Syntactic Variety in Boundary Logic

William Bricken

Boundary Institute, 18488 Prospect Road, Suite 14, Saratoga CA 95070 USA

Abstract. *Boundary logic* is a formal diagrammatic system that combines Peirce's Entitative Graphs with Spencer Brown's Laws of Form. Its conceptual basis includes boundary *forms* composed of non-intersecting closed curves, *void-substitution* (deletion of irrelevant structure) as the primary mechanism of reduction, and *spatial pattern-equations* that define valid transformations. Pure boundary algebra, free of interpretation, is first briefly described, followed by a description of boundary logic. Then several new diagrammatic notations for logic derived from geometrical and topological transformation of boundary forms are presented. The algebra and an example proof of *modus ponens* is provided for textual, enclosure, graph, map, path and block based forms. These new diagrammatic languages for logic convert connectives into configurations of containment, connectivity, contact, conveyance, and concreteness.

1 Introduction

Since antiquity, logical connectives have been presumed to be abstract; they are the *syncategoremata*, words that refer to nothing but themselves yet function to connect words that do have referents [1, p233]. Since logical connectives have no explicit form, they are represented by meaningless tokens. Peirce's Alpha Existential Graphs (AEG) [2 (1896)] introduces a radical re-conceptualization: the connectives of formal logic can take the form of diagrammatic structures consisting of closed non-intersecting curves, or *boundaries*. Composition of boundaries sharing the same space, and nested within each other, creates a spatial pattern language that is sufficient to express the sentences of propositional calculus. Traditionally, propositional rules of inference permit new sentences to be added to the collection of valid sentences, a strategy of accumulation. The diagrammatic reasoning in AEG follows a fundamentally different strategy: boundary patterns are transformed by rules that *add and delete* boundary structure. Peirce shows that inference can be achieved by creation and destruction, rather than by accumulation.

The *boundary logic* presented herein was introduced by G. Spencer Brown in *Laws of Form* (LoF) [3]. The mathematics is equational rather than inferential; like Boolean algebra, valid transformations are specified by equations. Boundary logic uses the boundary pattern language introduced by Peirce, but the add-and-delete rules of AEG are incorporated into *pattern-equations* that permit deduction to proceed solely though rewrite rules that delete structure. Pattern-equations define equivalence sets on boundary forms; the algebraic formulation replaces one-directional inference with the familiar bidirectional algebraic rules of substitution and replacement of equals.

The pure algebraic mathematics of boundaries [4] is based on the concept of *distinction*, or difference. It is constructed *de novo*, without reference to logical, set theoretic, relational, numeric, or categoric objects. Boundaries are strictly structural,

D. Barker-Plummer et al. (Eds.): Diagrams 2006, LNAI 4045, pp. 73 – 87, 2006.

representing only the abstract concept of difference, without requiring identification of the type of object being differentiated. Thus, boundary mathematics differs substantively from the conventional mathematics of strings.

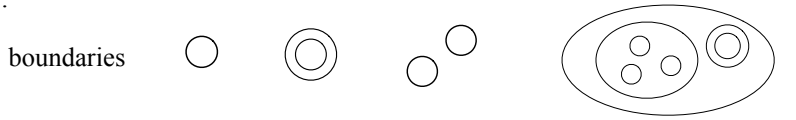
The abstract structure of boundary algebra is described first in Section 2, including the basic concepts of spatial operators with arbitrary numbers of arguments, spatial pattern-equations, and permeable boundaries. Boundary logic, the application of boundary algebra to logic, is then described in Section 3. Two new tools for deduction are introduced: void-substitution and boundary transparency. Section 4 shows a sequence of geometrical and topological transformations of boundary forms that generate over two dozen new diagrammatic notations for logic. Each of these notations provides potential new tools for Cognitive Science and for Computer Science. The structure of each notation suggests unexplored models of how we might read, analyze, manipulate, compute with, and think about deductive logic. The notations also suggest a wide diversity of data structures and algorithms for both hardware and software implementation of logic. Other than Peirce's original concept of a diagrammatic logic expressed as boundaries, and Spencer Brown's equational axiomatization of the same logic, Sections 2 and 4 are new. The axiomatization in Section 3 is the author's.[1]

2 Boundary Algebra

Composition of closed, non-intersecting planar curves, called *boundaries*, constructs a formal diagrammatic language, independent of an interpretation as logic. The alphabet is a singleton set of symbols consisting of the *empty boundary*, { ○ }, which is called a *mark*. A word consists of replicates of marks composed in a non-conventional manner. Since boundaries have both an inside and an outside, replicate symbols can be juxtaposed in two ways: on the inside of the original boundary and on the outside of the original boundary. Rather than one "concatenation" operator, there are two: `SHARING` is composition on the outside, while `BOUNDING` is composition on the inside. The formal language consists of the set of composable boundary forms.

2.1 Parens Notation

Delimiting tokens such as parentheses, brackets, and braces can be used as a shorthand notation for closed planar curves. Herein, parentheses that stand in place of boundaries are called *parens*. Some boundary structures and their parens abbreviations follow:



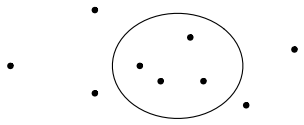
Above, the first form is the empty boundary; the second form is an elementary application of `BOUNDING`; the third, an elementary application of `SHARING`. The fourth

[1] This axiomitization of boundary logic has been extensively tested and applied to computer science problems over the last two decades, including theorem proving and satisfiability [5], minimization of software and knowledge-bases, parallel processing [6], visual languages [7], 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`.

parens form is compound. Well-formed parentheses are well understood; well-formed parens are identical but for the semantic constraint that parens do not have an addressable "left" and "right" portion, since this would violate the closure of the boundary. One accidental property [8, p3] of a parens is that it appears to be fragmented into two portions, another is that parens line up in an apparent sequential order, whereas boundaries can rest in any portion of the space they are drawn in.

2.2 Variary Operators

The functional basis set of the language of boundaries consists of two constructive operators, `BOUNDING` and `SHARING`. These diagrammatic operators are quite unconventional. An enclosing boundary can *bound* any number of forms, including none. Any number of forms can *share* a space, including none. In the example below, dots represent other single bounded forms. The explicit boundary encloses several forms, and while several others share the same external space.



The operators `BOUNDING` and `SHARING` do not have a specific arity, both are *variary*. Absence of a specific arity strongly differentiates the boundary formalism from modern algebra. The closest conventional description for a variary operator is that of a "single argument set function". Conventionally, relations are defined by a set of ordered pairs. Relational properties such as commutativity and transitivity describe the symmetries that the relation imposes upon its ordered arguments. A boundary, however, can contain any number of forms. The collection of forms contained within a boundary is *a priori* neither countable nor orderable. Since forms are not taken two-at-a-time, there is no concept of associativity. By construction, boundary mathematics does not support the numerical concept of arity nor the proximal concepts of commutativity and associativity. Boundaries are neither functions nor relations.

The structure of a boundary form is defined explicitly by the boundaries themselves. The space upon which boundaries are imposed is void; *void space* is neither metric, nor geometric, nor topological. Thus, there is no concept of geometric or topological localization that applies to boundaries. Boundaries can reside anywhere so long as they do not intersect other boundaries. Another consequence is that the boundary language includes no "empty word" other than the entire space that supports all boundaries. Since there is no specifically defined relative position for forms `SHARING` a space, an empty placeholder is not necessary. The idea of a null boundary is wrapped up within the ground symbol; absence of form is simply the inside of the mark, `()`.

Seen as a relation, a boundary distinguishes what is inside from what is outside. The most natural conventional interpretation of boundary forms is as a partial ordering, with `BOUNDING` providing a strict ordering, and `SHARING` providing an equivalence class of forms in the same space. Even so, since both `BOUNDING` and `SHARING` are variary, a relational interpretation is difficult. The conceptualization represented by the boundary language does not support conventional relational properties such as reflexivity, symmetry and transitivity. Conventionally, variary relations are universal.

2.3 Pattern-Templates and Pattern-Equations

Let the set of capital letters, {A,B,...}, provide variables that can stand in place of any boundary form, including many forms and no forms, and the set of small letters, {a,b,...}, provide variables that can stand in place of either a mark, or the absence of a mark. When variables are included within boundary forms, for example (A ((B))), forms can be used as *pattern-templates* to identify specific structure within other forms. The pattern-template (A ((B))) matches ((()) (()())), with A=(()()) and B=*void*. Pattern matches are spatial rather than textual, while pattern variables can match many forms within a space, as well as matching no forms at all[2].

Similar to the idea of extensionality in set theory, two bounded forms are identical only if they enclose identical contents. A *pattern-equation* is an assertion that two structurally different patterns are equal; forms that match either pattern have the same value (modulo the asserted equation), partitioning the set of forms into equivalence classes. The *semantics* of the boundary language is defined by pattern-equations that assert specific patterns, or pattern-templates, to be equivalent. A set of pattern-equations create a particular boundary algebra.

Consider the following two pattern-equations from Laws of Form as providing an evaluation function for forms not containing variables. The two equations reduce any form either to a mark, (), or to the absence of a mark, thus establishing two equivalence classes, mark-equivalence and void-equivalence[3].

```
( ) ( ) = ( )          SHARING EVALUATION
 (( ))  =              BOUNDING EVALUATION
```

2.4 Boundary Permeability

Since boundaries have two sides, they support two types of crossing over. *Impermeable* boundaries block crossing both from the inside outward and from the outside inward. *Semipermeable* boundaries block crossing from one side only, the other side is *transparent*. Boundaries are taken to enclose their contents, establishing the convention that semipermeable boundaries block crossing from inside outward. Fully permeable boundaries are imaginary, since a permeable boundary is transparent on both sides, and is thus indistinguishable from void space. Semipermeability is defined by the PERVASION pattern-equation, described below.

Thus far, features of a pure boundary algebra have been identified without placing an interpretation for conventional mathematics or logic on boundary forms. As is the case with any mathematics, an interpretation can be constrained to entry to and exit from the formalism. The mechanism of forms and pattern-equations can thus be used to compute over the structures of the interpretation.

---

[2] These same pattern matching options are provided in the mathematically comprehensive technical computing environment, *Mathematica*.

[3] The presentation in this paper is informal. Metatheory has been developed by Peirce [2], Spencer Brown [3], Kauffman [9], and Bricken [5]. Recent pioneers have legitimized the formal diagrammatic reasoning of AEG [10][11][12]; metatheory is invariant under provable equivalence. The equational structure of boundary logic supplies algebraic metatheory [13]; substitution and replacement are domain independent [14]. Void-equivalence is at the basis of boundary algebra, providing secure syntactic metatheory as a rewrite system.

3 Boundary Logic

Boundary logic is an interpretation as propositional logic of the abstract algebra of boundaries described above, using the following map:

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

The map from boundaries to logical connectives is *one-to-many*, implying that boundary logic is not isomorphic to propositional calculus or to Boolean algebra. This observation is also supported by comparing the basis constants of both systems; boundary logic has one while propositional logic has two. Transcribing the boundary void into a token (which is similar to representing the empty set as a set, {}) does establish isomorphism, however such a transcription undermines the fundamental mechanisms of boundary logic described below, those of void-equivalence and boundary transparency. Expressions that are structurally different in conventional logic may not be different in boundary logic. The following three transcriptions illustrate the condensation of logical structure into less boundary structure due to the one-to-many map:

Logic:	¬f	((f ∨ f) → f) ∨ f	A ∧ B = ¬(¬A ∨ ¬B)
Parens:	( )	( )	((A)(B)) = ((A)(B))

Boundary logic is an amalgam of Peirce's Entitative Graphs [2, 3.456-552] and Spencer Brown's Laws of Form. Entitative Graphs are a dual variety of Existential Graphs, for which forms occupying the Sheet of Assertion (i.e. the blank page) are joined in disjunction rather than in conjunction. Forms in LoF map directly onto Entitative Graphs, however the transformation system in LoF is equational rather than implicative as in AEG. The axiomatization of boundary logic improves upon that of LoF by using only rules based on *void-substitution* (deletion of structure). The two pattern-equations that define valid transformations in boundary logic are:

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

`OCCLUSION` identifies void-equivalent structure. The *curly brace* in `PERVASION` is a *meta-boundary* indicating any number of intervening boundaries, including none. Curly braces represent semipermeable boundaries, permitting any form, `A`, on the

outside to be arbitrarily anywhere on the inside, regardless of depth of nesting. Form B stands in place of the contents of the particular space containing the replicate of form A. PERVASION has no analogs in conventional mathematics. Two theorems that make the presentation of boundary proofs more succinct are:

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

These pattern-equations map onto Peirce's rules of transformation for Alpha Graphs, with the exception that OCCLUSION combines the AEG rules "Erase at even depths" and "Insert at odd depths" into a single equation that eliminates depth specific transformation rules. The curly braces of PERVASION render boundaries transparent to outer forms, capturing the iteration/deiteration rules of AEG. INVOLUTION maps directly onto the double cut rule of AEG. DOMINION is the termination condition for algebraic proof, and is derived from OCCLUSION by calling upon Leibniz' inference rule for equational logic (functional substitution).

To use boundary logic, propositional sentences are first transcribed into their boundary form, then the algebraic pattern-matching mechanism of boundary logic is used to reduce the transcribed form. After reduction, what remains of the form is transcribed back into a propositional sentence. The boundary pattern-equations above are not inferential logic and they are not equational logic. Consider the analogy of matrix logic algebra [15]. The sixteen binary Boolean relations can be transcribed into 2x2 binary matrices. Deduction then occurs via the rules and structures of matrix algebra, with logical constants corresponding to scalars, variables corresponding to 2-vectors, and connectives corresponding to 2x2 matrix operators. Intermediate vectors that occur during matrix evaluation represent imaginary logical values, making the expressibility of matrix logic inherently greater than that of Boolean logic. Similarly, boundary logic uses extra-logical mechanisms to achieve deduction. Matrix logic provides *more* mechanism than inference to achieve deduction, boundary logic provides *less* mechanism than inference to achieve the same deductive results.

The pattern-equations of boundary logic have remarkable properties:

- An an axiomatization of elementary logic, the two equations are more succinct than any string-based axiomatization. Boundary logic itself maps *one-to-many* onto conventional string notation, making it a formally simpler system than conventional logic.
- Reduction takes place by *void-substitution*. The right-side of each pattern-equation is the same as the left-side but for some structure removed. The missing structure on the right-side is *void-equivalent*.
- Curly-braces render all intervening boundaries *transparent* to outside forms. PERVASION identifies a relativistic void-equivalence: relative to an outside form, both intervening boundaries and inner replicates are void-equivalent.
- Boundary forms can be *geometrically and topologically transformed* to generate different varieties of syntax. Reduction in each syntactic variety still consists of deletion of specific structures.

4 Syntactic Variety

The remainder of the paper presents over two dozen notations for boundary logic, most of them new[4]. The notations are purely syntactic varieties derived from geometric and topological transformation of boundary forms. Seven families of syntax (parens, circles, distinction networks, steps, centered maps, distinction paths, and blocks) differ topologically, the rest are simpler geometric reconfigurations. Each family could potentially shed light on the sub-structure of logic and of cognition.

Textual forms are one-dimensional token strings. Propositional calculus, Boolean algebra, and parens notation are examples. *Enclosures* are two-dimensional, AEG is the primary example. *Graph* and *map* forms are three-dimensional, requiring the use of depth cues such as crossing graph links. Distinction networks and rectangle maps are examples. *Steps* and *rooms* are anthropomorphized maps. *Paths* are one-dimensional forms spread over a two-dimensional space and may be seen as temporal or spatial traversal; distinction paths is the primary example. *Blocks* are three-dimensional spatial forms that can be physically manipulated. Importantly, each variety of syntax is accompanied by a mechanism for valid diagrammatic reasoning.

4.1 Display Conventions

Each syntactic variety is illustrated by a single example, the binary exclusive-or, XOR. Exclusive-or can be decomposed into simpler connectives, thus illustrating other logical connectives. The alignment between decomposed XOR and its parens form is:

```
p XOR q = ¬(¬(p ∨ q) ∨ ( p ∧ q ))      Conventional logic
          ( (p   q)   ((p) (q)))      Parens notation
```

Syntactic varieties are loosely organized into topological families. The sequence of intermediate transformations that generate the new notations from prior notations is illustrated within each family. For one member of each family, the transcription of the boundary logic rules OCCLUSION, INVOLUTION and SHALLOW PERVASION into the new notation is shown, followed by a proof of *modus ponens* using the new notation. For all varieties, rearrangement after the application of a reduction step is not required, since each reduction step simply deletes structure. Some figures that follow have been rearranged slightly, solely to improve the aesthetics of the presentation. To reduce complexity and to improve readability, the deep boundary semipermeability indicated by curly braces has been simplified into a single semipermeable boundary[5]. The pattern-equation for SHALLOW PERVASION is:

```
A (A B) = A (B)              SHALLOW PERVASION
```

Several notational guides have been incorporated:

- The meta-token indicates absence of form, and is used solely as a visual convenience. It has absolutely no interaction with forms.
- Square brackets, [ ], are highlighted parens. They are identical to rounded parens in all other respects.

[4] Many of the notations were inspired by the work of, and conversations with, Louis Kauffman.

[5] SHALLOW PERVASION can be composed in steps to generate PERVASION. LoF uses SHALLOW PERVASION; AEG uses PERVASION; neither fully develop the idea of permeable boundaries.

• A large dot, • , identifies the shallowest space (the outside) of a form. For some notations, this point-of-reference is embedded within the form.

• Multiple forms SHARING the shallowest space are always explicitly bounded, so that forms in the outermost space are singular. Multiple forms in the shallowest space are enclosed using INVOLUTION:

```
A B C  =  ((A B C))        Packaging via INVOLUTION
```

4.2 String Varieties

For comparison, the pattern-equations of boundary logic are expressed below in parens notation and in conventional textual syntax:

Notation	OCCLUSION	INVOLUTION	SHALLOW PERVASION
Parens	(( ) A) = ><'	((A)) = A	A (A B) = A (B)
Implicative logic	(f→A)→f ⊨ f	(A→f)→f ⊨ A	((A→f)→B)→A ⊨ B→A
Boolean algebra	(t+A)' = f	A'' = A	A+(A+B)' = A+B'

In conventional logic, *modus ponens* is: $A \wedge (A \rightarrow B) \models B$

Modus ponens transcribed into parens notation is: `[ ((A) ( (A) B )) ] B = ( )`

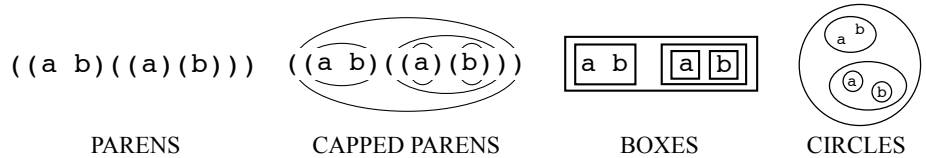
The double turnstile of logical implication becomes an assertion of equality in equational logic systems; both the Deductive Theorem and the steps of inference are absorbed into the match-and-substitute strategy of algebra. The directionality of implication is mitigated by collecting all forms on one side of the equation. The square bracket above highlights the transcribed logical implication. Conventional syntactic proof steps become a sequence of boundary logic void-substitution steps. Should the left-side reduce to mark, the asserted equality is logically valid.

In the following parens notation proof of *modus ponens*, a rule application on each line deletes void-equivalent structure. Since void-equivalent structure cannot impact values, the order in which structure is removed is irrelevant.

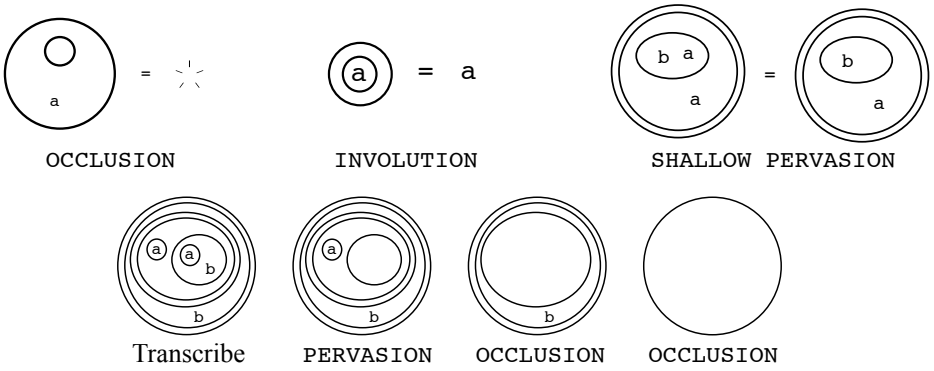
```
[ ((A) ( (A) B )) ] B = ( )      transcription
[ ((A) (       )) ] B = ( )      pervasion
[                  ] B = ( )      occlusion
[                  ]   = ( )      dominion, identity
```

4.3 Enclosure Varieties

AEG is almost always presented in the syntax of planar enclosures. The first two representations of XOR below show how parens delimiting tokens can be connected by *caps* to construct planar enclosures. The next two varieties geometrically modify the shape of the enclosures.

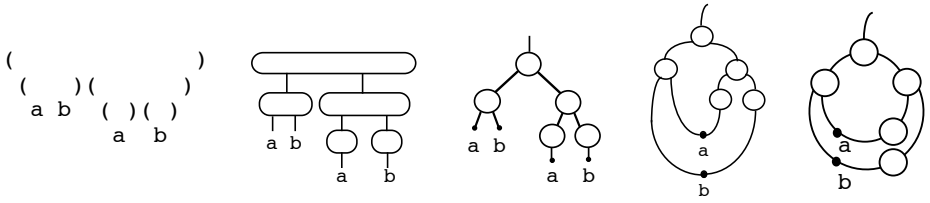


The three parens notation pattern-equations in the table above are textual abbreviations of boundary logic rules that are more naturally expressed in the diagrammatic syntax of enclosing circles below. A proof of *modus ponens* in circle notation then follows:



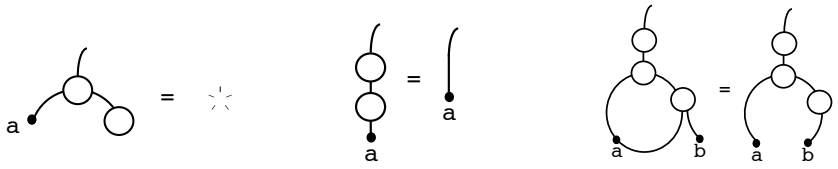
4.4 Graph Varieties

Extruding a parens form downward converts containment into graph connectivity. The first form below is an extruded parens form. In the second form, parens are capped to construct nodes. The third form shrinks capped parens to nodes of the same size, forming a boundary tree; nodes replace boundaries and directional links replace enclosure. The outside, or shallowest space, of the circle forms above becomes the top, or root, node of the tree. Replicates of variables are next joined together, constructing a directed acyclic graph that supports replication through multiple links rather than through multiple labels. The fifth form is a distinction network; the links have been aligned geometrically in circular patterns that emphasize graph sub-structures.



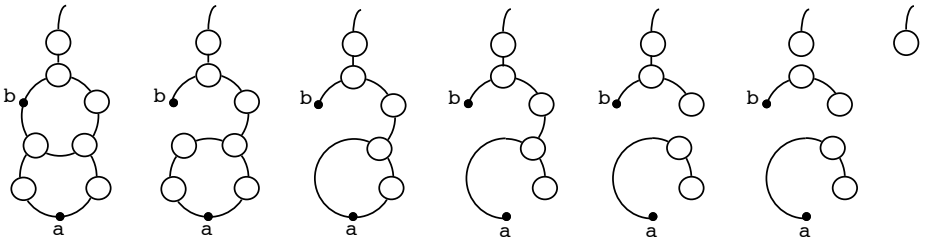
Two conventions identify the shallowest space of a graph. North-south orientation places the root node towards the top of the page, making it the "top" node. This convention is quite contextual, since it requires meta-information about orientation in space. An alternative technique is simply to extend a link out of the graph, locating the context explicitly. Each variable node then identifies the bottom of the graph, its deepest content. When reading graphs as processing networks, variables are inputs and the root node is output. Although trees are two-dimensional, graphs are three-dimensional, since graphs violate planarity whenever combining multiple occurrences of variables into single nodes requires connective links to cross.

The boundary logic transformation rules are expressed as distinction networks below. PERVASION is a path rule; when any two paths from a node join again at another node, the longer path can be detached from the lower node.



OCCLUSION INVOLUTION SHALLOW PERVASION

Each of the distinction network rules can be implemented by parallel asynchronous message-passing between nodes [6]. Global reduction occurs without global coordination. The distinction network proof of *modus ponens* displays characteristics of the localized message-passing regime. Reduction occurs primarily through deletion of links, with graph fragments that are not connected to the root node being treated as irrelevant. The implication for proof theory is that deduction is a parallel rather than a sequential process.

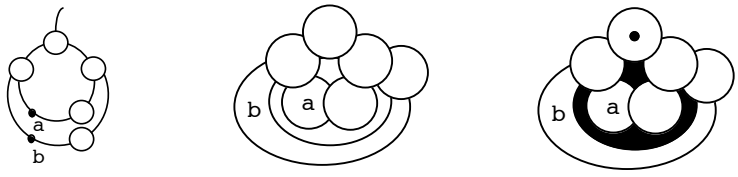


Transcribe PERVASION INVOLUTION PERVASION OCCLUSION OCCLUSION

4.5 Map Varieties

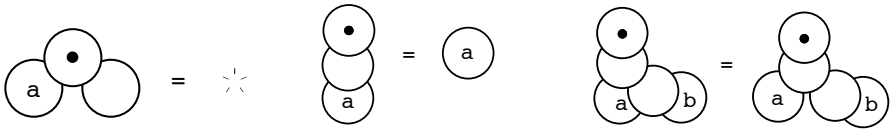
A distinction network can be converted into a map by enlarging each node until neighbors touch, constructing a common border in place of each link. Link connectivity becomes territorial contact. Unlike circle maps and rectangle maps, both steps maps and rooms maps are anthropomorphized.

Steps. The curvature of common borders in the steps map below incorporates visual cues in a third dimension that locate the top step, and show the relative depths of other steps. In the third form, background areas not part of the map but captured by the map are darkened to better contrast steps from background. A redundant point-of-reference dot is added to clearly identify the top step. Were the convex/concave depth cues to be lost by straightening the borders, the reference dot becomes the sole indicator of the top step.



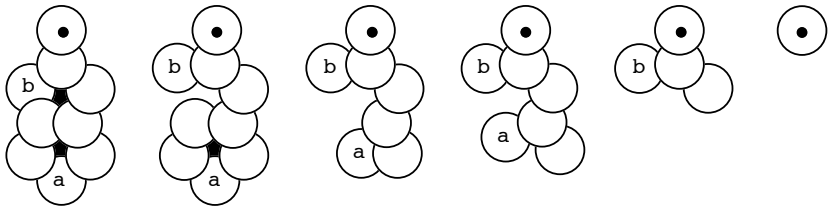
DISTINCTION NETWORK STEPS MAP ORIENTED STEPS

For map varieties, reduction deletes borders. The boundary logic reduction rules are expressed below as steps forms. SHALLOW PERVASION is represented by slipping the pervaded step out from under the lower step. In steps maps, PERVASION deletes a common border, rather than disconnecting a link, or erasing a form.



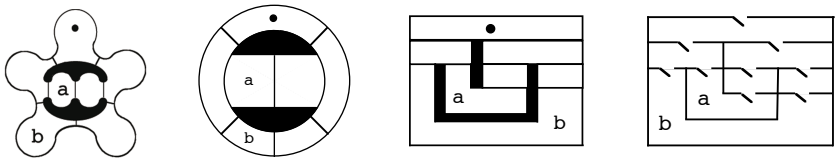
OCCLUSION INVOLUTION SHALLOW PERVASION

Although it is generally possible to construct a map representation of any form, it is not generally possible to maintain contiguity of territories in two dimensions. In particular, when a graph form requires crossing links, the associated map form requires a third, depth dimension. The proof of *modus ponens* for steps follows:



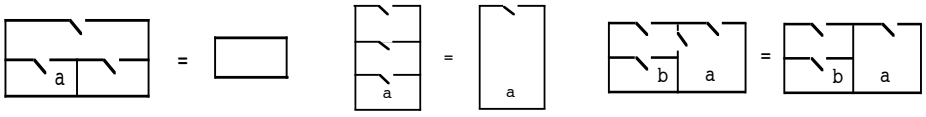
Transcribe PERVASION INVOLUTION PERVASION OCCLUSION OCC.

Below, the oriented steps map from above is geometrically rearranged in several ways. These map varieties offer visual rather than conceptual variation.



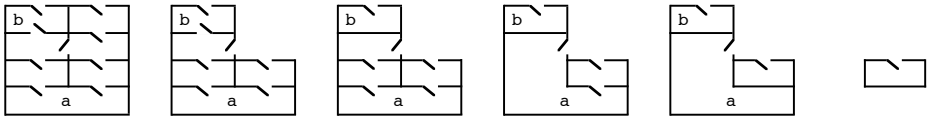
BLOBBY MAP CIRCLE MAP RECTANGLE MAP ROOMS

Rooms. Rather than enforcing separation of non-adjacent territories with captured space, the walls of rooms enforce non-adjacency, while open doors represent shared borders. PERVASION simply closes a door, while OCCLUSION closes off several rooms, creating an "empty room" by denying access. INVOLUTION deletes walls. A room map is TRUE whenever there is direct access to an empty room from the outside.



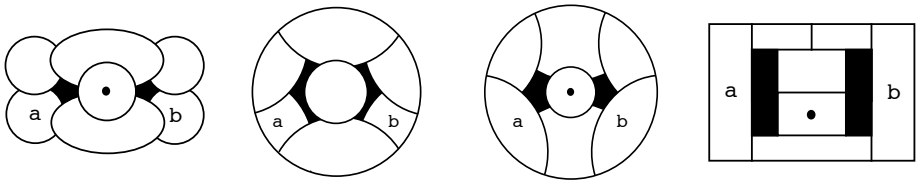
OCCLUSION INVOLUTION SHALLOW PERVASION

In the proof of *modus ponens* below, rooms could of course be extruded into a third dimension, constructing a visceral proof technique based on walking around. Door closures can be determined subjectively, from within the representation, without a global perspective, analogous to the asynchronous reduction of distinction networks. Subjective PERVASION might be expressed as: "When you are in a room, R, with two inwardly open doors, and one door leads to a room that allows you to return to R through the second door, then close the second door."



4.6 Centered Map Varieties

Centered maps put the reference point in an interior territory rather than a territory on the edge of the map. Centering often better displays symmetries. The first map below is a centered version of steps, the rest are geometric modifications of centered steps.



The two versions of the centered circle map above illustrate the use of semantic visual cues to convey depth of nesting. Centered circle map I is faithful to the three-dimensional overlap cues of centered steps. In the centered circle map II, the depth cues provided by overlap are contradictory, removing semantic interpretation. Overlap is completely abandoned in the centered rectangle map. These varieties illustrate that depth cues are not an essential component of logic maps; however, in the case of maps generated from non-planar graphs, some individual territories are not contiguous.

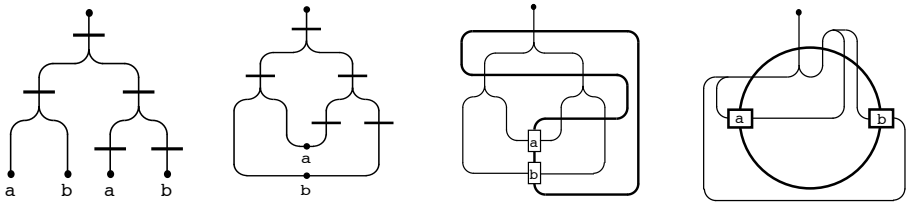
The topological transformation from oriented to centered maps calls upon an unusual type of spatial rearrangement. Compare the earlier oriented steps map to centered steps maps above. In the steps map, the external space (adjacent both to the top step and to step `b)` becomes a captured interior space of the centered steps map. The surrounding space commutes from outside to inside. This is accomplished by drawing the arc-like step `b` of the steps map around the root step rather than around step `a`. An accidental property of these logic maps permits two planar forms, one read from the outside and one read from the inside. Although the logical semantics does not change, the diagrammatic form permits two different cognitive perspectives that echo familiar distinctions between objective/subjective, extrinsic/intrinsic, absolute/relative, and global/local. Point-of-view is a theme throughout the syntactic varieties.

4.7 Path Varieties

In the path varieties, structural containment, connectivity and contact have been replaced by conveyance along a path. The model is one of spatial and temporal transversal rather than reading an objective structure.

The bar tree below appears to be identical to a parens tree (earlier above) but for the shape of the nodes. Bar graphs and distinction networks share the same resemblance. However, this solely visual modification identifies significantly different implementations: the node/link model of distinction networks changes into

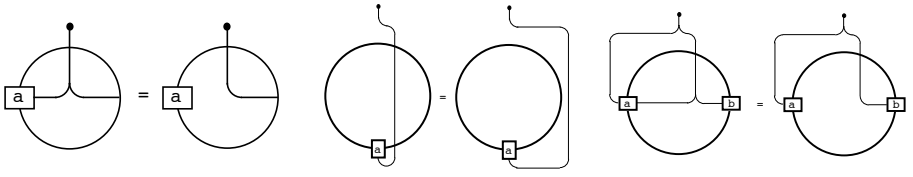
the barrier/path model of bar graphs. Distinction network nodes actively enact transformation processes, while links are simple communication channels between nodes. In a bar graph however, the barrier is a simple communication interface, while the path represents the dynamics of an evaluation of the form. Nodes in a distinction network connect and functionally transform convergent links, while the convergence of paths in a bar graph is independent of the bars. A semiconductor design analogy is that distinction networks represent the circuit schematic model with nodes as logic functions and directional links as input and output, while bar graphs represent the transistor-level model with bars as inverting transistors and paths implementing disjunctive logic through physically wired signal convergence (wire-OR).



BAR TREE BAR GRAPH PATH GRAPH DISTINCTION PATH

A bar graph converts into a path graph by connecting all bars by a single continuous loop[6]. The path graph is then modified geometrically so that the bar loop becomes the single circular boundary of a distinction path. Distinction paths convert connectivity into crossings and depth into number of crossings. All of the boundaries in the parens form collapse into the single distinction path boundary. `BOUNDING` by apparently distinct boundaries becomes crossing of a single boundary. Odd and even depths of nesting are replaced by the pure parity of locating a segment of the path on the inside or the outside of the distinction path boundary. `SHARING` of space by apparently distinct forms becomes divergence of traversal choices along the single path. Non-planar bar graphs require bridges (in a third dimension out of the plane of the paper) for paths to be able to cross over other paths.

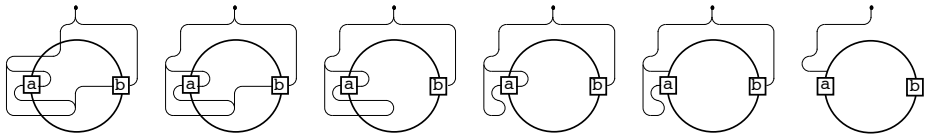
Rules for distinction paths involve shifting and deleting path segments. Paths that terminate at the distinction path boundary are ground states: termination on the inside is `FALSE` while termination on the outside is `TRUE`.



`OCCLUSION` `INVOLUTION` `SHALLOW PERVASION`

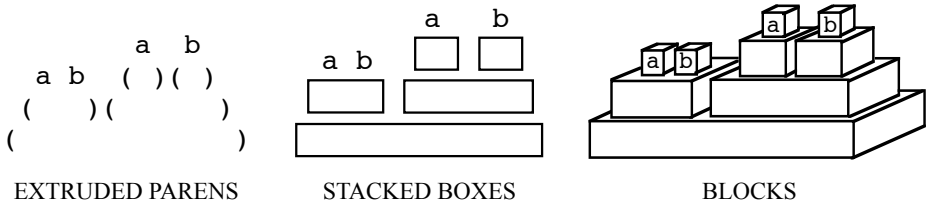
[6] Connecting bars into a single loop is an application of the Jordan curve theorem. The parens representation is first over-capped to convert each parens into a single bar. Then, rather than under-capping matching bar ends to construct the circle variety, under-caps are constructed iteratively from the deepest space to connect adjacent parens fragments: connecting `)(` in the case of `()()`, and `))` in the case of `(())`. This regime maps even depths to the outside of the curve and odd depths to the inside, with nesting depth counted by the number of transverse crossings of the Jordan curve [16, p 605ff].

A distinction path proof of *modus ponens* follows:

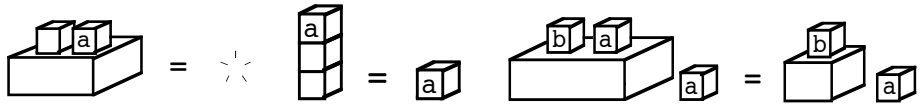


4.8 Block Varieties

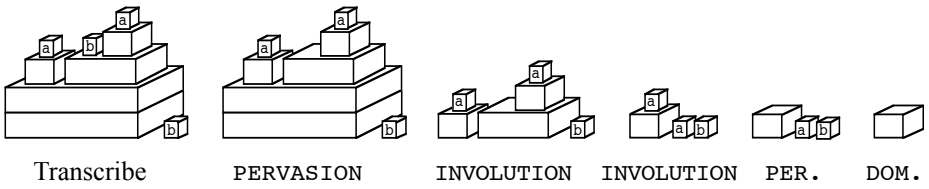
A parens form can be extruded upward and capped to construct stacks of boxes similar to parens trees, but with contact replacing connectivity. These boxes can then be extruded again into a third dimension, constructing a three-dimensional block representation that is physically manipulable. Containment becomes concrete contact. Block forms are not anthropomorphized, they are simply concrete structures in a physical environment.



Reduction rules identify configurations of stacks from which blocks can be removed.



The proof of *modus ponens* that follows can take place through physical manipulation of concrete objects standing in place of abstract logical forms.



Like Cuisenaire rods used to teach elementary numerical concepts to pre-schoolers through physical manipulation, logic blocks could be used to teach elementary logical concepts. The difficulty is that the concepts taught by logic blocks are those of diagrammatic boundary logic rather than the more familiar string-based logic embedded in language.

5 Conclusion

Algebraic formulation of Peirce's original Entitative Graphs provides a plethora of diagrammatic languages for logic. Diverse geometric and topological transformations of the spatial syntax result in several distinctly different two- and three-dimensional representations, all using the same three abstract pattern-equations to achieve form reduction. Underlying these syntactic varieties is a new set of mathematical concepts: void-equivalence, variary operators, boundary semipermeability, spatial pattern-equations.

That our familiar conversational logic and our formal typographic logic can both be rendered in a variety of structurally simpler diagrammatic and experiential representations raises interesting questions for cognitive science. How would a newly acquired ability to visualize or to physically manipulate logical form influence the quality of logical reasoning? More challenging, though, are the unfamiliar formal concepts that do not map onto conventional logic. The pattern of transformations that constitute reasoning in boundary logic is concise: a constructive proof can only proceed from mark to `DOMINION` to `PERVASION`, although the void-equivalent structure of `OCCLUSION` and the void-equivalent paired bounds of `INVOLUTION` can add syntactic complexity anywhere within a form, as illustrated below:

`( )` ⇄ `A ( )` ⇄ `A (A)` ⇄ `((A (A)))` ⇄ `((A (A))) (X (X))`

For valid forms, `PERVASION` asserts that context creates content. The familiar idea that the antecedent (content) validates the consequent (context) is reversed. More fundamentally, deduction proceeds by the deletion of irrelevant structure rather than by the accumulation of facts. Like any mathematics, boundary logic is a way of looking at problems, a way of thinking and seeing. As a technique, it may seem incomprehensible at first. After some practice, syntactic manipulation becomes second nature. But, like Venn diagrams, Existential Graphs and other diagrammatic formalisms, can the formal techniques of boundary logic claim to represent cognitive processes?

References

1. Kneale, W., Kneale, M. (1962) *The Development of Logic*. Oxford Univ Press.
2. Peirce, C.S. (1931-58) *Collected Papers of Charles Sanders Peirce*. Hartshorne, C. Weiss, P., Burks, A. (eds.) Harvard Univ Press.
3. Spencer Brown, G. (1969) *Laws of Form*. George Allen and Unwin.
4. Bricken, W. (2006) The Mathematics of Boundaries: A Beginning. Diagrams'06.
5. Bricken, W., Gullichsen, E. (1989) Introduction to Boundary Logic. *Future Computing Systems* 2:4 1-77.
6. Bricken, W. (1995) Distinction Networks. in Wachsmuth, I., Rollinger, C.R., Brauer, W. (eds.) *KI-95: Advances in Artificial Intelligence*. Springer 35-48.
7. James, J. Bricken, W. (1992) A Boundary Notation for Visual Mathematics, *1992 IEEE Workshop on Visual Languages*, Seattle, IEEE Press, 267-269.
8. Shin, S. (1994) *The Logical Status of Diagrams*. Cambridge Univ Press.
9. Kauffman, L.H. and Varela, F.J. (1980) Form Dynamics. *J. Soc. Biol. Structures* 3:171-206.
10. Barwise, J., Etchemendy, J. (1996) Heterogeneous Logic. In: Allwein, G, Barwise, J.(eds.) *Logical Reasoning with Diagrams*. Oxford Univ Press.
11. Shin, S. (2002) *The Iconic Logic of Peirce's Graphs*. MIT Press.
12. Hammer, E. (1995) *Logic and Visual Information*. CSLI Publications, Stanford.
13. Halmos, P. and Givant, S. (1998) *Logic as Algebra*. Mathematical Assoc. of America.
14. Birkoff, G. (1935) On the Structure of Abstract Algebras. *Proc. Cambridge Phil. Soc*, 31 417-429.
15. Stern, A. (1988) *Matrix Logic*. North-Holland/Elsevier.
16. Kauffman, L.H. (1993) *Knots and Physics* (2nd edition). World Scientific.