Methodology

Formal Definition of the Calculus

The Primitive: Distinction

The calculus of indications begins with a single primitive: the act of **distinction**. To distinguish is to create a boundary that separates two regions—an inside and an outside. This act is represented by the **mark** or **cross**:

$$\langle\ \rangle$$

{#eq:mark}

The mark creates a bounded region. Content placed inside the mark is **contained** within the boundary; content outside is in the **void**.

Definition 1: Form

A **form** is defined recursively:

1. The **void** (empty space) is a form
2. The **mark** $\langle\ \rangle$ is a form
3. If a is a form, then $\langle a \rangle$ (enclosure of a) is a form
4. If a and b are forms, then ab (juxtaposition of a and b) is a form

The Two Axioms

The entire calculus derives from two axioms:

Axiom J1: Calling (Involution)

$$\langle\langle a\rangle\rangle = a$$

{#eq:calling}

Interpretation: Crossing a boundary twice returns to the original state. This is the spatial analog of double negation: NOT(NOT a) $= a$.

Proof sketch: Consider being inside a region bounded by $\langle a\rangle$. The inner boundary places you "outside of a" relative to a. The outer boundary then places you "inside" relative to being "outside of a"—returning you to a.

Axiom J2: Crossing (Condensation)

$$\langle\ \rangle\langle\ \rangle = \langle\ \rangle$$

{#eq:crossing}

Reduction Algorithm

Definition 3: Canonical Form

A form is in **canonical form** if no reduction rule can be applied. The only canonical forms are: - The void $\emptyset$ - The mark $\langle\ \rangle$

Reduction Rules

The reduction engine applies rules in the following priority:

1. **Calling Reduction**: If a form matches $\langle\langle a\rangle\rangle$ where a has exactly one enclosed child, reduce to a
2. **Crossing Reduction**: If a form contains multiple simple marks $\langle\ \rangle$ in juxtaposition, condense to single mark
3. **Void Elimination**: Remove void elements from juxtaposition (void is the identity for AND)
4. **Recursive Application**: Apply rules to nested subforms

Algorithm: Reduce to Canonical Form

```
function REDUCE(form):
    while REDUCIBLE(form):
        if CALLING_PATTERN(form):
            form ← APPLY_CALLING(form)
```

Boolean Algebra Correspondence

The Isomorphism

Boundary logic is isomorphic to Boolean algebra:

Boundary Logic	Boolean Algebra	Propositional Logic
$\langle\ \rangle$ (mark)	TRUE (1)	T
void (empty)	FALSE (0)	F
$\langle a \rangle$	NOT a	$\neg a$
ab	a AND b	$a \wedge b$
$\langle\langle a\rangle\langle b\rangle\rangle$	a OR b	$a \vee b$
$\langle a\langle b\rangle\rangle$	$a \rightarrow b$	$a \rightarrow b$

Derivation of OR

The De Morgan form for disjunction:

$$a \vee b = \neg(\neg a \wedge \neg b) = \langle\langle a\rangle\langle b\rangle\rangle$$

{#eq:or}

Derived Theorems (Consequences)

Spencer-Brown derives nine consequences (C1-C9) from the axioms. We verify each computationally:

C1: Position

$$\langle\langle a\rangle b\rangle a = a$$

C2: Transposition

$$\langle\langle a\rangle\langle b\rangle\rangle c = \langle ac\rangle\langle bc\rangle$$

C3: Generation (Excluded Middle)

$$\langle\langle a\rangle a\rangle = \langle\ \rangle$$

This corresponds to $a \vee \neg a =$ TRUE.

C4: Integration

Evaluation Semantics

Definition 4: Truth Value

The truth value f of a form f:

$$\text{void} = \text{FALSE}$$

$$\langle\ \rangle = \text{TRUE}$$

$$\langle a \rangle = \neg a$$

$$ab = a \wedge b$$

{#eq:semantics}

Theorem 3: Soundness

Claim: Equivalent forms evaluate to the same truth value.
Proof: The axioms preserve truth value: - J1: $\langle\langle a \rangle\rangle = \neg\neg a = a$
- J2: $\langle\ \rangle\langle\ \rangle = \text{TRUE} \wedge \text{TRUE} = \text{TRUE} = \langle\ \rangle$

Implementation

The computational framework implements:

1. **Form Construction**: `Form` class with void, mark, enclosure, juxtaposition
2. **Reduction Engine**: `ReductionEngine` with step-by-step traces
3. **Evaluation**: `FormEvaluator` for truth value extraction
4. **Theorem Verification**: `Theorem` class with automatic proof checking
5. **Visualization**: Nested boundary diagrams for forms

All implementations achieve test coverage exceeding 70% with real data verification (no mock testing).