A BOUNDARY NOTATION FOR VISUAL MATHEMATICS

Jeffrey James and William Bricken
September 1992

Published in IEEE Visual Languages'92

ABSTRACT

Instead of traditional mathematical notation, we can describe formal mathematical systems in visual form. While traditional notation uses a linear sequence of symbols, visual mathematics uses boundary notation, which is comprised of objects and boundaries to enclose objects. Boundary notation is abstract, decoupling the underlying mathematics of a system from its visual representation. Once a system is defined in boundary notation, visual designs can be explored that optimize specific features. We demonstrate this approach with propositional logic and elementary algebra. Visual mathematics provides a robust foundation for visual languages much as linear mathematics provides a foundation for programming languages.

1. Introduction

Visual mathematics redefines mathematics spatially, so concepts such as numbers, variables, and expressions can be manipulated and understood visually. Expressions in visual mathematics are groups of objects with visible structure, where mathematically related components appear similar. Users manipulate these expressions not by the standard rules, but by creating and removing objects, by grouping and replacing parts of the expression, and by changing positions of objects and boundaries. Visual mathematics allows perceptual interaction rather than symbolic processing.

We present a notation for describing formal mathematical systems, called boundary notation. Boundary notation is comprised solely of labeled objects, spaces, and boundaries. Because boundary notation represents mathematics using only these spatial constructs, it allows interpretations of the underlying concepts that are completely visual [1]. This representation is visually interactive; an image completely describes an expression and computation occurs visually.

In the remainder of this paper, we will introduce boundary notation, its constructs, properties, and use. We then describe propositional logic and elementary algebra using boundary notation and provide visualizations of these systems.

2. Boundary Notation

The principles of boundary notation were first introduced by G. Spencer-Brown in his "calculus of indications" [2]. His system of mathematics has one fundamental concept, which is distinction. Distinction creates perspective by framing a space for observation, by forming the boundaries of objects, and by distinguishing objects out of an environment.

Boundary notation interprets Spencer-Brown's calculus of indications spatially. This spatial interpretation denotes a distinction as a boundary, which creates a space with content (inside) distinguished from its context (outside). The content of a bounded space has no ordering; the space is inherently commutative and associative. Sorting and rearrangement of items does not change the value of the space, provided items do not cross boundaries. A boundary distinguishes content from context; crossing the boundary changes that distinction. Boundaries and space are the fundamental constructs of boundary notation.

Boundary notation specifies a mathematical system with a set of equivalence rules. The system computes by matching a pattern from a rule and substituting an equivalent pattern. All boundary patterns include variables that will match any arbitrary configuration of objects and boundaries. This configuration must be confinable within a single space so that a boundary can encircle it, and when a boundary is matched, all of its contents must be matched also. An empty, or void, configuration can also be matched. Using boundary notation, we can describe the axioms of a mathematical system. The following applications are defined by such axioms.

3. Boundary Logic

Boundary logic is propositional logic interpreted using boundary notation. It uses two operators as a minimal basis for elementary logic: space provides the OR operation; boundaries provide the NOT operation.

a OR *b*	is	*a* *b*
NOT *a*	is	(*a*)

As space, OR implicitly has commutative and associative properties. Boundaries denote a space whose contents are logically inverted, the demarcation between what is and what is not. A space without contents, the void, represents the logical FALSE. Thus, a boundary around nothing represents NOT FALSE, which is TRUE. Other propositional connectives can be constructed from this basis:

IF *a* THEN *b*	is	(*a*) *b*

With the semantics of boundary logic understood, manipulating logic is simple. The axioms for boundary logic are shown in Figure 1 using parenthesis to draw boundaries [3]. These axioms were chosen for visual clarity; each requires only erasure to simplify.

Dominion	a ( )	<==> ( )
Pervasion	a (a b)	<==> a (b)
Involution	((a))	<==> a

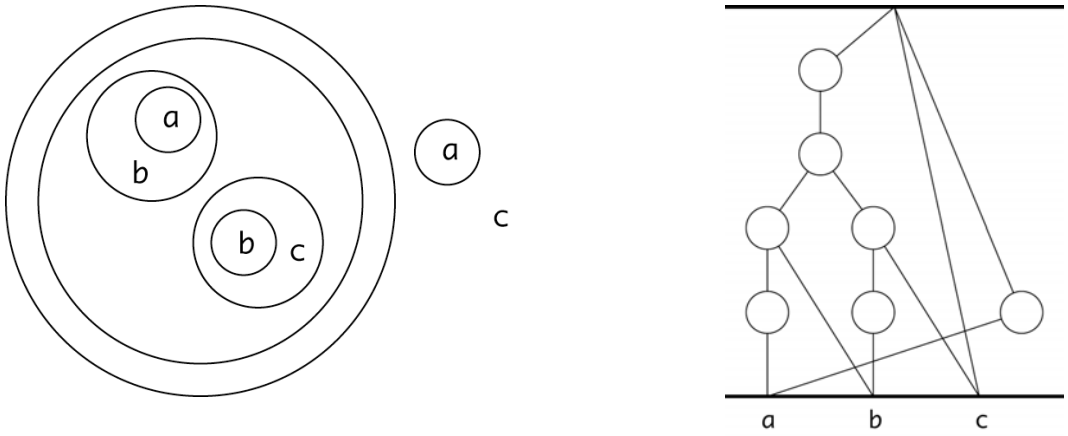
Figure 1. Axioms of boundary logic
a and b are arbitrary configurations

A proof in boundary logic is illustrated in Figure 2. Rules apply in parallel: steps 2 and 3 can occur simultaneously. Rule selection is arbitrary: step 3 could have been the involution of b, resulting in a different proof. Rules apply to arbitrary configurations: steps 3 and 4 can be combined into a single step where c and (b) match as a single configuration.

0.	((((a) b) ((b) c))) (a) c	Transcribe
1.	((a) b) ((b) c) (a) c	Involution of ((a) b) ((b) c)
2.	(b) ((b) c) (a) c	Pervasion of (a)
3.	(b) ((b)) (a) c	Pervasion of c
4.	(b) ( ) (a) c	Pervasion of (b)
5.	( )	Dominion of (b) (a) c

Figure 2. Proof of ((a->b AND b->c) -> (a->c)) in boundary logic

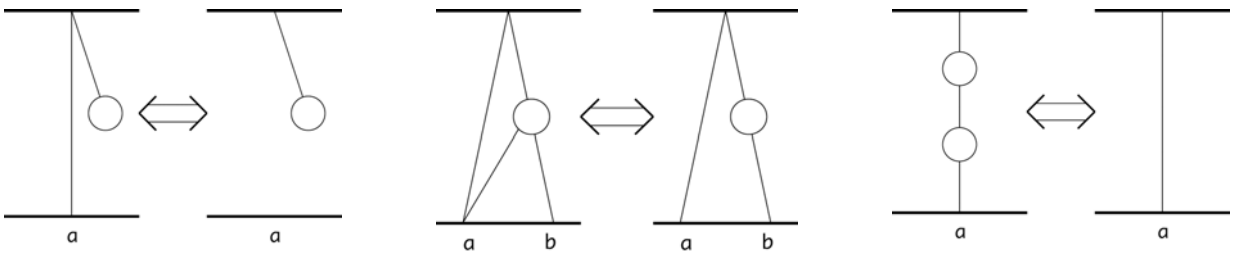
The spatial structure of boundary logic allows visual interpretations, including the two interpretations shown in Figure 3 [4]. When boundaries are drawn in two dimensions, they appear as encircling boundaries, appearing almost identical to the parenthesis notation. Objects are free to be rearranged within a space. Another representation, the distinction network, connects boundaries to contents as edges in a network. The network proceeds from the space of the entire expression at the top, to the variables at the bottom. The network form draws variables only once.



Encircling Boundaries *Distinction Network*

Figure 3. Visualizations of the boundary logic expression ((((a) b) ((b) c))) (a) c equivalent to ((a->b AND b->c) -> (a->c))

Axioms are animated differently for encircling boundaries than for distinction networks. The encircling boundaries fade in new patterns and fade out old ones, without a break in continuity. The distinction network, in contrast, is shown in Figure 4. When removing boundary nodes, the network connections meet and stretch into place like rubber bands. Other animations simply break connections. (See [4] for implementation details.)



Dominion *Pervasion* *Involution*

Figure 4. Visualization of boundary logic axioms

4. Boundary Algebra

Boundary algebra applies boundary notation to perform algebraic manipulation. Any notation that expresses elementary algebra requires more than a single distinction. For this purpose, we extended boundary notation by attaching descriptors to each boundary that define each distinction. The rest of boundary mathematics remains intact: space is still commutative and associative, and rules still apply in parallel.

The fundamental construct of boundary algebra for expressing numbers is the unit, expressed here by an asterisk, *. A unit is necessary to "add like things" and to perform multiplication. Fundamentally, space has the semantics of addition; configurations add by incorporating them in the same space. If they have the same unit, the result can be simplified. On the other hand, multiplication replaces the units of one expression with the entirety of the other.

We represent elementary algebra in boundary notation using four special purpose distinctions. The rules listed in Figure 5 define these distinctions. The first two rules accommodate numerical representation. Cardinality allows replacement of two identical configurations by a single configuration, distinguished as doubled. The inverse distinction indicates an inversion of the contents, which would cancel with a non-inverted copy when in the same space. Both of these distinctions can be distributed across their contents.

Cardinality	* *	<==>	Two[*]
Inverse	* Inverse[*]	<==>	
Lambda	{f[#] g[#]}[*]	<==>	f[*] g[*]
Composition	f[g[*]]	<==>	(f g)[*]
Distribution	f[a b]	<==>	f[a] f[b]

Figure 5. Rules of boundary algebra
a and b are arbitrary configurations
f and g are arbitrary distinction configurations

Two special-purpose distinctions allow the building of algebraic structure. The lambda distinction abstracts a common configuration out of an expression, replacing it with a place holder. The other special-purpose distinction

allows composition of distinctions by abstracting them into a separate space. Once in a separate space, distinctions themselves can be modified by boundary algebra rules, creating exponents. The distinction composition rule applies only to commutative and associative distinctions.

The special purpose distinctions above combine with additional distinctions representing algebraic unknowns. If we assume these distinctions represent quantities, then we can apply rules to them as such. Quantitative distinctions are commutative and associative, allowing composition. Quantitative distinctions can also be distributed across their content (the distribution rule). The examples in Figure 6 include a mixture of predefined and unknown distinctions.

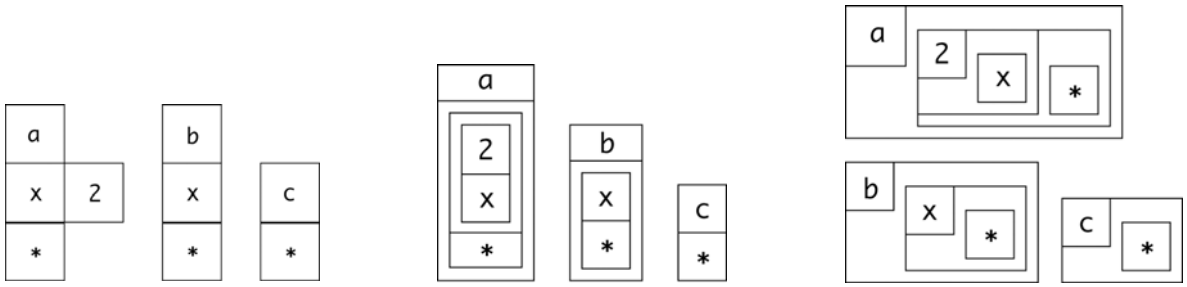


Figure 6. Visualizations of the boundary algebra expression a[Two[x][*]] b[x[*]] c[*] equivalent to ax2+bx+c

Figure 6 shows three visual interpretations of boundary algebra. Stacks use blocks for each object, and stack to specify distinction. The distinction spaces approach draws boundaries as pairs of spaces, one to describe the boundary, the other to specify its content. This representation is vertically dependent. The third interpretation attaches objects to boundaries, creating labeled boundaries.

These approaches provide visual interpretations of the boundary algebra rules. Figure 7 shows this visualization for labeled boundaries. Using just these visual rules, algebraic manipulation is possible; they are sufficiently powerful to visually derive the quadratic formula.

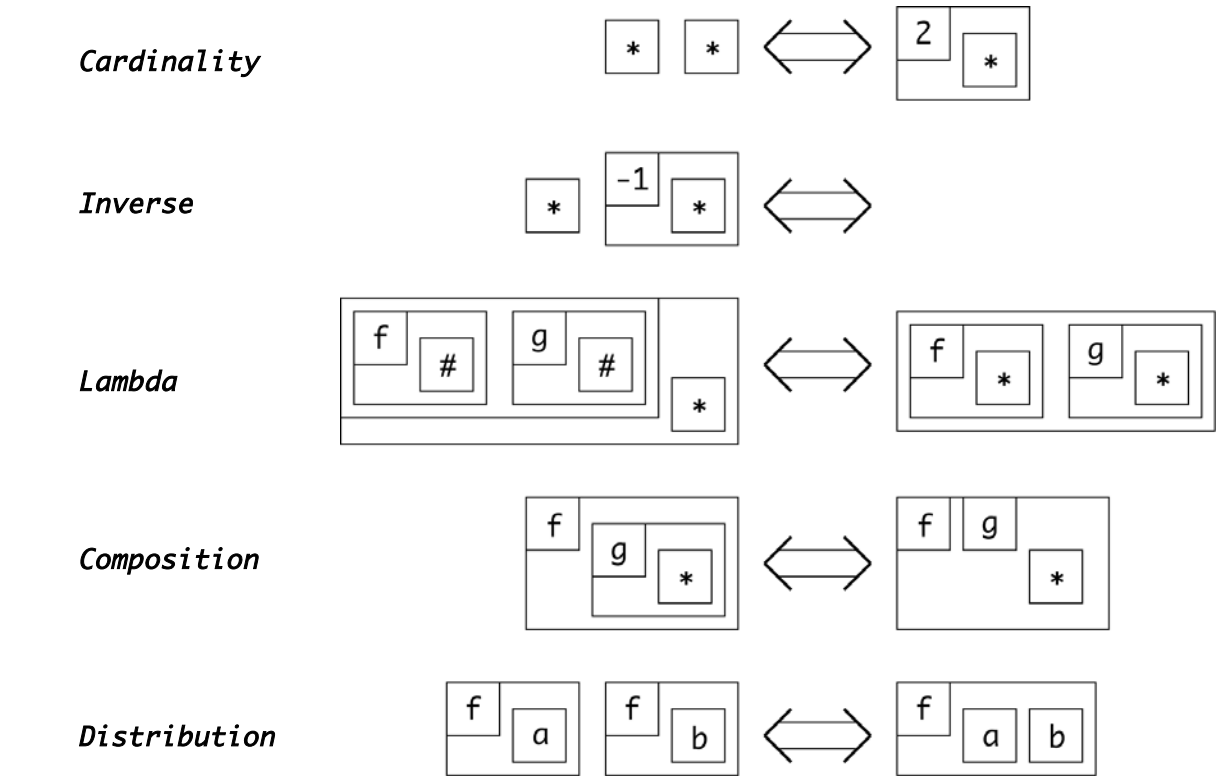


Figure 7. Visualization of boundary algebra axioms

5. Conclusion

Mathematics can be made visual with boundary notation. Boundary notation forms expressions using boundaries to separate spaces and distinguish objects. Computation occurs on these spatial structures rather than on typographical structures. The resulting mathematical interactions are visual and can be fully animated.

Future work in this area will focus on the implementation of boundary algebra. Our current work in boundary logic is complete, having shown the boundary rules to be axiomatic and having implemented software to perform and animate logic proofs [4]. The rules of the boundary algebra must be similarly refined and a software implementation built. This implementation will demonstrate the power and interactivity of visual mathematics using boundary notation.

REFERENCES

We thank Ann Miller, Meredith Bricken and Kimberly Osberg for their insightful comments on drafts of this paper.

[1] K. M. Kahn and V. A. Saraswat. Complete Visualizations of Concurrent Programs and their Executions. In Proceedings of the IEEE Workshop on Visual Languages, pages 7-15, 1990.

[2] G. Spencer-Brown. Laws of Form. Bantam: New York, 1969.

[3] W. Bricken. A Deductive Mathematics for Efficient Reasoning. Human Interface Technology Laboratory, Technical Report No. HITL-R-86-2, 1986.

[4] W. Bricken. An Introduction to Boundary Logic with the Losp Deductive Engine. Human Interface Technology Laboratory, Technical Report No. HITL-R-89-1, 1989.