Boundary Logic and Alpha Existential Graphs

William Bricken

Boundary Institute
18488 Prospect Road, Suite 14, Saratoga CA 95070 USA
bricken@halcyon.com

Abstract. Peirce's Alpha Existential Graphs (AEG) is combined with Spencer Brown's Laws of Form to create an algebraic diagrammatic formalism called *boundary logic*. First, the intuitive properties of configurations of planar non-overlapping closed curves are viewed as a pure boundary mathematics, without conventional interpretation. Void *representational space* provides a featureless substrate for boundary forms. *Pattern-equations* impose constraints on forms to define semantic interpretation. Patterns emphasize *void-equivalence*, deletion of structure that is syntactically irrelevant and semantically inert.

Boundary logic maps one-to-many to propositional calculus. However, the three simple pattern-equations of boundary logic provide capabilities that are unavailable in token-based systems. *Void-substitution* replaces collection and rearrangement of forms. Patterns incorporate *transparent boundaries* that ignore the arity and scope of logical connectives. The algebra is isomorphic to AEG but eliminates difficulties with reading and with use by substituting a purely diagrammatic formalism for the logical mechanisms incorporated within AEG.

1 Introduction

Barwise [1], Shin [2][3], Hammer [4], Jamnik [5] and others have shown that Peirce's Alpha Existential Graphs (AEG) and other diagrammatic systems for logic are indeed sufficiently rigorous to be relied upon as formal reasoning systems. Shin identifies the next level of criticism of Existential Graphs (EG): "The graphs of EG are difficult to read off, and the rules of EG difficult to use." [3, p.3] Shin then provides two formal improvements to EG that address these weaknesses: a new reading algorithm that allows a greater flexibility of interpretation, and reformulated transformation rules that better expose symmetries in Peirce's original system.

Spencer Brown's *Laws of Form* (LoF) [6] presents a version of diagrammatic logic that maps to Peirce's Entitative Graphs [7, 3.456-552], the dual interpretation of Existential Graphs. Spencer Brown improves upon the rule sets of both Peirce and Shin by formulating diagrammatic logic as an algebraic rather than an implicative system. LoF also introduces an arithmetic of logic based on spatial containment (cuts in AEG). The single ground value is a closed planar curve. The absence of a form, *void*, becomes the second logical value, just as The Sheet of Assertion in AEG absorbs the logical value TRUE into the unmarked page.

Herein, Alpha Existential Graphs is combined with Laws of Form to create an algebraic diagrammatic formalism called *boundary logic*. The axioms of boundary logic are similar to those of both Peirce and Spencer Brown, however they are particularly formulated for ease of use.[1]

The notation of boundary logic (and of AEG and LoF) maps one-to-many onto that of Boolean algebra and conventional implicative logic. Shin's alternative readings of AEG incorporate this one-to-many mapping. Boundary logic transformations are achieved by valid substitutions defined by pattern-equations. The algebraic technique of boundary logic continues Shin's refinement of the rules of AEG to its conclusion, but is derivative of Spencer Brown rather than Peirce.

A one-to-many mapping between formal systems denies isomorphism and affirms the smaller system to be more elegant. This elegance shows up as improved notational and transformational techniques. The one-to-many mapping provides a formal basis for the observation that boundary logic is both easier to read and easier to use than conventional logical systems.

The following presentation is informal, relying on what Shin has called the "intuitively homomorphic visual facts" [3, p.27] of closed curves. A diagrammatic grammar, boundary operators, and an axiomatic basis of algebraic equations over boundary patterns are developed first. Then boundary logic is compared to both propositional calculus and to Peirce's Alpha Graphs.

2 First Class Diagrammatic Formalism

In order to discover the inherent formal properties of configurations of closed boundaries, a pure boundary mathematics is first presented, without anchors to conventional mathematical interpretation. The approach echoes Kempe's goal: "...to separate the necessary matter of exact or mathematical thought from the accidental clothing -- geometrical, algebraical, logical, etc." [11, p.2]

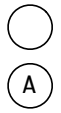
Boundary logic is not isomorphic to propositional calculus. Isomorphism shows mathematical equivalence, but it does not show the ways in which diagrams can be formally superior to token strings. The pattern-equation approach of boundary logic incorporates types of structure that are simply not available within a string-based notation, providing new tools to simplify our understanding of logic.

---

[1] Boundary logic has been extensively tested and applied to Computer Science problems over the last two decades, including theorem proving and satisfiability [8], minimization of software and knowledge-bases, parallel processing [9], visual languages [10], and logic synthesis of the over 200 small and large semiconductor designs included in the ISCAS'89 benchmarks. See the boundary mathematics section of www.wbricken.com.

2.1 Forms

A configuration of closed curves is called a boundary form, or more simply, a *form*. The empty closed curve is called a *mark*. The symbols of boundary algebra consist of one ground form (the empty enclosure or mark), and sufficient capital letters {A,B,C,...} to stand in place of arbitrary forms. The mark is not a token, since it has an inside; however, it is the only symbol in a formal language of boundary forms. There is no "empty form", only an empty mark that serves to ground the spatial grammar. These and no other structures are forms:



is a form. GROUND

is a form iff A is an existent form. BOUNDING

is a form iff both A and B are existent forms. SHARING

This definition of structure embodies two operations for composing forms: BOUNDING and SHARING. The operations are defined to be bidirectional, and thus can be used both constructively and deconstructively to add and remove boundary structure. An enclosure, or boundary, can be constructed in any space, and around any contents. Multiple forms can be constructed in any space, in which case the forms *share* that space. In the diagrammatic language above, labeled forms are required to exist in order to avoid defining operations on void, that is, on the absence of form.

2.2 Representational Space

The space upon which forms are recorded is called the *representational space*. There is a simple perspective that characterizes this space:

Representational space has absolutely no properties.

The representational space of strings has the accidental property of total ordering, requiring rearrangement and precedence rules to express abstract mathematical concepts. One might say that the syntax of string notation interferes with the natural semantics of symmetry by being unidimensional. In contrast, forms are a partial ordering that provides both equivalence (SHARING) and total ordering (BOUNDING).

Here, representational space is identified with void. The emptiness of the underlying substrate does not support metric concepts such as size, shape, location and origin, and does not support topological concepts such as points, proximity, connectedness and continuity. The accidental properties 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 mathematically abstract mark, divorced from its visceral semantics, simply distinguishes an interior and nothing more. As a diagrammatic form, a mark looks like what it means.

Within a featureless space, all structure is directly visible as boundary forms. By construction, forms do not have an implicit structure that requires enforcement by transformation rules. From a diagrammatic viewpoint, the conventional properties of associativity, commutativity, transitivity, and arity are superfluous, indeed misleading, when attributed to forms SHARING representational space. For example, void space does not possess the relational property of commutativity defined for string notation as a rearrangement in the ordering of arguments. More abstractly, commutativity is a type of operator symmetry: ordering of arguments is irrelevant, the operator applies to unordered rather than ordered pairs. The freedom of two-dimensional space permits a direct representation of unordered pairs as two forms occupying an unstructured space. SHARING is not limited to apply to only two forms at a time. Both SHARING and BOUNDING are *variary*: any number of forms, including none, can share a representational space, and can be enclosed by a boundary.

It is important to realize that a boundary does not necessarily separate space into two independent portions. The outermost space could also be seen to pervade all bounded forms within it, just as physical space pervades all objects within it. Unlike the cuts of AEG, an underlying representational space need not be fragmented. Intuitively, representational space supports every boundary, and every form inside and outside of each boundary. Boundaries that separate space into exclusive portions are impermeable; boundaries that permit pervasive spaces are semipermeable.[2]

2.3 Parens Notation

Parens notation is a convenient representation for forms. It is essentially well-formed parenthesis structures with the delimiters interpreted as closed boundaries. Parens notation is used solely as a shorthand for the spatial structure of boundary forms. An example of a parens abbreviation and the form that it stands for follows:

The set of well-formed parentheses is a well-known mathematical structure, since operator precedence is conventionally expressed using parentheses. Unfortunately, the conventional definition of well-formed parentheses violates the closure property of a boundary by addressing the "left" and the "right" delimiter independently, an excellent example of how string formalism can modify the semantics of spatial forms.

---

[2] Boundaries that are fully permeable are not boundaries, since they fail to make a distinction.

Parens notation has the accidental properties of syntactically fragmented boundaries and of ordering forms in one dimension. Placing forms in a canonical order greatly improves the efficiency of parens processing, but only adds unwanted syntactic complexity to the abstract idea of spatial enclosure. Parens forms are foremost data structures, they bear only partial resemblance to their intended meaning.

2.4 Equality

Comparing boundary forms for equality is restricted at this point to forms that are structurally identical, since pattern-equations have not yet been incorporated. Intuitively, two forms are identical when every form within each is identical. A recursive definition of equality is supported by the deconstructive direction of the boundary form grammar.[3]

```
( ) = ( )
(A) = (B)   iff   A=B
A B = C D   iff   A=C  and  B=D
```

Equality includes the common transformations associated with equations:[4]

Form Substitution: `A=B` implies `subst[W,X,A]=subst[W,X,B]`

Form Replacement: `A=B` and `W=X` implies `A=subst[W,X,B]`

Functional Substitution: `A=B` implies `F[A]=F[B]`

Substitution and replacement furnish the entire transformational mechanism of boundary logic. Pattern-equations use form replacement; evaluation and renaming use form substitution. Functional substitution permits `BOUNDING` and `SHARING` to be applied constructively to both sides of a valid equation.

2.5 Pattern-Equations

Forms consisting of boundaries, with or without form variables, are patterns. In a diagrammatic formalism, spatial patterns take the place of relational properties such as transitivity and symmetry. Patterns that can be validly substituted for one another are expressed as *pattern-equations*. Pattern-equations define pre- and post-conditions for transformations. For instance, permission to construct and to eliminate replicate

---

[3] To avoid implicating a specific mechanism for testing pairs of subforms, variable matching in the decomposition of `SHARING` implicitly includes backtracking in case of a poor selection of which pairs to compare.

[4] `subst[X,Y,Z]` reads "substitute `X` for every `Y` in `Z`".

bounded forms SHARING a space might be asserted as the pattern-equation B B = B, where B stands for any boundary form.

2.6 Void-Equivalence

Computation over boundary forms involves the use of void in fundamental ways.

A featureless void space assures that the only relational property ascribed to forms SHARING space is universality. SHARING is a universal relation that in conventional terms is trivially reflexive, associative, symmetric and transitive. Abstractly, many relational properties reflect characteristics of the substrate representational space. Conventional string-based relational properties are not "implicit" or "tacit" in representational space, they are simply not relevant.[5]

Most pattern-equations identify forms with structure that can be deleted. Structure that is equal to no structure is *void-equivalent*. Strictly, structures that reduce to nothing are imaginary. The idea of void-equivalence emphasizes that boundary pattern transformations are achieved by *void-substitution* (that is, by deletion or erasure), in contrast to transformation by rearrangement and evaluation that is characteristic of string-based systems. Deletion operations confer excellent behavior on boundary computation; for instance, rapid convergence is assured.

Void-equivalent forms can be brought into syntactic existence in any representational space. Like the virtual particles of the physical vacuum, void-equivalent forms permeate the entire representational space. Such forms are virtual, they can interact with existent forms as catalysts and then subside back into non-existence. To Peirce, "Every blank part of the sheet is a partial graph." [7, 4.397].

Void-equivalence strongly impacts the conventional approach to semantics. Intuitively, syntactic structures that turn out to be equivalent to no structure at all cannot influence the value of a form. Thus, void-equivalent forms can be manipulated by pattern-equations without concern for validity. This is the Principle of Void-Equivalence:

> Void-equivalent forms are syntactically irrelevant and semantically inert.

Each of the inference rules in AEG incorporates addition or deletion of structure. Other conventional forms of inference rely upon rearrangement and accumulation of

---

[5] "Operations of commutation, like xy therefore yx, may be dispenced [sic] with by not recognizing any order of arrangement as significant. Associate transformations, like (xy)z therefore x(yz), which is a species of commutation, will be dispenced with in the same way..." Peirce, [7, 4.374].

structure. Inference (and computation) via void-equivalence is a most important innovation in AEG, other features such as the Sheet of Assertion and multiple readings derive directly from this fundamental change in the mechanism of valid reasoning.

2.7 Laws of Form

In contrast to Peirce's implicative approach, Spencer Brown casts the diagrammatic logic of enclosure within an algebraic formalism. One of Spencer Brown's most important contributions is the semantic use of the absence of a mark. While Peirce treated non-representation tangentially, Spencer Brown made it central to the understanding of LoF. He provided a modern axiomatization of diagrammatic logic, whereas Peirce struggled to identify components of the yet-to-be-formulated axiomatic method.

Another fundamental contribution of LoF is the specification of the axioms of an arithmetic of enclosures. The boundary arithmetic presented in LoF consists of two initial equations relating replications of the one constant.

```
( ) ( ) = ( )          CALLING
 (( ))  =              CROSSING
```

The left-sides of these equations are quite natural, they identify the two configurations that are possible when a replicated mark is constructed in the same representational space as the original mark. The replicate can be drawn outside of the original, in which case the two marks are SHARING space. The replicate can be drawn on the inside of the original, in which case the two marks are nested, the outer boundary is BOUNDING the inner mark.

Although the two configurations of replicates are structurally determined, the value of each configuration on the right-side is a choice, defining a particular boundary arithmetic. These two particular rules partition the set of forms into two equivalence classes, those that reduce to mark and those that reduce to nothing at all.

3 Interlude

Boundary algebra is defined by the set of forms, the structural operators BOUNDING and SHARING, the mechanisms of substitution and replacement, and specific assumed pattern-equations that align with the intended semantics of forms. Pattern-equations can be used to express the result of operations, the presence of void-equivalent forms, the constraints of a particular interpretation, the rules of inference, and other desired theorems used to transform and manipulate forms.

The language of well-formed boundaries has a natural and visually apparent semantics, that of partial ordering, or containment. Containment (as membership) is the fundamental relation of set theory, partial ordering is the fundamental relation of order and lattice theory, and since the logical conditional is an inclusion relation, the concept of containment is also sufficient to express propositional calculus and Boolean algebra, as AEG shows.

The pure mathematics of boundary logic has been addressed thus far. Operations can be applied to boundary structures without having to consider the meaning of a form. A specific boundary algebra consisting of a set of pattern-equations is defined next. Pattern-equations define valid transformations within an interpreted application of boundary mathematics (such as logic). As with any mathematical system, the representation and mechanism of the system is independent of the particular application. Thus, an interpretation of boundary algebra for logic is constrained to entry into the boundary formalism, via transcription from logic to forms, and to exit from the formalism, via interpretation of forms for logic. Algebraic manipulation of forms via pattern-equations is then free of logical semantics. This in turn facilitates the removal of difficulties encountered while reading and using AEG.

The relationship between boundary forms and conventional logic can be viewed from many perspectives, each of which requires a longer discussion. They include:

- from first principles, the nature of void and mark
- from the nature of boundaries
- from the semantics of containers and containment
- from the arithmetic and algebra of boundaries in LoF
- from the implicative boundaries in AEG
- from the mapping of forms to logic connectives
- from partial order theory
- from properties of the `BOUNDING` and `SHARING` relations
- from viewing `BOUNDING` and `SHARING` as properties of boundaries
- from viewing `BOUNDING` as a binary relation between spaces

Here, Peirce's lead is followed, his transformation rules are simply transcribed into boundary pattern-equations. For an application to logic, pattern-equations partition the set of forms into one of two equivalence classes, which are interpreted as the class of `TRUE` forms and the class of `FALSE` forms. The structures and transformations of boundary logic, however, are expressed in terms of diagrammatic concepts such as boundary forms, void-equivalence, and boundary semipermeability.

4 The Map to Logic

The following map is a bridge between the diagrammatic semantics of forms and the linguistic semantics of logical connectives. This map reduces propositional calculus

and two-valued Boolean logic to a unary boundary logic with one ground value (the mark or the empty boundary). The other ground value is imaginary, FALSE forms are cast into void and thus into non-existence.

LOGIC	BOUNDARY	ABSTRACTION
false		void
true	( )	ground
not A	(A)	BOUNDING property
if A then B	(A) B	BOUNDING relation
A or B	A B	SHARING relation
A and B	((A)(B))	a compound form
A iff B	(((A) B)((B) A))	equivalence relation

It is illustrative to transform DeMorgan's law into parens notation:

$$A \wedge B = \neg(\neg A \vee \neg B) \qquad ((A)(B)) = ((A)(B))$$

Expressions that are structurally different in conventional logic may not be different in boundary logic. To understand void-equivalence, the following transcription is useful:

$$((\text{false} \vee \text{false}) \rightarrow \text{false}) \vee \text{false} \qquad ( )$$

4.1 Axioms of Boundary Logic

The mechanisms of logical inference, of Boolean transformation, and of proof strategy reduce to three pattern-equations that can be considered to be the axioms of boundary logic. The right-side of each equation identifies void-equivalent structure that can be deleted from the left-side.

(A ( ))	=		OCCLUSION
((A))	=	A	INVOLUTION
A {A B}	=	A {B}	PERVASION

Capital letters refer to any form regardless of complexity, including multiple forms SHARING a space, and the absence of any form. Capital letters can also refer to propositions, forms that, when interpreted for logic, take on one of two ground values.

The curly brace in PERVASION is a *meta-boundary*, standing in place of any intervening content, including none. What is outside can be deleted or inserted inside, independent of the depth of nesting within a form. Conventionally, PERVASION would be called an axiom schema, a rule abstract standing in place of a set of specific rules that apply to specific forms. In terms of boundary logic, however, PERVASION is a single rule that expresses the semipermeability, or transparency, of boundaries

from the outside. Strictly, semipermeable boundaries are invisible from their transparent side; they do not support counting depth of nesting or taking steps sequentially through each level. Thus PERVASION is yet another application of void-equivalence: relative to an outside form, both intervening boundaries and inner replicates are void-equivalent.

4.2 Alpha Graphs

Peirce 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 a transformed graph would result in an "absurd" configuration, such as an 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 alternating types of 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.

Peirce defined his transformation rules (permissions) to be the "Pure Mathematical Definition of Existential Graphs, Regardless of their Interpretation". That is, he considered the rules to be structural rather than semantic. Peirce developed AEG for logic, and did not call upon the notational advantages of an algebra. Equality does not play a role in Alpha Graphs, leading Peirce to state bidirectional transforms as two separate rules, one for each implicative direction.

Expressing the scope of Peirce's diagrammatic rules in conventional string notation, or in conventional binary operator interpretation, is quite difficult. Logical interpretation of AEG generally generates a shallow approximation of the strength of Peirce's diagrammatic rules, since a logical interpretation must limit the concept of boundary semipermeability expressed by PERVASION to impermeable binary connectives. 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. This rule has been transcribed by Norman [12] as

$$\varphi \rightarrow \psi \quad \models \quad \varphi \rightarrow (\varphi \wedge \psi)$$

The transcription captures little of the intent of the rule of iteration. In the OR-logic of Entitative Graphs, for example, all four forms in the following table are equivalent in value. The final three forms are derived from the first by a single application of iteration. The iterated form is highlighted.[6]

---

[6] Square brackets are used to highlight specific parens boundaries. Otherwise, they are identical to round parentheses.

	CONVENTIONAL LOGIC	ENTITATIVE GRAPH
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](D))))
S4.	A→(B∧(C→ ¬(A→ D))))	(A)((B)((C) ([A] D))))

S2 is easily derivable from S1; if A implies a conjunction, then A also implies that conjunction with A added. In S3, ¬A jumps over two connectives to build a negation with a new disjunction. In S4, A jumps over two connectives to build a new conditional, while shifting the negation of D to the outside of the new conditional.

Not only is it challenging to prove the equivalence of forms derived through deeply nested iteration by conventional means, it is entirely foreign to skip over connectives, to add new connectives deep within an expression, and to indirectly move negations within an expression. The central point, though, is that iteration is a single, succinct rule for diagrammatic forms, and cannot be expressed as a rule using string-based concatenation of binary connectives. The diagrammatic concepts of transparency and semipermeability are not included in the concepts of conventional mathematics.

4.3 Boundary Logic and Alpha Graphs Compared

AEG applies the diagrammatic structure of enclosure specifically to logic. The representations of boundary logic and AEG are isomorphic, while the systems of transformation rules are remarkably close to being the same. The primary differences between the two systems are:

- AEG uses the dual interpretation of boundary logic, for which empty space is deemed TRUE as opposed to FALSE. Peirce earlier developed Entitative Graphs [7, 3.456-552] that use emptiness as FALSE and space as disjunction, thus aligning with boundary logic.

- AEG is assertion-based; forms are taken as premises. Boundary logic is structure-based; forms are algebraic and do not require an interpretation as logic.

- Boundary logic provides a more modern algebraic transformation system, uniting pairs of asymmetrical implicative rules into symmetrical equations.

- Boundary logic assumes a homogeneous empty page, while the cuts of AEG are taken to severe the page into discrete portions.

- The implicative add and erase rules of AEG fail to provide a clear termination goal. Boundary logic uses a single void-equivalence rule.

- The highly efficient algebraic axiomatization of boundary logic alleviates pragmatic difficulties of transforming forms in AEG.

These differences are overshadowed by the profound differences between conventional string-based and diagrammatic logics. Both AEG and boundary logic take a diagrammatic formalism as foundational; both assign a semantics to empty space; both use PERVASION as a pattern rule rather than a functional transformation; and both incorporate void-equivalence into every transformation rule.

4.4 Boundary Logic and Alpha Graph Rules

The rules of AEG (AND-logic), Entitative Graphs (OR-logic), and boundary logic are transcribed into symbolic notation below.[7]

RULE	ALPHA	ENTITATIVE	BOUNDARY
R1. Erase	((A) B) ⊨ (( ) B)	((A) B) ⊨ ((A))	(A ( )) =
R2. Insert	(A) ⊨ (A B)	A ⊨ A B	
R3. Iterate	A (B) ⊨ A (A B)	A (B) ⊨ A (A B)	A {B} = A {A B}
R4. Deiterate	A (A B) ⊨ A (B)	A (A B) ⊨ A (B)	
R5. Double cut	A = ((A))	A = ((A))	A = ((A))

Alignment of Rules. Double cut, R5, is INVOLUTION in boundary logic. INVOLUTION is the only rule that has a clear isomorphic interpretation as a conventional rule, that of double negation. In boundary logic, this theorem is not fundamental, it can be derived from OCCLUSION and PERVASION alone, given an antisymmetry property of boundaries that provides a bridge between equational and implicative forms.

Rules R1 and R2 give permission to erase and to insert forms in an Alpha Graph, with the exception that an empty boundary cannot be erased. This grounds the representation of AEG with two halting conditions, the empty page taken as TRUE and the empty boundary taken as FALSE. Rather than inserting forms to achieve a

---

[7] The numbering of rules follows Roberts [12]. For readability, the rules for AEG and for Entitative Graphs do not include a notation for application at arbitrary depths. For R1 and R2, the representation of deep rules would require identification of odd and even depths of nesting.

reduction pattern as in R2, boundary logic begins by recording the entire problem, including the postulated conclusion when available. Forms are then deleted until either void, a single mark, or a form contingent on variables remains.

The boundary logic rule of OCCLUSION identifies the void-equivalent pattern that unites R1 and R2:

(( ) A) = OCCLUSION

DOMINION is a variation of OCCLUSION that is used only once, as the final terminating step of a valid deduction. It can be derived from OCCLUSION by BOUNDING each side (functional substitution) and applying INVOLUTION.

( ) A = ( ) DOMINION

Insert/erase rules in AEG 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.

	R1: ERASE	R2: INSERT	
Conventional logic:	simplification $A \wedge B \vDash A$	addition $A \vDash A \vee B$	
Entitative graphs:	erase odd ((A)(B)) ⊨ ((A))	insert even A ⊨ A B	
Boundary logic:	[((A)(B))] ((A))	[A] A B	
	(A)(B) A	[A] A B	involution
	( )(B) A	[ ] A B	pervasion
	( )	[ ]	dominion

In boundary logic, R1 and R2 are simply specific applications of PERVASION combined with DOMINION. This is easily seen above by converting the implicative forms of R1 and R2 into boundary form and reducing using boundary logic axioms. Essentially, R1 and R2 combine into the single rule of DOMINION in OR-logic, and into the rule of OCCLUSION in Peirce's preferred AND-logic.

The separation of Alpha Graphs into two discrete domains (odd and even depths of nesting) is undermined by PERVASION, creating subtle inconsistencies in the description of AEG rules. This leads to an inappropriate embedding of logic into the manipulation of Alpha Graphs. Boundaries are seen to require alternate interpretations via varieties of DeMorgan laws, creating an overemphasis on counting the depth of nesting. PERVASION requires no count of depth; in fact, the notion of depth of nesting is irrelevant to each boundary logic rule.

Mathematical Abstraction. The differences between the AND-logic and OR-logic interpretations of boundary forms can be strictly limited to transcription from logic to boundaries and from boundary logic back into conventional logic. Thus, there is no need to carry 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.

Within boundary logic, BOUNDING is not associated with logical negation, it is simply adding an enclosure. Double cut/INVOLUTION is a structural void-equivalence having no association with double negation, although it does keep track of the logical parity of spaces structurally. Similarly, SHARING is not associated with disjunction or with conjunction, it is simply a property of forms contained within the same outer boundary. But independence from logical interpretation is most prevalent in PERVASION, since it calls upon the semipermeability of boundaries to transcend the scope and arity of logical connectives by rendering them transparent.

Entailment. In the table of AEG and boundary logic rules above, pairs of logical entailments, $\models$, are combined into a single pattern-equation. In conventional logic, syntactic proof, $\vdash$, is a series of syntactic steps grounded by axioms that define the validity of each step solely in terms of material implication. Syntactic proof is related to semantic entailment by the Deduction Theorem, which establishes that the two are equivalent. In an algebra, proof rests on a sequence of syntactic substitution steps grounded by axiomatic equations that define valid substitutions, while "entailment" rests on the validity of substitution and replacement themselves. The equivalence of proof and validity is built into the algebraic mechanism.

In boundary logic, proof consists of a sequence of void-substitution steps grounded by axiomatic pattern-equations that define valid void-equivalent forms. The Principle of Void-Equivalence, which asserts that algebraic manipulation of void-equivalent forms is independent of semantics, achieves the objectives of the Deduction Theorem, that of assuring that symbol manipulation does not undermine intention. Comparative entailment then reduces to the mapping between proof steps and pattern substitutions:

$$X = Y \quad \text{iff} \quad X \vdash Y \text{ and } Y \vdash X$$

This definition bridges between logical and algebraic forms. The parens version is:

$$A = B \quad =\text{def}= \quad (([A]\ B)([B]\ A)) = (\ )$$

with the highlighted boundaries standing in place of syntactic proof steps. That is,

$$X \vdash Y \quad \text{iff} \quad [X]\ Y = (\ )$$

Here, the steps of syntactic proof have been transcribed into an algebraic constraint that a sequence of void-substitutions must reduce a form to a mark, and in particular, that the consequent form, Y, must render the antecedent form, X, to be void-equivalent through PERVASION. Logical implication is simply an assertion about two forms, potentially compound, separated by a simple boundary. The boundary that represents syntactic proof is the same boundary used to represent syntactic form. The multiple steps in a proof become multiple applications of pattern-equations. The mechanism of PERVASION (and the other boundary logic pattern-equations) is functionally identical whether representing the conventional concepts of proof, logical implication, or material implication. Only the form of the contents, X, and the context, Y, differ.

Distribution. In contrast to Boolean algebra, DISTRIBUTION is not an axiom of boundary logic. DISTRIBUTION is required to convert deeply nested forms into conjunctive normal form (CNF) with only two levels of nesting. CNF, in turn, is used as a canonical representation underlying the proof of metatheorems of logic.

DISTRIBUTION is an axiom in LoF, however in boundary logic, DISTRIBUTION is a consequence of PERVASION. This has significant impact on how to think about proof in diagrammatic logic. CNF trades multiple reference to variables for a shallow nesting of boundaries in order to accommodate a conventional binary connective reading. 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 with descent into nested connectives. An appropriate (semi)canonical representation for boundary logic has minimal replication of variable labels and maximal nesting of boundaries.

The boundary rule of DISTRIBUTION can be rewritten to emphasize the relationship between variable replication and depth of nesting:

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

(A ((B)(C))) = (((A B)(A C)))          functional substitution
             =   (A B)(A C)            involution
```

The left-side has three levels of nesting while the right-side has only one. The right-side de-emphasizes applications of PERVASION, while the left-side de-emphasizes replication of labels.

A Proof Example. The *self-distributive law of the conditional* serves as an example of the deductive procedures of AEG and of boundary logic. Two proofs follow. The first AEG proof from Roberts [13, p46] illustrates how the directionality of the rules

of AEG interferes with the maximal use of void-equivalence to achieve efficient transformation sequences.[8] The second proof is that of boundary logic..

Implicative theorem: `(p → (q → r)) → ((p → q) → (p → r))`

Alpha Graph proof (the outermost scroll is highlighted):

```
      [p [(q (r))]]                                          antecedent A
                        [(p (q))  [(p (r))]]                 consequent C
   [ (p ((q (r)))) [ ((p (q))  ((p (r)))) ]]                 theorem (scroll)

1.                                                           sheet of assertion
2.  [               [                      ]]                R5
3.  [ (p ((q (r)))) [                      ]]                R2
4.  [ (p ((q (r)))) [ ( p     ((q   (r)))) ]]                R3
5.  [ (p ((q (r)))) [ ( p       q   (r)  ) ]]                R5
6.  [ (p ((q (r)))) [ ( p   (  (q)) (r)  ) ]]                R5
7.  [ (p ((q (r)))) [ ( p   (p (q)) (r)  ) ]]                R3
8.  [ (p ((q (r)))) [ ((p (q))  ((p (r)))) ]]                R5, QED.
```

Step 3 of the proof inserts the antecedent `A` at an odd depth. Step 4 migrates `A` into the position of the consequent in the scroll. The task is now to transform the migrated form `A` into the form of the consequent `C`. `A` is first partially deconstructed in Step 5, and then rebuilt in Steps 6, 7 and 8. Presumably Steps 5-8 require some thought and planning. The plan for Steps 5-8 can most easily be developed backwards, converting `C` into `A`. It would be more natural, in fact, to reverse the entire procedure, but this is blocked by the (avoidable) directional Step 3. The proof may just as well have begun at Step 4, by introducing the tautological form `(A (A))` onto the sheet of assertion. Even more straightforward would be to strip both `A` and `C` of both double cuts and iterated forms, both bidirectional transformations in AEG, which then leaves `A` and `C` the same. A maximal deletion strategy above would proceed as follows:

```
   [ (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
```

But here, Peirce's approach would hang up, since deletion at odd depths is not permitted. Peirce was focused on logical implication as a technique of thought. The form `(X ( ))` could be generated but not deleted. That is, proof steps are not reversible. When read for logic, this form is: `X → true`, certainly palatable as an

---

[8] Proofs by Roberts [13, p46], Shin [3, p95], Hammer [4, p104], and to a lesser extent Dau [14, p215] contain unnecessary steps that can be attributed to the implicative approach of AEG.

assertion. But apparently Peirce did not avail himself of the safe tautology (X (X)), which reads X → X, and can be freely added to or erased from the sheet of assertion.

The boundary logic proof uses the dual space of AEG that emphasizes PERVASION and directly implements the maximal deletion strategy suggested above:

((p) (q) r) ((p) q) (p) r	transcribe theorem
((q)) (q) (p) r	pervasion (R4)
( ) (q) (p) r	pervasion (R4)
( )	dominion, interpret as TRUE

5 Conclusion

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
- entanglement of logic in purely diagrammatic pattern reduction

However, the dominant computational clumsiness has little to do with Peirce's formulation. There has been a general failure to understand the power of deiteration/ PERVASION as a mechanism to disentangle the structure of logical problems. This failure can be traced back to a predisposition to construct diagrammatic systems as representations for specific conventional mathematical systems, rather than to establish a pure diagrammatic mathematics based on the intuitive structure of boundaries themselves.

Boundary logic eliminates the sources of clumsiness in AEG by using mathematical abstraction to remove logical inference from diagrammatic pattern substitution. A cost is to introduce the unfamiliar formal diagrammatic concepts of boundary forms, void-equivalence, and semipermeable boundaries. Each of these concepts is available within both AEG and LoF. Both systems use boundary forms. It takes Spencer Brown's algebraic approach to bring out the power of void-substitution. LoF however, treats crossing each boundary by PERVASION as a separate step. It takes AEG's sweeping use of deep iteration and deiteration to bring out the power of semipermeable boundaries. But AEG fails to take the final step in the use of PERVASION, that of permitting OCCLUSION to terminate a deduction. Alternating logical depths in the AEG boundary form coupled with unidirectional implication prevent this. Therein is the contribution of boundary logic: to combine an algebraic formalism with deep PERVASION of diagrammatic boundary forms.

More fundamentally, diagrammatic mathematics can stand alone, without an interpretation as a conventional mathematical system. Indeed, the properties of BOUNDING and SHARING assure that boundary mathematics does not mesh well with conventional techniques. But it is not uncommon to find an unexpected formal technique that introduces new types of imaginaries in order to enhance understanding of a conventional system. For example, angular orientation in physical space is described by Euler angles (1752) in a natural, vector-oriented way. Hamilton's quaternions (1843) introduce new types of imaginary units within an independent algebra. Quaternions can be *interpreted* as orientation in physical space [15]. Peirce took the first step with the diagrammatic logic of AEG. Spencer Brown incorporated void-equivalence into an algebra that stands independent of logic. Boundary mathematics is built upon this new type of imaginary: void-equivalent form in a featureless space. It provides computation via void-substitution that eliminates imaginary diagrammatic structure. And it can be interpreted as logic. Courant and Hilbert (1953) provided a natural spatial understanding of quaternions. The challenge issued by boundary logic is to find a natural understanding of logic based on the idea that one-half of each logical dual is non-existent.

References

1. Barwise, J., Etchemendy, J. Heterogeneous Logic. In: Allwein, G, Barwise, J. (eds.) *Logical Reasoning with Diagrams*. Oxford University Press (1996).
2. Shin, S. *The Logical Status of Diagrams*. Cambridge University Press (1994).
3. Shin, S. *The Iconic Logic of Peirce's Graphs*. MIT Press (2002).
4. Hammer, E.M. *Logic and Visual Information*. CSLI, Stanford (1995).
5. Jamnik, M. *Mathematical Reasoning with Diagrams*. CSLI, Stanford (2001).
6. Spencer Brown, G. *Laws of Form*. George Allen and Unwin (1969).
7. Peirce, C.S. *Collected Papers of Charles Sanders Peirce*. Hartshorne, C. Weiss, P., Burks, A. (eds.) Harvard University Press (1931-58).
8. Bricken, W., Gullichsen, E. An Introduction to Boundary Logic with the Losp Deductive Engine. *Future Computing Systems* 2(4) (1989) 1-77.
9. Bricken, W. Distinction Networks. in Wachsmuth, I., Rollinger C.R., Brauer, W. (eds.) *KI-95: Advances in Artificial Intelligence*. Springer (1995) 35-48.
10. James, J. Bricken, W. A Boundary Notation for Visual Mathematics, *1992 IEEE Workshop on Visual Languages*, Seattle, IEEE Press, (1992) 267-269.
11. Kempe, A.B. Memoir on the Theory of Mathematical Form. Philosophical transactions of the Royal society of London 177 (1886) 1-70.
12. Norman, A.J. Diagrammatic Reasoning and Propositional Logic. MPhil Thesis, University College, London (1999).
13. Roberts, D.D. *The Existential Graphs of Charles S. Peirce*. Mouton (1973).
14. Dau, F. Mathematical Logic with Diagrams. http://www.dr-dau.net/ (2005).
15. Shoemake, K. Animating Rotations With Quaternion Curves. Proceedings of SIGGRAPH '85, p.245-252.