is equal to void is called a void-equivalent. Strictly, structures that reduce to nothing are imaginary.

The idea of void-equivalence emphasizes that spatial pattern transformations are achieved by void-substitution (that is, by deletion or erasure), in contrast to transformation by rearrangement that is characteristic of string-based systems. Deletion operations confer excellent behavior on boundary computation; for instance, rapid convergence is assured.

2.11 EXPRESSIBILITY

At this point, we have had no concern about types of structural elements, there is only one. Compositions of boundaries, however, are non-trivial.

The language of well-formed boundaries has a natural semantics, that of partial ordering, or containment. A spatial form not only represents a partial ordering, it has a visual semantics of partial ordering, thus escaping the purely arbitrary assignment of tokens to meanings that characterize string-based systems.

Containment is the fundamental relation for finite set theory; partial ordering is the fundamental relation for order and lattice theory; and since the logical conditional can be represented by an inclusion relation, containment is also sufficient to express propositional calculus, as Peirce has shown in his Alpha graphs.

Boundary notation can also be mapped to other diagrammatic techniques used to convey the structure of partial orders such as Venn diagrams for subsets (intersecting boundaries are prohibited), Hasse diagrams, graphs of a relation, operator tables and binary matrices.

3 BOUNDARY ALGEBRA

Substitution patterns are defined to be valid when they embody a semantic intention. As yet, this intention has been guided only by the construction of a simple, purely diagrammatic formal representation. We now examine the potential mappings of boundary forms to conventional mathematical structures. Are spatial SHARING and BOUNDING patterns morphic to conventional string-based formal structures?

In bridging between diagrammatic and string systems, we will be using two quite different languages: pure diagrammatic forms and conventional string notation that corresponds (at times not rigorously) to these pure forms. Although parens notation provides the convenience of drawing diagrams on a line, parens forms are markedly different than token strings.

We consider the two diagrammatic operations of SHARING and BOUNDING. SHARING is a relation between bounded forms and a space, while BOUNDING is both a property of an enclosed space, and a relation between two spaces, the enclosed space and the space outside of the enclosed space.

3.1 THE SHARING RELATION

The special properties of relations are conventionally expressed using logic; they can also be viewed as structural patterns within a relation table. The concept of SHARING, however, is most naturally and succinctly expressed in diagrammatic notation as a capability of space.

The SHARING relation applies to bounded forms in a bounded space. As an example:

B = ( ) (( )( )) ((( )))

B can also be labeled as

B = a b c with a = ( ), b = (( )( )), and c = ((( )))

We say that a, b, and c SHARE the same space. In the diagrammatic language, nothing more need be added, the properties of sharing are embodied in the existence of boundary forms in an unstructured space.

We now explore the conventional consequences of committing to a paradoxical definition of SHARING as a relation without attributes. Both set theory and logic are as yet of no use. We have only the ability to construct spatial enclosures (inclusions) and the permissions to label both spaces (via capital letters) and bounded spaces (via small letters). Forms define patterns that can be attached to a variety of mathematical semantics.

3.1.1 PROPERTIES OF SHARING

In conventional terms, SHARING is an equivalence relation on bounded forms in a particular space. We will use the infix token tilde, ~, to indicate the conventional interpretation of SHARING in string notation:

a~b with a, b in the set of well-formed bounded forms

Calling SHARING a binary relation defines it as a collection of ordered pairs. This is potentially misleading, since the semantics of the particular shared space is completed captured by the set of boundaries adjoining it. These consist of an outer boundary that names the shared space and a collection of inner boundaries that name the spaces they contain. This way of seeing the form is "object-oriented" in that it collects all the single boundaries that access the space. A spatial or diagrammatic view might be "space-centered", characterizing each distinct boundary abutting the space as door. For the purpose of accessing any space next door, all the doors would form an equivalence class: all doors are equally preferred.

SHARING is a relation on a Set, the set consists of the contents of a particular space; the space is identified by its outer boundary.

does not permit an ordering concept or a pairing concept.

SHARING is reflexive. A single bounded form in a space, such as (a), SHARES that space with itself.

Conventional notation usually requires a replicate to express the idea of reflexivity. Two identical forms would SHARE one space. Diagrammatic notation requires singular existence; a single form SHARES a space with itself since SHARING does not have a spatial arity. Two replicate forms SHARING a space expresses cardinality rather than reflexivity.

SHARING is symmetric, since two forms SHARING the same space have the same SHARING relation to each other. SHARING is transitive: when three forms share the same space, all three possible sharing relations are present. Thus, SHARING is a type of EQUIVALENCE relation; all forms within the same space are sharing that space equivalently.

As well, SHARING is associative. It does not matter which pairs we consider first. More fundamentally, space does not as yet support pairing or counting. A comparison of relational concepts follows:

CONVENTIONAL		DIAGRAMMATIC	
<u>structure</u>	<u>notation</u>	<u>structure</u>	<u>notation</u>
reflexive	a~a	existence	a
symmetric	a~b = b~a	no location	a b
transitive	a~b & b~c --> a~c	no grouping	a b c
associative	a~b.~c = a~.b~c	no pairs	a b c

The dot is Quine's notation for reinforcing which pairings have sequential priority. We use it here since parentheses are reserved for BOUNDING.

In fact, SHARING is a universal relation, all forms in a space SHARE with all other forms in that space. A universal relation is trivially reflexive, symmetric, transitive and associative.

3.1.2 LOGIC WITHIN SPACE

Conventionally, when a relation R is reflexive, we say that it is TRUE that aRa belongs to the set of valid relations on R. It appears to be fairly innocuous to introduce a truth functional evaluation of membership within a relation, because relational properties in general incorporate logical connectives such as AND and IMPLIES.

Consider the definition of transitivity, which requires in turn a definition of the meanings of the conjunction of a~b and b~c, and of the implication of such a conjunction. From a conventional viewpoint, these are well-defined, AND means that both relations belong to the set of valid relations for ~, while IMPLIES means that if both are in the valid set, then so is another, specifically a~c.

In the case of diagrammatic SHARING, transitivity does not impose a logical AND interpretation onto space. AND requires only that two pairs of bounded forms be present in the same space. However, since space does not support pairing, this requirement is vacuous. IMPLIES is visually obvious in the constitution of forms in space. When the three bounded forms share a space, they share in all possible combinations. That is, spatial transitivity does not require us to travel between (to make a transit of) paired relations to find a third.

Spatial symmetry does not require us to locate one relation and then find another in a symmetrical part of the relation table. Two bounded forms simply co-exist in the same space. Spatial associativity, as well, does not impose logic. Conventionally, associativity imposes a structural relation between two

~ relations. That relation is of a different type, one of sequencing. Spatially, the bounded forms exist in parallel, there is no necessity for, or concept of, sequential SHARING.

3.1.3 IS SHARING FUNCTIONAL?

SHARING is not a function. A universal relation permits too many multiple mappings between forms. Technically, SHARING is not unique, every form shares with every other form.

We can, however, formulate SHARING as a set function induced by the SHARING relation. A set function is a mapping between two sets. The idea is to preserve uniqueness by mapping single sets of sharing forms in the domain uniquely onto sets of forms of the range. Generally a set function is a mapping between every member of the power set of bounded forms sharing space. This is much closer to the diagrammatic intent of the SHARING relation. For example, the form {a, b, c} has eight members of the power set, and thus 64 pairs of sets for the set function. Schematically, these are

({ },{ })	({a}, { })	...	({a,b}, { })	...	({a,b,c}, { })
({ },{a})					
...					
({ },{a,b})	...	...		...	...
...					
({ },{a,b,c})	({a}, {a,b,c})	...	({a,b},{a,b,c})	...	({a,b,c},{a,b,c})

This awkward formulation conveys completely the possible SHARING subsets. The ungainly size comes from attempting to convert simple diagrammatic SHARING into a linear binary relation between subsets. However, given this concession, we can now pursue the group theoretic structure of the SHARING set function.

3.1.4 ALGEBRAIC STRUCTURE OF SHARING

Converted to a binary set function, SHARING can be considered to be an algebraic magma (a set together with a closed binary function). SHARING is associative, making it a semigroup. We now examine the ideas of identity and inverse.

An identity element meets the condition that any form SHARING space with the identity element is simply that form. Diagrammatically, this is the case with void, which we could (non)represent by a vacancy. Using the notation of boundary algebra,

$$a \qquad = \; a$$

The unrecorded void is a trivial identity, it results in an identity equation for a. In conventional notation, we could let void be represented by an underbar:

a~_   = a

To rigorously stay with set functions, we let the variable a stand in place of any member of the powerset of forms SHARING the space, and the underbar _ stand in place of { }.

The set function notation requires two sets to share a space. For example, in a universe of four elements, each must remain unchanged when SHARING space with the identity. This places an interpretation on space to perform a specific functional transformation on sets.

{a,b,c}   ~   { } = {a,b,c}

By mapping boundary notation onto that of the set function, we can see that one possibility is for SHARING to perform the UNION operation, with the empty set as the identity.

{a,b,c} union { } = {a,b,c}

The difficulty is that the set theoretic interpretation imposes an operation that space performs. The intent of diagrammatic SHARING is simply to place two spatial objects within the same space, not to have them interact.

String notation is clear about pre- and post-operation expressions:

{a,b,c} union {b,c,d} = {a,b,c,d}

Union with the empty set is imposed as an interpretation primarily because the empty set is an explicit token in set theory. However, an identity and its operation are co-defined; the empty set comes into existence to maintain the binary character of an operation that results in no change.

Diagrammatically, the void does not exist, nor does it need to exist in order to support semantic structure. Although the set function for SHARING permitted us access to modern algebra, it quickly becomes too cumbersome for the essential purpose of defining the structure of spatial SHARING.

An unintended consequence of imposing the union operation on space is that the equal sign becomes limited to the definitions of set theory: equality acquires the intention of extensionality, and becomes defined as two sets with identical membership. From the set theoretic perspective, the equivalence class of bounded forms SHARING a space is universal to that space, incorporating all possible pairings from the elements of the powerset of the set of bounded forms. From a diagrammatic perspective, the bounded forms in a space have no connective structure; SHARING is nothing more than the equivalence class of those bounded forms not separated by another explicit boundary. Both the concept of operation and the concept of identity violate the diagrammatic intent of space to remain inert and featureless.

3.1.5 SHARING WITH VOID

Conventionally, we must address SHARING relations with void. Since void is everywhere throughout a space, it is somewhat of a distortion to localize it as a unique member of a SHARING set function, but this is the cost of introducing sets that have an explicit null member.

In diagrammatic notation, SHARING with void is trivial, since void is non-existent. In conventional notation, we have

a~b = a~_.~._~b = a~_._~_._~b

The SHARING identity is void, as is null SHARING:

_~a = a~_ = a   _~_ =

Having identified an identity, SHARING becomes a monoid. We hope now to define the SHARING group by identifying an inverse for each bounded form.

3.1.6 SHARING IS NOT A GROUP

A SHARING inverse is defined as

$$a\sim a^{-1} = a^{-1}\sim a = i$$

From the perspective of the SHARING set function, the set of all bounded forms in a space is a Universe. Each form has an inverse defined by the universal set compliment. However, the meaning of an induced set SHARING a space with its inverse remains ambiguous. Consider for example, a set element SHARING space with its inverse in a universe of four bounded forms:

{a} {b,c,d}

The question is, do the contents of this space equal the SHARING set identity?

{a} ~ {b,c,d} =?= { }

As mentioned, should space be identified with set union, then the identity is the empty set. This does not work in the above equation, since union would yield the universe of elements.

{a} union {b,c,d} = {a,b,c,d}

Similarly, should we go back to alter the interpretation of space to be, say, set intersection, then we would have a solution to the identity equation as

{a} intersection {b,c,d} = { }

However, SHARING-as-intersection generates a different identity on the way to monoid-hood. For set intersection, the universal set is the identity:

{a,b,c}	intersection	{ }	=	{ }	not an identity
{a,b,c}	intersection	{a,b,c,d}	=	{a,b,c}	U is the identity

We have encountered a contradiction and as a consequence, SHARING (degraded by an explicit void in the empty set, by set functions on subsets of sharing forms, and by interpreted operations on space) is not a group. The reason is somewhat surprising. It is not because an inverse and an identity do not exist, but rather because they are inconsistent across their interdependent definitions.

3.2 THE BOUNDING PROPERTY

We now consider BOUNDING. Every bounded form has the BOUNDED property; however, BOUNDING is not a property of structural forms, since it is applied to a space and all it contains. BOUNDING is a property of a space, not of forms. Every space is bounded, whether the contents of that space are singular, multiple, or void. The property that a boundary confers on a space is that of framing.

Capital letters stand in place of the contents of a space, so BOUNDING is a universal property over capital letter forms:

(A)   for all A

SHARING and BOUNDING are co-defined:

> A boundary distinguishes the space shared by forms in that space.

The bounded forms within each space are necessarily composed by the SHARING relation. Specific boundaries label the space they contain.

3.3 THE BOUNDING RELATION

When a boundary is viewed as a property of the space it contains, it frames that space, and nothing more. However, BOUNDING can also be viewed as a relation between the spaces that the boundary stands between:

A (B)

That is, a boundary identifies two sides, an inside and an outside, and might be construed to be a strictly binary relation between each of its sides. In contrast to SHARING, which is a universal equivalence relation over a space, BOUNDING denies that two forms share the same space. The intervening boundary distinguishes the space occupied by A from the space occupied by B. We will adopt the perspective of reading from the outside, as discussed in Section 2.

We will use a vertical bar, |, as the conventional token for the BOUNDING relation. A|B means A is outside of B.

3.3.1 TWO READINGS

For now, there is no interpreted operation or behavior associated with the boundary, other than containment of a space. However, A (B) can be read in two substantively different ways:

A~(B)	A SHARING with (B); the boundary frames B
A|B	A BOUNDING B; the boundary relates A and B

BOUNDING and SHARING space are aspects of the same single spatial structure. This is an aspect of boundary forms that is purely diagrammatic, and cannot be finessed by conventional means. A boundary both separates and connects.

When the boundary is read as a property of the contained space B, it separates B from A; the form (B) is seen to be sharing a space with the form A. Interpreting a boundary as a frame leaves the outer space unencumbered by imposed structure. This maintains the intended semantics of featureless space capable of SHARING and nothing more.

When the boundary is read as a relation, it connects A to B; the form A (B) represents a possible binary ordering relation between spaces. Interpreting a boundary as a relation imputes a structure to both the inside and the outside space. Space is no longer inert, instead it supports differentiation. This attribution has surprising consequences.

3.3.2 ORDER STRUCTURE OF THE BOUNDING RELATION

Like SHARING, we will seek a semantics of BOUNDING based upon appeal to the visually obvious. In common language, we can see that A is on the outside of B. We seek to capture the concept of "being outside of" while introducing no rules of transformation. The idea is to examine BOUNDING in the context of the structural properties of orders.

We adopt the convention that outside is abstractly greater-than inside, providing an initial correspondence between boundary and conventional concepts. The intention is that the enclosure of the boundary limits its contents, a limitation not encountered outside of the enclosure:

A (B) might be interpreted as $A > B$

3.3.2.1 Symmetry

In Section 2, we made an initial concession to choose a privileged side of the boundary by placing our perspective on the outside. This choice introduces an asymmetry.

Asymmetry	$A\|B \neq B\|A$	A (B) ≠ B (A)

Distinction, without a privileged perspective, is symmetric. A is distinguished from B, and on equal footing, B is distinguished from A. A choice of perspective immediately denies that BOUNDING is symmetric. If B is indeed limited relative to A, then that limitation is lifted when B is placed outside. Further, if A is unlimited as the context of B, that absence of limitation cannot be maintained by placing A inside the enclosure.

We have not excluded any forms for A and B; there is no structural reason to deny construction of the form A (A). Regardless of whether or not A (A) is valid, the fact that it equal to itself makes BOUNDING antisymmetric.

Antisymmetry	A|B & B|A --> A=B	A (B) = B (A) is A=B

The conventional notation for antisymmetry contains logic, which thus far we cannot transcribe. Besides, the endeavor is to develop a semantics of boundaries without recourse to logic or set theory. Confined to boundary notation, we can only say that the equality of two asymmetric forms defines equality of the components of those forms. Two forms are equal if each can bound the other.

By accepting A (A), we must also convert the original interpretation of BOUNDING to be greater-than-or-equal-to:

A (B)   is interpreted as   A ≥ B

This does not yet imply that BOUNDING is any type of ordering.

3.3.2.2 Reflexivity

Reflexivity of BOUNDING accepts that a form can be outside of itself, A (A). Obviously such a form can be constructed, but how do we know that it is not void-equivalent? That is, how do we assert A (A) prior to the introduction of relational tables, logic and transformation?

For SHARING, we finessed this issue by arguing that reflexivity (as well as symmetry and transitivity) was obvious by definition and by visual inspection. It is not obvious that the form A (A) should be accepted; a form distinguished from itself might be interpreted to contradict the intention of BOUNDING, rendering the boundary void-equivalent. A boundary used to distinguish something that is structurally indistinguishable is not a boundary.

Conventionally, assertion of reflexivity calls for the addition of truth values to the evaluation of forms. A valid relational property is interpreted "as TRUE"; for reflexivity, it is TRUE that the relational table includes the application of the binary relation to replicates of the same form.

One possible approach to boundary reflexivity is to permit truth, which has not yet been introduced, to default to existence. Void does not exist, it supports no interpretation or evaluation. Void-equivalent forms may exist syntactically, but their value is non-existent. Void-equivalent forms stand in place of imaginary structure. This approach is inadequate, since we as yet have no methods of determining whether or not A (A) is void-equivalent.

Another possible approach is to define reflexivity to exist, that is, to not equal void: A (A) ≠   . This merely adds the logical concept of NOT to the meta-language, however we have not yet defined semantic equivalence (although we have defined structural identity).

A third approach is to rely upon the original conceptualization of space as pervasive. That is, in the form A (B) the space labeled A is not fragmented by the space labeled B. The space A continues to pervade the space B, leading to the equation:

A (A B) = A (B)   PERVASION

A strength of this approach is that the empty boundary, ( ), is a positive example of reflexivity, void is on both sides. The empty boundary is also a positive example of pervasion, in that void on the outside permeates the inside. A weakness of this approach is that BOUNDING might still be interpreted as being exclusionary.

3.3.2.3 Transitivity

A preorder is a relation that is reflexive and transitive. A preorder with a least element is a well-ordering, sufficient to support the Mathematical Induction Principle. Instead of grounding an induction in zero or in the empty set, boundary induction might be grounded in void. This implies that inductive transformations identify and erase syntactic structures that are void-equivalent.

Conventionally, an ordering is at least transitive. Types of orders are defined by adding varieties of reflexivity and symmetry properties. Is BOUNDING obviously transitive? That is, if A is outside of B and B is outside of C, is it necessarily true that A is outside of C?

When a boundary form consists only of nestings, as in (A (B (C (D)))), transitivity is visually and structurally obvious. In the form (A (B)(C)), however, the spaces B and C are not nested. SHARING space is an equivalence relation that denies complete nesting, and thus denies linear ordering.

Consider two bounded forms SHARING a space, with B participating in both forms: (A (B)) (B (C)). In one bounded form, the space of B does not contain (C); in the other, it does. Here transitivity is not obvious.

3.3.3 DIAGRAMMATIC BOUNDING

We have now come to the diagrammatic core of the interpretation of BOUNDING as a type of ordering. BOUNDING does not construct an order without additional structure placed on space. This theme occurs for every relational property that can be assigned to BOUNDING. We must step outside of pure diagrammatics to anchor diagrams to conventional mathematical structure.

We have mentioned in passing several ideas as to the source of additional interpretive constraint:

- -- Abandon SHARING in space, to create a linear order or nesting.
- -- Assert that some forms, such as reflexivity, do exist.
- -- Place a logical interpretation on diagrammatic structure.
- -- Adopt PERVASION, that the outside permeates the inside.

To foreshadow what follows, without abandoning SHARING, the other three ideas amount to the same thing. Prior to examining the consequences of these concessions, we add two more appealing considerations as obvious: using void as a lowest bound, and permitting boundaries to be semi-permeable.

3.3.4 LOWEST BOUND

Empty space can be taken to be an intuitive least element for BOUNDING forms. Diagrammatically and visually, void is underneath all boundaries. Void is also innermost in every form, since the most deeply nested space in any form is necessarily empty. Conventionally, void is the infimum, the greatest lower bound for all forms. However, an infimum is considered to exist as an element in an ordered set while void does not exist. The problem could be approached as one of notation, i.e. simply label the void. We avoid this approach since it converts space into a token, objectifying and falsely localizing void, and undermining conceptual understanding of the diagrammatic semantics of nothing.

3.3.5 PERMEABILITY

We have mentioned the possibility that boundaries might be interpreted as exclusionary. Such boundaries are impermeable. We have also mentioned pervasion, that what is outside is also inside. In such a case, boundaries are semi-permeable; a semi-permeable boundary is transparent in one direction but not in the other. Finally, should a boundary is fully permeable, then it is not a boundary but a space. A permeable boundary is imaginary (i.e. void-equivalent), since it fails to bound.

The impermeable interpretation is like logical exclusive OR and set complement, each of A and B excludes the other. This interpretation aligns with the separating property of boundaries. The semi-permeable interpretation aligns with the connecting property of boundaries, logical inclusive OR and the subset relation. Thus A (B) might be either:

Semi-permeable:	$A \geq B$	and	$B < A$	WEAK ORDER
Impermeable:	$A > B$	and	$B < A$	STRICT ORDER

A weak order (semi-permeable boundaries) admits reflexivity by definition while a strict order (impermeable boundaries) does not. The example of boundary numbers above uses impermeable boundaries, while the example of boundary logic to come uses semi-permeable boundaries.

The foray into an interpretation as ordering has lead to significant questions about transitivity and reflexivity. Antisymmetry was established by adopting a specific perspective for reading. We now consider transformation rules that define equivalence classes of boundary forms.

3.3.6 TRANSFORMATION

Boundary forms constitute a pattern language of BOUNDING and SHARING in a featureless space. Interpretations of this language can be imposed by equations that identify equivalent patterns. In the example of integer numerics, boundary place notation is defined by two transformation rules:

```
 A  A  = (A  )              DOUBLE
(A)(B) = (A B)              DISTRIBUTE
```

In the example of order theory, we identified a need to impose logical interpretations on boundary forms. This can be achieved by asserting that reflexivity "IS TRUE", by translating the logical forms that define relational properties into boundary notation, or by adopting boundary pervasion.

4 BOUNDARY INTEGERS

When the boundaries constituting a form are labeled by indices, nested boundaries can be viewed as an ordinal structure on nested spaces. Bounded forms SHARING a space provide the cardinal structure of an equivalence class that commingles with the ordinal structure of bounding to construct a partial order of bounded forms. A count of sharing forms or of nesting boundaries is not required by transformations on patterns of these structures (as introduced later). Thus, the numeric aspect of the partial order interpretation is an extension to the boundary structure of partial orders.

4.1 NUMERICAL FORMS

There is an interesting variety of interpretations of parens forms for numerics, three simple examples follow:

```
      CARDINAL                ORDINAL                 VonNEUMANN

  0 --                    0th --                 0 --  {}
  1 --  ( )               1st --    ( )          1 --  { {} }
  2 --  ( ) ( )           2nd --   (( ))         2 --  { {}, {{}} }
  3 --  ( ) ( ) ( )       3rd --  ((( )))        3 --  { {}, {{}}, {{}{{}}} }
                ...
```

4.2 BOUNDARY PLACE NOTATION

The simple operations of addition and multiplication of boundary integers can be implemented by attaching specific transformation rules to forms. For example, a cardinal arithmetic of boundaries might have a semantics of unit-blocks, with addition achieved by "pushing together" and counting the resulting pile. More efficiently though, the unitary boundary integer notation can be converted into a spatial analog of place notation by using depth of nesting as a base [Kauffman]. The following interpretation is a binary arithmetic of boundary forms. Units are represented by a centered dot, •, for visual ease. In a computational regime, the empty boundary, ( ), is sufficient to represent the unit.

```
            BOUNDARY-BINARY                                 RULE

      0 --
      1 --          ( ) = •
      2 --       ( )( ) = •• = (•)                   A  A  ==>  (A)    DOUBLE
      3 --    ( )( )( ) = (•)•
      4 -- ( )( )( )( ) = (•)(•) = (••) = ((•))    (A)(B) ==> (A B)   MERGE
                                ...
```

The transformation rule for boundary place notation is a single recursive equation:

```
            ( )( ) ==> (( ))                         UNIT DOUBLE  (base case)
            (A)(B) ==> (A B)                         MERGE   (inductive case)
```

Conventionally, the DOUBLE rule is the simplest version of the rule of distribution for integers, distribution of units:

$$1*1 + 1*1 = 1*(1 + 1) = 1*2 = 2$$

MERGE, the more general inductive case, is equivalent to distribution of doubles:

$$2*A + 2*B = 2*(A + B)$$

DOUBLE can also be generalized to apply to equal configurations, constructing an intermediate case between UNIT DOUBLE and MERGE:

```
             A  A ==> (A)     with A ≠ void          DOUBLE
```

$$1*A + 1*A = (1 + 1)*A = 2A$$

Different bases are achieved by changing DOUBLE to apply to an accumulation of units equal to the specific base. For example, base 5 would be

```
A A A A A ==> (A)            BASE 5
```

Finally, in the diagrammatic formalism, MERGE is not binary, but rather applies to any number of bounded forms sharing a space:

```
(A)(B)...(Z) ==> (A B ... Z)
```

4.3 NUMERICAL OPERATIONS

Addition is simply placing forms in the space. To MERGE is to concurrently remove all frames at one level, and is thus a parallel, variary operation:

```
A + B + C + D + E:         (A)(B)(C)(D)(E) ==> (A B C D E)
```

OPERATION	NOTATION		PROCESS	
	conventional	boundary	conventional	boundary
ADD:	A + B	A B	look-up in table	put in same space
MULTIPLY:	A * B	A into B	look-up, add	substitute for unit

Multiplication of two forms is achieved by substitution. One form is substituted for every unit • in the other form. For example, 6 * 5:

```
5 = ((•))•
6 = ((•)•)             6*5 = ((  ((•)•)  ))  ((•)•)
```

The above result is immediately equal to 30, no further computation is required. However, boundary place notation reduction rules can be applied to reach a canonical shortest form of the result.

```
30  =   ((((•)•))) ((•)•)          ((•)•) times ((•))•
        ((((•)•))   (•)•)          merge
        ((((•)•)     •)•)          merge
```

In summary, boundary forms can be interpreted as integers. Operations are defined by combining forms as patterns. Addition combines in space, while multiplication combines by substitution into the deepest spaces.

5 BOUNDARY LOGIC

We continue to seek minimal interpretive structures. For example, the empty boundary, ( ), is the minimal case of reflexivity with void on both sides. Here, we seek to connect boundary forms to logical connectives and to the assertion of logical truth. One elegant but arbitrary approach is to take the empty boundary as a representation of logical truth.

( ) =map= TRUE

This is appealing since the empty boundary is the original diagrammatic frame, the first mark. We let the first mark be valid, or more simply, TRUE. The initial frame is not void-equivalent, if it were then we would not be able to begin to draw a valid diagram.

5.1 THE MARK OF TRUTH

For purposes of logical assertion, and to differentiate existent from void-equivalent forms, we can now assert that BOUNDING is reflexive:

A (A) = ( )   REFLEXIVITY

This equation also asserts that in the configuration of reflexivity, all forms A are void-equivalent.

Due to the antisymmetry of BOUNDING, we know that accepting A|A is equivalent to accepting A = A. For boundary forms, equational identity is co-defined with relational reflexivity.

The rest of the logical connectives can be deduced from one further observation: ( ) is not void-equivalent.

( ) ≠

When void is bounded, we see an empty mark, the diagrammatic form of TRUE. For a logical interpretation, the only value that TRUE does not equal is FALSE. We can take BOUNDING to be logical negation by reading the empty bound as both TRUE and as NOT FALSE. TRUE = NOT FALSE. This neatly finesses the problem of mapping between a boundary system with one ground form, and a logical system with two ground values. A mark is TRUE, the absence of a mark is FALSE.

5.2 THE MARK OF IMPLICATION

We can use the technique of multiple readings of diagrammatic forms for logic to construct the rest of the logic connectives. For example, here is the boundary rule of INVOLUTION:

ft[INVOLUTION is not necessarily dependent on a logical interpretation. or on the permeability of boundaries.

(( )) =	not not false = not true = false
((A)) = A	not not A = A

The empty space on the right-side of the first equation declares that the form on the left-side is void-equivalent. We can then use void-substitution to establish:

(( )) ( ) = ( )

This is the instance of reflexivity for which A = ( ).

We can also take the boundary to be the logical conditional. We already have most of the truth table for establishing the BOUNDING relation as if-then:

<table>
<tr><th>FORM EQUATION</th><th>READING</th><th>INTERPRETATION</th></tr>
<tr><td>( ) = ( )</td><td>void|void = mark</td><td>if false then false = true</td></tr>
<tr><td>(( )) =</td><td>mark|void = void</td><td>if true then false = false</td></tr>
<tr><td>(( )) ( ) = ( )</td><td>mark|mark = mark</td><td>if true then true = true</td></tr>
<tr><td>( ) ( ) = ( )</td><td>void|mark = mark</td><td>if false then true = true</td></tr>
</table>

The last form equation can be derived from an alternative form of the conditional:

if A then B = (not A) or B

Using (A) as NOT A, the form (A) B assigns the interpretation of OR to SHARING. As well as reading the last form equation as a conditional, it can also be read as "TRUE OR TRUE = TRUE"

5.3 THE MAP TO LOGIC

The rest of the logical connectives are easily mapped, leading to this table:

BOUNDARY	LOGIC
<void>	false

( )	true
(a)	not a
(a) b	a implies b
a b	a or b
((a)(b))	a and b
(((a) b)((b) a))	a iff b

This mapping is a bridge between the spatial semantics of bounding diagrams and the linguistic semantics of logical connectives. Thinking diagrammatically, we have reduced Boolean logic to a unary boundary logic of one existent ground value (the empty boundary). The other ground value is imaginary, FALSE is cast into void and thus into non-existence. In summary,

MAPPING	LOGIC	BOUNDARY
SHARING is disjunction	A v B	A B
BOUNDING property is negation	¬A	(A)
BOUNDING relation is the conditional	A-->B	(A) B

5.4 A DECISION PROCEDURE

The idea of pervasion, that boundaries are semi-permeable from the outside, is sufficient, along with the definition of reflexivity, to axiomatize boundary logic. Boundary logic is equational rather than implicative. Since boundary forms map one-to-many to conventional sentences, boundary logic is formally more succinct than conventional logical calculi.

Given semi-permeability, the boundary reflexivity equation can be reduced by applying PERVASION. When the same form is both inside and outside, PERVASION permits deletion of the form inside. Applying this to reflexivity yields the rule of DOMINION.

A (A) = ( )	REFLEXIVITY
A ( ) = ( )	DOMINION
A (A B) = A (B)	PERVASION

DOMINION and PERVASION axiomatize propositional calculus. The rule of INVOLUTION, which can be derived from the antisymmetry equation, is also needed for efficiency:

((A)) = A	INVOLUTION

5.5 BACK TO ORDER

The conventional definition of an order relation is co-defined with the definition of the logic connectives. Each requires the other. Although logic is not usually phrased in terms of ordering, the conditional connective itself forms a partial ordering.

The decision procedure above can be interpreted for logic; it can also be interpreted for orderings through the conventional properties of order. Boundary reflexivity is the source of the DOMINION rule. When antisymmetry is converted into boundary form, it confirms the boundary structure of equality. A stepwise transcription follows:

```
          A|B                 (A) B
          B|A                       (B) A
        X and Y             ( (  X  ) (  Y  ) )
      A|B and B|A           ( ((A) B) ((B) A) )

A|B and B|A --> A=B         ( ((A) B) ((B) A) )  is  A=B
```

Finally, the form of transitivity can be constructed using only PERVASION, showing that PERVASION is a generalized combination of both reflexivity and transitivity:

```
(     )                      true
(     ) (B    ) A (C)        +dominion
(A (B)) (B    ) A (C)        +pervasion
(A (B)) (B (C)) A (C)        +pervasion, form of transitivity
```

In relational notation, parentheses are used to conventionally group operations:

```
((A|C)|(B|C))|(A|B)
```

The conventional form of transitivity mixes logic and order relations:

$$((A \geq B) \ \& \ (B \geq C)) \dashrightarrow (A \geq C)$$

The reading for logic simply converts ≥ into the conditional, -->:

```
((B-->A) & (C-->B)) --> (C-->A)
```

Our excursion into diagrammatic formalism has yielded a condensed diagrammatic language that cross-translates both to logic and to order, while providing simpler computational mechanisms for both. Different mechanisms yield integer numerics.

5.6 COMPARING TO ALPHA EXISTENTIAL GRAPHS

We have derived a boundary logic purely from the intuitive visual structure of nesting and sharing boundaries. The semi-permeable interpretation of BOUNDING led to the diagrammatic rule of PERVASION. PERVASION and reflexivity axiomatize boundary algebra. Antisymmetry derived from taking a privileged perspective for viewing boundaries connects logic to algebraic equality.

Peirce's Alpha existential graphs apply the diagrammatic structure of enclosure specifically to logic. How do boundary logic and Alpha graphs map to each other? As would be expected, the representations are isomorphic, while the system of transformation rules is remarkably close to being the same. The central differences fall into two categories, the meaning of space and the difference between an implicative and a algebraic mechanism.

EMPTINESS

-- Boundary logic assumes a homogeneous empty page, while Peirce's boundaries, called cuts, are taken to severe the page into discrete pieces.

-- Alpha graphs use the dual interpretation of boundary logic, for which the empty sheet is deemed TRUE as opposed to FALSE. Peirce earlier developed entitative graphs that use emptiness is FALSE and space as disjunction, thus aligning with boundary logic.

-- Boundary logic uses semi-permeable boundaries, while Alpha graphs use impermeable graphs.

LOGIC vs ALGEBRA

-- Alpha graphs are assertion-based; forms are taken as premises. Boundary logic is structure-based; forms are algebraic until reduced.

-- Boundary logic provides a somewhat more modern algebraic transformation system, uniting pairs of asymmetrical, unidirectional rules into single symmetrical, bidirectional rules.

-- Alpha graphs use one-directional add and erase rules that provide an unclear termination goal. Boundary logic uses a single OCCLUSION rule that is void-equivalent.

-- The use of parens notation to step back and forth between an implicative format consisting of one space, and an algebraic format consisting of two compared spaces alleviates pragmatic difficulties of manipulating Alpha graphs.

-- The highly efficient algebraic axiomatization of boundary logic alleviates pragmatic difficulties of transforming Alpha graphs.

These differences are overshadowed by the profound differences between conventional string-based logic and both Alpha graphs and boundary logic:

-- Both take diagrammatics as foundational.

-- Both use PERVASION as a pattern rule rather than a functional transformation.

-- Both rely on void-equivalence in every transformation rule.

-- Both subsume conventional logic, however pure boundary mathematics does not rely on a logical interpretation.

To Peirce, the formal structure of logic was a geometrical not a textual property. Geometric properties can be observed directly. This structure is obscured by textual forms, since text cannot directly represent some essential concepts of diagrammatic logic. Peirce's philosophical perspectives are closely aligned to those expressed in this paper.

5.6.1 ALPHA GRAPH REPRESENTATION

Pierce considered "The Sheet of Assertion" (i.e. the blank page) to be conjunctive, so that any form written on it is asserted to be TRUE. The Sheet of Assertion represents the universe of discourse; each form SHARING the empty page is a TRUE form. Should a set of assertions be NOT TRUE, then the graph would result in an "absurd" configuration, such as the empty enclosure.

Peirce's boundaries are "cuts". The area enclosed by a cut is considered to be NOT on the sheet of assertion. Peirce structured the page into two separate spaces, those areas nested at even depths and those at odd depths. Forms at even depths are TRUE, while those at odd depths are FALSE. The two domains alternate as complements.

The calculus of Alpha graphs is implicative, however the power of the representation permits abstract conclusions to be drawn simply from recording the premises. Should a conclusion be known, it is tested by placing its negation on the Sheet of Assertion. In such cases a TRUE conclusion enables all forms to be erased from the sheet. During graph deduction, forms are constructed and deleted until the graph matches the desired result. This process takes some planning and intuition, but not nearly as much as, say, natural deduction.

5.6.2 ALPHA GRAPH TRANSFORMATION

Peirce defined his transformation rules (permissions) to be the "Pure Mathematical Definition of Existential Graphs, Regardless of their Interpretation". This is, he considered the rules to be structural rather than semantic. Peirce developed Alpha graphs for logic, and did not call upon the notational advantages of an algebra. Logic rests on implication, while algebra rests on bidirectional implication, that is, on equivalence. Equality does not play a role in Alpha graphs, leading Peirce to state bidirectional transforms as two separate rules, one for each implicative direction. Also, Peirce was working prior to the development of the axiomatic method. Thus his rules, although provably complete and consistent, are phrased in an intuitive style.

It is not possible to express the scope of Peirce's diagrammatic rules in conventional string notation, or in a conventional binary operator interpretation. Thus the logical interpretation of Alpha graph transformations generates a shallow approximation of the strength of Peirce's diagrammatic rules.

Iteration, for example, gives permission to insert a replicate of any graph into any level of nesting deeper than (or at the same level as) the replicated graph. In OR-logic, the following insertions are valid:

```
            BOUNDARY LOGIC                       CONVENTIONAL LOGIC

S1.  (A) (   (B)((C)   (    D))))       A-->(    B & (C-->       ¬     D  ))
S2.  (A) ((A)(B)((C)   (    D))))       A-->(A & B & (C-->       ¬     D  ))
S3.  (A) (   (B)((C)(A)(    D))))       A-->(    B & (C-->(¬A v ¬     D )))
S4.  (A) (   (B)((C)   ((A) D))))       A-->(    B & (C-->      ¬(A-->D))))
```

We emphasize that all four forms are equivalent, and that the last three boundary forms are each derived from the first one by a single application of iteration.

S2 is not directly derivable from S1 by a single conventional logic rule, but it is easy to see that if A implies a conjunction, then A also implies that conjunction with A added. In S3, ¬A jumps over two connectives (& and -->) and builds a new disjunction where a negation used to be. In S4, A jumps over two connectives and builds a new conditional, while shifting the negation of D to outside the new conditional.

Not only is it difficult to prove the equivalence of forms derived through deeply nested iteration by conventional means, it is entirely foreign to skip over operator boundaries, to add new operations deep within a form, and to move negations within a form. The central point, though, is that iteration is a single, succinct rule for diagrammatic forms, and cannot be expressed as a single rule using string-based concatenations of binary connectives.

5.6.3 ALPHA GRAPH AND BOUNDARY LOGIC RULES

Peirce's rules are transcribed into symbolic notation below. ref[ROBERTS] Peirce's existential graphs are cast in AND-logic; his entitative graphs ref [1897] are based on OR-logic, the dual of AND-logic. We provide representations of Peirce's rules in both AND-logic and OR-logic to illustrate alignment with boundary logic forms which have the most direct interpretation as OR-logic.

RULE	AND-LOGIC(even)	OR-LOGIC(odd)	BOUNDARY
R1. erase even/odd	((A) B) ==> (( ) B)	((A) B) ==> ((A))	( ) = A ( )
R2. insert odd/even	(A) ==> (A B)	A ==> A B	
R3. iteration	A (B) ==> A (A B)	A (B) ==> A (A B)	A {B} = A {A B}
R4. deiteration	A (A B) ==> A (B)	A (A B) ==> A (B)	
R5. double cut	A <==> ((A))	A <==> ((A))	A = ((A))

5.6.3.1 ALIGNMENT OF RULES

Rules R1 and R2 give permission to record and erase forms from the page, with the exception that an empty boundary cannot be erased. This grounds the representation with two halting conditions, the empty page taken as TRUE and the empty boundary taken as FALSE. Recorded forms are taken as logical assertions.

The boundary logic rule of DOMINANCE is used only once, as the final terminating step of a valid deduction. It is better understood as a void-equivalent rule:

(( ) A) = OCCLUSION

PERVASION works to remove deeply nested replicates; whenever a boundary becomes empty, it and its context are void-equivalent. Proofs by Roberts, Shin, and Dau refs[], contain unnecessary steps that might be attributed to an inefficient use of adding and erasing forms.

Rather than inserting forms to achieve a reduction pattern as in R2, boundary logic begins by recording the problem, in particular all of the premises and the conclusion. Forms are then deleted until either void, a single mark, or a contingent expression remains.

Double cut, R5, is INVOLUTION in boundary logic. INVOLUTION is teh only rule that has a clear isomorphic interpretation in conventional notation, as double negation. This theorem is not fundamental, it can be derived from OCCLUSION and

PERVASION alone, given an antisymmetry rule that provides a bridge between equational and logical forms. INVOLUTION is required often to remove void-equivalent boundary pairs. In boundary logic, it is seldom used constructively.

PERVASION is bidirectional iteration, R3 and R4 combined. The curly braces in the PERVASION equation are meta-tokens standing in place of any number of nested boundaries. Intervening forms are irrelevant due to the dissociation of forms created by the definition of SHARING.

PERVASION is the workhorse of boundary reduction. Relatively difficult deductive problems usually have a sequence of necessary extractions/deiterations during the course of reduction. Trivial applications of INVOLUTION and OCCLUSION efficiently remove structure remaining after applications of PERVASION.

Peirce may have recognized the power of this rule. When we examine his five rules, iteration/deiteration stand out as the only transformations of any complexity. The separation of Alpha graphs into two discrete domains (odd and even depths of nesting) is undermined by PERVASION, creating subtle contradictions in Peirce's descriptions. This leads to an embedding of logic into the reading of Alpha graphs by modern scholars. Forms crossing boundaries are seen to alternate interpretation via varieties of DeMorgan Laws, and to an overemphasis on counting the depth of nestings.

5.6.3.2 GENERALITY

The differences between AND-logic and OR-logic interpretations of boundary forms can be strictly limited to transcription from logic to boundaries and from the boundary logic conclusion back into conventional logic. Thus, there is no need to carry any logical interpretation into the transformation rules themselves, or their applications. Once a transcription is completed, the diagrammatic rules are not logical rules, but rather a much stronger set of spatial pattern transformations that apply independently of logical interpretation (AND- or OR-logic).

Double cut/INVOLUTION can be seen as a structural modification having nothing to do with double negation. However, the independence of the logical interpretation is most prevalent in PERVASION, which applies regardless of depth of nesting and regardless of odd or even nested spaces. PERVASION is the semipermeable property of boundaries, conditioned on the existence of outer forms. It shows clearly that the depth of Alpha graph boundaries does not necessarily alternate between positive and negative interpretations for logic.

Peirce's insert/erase rules are anchored to the parity of depth of nesting. In OR-logic, erasure (R1) is a variant of the logical rule of addition, while insertion (R2) is a variant of logical simplification.

```
            OR-LOGIC BOUNDARIES                  CONVENTIONAL LOGIC

R1.   erase odd   ((A)(B)) ==> ((A)  )       simplification   A & B ==> A
R2.   insert even        A ==> A B           addition             A ==> A v B
```

In boundary logic, R1 and R2 are nothing less than specific applications of PERVASION combined with DOMINION. This is easily seen by converting the implicative forms of R1 and R2 into a fully boundary form:

```
                             R1                        R2

OR-logic             ((A)(B)) ==> ((A)  )           A ==> A B

Pure boundaries    [ ((A)(B)) ]   ((A)  )         [ A ]   A B
                      (A)(B)        A             [ A ]   A B      involution
                      ( )(B)        A             [   ]   A B      pervasion
                      ( )                         [   ]            dominion
```

The essence of R1 and R2 is the introduction of the DOMINION rule. The erasure of R1 is the erasure permitted by deiteration, while the insertion of R2 is the construction permitted by iteration. The degenerate void cases of each rule (for which A is void) expose the necessity of DOMINION:

```
                             R1                        R2

OR-logic              (( )(B)) ==> (( )  )             ==> B

Pure boundaries     [ (( )(B)) ]   (( )  )          [  ]   B
                       ( )(B)                                      involution
```

The OR-logic permission associates the form ( ) A with TRUE, which is ( ) in OR-logic. This permission can thus be written as:

```
                   ( ) A = ( )                            DOMINION
```

The AND-logic interpretation of Alpha graphs effectively hides the rule of DOMINION in the Sheet of Assertion, thus obscuring the clear termination pattern.

```
                             R1                        R2

AND-logic                  B ==>                  ( )  ==> (B)
```

Pure boundaries	[B [ ]]	[( ) [(B)]]	scroll
		[( ) B]	involution

When the AND-logic form (( ) A) is asserted, it is taken to be TRUE, which is void in AND-logic. This permission can thus be written as:

(( ) A) = OCCLUSION

5.6.3.3 DISTRIBUTION

Rules such as DISTRIBUTION and CONSENSUS are theorems based on PERVASION. In contrast to Boolean algebra and to axiomatizations of propositional calculus, DISTRIBUTION is not fundamental to boundary systems. DISTRIBUTION is required to convert deeply nested forms into conjugate normal forms (CNF) with only two levels of nesting of literals. CNF, in turn, is used extensively as a canonical representation underlying the metatheorems of logic.

In boundary systems, DISTRIBUTION is a consequence of PERVASION. This has significant impact on how to think about diagrammatic logic. CNF trades multiple reference to variables for a shallow nesting of boundaries. CNF accommodates a conventional binary connective reading of logical forms. However, boundary transformations, especially PERVASION, are most efficient as rules for transforming deeply nested forms, forms for which a conventional interpretation requires the many steps associated nested functions. CNF is exactly the wrong way to approach transformation of diagrammatic forms.

The appropriate (semi)canonical representation for boundary systems is minimal replication of variable labels and maximal nesting of boundaries. In a potentially misleading interpretation for logic, maximally nested boundary forms might be called Implicate Normal Form (INF), since each nested implication adds a level of boundary nesting.

The boundary rule of DISTRIBUTION,

A ((B)(C)) = ((A B)(A C)) DISTRIBUTION

can be rewritten to emphasize the relationship between variable replication and depth of nesting:

(A ((B)(C))) = (((A B)(A C)))
= (A B)(A C) involution

The left-side is INF while the right side is CNF. The CNF format deemphasizes applications of PERVASION, while the INF format deemphasizes replication of labels.

INF is canonical up to irreducible varieties of DISTRIBUTION. The advantage of CNF is that [VERIFY] identification is tautologies is easier. However, conversion to CNF is an exponential process. INF has the same advantage for identification of tautologies, at the cost of not necessarily minimal forms.

5.6.4   ALPHA GRAPH AND BOUNDARY LOGIC PROCESS

The self-distributive law of the conditional serves as a common example of the deductive procedures envisioned by Peirce. It's proof is presented below for Alpha graphs and for boundary logic. Brackets are highlighted parens.

Logic theorem: (p-->(q-->r)) --> ((p-->q)-->(p-->r))

Alpha-graph proof:

```
        [p [(q (r))]]                                transcribe antecedent
                       [(p (q))  [(p (r))]]          transcribe consequent

      [ (p ((q (r)))) [ ((p (q))  ((p (r)))) ]]      transcribe theorem
      [ (p ((q (r)))) [ ( p   (p (q)) (r)  ) ]]      R5, rearrange text
      [ (p ((q (r)))) [ ( p   (  (q)) (r)  ) ]]      R3
      [ (p ((q (r)))) [ ( p       q   (r)  ) ]]      R5
      [ (p ((q (r)))) [ ( p ((    q   (r)))) ]]      R5
      [ (p ((q (r)))) [                      ]]      R3
      [               [                      ]]      R2
                                                     R5
```

Improved alpha-graph proof:

```
      [ (p ((q (r)))) [ ((p (q))  ((p (r)))) ]]      theorem
      [ (p   q (r)  )    (p (q))    p (r)     ]      R5
      [ (    q      )    (  (q))    p (r)     ]      R4
      [ (    q      )    (     )    p (r)     ]      R4
      [                  (     )              ]      R2
                                                     R5
```

Boundary logic proof:

```
            ( (p) (q) r ) ((p) q) (p) r              transcribe theorem
            (     (q)   ) (    q) (p) r              pervasion
            (           ) (    q) (p) r              pervasion
```

```
          (           )                             dominion
```

The first proof in the example ref[Dau] illustrates how the AND-logic scroll might interfere with the use of PERVASION to achieve an efficient transformation sequence, such as the second proof. The third boundary logic proof can be interpreted as OR-logic, but it is more appropriately viewed as a diagrammatic proof emphasizing PERVASION, with only the transcription into and out of the parens form having an intentional logical interpretation.

=====
Another example is Leibniz' %%%, ref[dau]

```
      (a-->c & b-->d) --> ((a & b)-->(c & d))

        (a (c)) (b (d))                             transcribe antecedent
                         (a b (c d))                transcribe consequent

      [ (a (c)) (b (d)) [ (a b (c d)) ]]            transcribe theorem
      [ (a (c)) (b (d))    a b (c d)   ]            R5
      [ (  (c)) (  (d))    a b (c d)   ]            R4
      [     c       d      a b (c d)   ]            R5
      [     c       d      a b (   )   ]            R4
      [                        (   )   ]            R2
                                                    R5


    [ ( ((a) c) ((b) d) ) ] ( ((a)(b)) ) ((c)(d))   transcribe theorem
        ((a) c) ((b) d)        (a)(b)    ((c)(d))   involution
        (    c) (    d)        (a)(b)    ((c)(d))   pervasion
        (    c) (    d)        (a)(b)    (      )   pervasion
                                         (      )   dominion
```
=====

Pierce's preference of an AND-logic interpretation for the Sheet of Assertion was partially based on a distaste for interpreting forms SHARING space as alternation. Adopting a literalist position, he wished to confound the existence of a form with its assertion. However, from a computational viewpoint, it is preferable to maximize the use of void-equivalence, eliminating irrelevancies as opposed to adding and erasing implied forms as supported by the mathematical style of constructing rather than deconstructing a proof. Due to their implicative approach, Alpha graphs fail to provide a clear deductive strategy, instead requiring active consideration of which forms to construct.

The complete decoupling of forms SHARING space permits efficient manipulation of bounded forms. The AND-logic form of implication bounds antecedent and consequent in what Peirce called a scroll, (p (q)). A featureless space permits antecedent and consequent to be independent, as (p) q, abandoning the structure of implication in favor of no structure at all.

Another difficulty with AND-logic is that double-cuts are implicated in scrolls. Since the form of implication has an outer boundary, to make progress in a proof, it is necessary to deconstruct (or construct) the scroll boundary by making a double cut. Since the number of rules is small, it is important that each be maximally intuitive, from a diagrammatic rather than a logic perspective. The cost to process clarity imposed by AND-logic does not balance the purely interpretative gain of working with assertions rather than with possibilities.

In summary, Alpha graphs have been widely perceived to be too clumsy and inefficient for general use. The sources of clumsiness are:

-- implicative rather than algebraic style
-- implicative insert/erase rules rather than void-equivalent deletion
-- constructive proof rather than deconstructive deletion

However the dominant computational clumsiness has little to do with Peirce's formulation: a general failure to understand the power of iteration/PERVASION as a mechanism to disentangle the form of logical problems. This failure can be traced back to a predisposition to construct diagrammatic systems to representation specific conventional mathematical systems, rather than to establish a pure diagrammatic mathematics based on the semantic structure of diagrams themselves.

6 SUMMARY

In this paper, we have suggested that it is not a good idea to confine mathematical notation to lines on a page. The benefits of a notation supported by a simple, non-structured space include the elimination of binary scope, duplicity of representation, commutativity and associativity. When such a notation is interpreted as propositional logic, the foundations of traditional symbolic representation can be seen to be overly elaborate.

The key concepts proposed in this paper follow:

\begin{itemize}
\item Non-linear representational spaces free elementary mathematical systems from linear irrelevancies.

\item The empty simple space and the simple token provide a foundational model for formal symbol systems. They refer to the two types of void.

\item The maintenance of simplicity requires constraints both on the duplicity of tokens and on the replication of operations.

\item Functional spaces and boundary-tokens permit the representational collapse of the object-process dichotomy.

\item Representational incompleteness permits notational elegance.

\item Graph notation permits non-duplicity of representation.

\item Elementary logic can benefit from a shift in perspective.
\end{itemize}

To foreshadow the potential utility of parens and graphic notation interpreted as logical expressions, consider the construction of computer programs that perform inference. Such programs are the heart of expert system technology, and are fundamental to the field of Artificial Intelligence. An inference engine based in parens notation can represent deductive tasks more efficiently and reach deductive conclusions in less steps than inference engines based in traditional techniques.\footnote{The formalization of these simplified processes is presented in W. Bricken, A Deductive Mathematics for Efficient Reasoning, unpublished.} Linear Losp is a parens-based engine that uses list processing as its implementation paradigm. An inference engine based in graph notation performs parallel deduction over a network. Parallel Losp implements this facility using a message-passing paradigm in which each node in the network is a small computing system. Both versions of Losp demonstrate the efficiency and elegance of computation in a simple space.

=======================
FT:
It is also interesting to note that capital letter forms have an interpretation as the disjunction of bounded forms. The mapping to logic therefore accepts a disjunction of forms in the place of every spatial name (capital letter variable). The diagrammatic formalism shows us a broader interpretation of all logical connectives, applying equally to the absence of propositions, to single propositions, and to disjunctions of propositions.
=====