Boundary Logic:
A Massively Parallel Model of Deduction

William Bricken

April 1989

Summary: Boundary logic, a new formalism based on diagrams rather than tokens, provides a model of deductive reasoning that is elegant, efficient, inherently parallel, and simpler than elementary logic.

Context: The mathematical foundation of the study of intelligence is *Boolean algebra*. Research in cognitive and developmental psychology (Piaget, Bruner, Suppes, Newell and Simon, Johnson-Laird) assumes that elementary logic (an interpretation of Boolean algebra) forms a minimal basis upon which understanding is built. Most computational models of cognition utilize the control structures of elementary logic. By eliminating the linear notational constraints of string representation, we develop a formalism that is foundational to Boolean algebra.

Mathematical expression in general, and elementary logic in particular, rests upon a notational convention explicitly introduced by Boole. Discrete ideas can be represented by *tokens*. This formal innovation was coupled with the premise that operators are also represented as tokens (signs of operation). In contrast, Venn showed that the elementary concepts of Boolean algebra can be represented by *diagrams*. However, the traditional techniques of formal systems have evolved as the manipulation of *typographical strings*, resulting in mathematical proofs and analyses (formal languages, complexity theory), computational implementations (sequential processing, Turing machines) and cognitive theories (natural deduction, symbol processing hypothesis) that are *string-based*.

The advent of graph theory, network models, and parallel processors has introduced models that are based in diagrams rather than strings. By axiomatizing transformations on diagrams, a *formal diagrammatic system* is constructed. Computation can then be implemented as manipulation of diagrams rather than strings. Such a formalism relies on the creation, destruction, and rearrangement of links and nodes (boundaries) rather than tokens. Of course, pointer arrays (LISP cons-cells) do this at the machine level, but what has been missing is an algebra of boundaries.

Distinction: In *Laws of Form*, Spencer-Brown provides a formal diagrammatic system that interprets elementary logic as configurations of boundaries. We have implemented several boundary-based systems, with startling gains in representational and computational efficiency. *Boundary mathematics* provides an innovative theory of rep-

resentation which permits efficient parallel processing and which in turn suggests a radical new theory of cognitive modelling.

Boundary logic places the meaning of logical tokens on boundary operators. Boundaries can be represented typographically by delimiting symbols (parentheses, brackets, braces). Interestingly, only one boundary symbol is needed to embody all the constants and operators of Boolean algebra (elementary logic). Let us call that boundary the *parens*, `( )`. The parens provides a representational facility not available with tokens: it indirectly refers to its inside, its contents. When the contents are empty, non-representation can be assigned a meaning.

A difficulty of parens notation is that it still enforces some characteristics of linear notation. In particular, typography requires grouping and ordering rules, and forces duplication of tokens to achieve multiple reference. Infix notation encourages the conceptualization of operators as binary. Boundaries, in constrast, provide an unlimited capacity for arguments. LISP, for example, extends AND to multiple arguments by enclosing the argument list in a delimiting set of parentheses.

The perspective of boundary logic is that operators are inherently variary (accommodating any number of arguments, including none). Linearization is specified as a constraint (non-commutativity, for instance) rather than as an axiom. The binary interpretation of AND and the consequent necessity for associative and commutative rules is seen an artifact of linear notations. Parens, then, acts like the curly braces of set theory, it contains sets of arguments. The difference between parens and curly braces is that the parens is an *operator* as well as a container. An empty parens is an operator without arguments (a constant). In boundary notation, ground objects are represented only once, by a single unique node. Multiple reference is by contact (shared boundary, multiple pointers).

A consequence of the non-linear features of boundary logic is that computational parallelism is easily achieved. Our implementation demonstrates massively parallel deduction for Boolean expressions. Parens are represented by *distinction nodes* in a network; nesting and concatenation of parens are represented by links. Each distinction node is a processor in a fine-grained architecture. The sparsely connected network is a distributed representation of the input expression. To achieve deduction, each distinction node examines its local connectivity, and initiates actions via message passing. The global minimum (logical solution) emerges out of local reduction of network structure. Message passing is imperative, asynchronous, and fully local; global information, locks, semaphores and blocking are unnecessary.

Content: A minimal token-basis for logic is FALSE and IF. Thus one possible map from token logic to boundary logic is as follows. Let FALSE be not represented. Non-representation, emptiness, is everywhere within an expression, it is literally the page under the symbols, and is the empty space bounded by a parens. Let IF be represented by the boundary delineated by the parens. The contents (inside) of a parens imply its context (outside). The map:

This map is many-to-one; parens notation condenses linear notation by rendering redundancies irrelevant. The contents of a parens are not ordered, thus they can be read into linear notation in any order.

Transformation rules for parens configurations are also surprisingly efficient. The axioms of boundary logic follow:

Logic	**Parens**	**Readings**
FALSE	*nothing*	
TRUE	`( )`	(IF FALSE FALSE), (NOT FALSE)
(NOT A)	`(A)`	(IF A FALSE)
(IF A B)	`(A) B`	((NOT A) OR B)
(A OR B)	`((A))B`	(IF (NOT A) B), (IF (IF A FALSE) B)
(A AND B)	`((A)(B))`	(NOT (IF A (NOT B))), (IF (IF A (IF B FALSE)) FALSE)

The algebra which is axiomatized by the above transformation rules is homomorphic to Boolean algebra. (Published reviews of Spencer-Brown's work have established isomorphism by filling the non-represented space with a token. This is a misunderstanding of boundary notation which degrades the elegance of the formalism.) The axioms provide Boolean minimization solely by erasure and creation of structure. Rearrangement, which is prevalent in token-based systems, is unnecessary. The equational (algebraic) formalism of boundary logic permits reversable deductive steps, as opposed to the unidirectional nature of implicational systems.

Dominion:	`( ) A = ( )`
Involution:	`((A)) = A`
Pervasion:	`(A) A = ( ) A`

Boundary logic as a cognitive processing model provides these features:

- efficiency and elegance of representation (reduction of storage constraints for cognitive models),
- non-representation (knowledge without storage),
- expressability equivalent to that of Boolean algebra (rational behavior),
- transformation by erasure and creation (reasoning without copying and storage overhead, Piagetian constructivism),
- efficiency and elegance of transformation and deduction (reduction of computational constraints),
- massive asynchronous parallelism (suggestive of physiological architectures), and
- an innovative formalism that is virtually unexplored (a potential breakthrough).

Boundary mathematics provides techniques and models wider than logic, suggesting a body of fascinating questions about the roles of distinction, labelling, erasure, deduction, topology, and models of thought in our culture.