Design and Implementation of a Boundary Arithmetic Calculator

William Bricken
Lake Washington Technical College and The Boundary Institute
william@wbricken.com

Abstract

Elementary mathematics education teaches both hands-on manipulation and the familiar symbolic abstraction. Concrete, human-centric addition is characterized by the Additive Principle: a sum is represented directly by its parts. Boundary arithmetic is a formal system that combines additive (rather than symbolic) transformations with an efficient representation of number. We implemented a calculator that displays boundary arithmetic computation as a tool to facilitate comparative axiomatic study of math learning. The calculator has a standard keypad and includes four modes for numeric base (unary, binary, decimal units, decimal digits) and three spatial visualization modes (1D, 2D, 3D). Formal system design decisions focus on the consistency, transparency, and expressibility of a notation for arithmetic as a physical activity. Calculator interface design decisions focus on rigorous adherence to transformation axioms and on the visual and interactive animation of the operations of arithmetic.

1. Introduction

We are currently constructing a tool to support study of the ways children are taught to think with mathematics and about mathematics. The intent is to explore patterns of errors associated with different axioms and axiom systems that define elementary arithmetic. The Rules of Algebra, for example, are a collection of axioms specified by group theory (the familiar concepts of commutativity, associativity, zero, inverse, and arity) that pervade elementary and secondary mathematics education. But prior to the introduction of symbolic forms, preschool mathematics emphasizes interactive manipulation, embodiment of concepts rather than abstraction of concepts. The tension between these two approaches "...is a fundamental and unavoidable challenge for school mathematics" [1].

Research in mathematics education recognizes the necessity of multiple modes of representation and multiple theoretic perspectives, and places mathematics learning in a pluralistic human context [2]. In contrast, the formal agenda of mathematics, called Hilbert's Program, includes removal of gross intuition, of psychological necessity, of physical interaction, and of concrete manipulation from the operations of mathematics [3]. "... numbers have neither substance, nor meaning, nor qualities. They are nothing but marks..." [4].

The symbolic model of arithmetic trades the visual and physical intuition that arises from direct experience for memorization of the rules of manipulation of structured strings of abstract tokens, explicitly divorcing representation from meaning in order to protect rigor. The goals of advanced mathematics do not necessarily align with the needs of novice learners nor with the objectives of mathematics education [5]. For example, "...despite the obvious importance of visual images in human cognitive activities, visual representation remains a second-class citizen in both the theory and practice of mathematics" [6]. Kaput [7] is directly critical of the emphasis of form over content, and attempts to steer mathematics education toward representational diversity. The advent of computer graphics and web-based virtual manipulatives [8,9,10] has reinforced visual and manipulative techniques at all levels of math education, but arithmetic itself is still characterized by a single symbolic theory, the Properties of Arithmetic, to the exclusion of other conceptualizations of number.

2. Comparative Axiomatics

Comparative axiomatics recognizes that the same mathematical concepts can be defined by a diversity of structural and transformational axioms, or rules. The choice of formal systems is based on pragmatic considerations. For example, integer computation can proceed in base-1, tallying marks in one-to-one correspondence with objects; in base-2, useful for computation within computers; in base-10, consistent with the current cultural emphasis on decimal notation; and even in mixed bases of 60, 12, 7, 28, and 365, reflecting models of time that have accrued over millennia. The underlying mathematical ideas of addition and multiplication of integers remain the same, but the specific algorithms, implementation strategies, and cognitive models vary considerably.

The history of formal logic, as another example, traces humanity's struggle to define rational thought, progressing through syllogistic reasoning, scholasticism, Boolean algebra, truth tables, natural deduction, resolution, and boundary logic [11,12,13,14]. Although each of these systems describes the structure of elementary logic, each provides a distinctively different strategy about how to implement it, both computationally and cognitively. The differences are more than notation, they are also instrumental in defining what rationality means.

A pivotal issue that can be addressed by comparative axiomatics is the respective roles of manipulative and symbolic models of arithmetic. In particular, is the separation of representation (syntax) from meaning (semantics) a desirable goal for math education?

In the following, we review the structure of natural numbers, with emphasis on the pragmatic necessities of notations that support computation and comprehension. We then describe a formal structure for additive systems, those original forms of arithmetic that conform to direct rather than symbolic manipulation [15]. We provide formal axioms, and a depth-value notation that exhibits the efficiency of place-value numerals while behaving additively rather than symbolically. This formal system is called *boundary arithmetic*.

We then describe the implementation of a boundary arithmetic calculator, a tool that displays arithmetic computation based on additive transformations. The primary design goal for the calculator is to rigorously convey the rules of additive arithmetic, to visually model computation as direct manipulation. A future goal is to show side-by-side comparison of transformations required by different axiom systems on the same problems, in order to identify through protocol analysis precisely which axiomatic structures are associated with errors of understanding by students. From this, we may be able to select representations for concepts and strategies for operations that have the pragmatic value of being friendly to human understanding. An immediate objective is to demonstrate that physical manipulation has an axiomatic foundation, that arithmetic can be formally defined by concrete action. The ultimate objective is to explore whether or not the reintroduction of elements of human learning, human psychology, and human physiology into the structures and processes of mathematics is a good idea.

3. The Structure of Natural Numbers

The earliest arithmetic was unit-based, sometimes called tally or *stroke arithmetic* [16,17].

```
12:          ////////////
```

Strokes map one-to-one onto a collection of objects, such as sheep or bananas or buckets of water. Stroke arithmetic is unary (base-1), obviating the need for different tokens, for management of token position, and for collection of groups. Addition in unary arithmetic is particularly simple: to add two stroke-numerals, put them together in a shared space:

```
5 + 7 = 12:    /////  ///////  ⇒  ////////////
```

Stroke-numerals however are difficult to read. It is necessary to count the strokes that are added together in order to read the value of their sum.

The Egyptians and Babylonians introduced names for collections of a specific cardinality. Roman numerals are a familiar example of a grouping system. For example, five strokes are structurally converted into the shape "V".

```
5:          /////  ⇒  V
```

The problem of readability is addressed by introducing iconic tokens for larger groups, such as `M` for `1000`. Like stroke-numerals, Roman numerals add by being placed together in a shared space. Ordering tokens by cardinality is convenient but not necessary:

```
521 + 235:   DXXI   CCXXXV  ⇒  DCCXXXXXVI
```

Specific "Roman number facts", such as `XXXXX = L`, contribute to the ease of reading at the cost of increasing cognitive load:

```
756:          D CC XXXXX V I  ⇒  DCCLVI
```

The Indo-Arabic decimal system incorporates a common rather than a mixed base. Individual number-tokens are needed only for the digits `0` through `9`. Place-value notation relies upon increasing multiples of the base 10 to express larger integers as polynomials. Places implicitly represent powers.

$$2756: \quad (2 \times 10^3) + (7 \times 10^2) + (5 \times 10^1) + (6 \times 10^0)$$

Indo-Arabic numerals exchange a great gain in readability for moderate losses in both computability and comprehension. Numerals no longer add by direct combination, instead adding requires both memorization of number facts (such as 4 + 5 = 9) and tracking the place-value of each digit. Digits are maintained in a strictly sequential position (imitating spoken language); calculation then includes techniques for interfacing adjacent positions, commonly called "carrying" and "borrowing". It takes several years of schooling for young students to master this representation.

Boundary numerals use depth-value notation to enhance readability. The techniques were first published for base-2 in 1995 by Louis H. Kauffman [18]. A boundary numeral replaces place-values by successively nested enclosures:

```
2756:          (((2) 7) 5) 6
```

Each boundary is read as multiplying its contents by the base of the system. Depth-value is a maximally factored form, in contrast to the polynomial representation of place-values:

$$2756: \quad 10 \times (\ 10 \times (\ 10 \times (\ 2\ ) + 7\ ) + 5\ ) + 6$$

In this notation, shared space is implicitly additive, while crossing boundaries is implicitly multiplicative. Addition is achieved by placing two or more numerals into the same space and then merging boundaries at the same depths.

Each type of number system represents a different perspective on what arithmetic means and how it works. Each affords particular errors in computation and each enforces particular limits on cognition.

4. Principles of Unit Ensemble Arithmetic

A *unit* is a mark, stroke, notch, pebble, shell, or other discrete singular distinction. Replicate units are intended to be indistinguishable in order to reduce the idea of cardinality to its foundation of one-to-one correspondence between units and objects. An *integer* is an ensemble of identical units. There is no zero unit. Unit ensembles support the Additive Principle: the ensembles contributing to a sum together represent the sum. Unit ensembles, and additive systems in general, can be either concrete or abstract. For example, a pocket full of pennies is concrete, while making a mark on a piece of paper for each penny is abstract. Both pennies and marks act additively.

Additive computation is achieved by relocation of ensembles, or it can be achieved simply by cognitive refocusing of perspective. In either case, addition is the consequence of the removal of spatial partitions, whether they be explicit or implicit. Residuals of additive notation are still present in today's arithmetic. Mixed numbers, for example, place whole numbers and fractions in the same space, implicitly embedding addition into the shared space. 1 3/4 means 1 + 3/4.

Unary additive systems have a substantively different axiomatic foundation than group-theoretic symbolic systems. Addition in a group-theoretic system is represented by symbolic rules that map ordered pairs onto single objects, together with rules that permit ordering and grouping to be altered. The zero place-holder that supports place-value notation is also the group-theoretic additive identity. Proof is supported by induction over successive integers. Within the additive concept of units sharing the same space, there are no notions of ordered pairs (relational structure), grouping (associativity), ordering (commutativity), arity (specific cardinality of operator arguments), or zero (identity for addition). Proof is by induction over ensembles. Unit ensembles also differ significantly from sets: there is no empty ensemble, identical individual units cannot be differentiated by labeling or by indexing, there is no distinction between a single unit and an ensemble of one, and no unit participates in more than one ensemble. As well as being grounded in experience, additive systems are also conceptually simpler.

Teacher training texts recognize the importance of additive systems throughout lower elementary mathematics. However, these texts explain the meaning of addition in terms of symbol manipulation, not in terms of the spatial intuitions of the Additive Principle. For example, commutativity of unit addition is achieved by fiat: "We may associate 3 + 5 with putting a set of 3 members in a dish, and then putting a set of 5 members in a dish to form the union of the sets. We associate 5 + 3 with putting the 5 set in a dish and then putting in the 3 set." [19] The unit ensemble definition of addition, in contrast, does not impose spatial or temporal ordering on actions. Children can add several unit ensembles by pushing them together all at the same time.

Unit ensemble arithmetic has two distinct drawbacks: mathematically, its formal structure seems not well understood, and pragmatically, it is very inconvenient to read the cardinality of large ensembles.

5. A Formal Model of Additive Arithmetic

The formal structure of unit ensemble arithmetic incorporates mereological *fusion* [20] as addition, and *substitution* based on one-to-one correspondence as multiplication. Table 1 shows a symbolic foundation for unit ensemble arithmetic.

Mereology is a formal theory of parts and wholes that does not include set theory. In general, an *ensemble* is a mereological whole with parts that do not overlap.

Table 1. A formal structure for unit ensemble arithmetic.

<table>
<tr><td colspan="2">Language</td></tr>
<tr><td>● is an ensemble</td><td>(interpreted as 1)</td></tr>
<tr><td>◊ is an ensemble</td><td>(interpreted as −1)</td></tr>
<tr><td colspan="2">if A and B are ensembles, so is A B</td></tr>
<tr><td colspan="2">Notation</td></tr>
<tr><td>A|B|C</td><td>ensembles in separate spaces</td></tr>
<tr><td>{A|B|C}</td><td>apply fusion</td></tr>
<tr><td>[A C E]</td><td>apply substitution</td></tr>
<tr><td colspan="2">Operations</td></tr>
<tr><td>{A|B|C} = A B C</td><td>(interpreted as addition)</td></tr>
<tr><td>[A ● E]</td><td>(interpreted as multiplication)</td></tr>
<tr><td>[● C E]</td><td>(interpreted as division)</td></tr>
<tr><td colspan="2">Axioms</td></tr>
<tr><td>● ◊ =</td><td>Cancellation</td></tr>
<tr><td>[A C E] = [E C A]</td><td>Symmetric Substitution</td></tr>
<tr><td>[A|B C E] = [A C E]|[B C E]</td><td>Distribution</td></tr>
<tr><td colspan="2">Polarity</td></tr>
<tr><td>Δ● = ◊</td><td>Change-polarity</td></tr>
<tr><td>Δ◊ = ●</td><td>Change-polarity</td></tr>
<tr><td>[◊ ● ◊] = ●</td><td>(interpreted as −1 x −1 = 1)</td></tr>
<tr><td>ΔA = [◊ ● A] and [● ◊ A]</td><td>(interpreted as −1 x A)</td></tr>
<tr><td colspan="2">Induction</td></tr>
<tr><td>A = B iff ΔA B =</td><td>1-to-1 correspondence</td></tr>
<tr><td>A = B iff A C = B C</td><td>uniqueness</td></tr>
<tr><td>A = ● xor A = ◊ xor A = B|C</td><td>decomposition</td></tr>
</table>

Different ensembles in separate spaces model different integers. Different spaces are represented by bars:

A|B|C|D|E ensembles in different spaces

Fusion is the deletion of partitions between ensembles. It is a variary, flat operation that applies to an arbitrary number of ensembles concurrently and does not support nested application. The fusion instruction is represented by placing partitioned ensembles in curly braces. The result of fusion is represented by ensembles sharing the same space. In the notation, spaces between fused ensembles have no meaning, they are visual highlights.

{A|B|C|D|E} ⇒ A B C D E FUSE ensembles (add)

For example,

3 + 1 + 2 = 6: {●●●|●|●●} = ●●● ● ●● = ●●●●●●

Textual delimiters (parentheses, brackets, braces) provide a convenient typographical notation for boundary arithmetic operations, although typography does impose an unnatural sequencing on the representation of additive concepts that is not introduced at higher dimensions.

Multiplication is modeled by one-to-one substitution of ensembles for units. All substitutions occur in parallel. The SUBSTITUTE instruction [A C E] reads: concurrently substitute the ensemble A for each occurrence of C within the ensemble E. In multiplication, C is a single unit.

[A ● E] substitute A for each ● in E (multiply)

Although substitution itself is a directional process, the type of substitution that models the commutativity of multiplication is insensitive to direction.

[A C E] = [E C A] symmetric SUBSTITUTION

For example, substitute ●● for each ● in ●●●:

2 x 3 = 6: [●● ● ●●●] ⇒ ●● ●● ●●

Substitute ●●● for each ● in ●●:

3 x 2 = 6: [●●● ● ●●] ⇒ ●●● ●●●

Fusion distributes over substitution:

[A|B C E] = [A C E]|[B C E] DISTRIBUTION

To incorporate subtraction, a second type of unit is provided, with opposite polarity. ● is interpreted as 1 and ◊ is interpreted as −1. When numerals of different polarity occupy the same space, the Cancellation Axiom models subtraction as *void-substitution*:

● ◊ = CANCEL units (subtract)

Additive systems have no explicit zero, permitting absence and deletion to be used computationally. For example:

3 + 1 + −2 = 2: {●●●|●|◊◊} = ●●●●◊◊ = ●●

The auxiliary operator CHANGE-POLARITY, Δ, is useful for algebraic proofs. Equivalence, for example, can be defined in terms of complete cancellation. Characteristic of most boundary systems, computation proceeds via deletion (void-substitution), rather than by rearrangement. CANCEL, for example, is implemented as [<void> ●◊ E]. Decomposition permits induction over ensembles.

Polar units integrate into the substitution mechanism to model multiplication of signed numbers:

−1 x −1 = 1: [◊ ● ◊] = ●

The form being substituted *for* plays the role of a reciprocal, integrating a model of division into the substitution process. In division, A is a single unit.

[● C E] substitute ● for each C in E (divide)

Substitute ● for each ●●● in ●●●●●●:

6 ÷ 3 = 2: [● ●●● ●●●●●●] ⇒ ● ●

In general, [A C E] is interpreted numerically as AE/C. Some consequences of this model of multiplication:

[● C ●] = 1/C [A ● A] = A^2

[A A E] = E ΔΔA = A

[A C E] = [B C E] iff A = B

[A C [E F G]] = [A [C E F] G] = [[A C E] F G]

6. Depth-value Notation

Unit ensembles can be rewritten into an efficient *depth-value notation* by a standardization process that results in a boundary numeral with minimal structure, and corresponds to maximal factoring of the base for a conventional integer. The rewrite process consists of two transformation rules, GROUP units (interpreted as multiply by the base) and MERGE boundaries (interpreted as distribution).

● ● = (●) GROUP BASE-2

●●●●● ●●●●● = (●) GROUP BASE-10

(●)(●) = (● ●) MERGE/SPLIT

These rules apply in both directions, left-to-right to minimize structure after addition and multiplication, and right-to-left to access structure during subtraction and division.

7. Calculator Implementation

The boundary arithmetic calculator is a prototype implementation, in Mathematica 7.0, intended to illustrate and explore additive arithmetic with children. The animation of boundary computation has both algorithmic and human-interface display components. The calculator shows computational steps as applications of specific transformation rules, unlike conventional calculators that show input and output but not process. A human-centric computational process is necessarily observable and interactive, both for comprehension of procedures and for verification of results.

The concepts of addition and multiplication inherently require little computational effort. Algorithmic effort arises directly from the use of non-unary bases that facilitate convenient reading of large ensembles. Boundary arithmetic calculation, like all calculation in arithmetic, is dominated by the standardization of notation that permits ease of reading, rather than by the conceptually transparent processes of addition and multiplication.

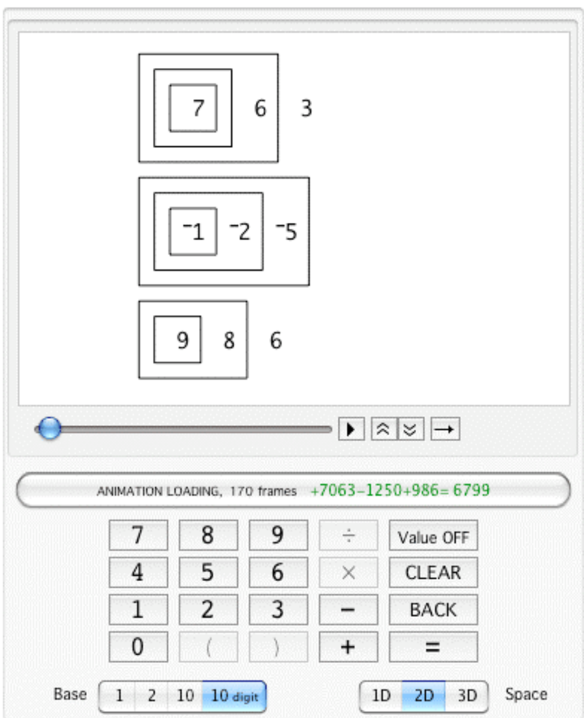


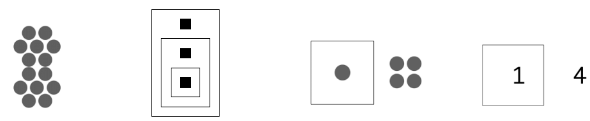
Figure 1. The boundary arithmetic calculator interface

7.1. Interface

The calculator interface incorporates the familiar input keypad with a large display screen (Figure 1). Only some of its functionality is currently implemented. The keypad permits positive and negative integers to be added, subtracted, multiplied, and divided. Keypad interaction is standard, digits are entered in conventional base-10 notation, separated by operation signs and grouped by parentheses. A textual information display directly above the keypad shows the current state of input; animation readiness, status, and duration; the expected result in standard notation; and error messages from human keypad slips and from software implementation bugs. The keypad EQUAL sign triggers an interactive animation of the desired computation. The BACK key permits correction of input mistakes and the CLEAR key clears the animation display.

A feature not found on conventional keypads is the VALUE key, which turns cursor-sensitivity of the display on and off. When on, the cursor will display in conventional notation the value of the form directly under it. This translation is useful for display of non-conventional base systems, and for translation of unfamiliar boundary numerals.

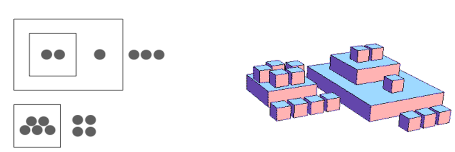
The calculator incorporates four computational BASE modes and three display SPACE modes. The 2D representation of the integer 14 is shown below for each BASE mode:



BASE-1 displays unit ensemble arithmetic, using solid discs as 1 and hollow discs as -1. BASE types other than BASE-1 use depth-value notation. Depth is represented by successive nesting of typographical delimiters in 1D mode, by nested rectangles in 2D mode, and by stacked rectangular blocks in 3D mode. The interpretation of each notation remains invariant; forms sharing space add, while nesting multiplies by the base. BASE-2 shows binary arithmetic. The representation is vertical, with solid squares representing 1 and hollow diamonds representing -1. BASE-10 shows unit ensembles that correspond to decimal digits. BASE-10-DIGIT shows aspects of conventional decimal notation, with digits from 1 from 9 in place of unit ensembles. Both base 10 forms display horizontally.

The three SPACE modes emphasize that boundary arithmetic is not confined to symbolic strings, but rather can be converted freely between linear, planar, and manipulative forms. Of course, the display of three dimensional forms is still representational rather than experiential. The 3D mode thus only indicates that computation can occur concretely, for example, by using stacks of physical objects such as blocks. Bricken [21] presents a diversity of spatial and experiential notations derived from geometrical and topological transformation of boundary forms, including textual, enclosure, graph, map, path and block notational systems. Examples of 1D, 2D, and 3D boundary notations for 213 + 54 (prior to fusion) are shown below in BASE-10.

((●●)●)●●● (●●●●●)●●●●



7.2. Animation

The animation display incorporates several conventional interactivity controls, including a slider for accessing the animation at any step, and buttons for start/stop, faster/slower, and forward/backwards (Figure 1). Animations of different BASE types follow the same algorithmic reduction steps, however the appearance of each differs. In this section, we describe the display of computation only in the 2D mode (additional examples of 1D display are presented in Sections 5 and 7.3).

In BASE-1, the display of addition first shows input integers as unit ensembles and then moves them together into a group. The VALUE key is necessary to count the units in the result. For the other three BASE types that use depth-value notation, addition is animated by first aligning nested depths visually, then moving adjacent boundaries together, and finally concurrently removing the partitions between boundaries. Figure 2 shows the animation of 213 + 54 in BASE-2 and Figure 3 shows the same for BASE-10. (In Figures that follow, the calculator interface is not shown.) The animation can be interpreted as placing ensembles that start out at the same depth but in different spaces into a single space of that depth. The shape of the resulting numeral is then normalized for ease of reading.

Should a space contain units of opposite polarity, polar units are matched one-to-one, and then each pair disappears into the background void, a visualization of the Cancellation rule, ● ◊ = . In BASE-10-DIGIT, digits first split in order to generate pairs of digits with equal cardinality and opposite polarity. Each pair merges and disappears. Figure 4 shows five concurrent cancellations in BASE-10-DIGIT (the initial boundary merging is not shown). Digit splitting is horizontal, digit merging is vertical.

This completes the computational phase for addition. Whatever remains represents the signed value at each particular depth. Although the calculator displays a consistent base over all spaces, in principle each space is independent. Different spaces within the same numeral can maintain different bases, different polarities, even different transformation strategies.

Standardization for readability comes next. All spaces convert to the same base and the same polarity. Local interaction across adjacent boundaries makes standardization a strongly parallel process [22]. When there are more units in a space than the cardinality of the base, units GROUP to form a new space. Grouping is displayed as units moving to form a base group, similar to cancellation, however when they converge they become a singular unit that grows a boundary around itself. Although the position of specific digits and ensembles does not enter into the axioms, from a visual perspective, when units form into groups there is a display dependence on proximity. Multiple boundaries within the same space, generated by grouping, next merge via the MERGE rule. Figure 5 shows a GROUP and MERGE sequence in BASE-10-DIGIT. In BASE-10-DIGIT mode only, multiple digits also fuse to form a single digit that represents their sum. BASE-10-DIGIT operations require knowledge of the facts of digit addition (e.g. 1 + 1 + 4 = 6). Digit facts are visual rather than symbolic in the other three BASE modes. Should all spaces have the same polarity, what remains is the simplest depth-value representation of the sum.

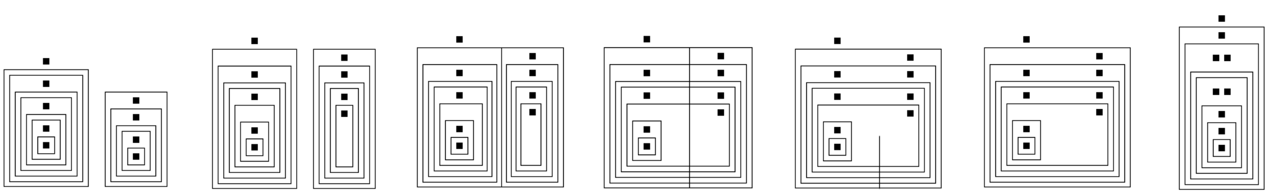


Figure 2. Animation of 213+54 in BASE-2 boundary notation, showing the Addition Principle

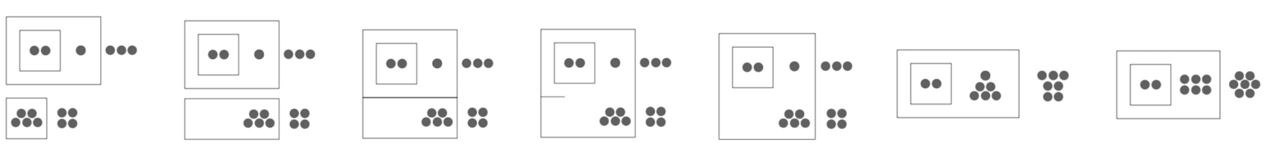


Figure 3. Animation of 213+54 in BASE-10 boundary notation, showing the Addition Principle

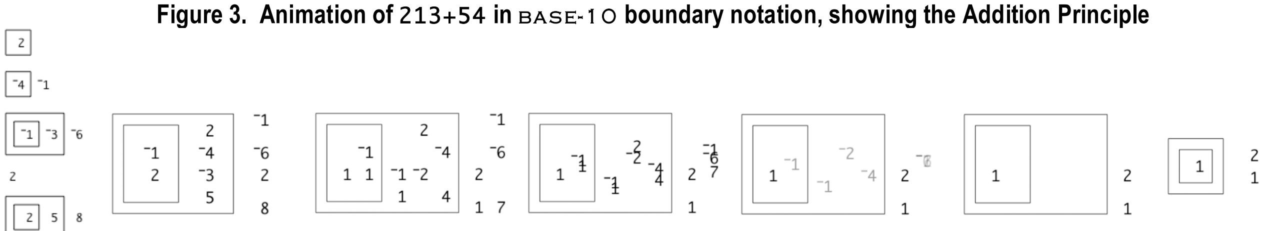
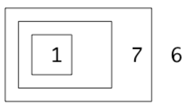


Figure 4. Animation of 20–41–136+2+258 in BASE-10-DIGIT boundary notation, showing Cancellation

The remaining possibility is that different spaces have different polarities. In this case, MERGE is applied in the reverse direction, to SPLIT boundaries. The boundary of the deeper space splits in two, with one boundary containing all but one unit, and the other containing a single unit. The boundary around the single unit collapses to reveal an ensemble of the appropriate base. The new units separate to match units of opposite polarity one-to-one, and are cancelled. What remains is a representation of the sum with consistent polarity. Figure 6 shows a SPLIT and CANCEL sequence in BASE-10-DIGIT.

When zero is added into spaces that do not contain units, and all boundaries are removed, what remains in BASE-10-DIGIT is a conventional place-value integer:



1 0 7 6

During multiplication, the numeral being substituted is replicated once for each unit in the numeral being substituted into. Visually, units being substituted for initially convert into miniature replicas of the numeral being substituted. The replicas then grow in place to match the scale of the host numeral. This completes the computational phase for multiplication. Next, units GROUP and boundaries MERGE, identical to the standardization phase after addition. Substitution itself maintains the appropriate polarity of units, so that no canceling occurs as the result of substitution.

7.3. Design Issues

An important design constraint is for the animation display to maintain rigorous consistency with the axioms of boundary arithmetic. Although the animation display is representational, it is intended also to serve as a set of dynamic instructions for conducting the same computation using physical manipulatives. For this, the 3D mode is most appropriate. In 2D, rectilinear containers were selected to represent boundaries for ease of graphics display. Earlier prototypes used circles, which interfered with the display of parallelism. For visual access, stacking rather than containment is used to represent depth in 3D. The intent of boundary notation is to maintain both the feel of manipulation and transparency to dimension of representation. Different display styles (varying shapes of units and orientation of numerals) are a result of experimentation rather than empirical validation of effectiveness.

Reviewing preschool math education concepts and playing with physical and virtual manipulatives helped the design process, however the central challenge was to unlearn how arithmetic is characterized. Review of the history of mathematics helped to validate that adding by "shoving together" is quite natural; the abacus is an example that has been in use for four thousand years. History also shows that dominant mathematical opinions during a given period usually suppress competing opinions, both antiquated and innovative. Prior to fractals being accepted as both beautiful and useful, for example, they were labeled "monster curves" and declared illegitimate [23,24]. The current debate over the legitimacy of diagrammatic proof echoes this struggle, for most mathematicians a proof must be symbolic [25,26]. Strangely, we were unable to find an existing formalization of additive systems.

There are three different choices available for subtraction: as an operation, as addition of opposite polarities, and as multiplication by -1.

$$A - B \quad = \quad A + (-B) \quad = \quad A + (-1)B$$

It seems a natural choice to use a negative unit, since that permits subtraction to be integrated with addition. The idea of several types of units was preferred over defining subtraction as a different type of operation. This approach is consistent with the group theoretic idea of an inverse, as well as being consistent with the tight integration of multiplication and division as substitution. However, such choices impose a specific model on the addition process. Negative units, for example, permit different polarities in different spaces, a consequence of parallelism but a significant deviation from conventional non-commutative subtraction. This is an example of the most

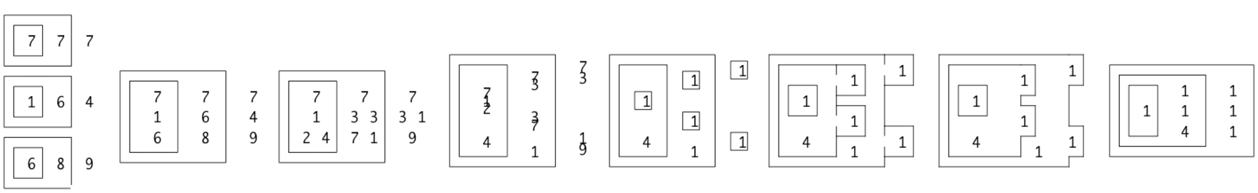


Figure 5. Animation of 777+164+689 in BASE-10-DIGIT notation, showing the GROUP and MERGE sequence

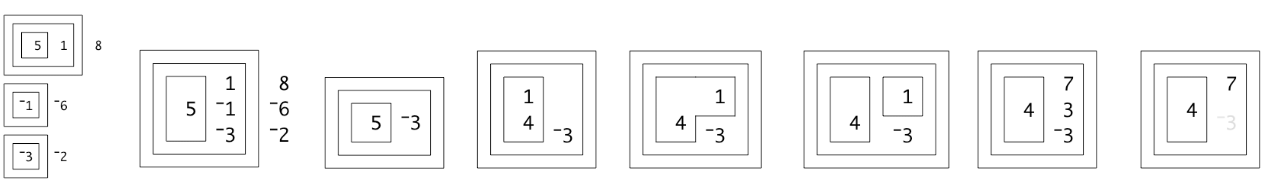


Figure 6. Animation of 5108–106–302 in BASE-10-DIGIT notation, showing the SPLIT and CANCEL sequence

challenging aspect of comparative axiomatics: how much consideration should be given to established convention? Is innovation within a universally accepted convention (linear, base-10, place-value numerals) excluded *a priori*?

Multiplication of signed numbers proved to be a difficult design decision. The form [◊ ● ◊] could be interpreted as: "there are no ●s in ◊, so substituting for ● in ◊ involves no change; the result should be ◊". CHANGE-POLARITY, Δ, provides a proof of the correct interpretation. The central idea is:

Δ[A C E] = [ΔA C E] = [A ΔC E] = [A C ΔE]

[◊ ● ◊] = ΔΔ[◊ ● ◊] = Δ[◊ ● Δ◊] = Δ[◊ ● ●] = Δ◊ = ●

Base systems introduce unavoidable sequential steps into standardization. Chains of grouping (carrying) can occur, as in 99 + 1. In 1D BASE-10-DIGIT mixed notation,

{(9) 9|●} = (9) 9 ● = (9)(●) = (9 ●) = ((●))

Chains of boundary splitting (borrowing) can occur across empty boundaries, such as in 200 – 1:

{((● ●))|◊} = ((● ●)) ◊ = ((●)(●)) ◊ = ((●) 9 ●) ◊ =

((●) 9)(●) ◊ = ((●) 9) 9 ● ◊ = ((●) 9) 9

From the perspective of visual animation, showing all potentially parallel processes overloads our ability to focus upon them all. A design decision was to limit display of concurrent processes to those of the same type across all spaces (with a few exceptions which were easily followed). After each parallel set of transformations, the shape of the single boundary integer is normalized, so that the eye can come to rest prior to the next set of transformations. Since the display area of the image varies widely, magnification is added selectively. The current implementation has stubs to accommodate color, sounds, and other multimedia effects, however these are yet to be explored.

8. Conclusion

The computational effort associated with arithmetic is purely notational, arising from a compromise between ease of concept and ease of reading. Unit ensemble arithmetic formalizes the conceptual ease of the Additive Principle. Boundary arithmetic combines ease of reading with ease of computation. This formal system provides the first step in a program of comparative axiomatics: having an alternative to compare. We have implemented a boundary arithmetic calculator as a tool to explore understanding and errors made by children as they learn arithmetic. Next we hope to contrast direct and symbolic manipulation, in an attempt to identify the appropriate roles of concrete and abstract mathematics instruction for children, and for adults burdened by math anxiety.

9. References

[1] J. Kilpatrick, J. Swalford & B. Findell [eds] (2001). *Adding It Up: Helping Children Learn Mathematics.* National Academy Press. p. 74

[2] J.G. Greeno & R. Hall (1997). "Practicing representation: learning with and about representational forms". *Phi Delta Kappan*, 78, 1–24. Available: http://www.pdkintl.org/kappan/kgreeno.htm

[3] M. Greaves (2002). *The Philosophical Status of Diagrams.* CSLI Publications.

[4] H. Weyl (1959). "Mathematics and the laws of nature". in I. Gordon & S, Sorkin [eds] *The Armchair Science Reader.* Simon & Schuster.

[5] M.S. Donovan & J.D. Bransford [eds] (2005). *How Students Learn Mathematics in the Classroom.* National Academy Press.

[6] J. Barwise & J. Etchemendy (1991). "Visual information and valid reasoning". in G. Allwein & J. Barwise [eds] *Logical Reasoning with Diagrams.* Oxford. p. 3

[7] J. Kaput (1987). "Representation systems and mathematics". In C. Janvier [ed] *Problems of Representation in the Teaching and Learning of Mathematics.* Erlbaum. p. 19–26.

[8] Utah State University (1999). National library of virtual manipulatives.Online:http://nlvm.usu.edu/en/nav/index.html

[9] P. Moyer, J. Bolyard & M. Spikell (2002). "What are virtual manipulatives?". *Teaching Children Mathematics*, 8(6) p. 373.

[10] D.H. Clements (1999). "Concrete manipulatives, concrete ideas". *Contemporary Issues in Early Childhood*, 1(1), 45-60. Available: http://www.gse.buffalo.edu/org/buildingblocks/Newsletters/Concrete_Yelland.htm

[11] W. Kneale & M. Kneale (1962). *The Development of Logic.* Oxford.

[12] G. Boole (1854). *The Laws of Thought.* Prometheus.

[13] C.S. Peirce (1931-58). *Collected papers of Charles Sanders Peirce* -- IV Ch 3 -- Existential Graphs, 4.397 - 4.417. C. Hartshorne, P. Weiss & A. Burks [eds] Harvard Univ Press.

[14] G. Spencer Brown (1969). *Laws of Form.* George Allen.

[15] IEEE Computer Society History of Computing. http://pages.cpsc.ucalgary.ca/~williams/History_web_site/timeline%203000BCE_1500CE/additive_number_system.html

[16] G. Ifrah (1981). *The Universal History of Numbers.* Wiley.

[17] R. Calinger (1999). *A Contextual History of Mathematics.* Prentice-Hall.

[18] L.H. Kauffman (1995). "Arithmetic in the form". *Cybernetics and Systems* 26: 1-57.

[19] P.B. Johnson (1975). *From Sticks and Stones.* SRA. p. 121

[20] R. Casati & A.C. Varzi (1999). *Parts and Places: The Structures of Spatial Representation.* MIT Press.

[21] W. Bricken (2006). "Syntactic variety in boundary logic". in D. Barker-Plummer et al [eds] *Diagrams 2006.* LNAI 4045 Springer-Verlag. p. 73-87.

[22] W. Bricken (1995). "Distinction Networks". in I. Wachsmuth et al [eds] *KI-95: Advances in Artificial Intelligence.* LNAI 981 Springer. p. 35-48.

[23] B. Mandelbrot (1982). *The Fractal Geometry of Nature.* Freeman.

[24] I. Lakatos (1976). *Proofs and Refutations.* Cambridge.

[25] S. Shin (1994). *The Logical Status of Diagrams.* Cambridge.

[26] G. Allwein & J. Barwise [eds] (1996). *Logical Reasoning with Diagrams.* Oxford.