The Set of Vanished Numbers $\mathbb{U}$

Dale Spencer
Zlan Technologies Ltd.
Email: dale@zlan.com

October 31, 2024

Abstract

We introduce a new mathematical structure $\mathbb{U}$, designed to track information that is lost during multiplication by zero while maintaining computational tractability. Beginning with a foundational singleton set containing a pure zero element, we develop a rigorous framework of mappings that preserve this otherwise lost information. The construction provides both theoretical insights into the nature of multiplication by zero and practical applications in computational systems where information preservation is critical. We establish the fundamental properties of $\mathbb{U}$ and demonstrate its relationship to standard number systems through explicit mappings. This work extends naturally to complex numbers and leads to a complete number system that maintains information typically lost in zero multiplication, with applications in data structure design and information theory.

Notation and Symbols

Number Systems

- $\mathbb{R}$ – Set of real numbers
- $\mathbb{Z}$ – Set of integers
- $\mathbb{C}$ – Set of complex numbers
- $\mathbb{U}$ – Set of vanished numbers (introduced in this paper)
- $\mathbb{CU}$ – Set of complete numbers (introduced in this paper)

Special Elements

- u – Fundamental unit of vanished numbers
- 0_p – Pure zero element
- 0_z – Zero element in standard integers
- au – General vanished number for $a \in \mathbb{R}$

Mappings

- $f_{0z} : \{0_p\} \to \mathbb{Z}$ – Pure zero to integer mapping
- $f_u : \{0_p\} \to \mathbb{U}$ – Pure zero to vanished number mapping
- $\phi : \mathbb{U} \to \mathbb{Z}$ – Zero-Connector Property mapping
- $\Phi : \mathbb{CU} \to \mathbb{C}$ – Extended Zero-Connector mapping for complete numbers

Complete Number Components

For a complete number $cu \in \mathbb{CU}$:

- a – Real component
- b – Imaginary component
- c_r – Real absorbed component
- d_i – Imaginary absorbed component
- $cu = a + bi + c_r u + d_i u$ – Full representation

Operations and Properties

- $(u \to 0_z)$ – Zero-Connector Property application
- $\times$ – Standard multiplication
- $\times_u$ – Multiplication in $\mathbb{U}$
- $+_u$ – Addition in $\mathbb{U}$

Functions and Transformations

- $\mathbf{A}_{ab}(a, b)$ – Area function in the ab plane
- $g_{\mathbb{R}} : \mathbb{R} \times \{0_p\} \to \mathbb{U}$ – Real vanishing mapping

Set Operations

- $\cap$ – Set intersection
- $\emptyset$ – Empty set
- $\exists!$ – Unique existence quantifier
- $\forall$ – Universal quantifier
- $\to$ – Mapping or implication

All other standard mathematical notation follows conventional usage. Specific definitions and properties of novel concepts are introduced as they appear in the text.

1 Introduction

The multiplicative property of zero stands as one of the most fundamental yet peculiar features of computational systems. When a number is multiplied by zero, the result is unambiguous - the product is zero. However, this apparently simple operation masks a subtle but significant loss of information. In the expression $a \times 0 = 0$, all information about the original number a vanishes in the result, leading to what we term *multiplicative absorption*.

This absorption is not merely a mathematical curiosity but has profound practical implications. Consider the following sequence of mathematical statements:

$$\begin{aligned} 5 \times 0 &= 0 \\ 3 \times 0 &= 0 \\ \therefore 5 \times 0 &= 3 \times 0 \end{aligned}$$

While logically consistent, this sequence demonstrates how distinct numerical values become indistinguishable after multiplication by zero, a property that can create significant challenges in computational systems and data structure design.

1.1 The Information Preservation Problem

The undefined nature of division by zero can be viewed as a direct consequence of information loss rather than an inherent limitation of arithmetic. To understand this distinction, consider the following points:

1. In standard computational implementations using integers $\mathbb{Z}$, when we write $a \times 0 = 0$, we have irrevocably lost the value of a
2. Subsequent operations cannot recover this information
3. The expression $0 \div 0$ is undefined because multiple valid answers exist - any number multiplied by zero equals zero

Remark 1.1 (Nature of Zero Multiplication). It is crucial to understand that multiplication by zero represents the vanishing of the underlying summation structure itself, not the elimination or removal of quantities. When we write $a \times 0$, we are describing a case where the recursive summation operation is performed zero times, resulting in a structural vanishing () that is distinct from the zero element in $\mathbb{Z}$. This subtle but fundamental distinction underlies our entire construction of $\mathbb{U}$.

This raises two fundamental questions:

1. Can we construct a mathematical system that preserves the information lost in zero multiplication while maintaining consistency with standard arithmetic?
2. How can such a system be practically implemented in computational contexts?

1.2 Our Approach

This paper introduces a new mathematical structure $\mathbb{U}$, the set of vanished numbers, designed to track and preserve information that would otherwise be lost during multiplication by zero. Our construction proceeds through three key stages:

1. **Pure Zero Construction**:

- We begin with a foundational singleton set containing a pure zero element
- This element exists independently of standard number systems
- It provides the basis for building information-preserving mappings

2. **Zero-Connector Property**:

- We establish explicit mappings between our new structure and standard numbers
- These mappings preserve both information and algebraic properties
- They enable practical implementation in computational systems

3. **Complete Numbers**:

- We extend our construction to a complete number system $\mathbb{CU}$
- This system combines standard numbers with vanished numbers
- It provides a framework for practical calculations requiring information preservation

1.3 Applications and Implementation

The structure we develop has immediate applications in:

1. **Data Structure Design**:

- Tracking value persistence through zero-mapping operations within $\mathbb{CU}$
- Maintaining provenance information through complete number calculations
- Enabling reversible computations involving zero within the set of complete numbers $\mathbb{CU}$

2. **Information Theory**:

- Preserving information through mappings traditionally considered destructive in $\mathbb{Z}$
- Maintaining state information in complete number zero-value conditions
- Enabling lossless computational tracking within $\mathbb{CU}$

3. **Practical Computing**:

- Implementation of complete number systems in programming languages
- Database systems requiring complete number value preservation
- Scientific computing applications demanding precision tracking in $\mathbb{CU}$

The remainder of this paper is organized as follows: Section 2 presents the foundational construction of $\mathbb{U}$. Section 3 establishes its key properties. Section 4 extends the construction to complex numbers. Section 5 discusses practical implementations and applications.

2 Foundational Construction

Our construction of the set of vanished numbers $\mathbb{U}$ begins with three foundational elements:

1. A pure form of zero that exists independently of standard number systems
2. Well-defined mappings that preserve algebraic structure
3. A connector property that allows controlled interaction between number spaces

We develop each of these elements systematically, ensuring rigorous foundations for later constructions.

2.1 The Pure Zero Set

The foundation of our construction begins with a single element that represents zero in its purest form, independent of any arithmetic operations or properties inherited from other number systems.

Definition 2.1 (Pure Zero Set)**.** We establish a foundational singleton set $\{0_p\}$ that exists independently of the integers $\mathbb{Z}$. This set contains exactly one element 0_p, which represents the purest form of zero:

$$\exists! 0_p \in \{0_p\} \text{ where } \{0_p\} \cap \mathbb{Z} = \emptyset \tag{1}$$

Proposition 2.2 (Non-existence of Arithmetic Operations)**.** *The pure zero set $\{0_p\}$ has no defined arithmetic operations. Specifically:*

1. *No addition operation*
2. *No subtraction operation*
3. *No multiplication operation*
4. *No division operation*

Remark 2.3. This restriction ensures that 0_p carries no inherent arithmetic properties. All subsequent operations will be defined through carefully constructed mappings to other number systems.

2.2 Mapping Functions

Having established a pure zero element with no inherent arithmetic properties, we now construct mappings that will allow us to connect this element to standard number systems while preserving information. These mappings must be carefully defined to maintain the purity of our construction while enabling meaningful arithmetic operations.

Definition 2.4 (Zero Mapping Function)**.** We first define a mapping function f_{0z} that connects the pure zero element to the integers:

$$f_{0z} : \{0_p\} \to \mathbb{Z}, \quad f_{0z}(0_p) = 0_z \tag{2}$$

where $0_z \in \mathbb{Z}$ represents zero in the integer number system.

Definition 2.5 (Pure Mapping Function)**.** We define a mapping function f_u that establishes the fundamental unit of our vanished number structure:

$$f_u : \{0_p\} \to \mathbb{U}, \quad f_u(0_p) = u \tag{3}$$

where u represents the fundamental unit of the vanished number set $\mathbb{U}$.

Remark 2.6. The notation u (rather than $0u$) is crucial here as it represents the fundamental mapping of pure zero, similar to how i represents the fundamental unit in imaginary numbers. Through the properties we will establish, we will see that u maintains an identity relationship with zero, ultimately showing that $u = 0u$, but the primary definition must establish u as the fundamental unit.

Theorem 2.7 (Mapping Independence)**.** *The mappings f_{0z} and f_u are independent in the sense that:*

$$f_u(0_p) \neq f_{0z}(0_p) \tag{4}$$

This independence is essential for maintaining information preservation in subsequent constructions.

Proof. The independence follows from:

1. $f_{0z}(0_p) = 0_z \in \mathbb{Z}$
2. $f_u(0_p) = u \in \mathbb{U}$
3. $\mathbb{Z} \cap \mathbb{U} = \emptyset$ by construction

Therefore, u and 0_z are distinct elements in different sets. □

2.3 Zero-Connector Property

Definition 2.8 (Zero-Connector Property)**.** The Zero-Connector Property establishes a fundamental mapping between $\{u\} \subset \mathbb{U}$ and $\{0_z\} \subset \mathbb{Z}$:

$$\phi : \{u\} \to \{0_z\}, \quad \phi(u) = 0_z \tag{5}$$

Theorem 2.9 (Fundamental Zero-Connector)**.** *The Zero-Connector mapping ϕ establishes the following fundamental properties:*

1. *$\phi(u) = 0_z$ preserves the zero element structure*
2. *The mapping is consistent with our pure zero construction: $\phi(f_u(0_p)) = f_{0z}(0_p)$*

2.4 General Vanishing Mapping

Definition 2.10 (General Form of Vanished Numbers)**.** For any $a \in \mathbb{Z}$, a vanished number has the form:

$$au = a \times (u \to 0_z) \tag{6}$$

where $(u \to 0_z)$ represents the application of the Zero-Connector Property.

Lemma 2.11 (Information Preservation)**.** *For any vanished number $au \in \mathbb{U}$, the original value a is preserved and can be recovered:*

$$\forall au \in \mathbb{U}, \exists! a \in \mathbb{Z} : au = a \times (u \to 0_z) \tag{7}$$

Proof. The preservation follows from our construction:

1. The coefficient a is explicitly maintained in the structure of au
2. The Zero-Connector Property ensures consistent mapping to $\mathbb{Z}$
3. The uniqueness of a follows from the well-defined nature of our mappings

□

2.5 Natural Identity Foundation

The construction of $\mathbb{U}$ fundamentally encodes a singular, well-defined case of absorption through its natural identity. This provides the theoretical foundation for all subsequent properties.

Definition 2.12 (Natural Identity)**.** The fundamental unit u serves as the multiplicative identity in $\mathbb{U}$ through the relation:

$$u = 0u \tag{8}$$

This identity is not proved but rather demanded by construction, encoding the single valid case of absorption in $\mathbb{Z}$.

Theorem 2.13 (Fundamental Absorption Encoding)**.** *The natural identity in $\mathbb{U}$ encodes the unique valid absorption case from $\mathbb{Z}$:*

$$\begin{aligned} u &= 0u \\ (u \to 0_z) &= 0 \times (u \to 0_z) \\ (0_z) &= 0 \times (0_z) \end{aligned}$$

where the final equality represents the singular valid absorption case $0 = 0 \times 0$ in $\mathbb{Z}$.

Remark 2.14. This encoding is fundamental to $\mathbb{U}$'s purpose: not to eliminate absorption entirely, but to reduce it to the single well-defined case where it behaves properly under multiplication and inverse operations. The parentheses in (0_z) are retained to emphasize the special properties we maintain through our construction.

Theorem 2.15 (Zero-Connector Operation Properties)**.** *For all operations constructed through the Zero-Connector Property mapping* $(u \to 0_z)$*, we maintain complete control over allowable operations on* 0_z *in* $\mathbb{Z}$*. This construction reveals that while information recovery is possible for any non-zero number multiplied by zero, the case of* 0×0 *where* 0 *originates from* $\mathbb{Z}$ *rather than* $\{0_p\}$ *cannot be recovered.*

Proof. Consider any arithmetic expression in $\mathbb{Z}$ that evaluates to $a > 0$. Suppose, by contradiction, that this expression contains at least one multiplication by zero. Let $b \times c$ be such a multiplication where either $b = 0$ or $c = 0$ (or both). Then $b \times c = 0$, and any further arithmetic operations on this result would yield 0. Therefore, no sequence of operations in $\mathbb{Z}$ involving multiplication by zero can result in $a > 0$.

Thus, when we encounter $a > 0$ in $\mathbb{Z}$, we can be certain that no information was lost through zero multiplication in its computation. However, when $a = 0$ in $\mathbb{Z}$, we cannot determine if this zero resulted from a multiplication by zero or was a primitive zero, as both paths lead to the same result in $\mathbb{Z}$. □

Remark 2.16 (Theoretical vs Practical Applications). While this limitation exists in the theoretical framework where we consider the uncountably infinite set of all possible operations and allow 0 to originate from $\mathbb{Z}$, practical applications operate within a finite set of well-defined operations. In computational contexts, such as tracking quantities (e.g., items in containers), we work exclusively with complete numbers through a finite sequence of arithmetic operations. In these cases, we can always determine if our 0 originated from $\{0_p\}$, ensuring recoverability of information for all values of a. This is precisely because practical computations, by definition, operate purely within $\mathbb{CU}$ using a finite sequence of operations, never encountering the theoretical limitation of 0 originating from $\mathbb{Z}$.

Remark 2.17 ('0' Literals and Pure Zero). The literal '0' in computational systems provides an interesting parallel to 0_p. In programming languages, such a literal exists as a pure symbolic representation before any arithmetic operations are applied, much like our $\{0_p\}$ construction. When we construct a complete number from this literal, as in `CompleteNumber(0)`, we are effectively following the same pattern as our theoretical construction: starting with a pure representation and explicitly mapping it into our arithmetic system. This alignment between the practical computational representation and our theoretical construction further reinforces the natural fit of complete numbers for practical applications. The literal '0' can be used directly to act on complete numbers because of this parallel to 0_p.

Theorem 2.18 (Multiplicative Properties)**.** *The multiplicative properties in* $\mathbb{U}$ *are defined piecewise for* $a \in \mathbb{Z}$*:*

1. *Natural Identity (fundamental):* $u = 0u$
2. *For operations* au/au*:*

- *When* $a \neq 0$*: Properties follow from ZCP mapping*
- *When* $a = 0$*: Natural Identity applies*

Remark 2.19 (Piecewise Identity). The piecewise nature of the multiplicative identity emerges directly from our construction using the Zero-Connector Property and our fundamental absorption encoding through $u = 0u$.

Corollary 2.20 (Natural Inverse Property)**.** *For all* $a \in \mathbb{Z}$*, the well-defined nature of the natural identity immediately implies the existence of a unique multiplicative inverse:*

$$au \times u/au = u \tag{9}$$

This follows directly from the singular nature of the absorption case we encode.

Remark 2.21. The visualization demonstrates how $\mathbb{U}$ resolves the discontinuity at $a = 0$ by encoding the natural identity. In $\mathbb{Z}$, a/a is undefined at 0, creating a discontinuity. In $\mathbb{U}$, $au/au = u$ holds for all $a \in \mathbb{Z}$, including $a = 0$, making the operation well-defined everywhere through our encoding of the singular valid absorption case.

Remark 2.22 (Identity Value of u). It is crucial to note that $u = 0u$ by construction, and one must resist the habit of assuming $u = 1u$. This assumption, while natural due to familiarity with the multiplicative identity in $\mathbb{Z}$, is only valid for $a > 0$ through the inherited identity property. For $a = 0$, we must use the natural identity where $u = 0u$. This distinction is clearly visible in the plot of au/au, which maintains the value $1u$ for all $a > 0$ but takes the value $u = 0u$ at $a = 0$.

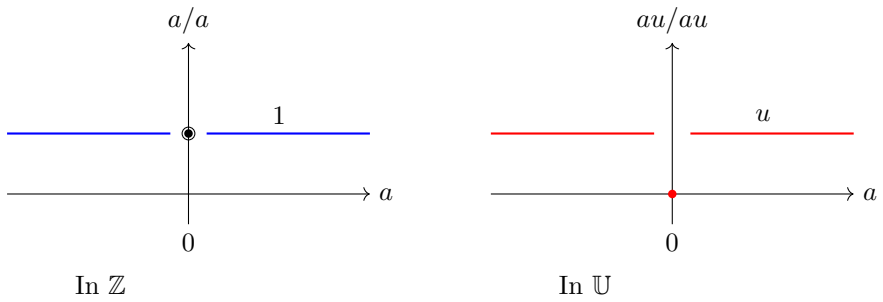

Figure 1: Resolution of the a/a discontinuity in $\mathbb{U}$. The left plot shows the discontinuity at $a = 0$ in $\mathbb{Z}$, while the right plot shows how $\mathbb{U}$ resolves this through the natural identity, making u the consistent value throughout.

3 Extending Analysis to $\mathbb{R}$ and $\mathbb{U}$

Theorem 3.1 (Extension to Real Numbers)**.** *The general vanishing mapping extends naturally to real numbers. For any $a \in \mathbb{R}$, we define:*

$$g_{\mathbb{R}} : \mathbb{R} \times \{0_p\} \to \mathbb{U}, \quad g_{\mathbb{R}}(a, 0_p) = a \times (u \to 0_z) \tag{10}$$

where the Zero-Connector Property maintains the same essential characteristics as in the integer case.

3.1 Visualizing $a \times b$ as Area A_{ab}

Definition 3.2 (Area Function)**.** Consider the following function which represents the area defined in the ab plane by the multiplication of a and b:

$$\mathbf{A}_{ab}(a, b) = a \times b, \quad \forall a, b : a, b \in \mathbb{R} \tag{11}$$

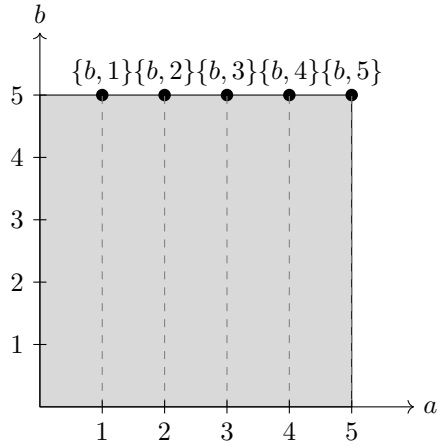

Figure 2: Some area $\mathbf{A}_{ab}(a, b)$ within the ab plane defined by $\mathbb{R}$

Theorem 3.3 (Area Limit Definition)**.** *The vanishing mapping can be understood through the limit of the area function as b approaches our mapped pure zero:*

$$\lim_{b \to (0u \to 0_z)} \mathbf{A}_{ab}(a, b) = g_{\mathbb{R}}(a, 0_p) \tag{12}$$

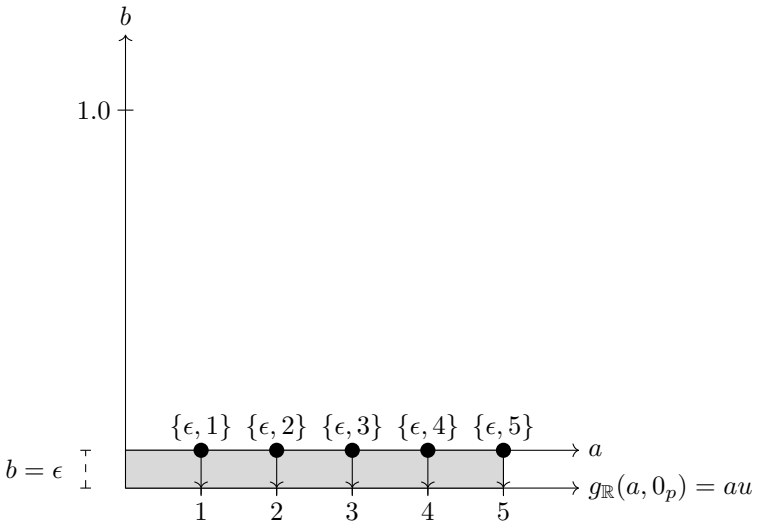

Figure 3: An infinitesimal area $\mathbf{A}_{ab}(a, b)$ as $b = \epsilon$ approaches $0u \to 0_z$

Theorem 3.4 (Mapping Equivalence)**.** *The limit approach and our direct mapping construction produce equivalent results:*

$$g_{\mathbb{R}}(a, 0_p) = au \iff \lim_{b \to (0u \to 0_z)} \mathbf{A}_{ab}(a, b) = au \tag{13}$$

This equivalence demonstrates the natural geometric interpretation of our algebraic construction.

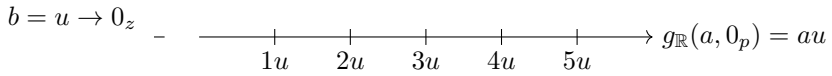

Figure 4: The number line au is what remains after b converges to $u \to 0_z$

4 Properties of $\mathbb{U}$

The following properties hold $\forall a \in \mathbb{Z}$:

1. **Additive Properties**:
   - **Closure**: $(a_1u + a_2u) \in \mathbb{U}$
   - **Associative**: $(a_1u + a_2u) + a_3u = a_1u + (a_2u + a_3u)$
   - **Commutative**: $a_1u + a_2u = a_2u + a_1u$
   - **Identity**: $au + u = au$
   - **Inverse**: $au + (-au) = u$
2. **Multiplicative Properties**:
   - **Closure**: $(a_1u \times a_2u) \in \mathbb{U}$
   - **Associative**: $(a_1u \times a_2u) \times a_3u = a_1u \times (a_2u \times a_3u)$
   - **Commutative**: $a_1u \times a_2u = a_2u \times a_1u$
   - **Distributive**: $a_1u \times (a_2u + a_3u) = a_1a_2u + a_1a_3u$
   - **Natural Identity**: $au \times u = au$ (by construction)
   - **Natural Inverse**: $au \times u/au = u$ (by construction)
   - **Inherited Identity**: $au \times 1u = au$
   - **Inherited Inverse**: $au \times 1u/au = 1u$
3. **Connector Properties**:
   - **Zero-Connector**: $u = 0_z$, where $0_z \in \{0\}$
   - **Desorber**: $au \times 1/bu = a/b$, where $\{1, a/b\} \in \mathbb{Z} \wedge b \neq 0$
   - **Zero-Connector Multiplicative Inverse**: $au \cdot 1/0_z = a$
   - **Recursive Definition**: $au = au \times 0_z$

Remark 4.1. The Inherited Identity and Inherited Inverse properties emerge from the seamless mapping of coefficients between $\mathbb{Z}$ and $\mathbb{U}$. These properties are explored in detail in Section 5.

5 Algebraic Structure of $\mathbb{U}$

The properties established in the previous section give rise to a rich algebraic structure with both natural and inherited characteristics.

Theorem 5.1 (Ring Structure)**.** *The set $\mathbb{U}$ forms a commutative ring with identity, where the identity arises from our fundamental absorption encoding rather than standard ring construction.*

Proof. The additive and multiplicative properties established earlier satisfy ring axioms with the crucial distinction that our multiplicative identity u emerges from the encoded absorption case rather than the standard construction. □

5.1 Inherited Properties

The seamless mapping between $\mathbb{Z}$ and $\mathbb{U}$ leads to an interesting duality in identities:

Theorem 5.2 (Identity Duality)**.** *The set $\mathbb{U}$ exhibits two distinct but compatible identity behaviors:*

1. *Natural identity: u acts as identity through absorption encoding*
2. *Inherited identity: $1u$ acts as identity through coefficient mapping*

Proof. For any $au \in \mathbb{U}$:

$$\begin{aligned} au \times 1u &= a \times (u \to 0_z) \times 1 \times (u \to 0_z)/(u \to 0_z) && (14)\\ &= a \times 1 \times (u \to 0_z) \times (u \to 0_z)/(u \to 0_z) \quad (a \times 1 \text{ occurs in } \mathbb{Z} \text{ before mapping back to } \mathbb{U}) && (15)\\ &= a \times (u \to 0_z) && (16)\\ &= au && (17) \end{aligned}$$

where the operation occurs in $\mathbb{Z}$ through coefficient mapping before returning to $\mathbb{U}$. □

Remark 5.3. This duality of identities is a unique feature of $\mathbb{U}$ arising from its construction. While u serves as the fundamental identity through absorption encoding, the structure also inherits the familiar behavior of multiplication by 1 through its coefficient mapping from $\mathbb{Z}$. The explicit grouping of $(a \times 1)$ in equation (15) exposes how this identity emerges: it is inherited because the operation occurs within $\mathbb{Z}$ where standard multiplication by 1 applies, before being mapped back into $\mathbb{U}$.

5.2 Homomorphic Properties

The relationship between $\mathbb{U}$ and $\mathbb{Z}$ extends beyond simple mapping:

Theorem 5.4 (Structure Preservation)**.** *The coefficient mapping between $\mathbb{U}$ and $\mathbb{Z}$ preserves algebraic structure while maintaining information typically lost in standard absorption.*

Proof. For $a_1, a_2 \in \mathbb{Z}$:

1. Addition: $(a_1u + a_2u) \to (a_1 + a_2)$ in $\mathbb{Z}$
2. Multiplication: $(a_1u \times a_2u) \to (a_1 \times a_2)$ in $\mathbb{Z}$
3. Zero mapping: $u \to 0_z$ preserves the special absorption case

□

6 Complete Numbers

Definition 6.1 (Complete Number Structure)**.** A complete number $cu \in \mathbb{CU}$ is an ordered quadruple of the form:

$$cu = (a, b, c_r, d_i) = a + bi + c_ru + d_iu \tag{18}$$

where:

- $a, c_r \in \mathbb{R}$ (real components)
- $b, d_i \in \mathbb{R}$ (imaginary components)
- $c_ru, d_iu \in \mathbb{U}$ (absorbed components)

Theorem 6.2 (Arithmetic Operations)**.** *For any two complete numbers* $cu_1 = a_1 + b_1 i + c_{1r} u + d_{1i} u$ *and* $cu_2 = a_2 + b_2 i + c_{2r} u + d_{2i} u$*:*

1. ***Addition****:*

$$cu_1 + cu_2 = (a_1 + a_2) + (b_1 + b_2)i + (c_{1r} + c_{2r})u + (d_{1i} + d_{2i})u \tag{19}$$

2. ***Multiplication****:*

$$\begin{aligned} cu_1 \times cu_2 =& (a_1 a_2 - b_1 b_2) + (a_1 b_2 + b_1 a_2)i + \\ & (a_1 c_{2r} + a_2 c_{1r} - b_1 d_{2i} - b_2 d_{1i})u + \\ & (a_1 d_{2i} + a_2 d_{1i} + b_1 c_{2r} + b_2 c_{1r})u \end{aligned}$$

Proof. 1. **Addition**: Follows directly from component-wise addition and the additive properties of $\mathbb{U}$.

2. **Multiplication**: We distribute terms using the field properties of $\mathbb{R}$, $\mathbb{C}$, and the multiplicative properties of $\mathbb{U}$:

$$\begin{aligned} &(a_1 + b_1 i + c_{1r} u + d_{1i} u)(a_2 + b_2 i + c_{2r} u + d_{2i} u) = \\ &\quad = (a_1 a_2 - b_1 b_2) \quad \text{(real terms)} \\ &\quad + (a_1 b_2 + b_1 a_2)i \quad \text{(imaginary terms)} \\ &\quad + (a_1 c_{2r} + a_2 c_{1r})u \quad \text{(real absorbed terms)} \\ &\quad + (a_1 d_{2i} + a_2 d_{1i})u \quad \text{(imaginary absorbed terms)} \\ &\quad + (-b_1 d_{2i} - b_2 d_{1i})u \quad \text{(cross absorbed terms)} \\ &\quad + (b_1 c_{2r} + b_2 c_{1r})u \quad \text{(remaining cross terms)} \end{aligned}$$

□

Theorem 6.3 (Closure Properties)**.** *The set* $\mathbb{CU}$ *is closed under addition and multiplication.*

Proof. Given $cu_1, cu_2 \in \mathbb{CU}$:

1. **Addition**: - Real components sum to a real number - Imaginary components sum to an imaginary number - Absorbed components sum within $\mathbb{U}$ by the closure property of $\mathbb{U}$ Therefore, $cu_1 + cu_2 \in \mathbb{CU}$

2. **Multiplication**: - Real/imaginary cross terms resolve to real/imaginary numbers - Terms involving u resolve to elements in $\mathbb{U}$ by closure in $\mathbb{U}$ - All components combine according to their respective field properties Therefore, $cu_1 \times cu_2 \in \mathbb{CU}$ □

Theorem 6.4 (Zero-Connector Compatibility)**.** *The Zero-Connector Property extends naturally to* $\mathbb{CU}$ *through the mapping:*

$$\Phi : \mathbb{CU} \to \mathbb{C}, \quad \Phi(cu) = \Phi(a + bi + c_r u + d_i u) = a + bi + \phi(c_r u + d_i u) \tag{20}$$

where ϕ *is the original Zero-Connector mapping.*

Theorem 6.5 (Information Preservation)**.** *For any operation sequence in* $\mathbb{CU}$ *involving multiplication by zero:*

$$\forall cu \in \mathbb{CU}, \exists! w \in \mathbb{CU} : cu \times 0 = w \tag{21}$$

where w *preserves all information about the original components of* cu *through the absorbed terms.*

Proof. For $cu = a + bi + c_r u + d_i u$:

$$\begin{aligned} cu \times 0 &= (a + bi + c_r u + d_i u) \times 0 \\ &= (a \times 0) + (bi \times 0) + (c_r u \times 0) + (d_i u \times 0) \\ &= 0 + 0i + au + (bi)u \end{aligned}$$

The result preserves a and b in the absorbed components while maintaining consistency with standard zero multiplication. □

6.1 Compatibility with Zero-Connector Property

Having established the structure of complete numbers, we now show how the Zero-Connector Property extends naturally to $\mathbb{CU}$ while preserving all essential properties.

Definition 6.6 (Extended Zero-Connector Mapping)**.** For complete numbers, we define an extended mapping Φ that preserves both structure and information:

$$\Phi : \mathbb{CU} \to \mathbb{C}, \quad \Phi(a + bi + c_r u + d_i u) = (a + bi) + \phi(c_r u + d_i u) \tag{22}$$

where ϕ is the original Zero-Connector mapping.

Theorem 6.7 (Complete Number Compatibility)**.** *The extended mapping Φ preserves:*

1. *The structure of complex numbers: $a + bi$ maps directly to $\mathbb{C}$*
2. *The vanished number properties through ϕ*
3. *Information about both real and imaginary components*

Proof. For any complete number $cu = a + bi + c_r u + d_i u$:

1. The complex part maps naturally: $(a + bi) \in \mathbb{C}$
2. The vanished components preserve information through ϕ:
$$\phi(c_r u) = c_r \times 0_z$$
$$\phi(d_i u) = d_i \times 0_z$$
3. Original values can be recovered through the established properties of $\mathbb{U}$

□

7 Developing an Intuition for Complete Numbers

When we count objects, like apples, we implicitly assume we're using integers from $\mathbb{Z}$. This unstated assumption leads to confusion about information preservation when multiplication by zero occurs.

Example 7.1 (Apple Counting in $\mathbb{Z}$). Let's represent the number of apples we have in a crate with the integer c and in a box with b, where $c, b \in \mathbb{Z}$. If we write:

$$b \times 0 = 0$$
$$c \times 0 = 0$$

The information about the original quantities is irrevocably lost due to absorption in $\mathbb{Z}$. This is a necessary fact to preserve consistency within $\mathbb{Z}$ by sacrificing completeness.

Theorem 7.2 (Complete Number Preservation)**.** *If instead we explicitly work with complete numbers from the start:*

$$b_c = b + 0i + 0u \in \mathbb{CU}$$
$$c_c = c + 0i + 0u \in \mathbb{CU}$$

Then multiplication with $0u$ preserves the original information:

$$b_c \times 0u = 0 + 0i + bu$$
$$c_c \times 0u = 0 + 0i + cu$$

Remark 7.3 (Practical Implementation). When working with quantities that might be multiplied by zero, we must explicitly:

1. Declare our numbers as complete numbers from the start
2. Perform operations in $\mathbb{CU}$ rather than $\mathbb{Z}$
3. Maintain the complete number structure throughout calculations

This is not merely a mathematical formality - it is essential for practical applications where information preservation is required.

Therefore, to properly solve our apple distribution problem:

$$\begin{aligned} b_c \times 0u &= 0 + 0i + bu \\ c_c \times 0u &= 0 + 0i + cu \\ (bu + cu)/0_z &= b + c \end{aligned}$$

Corollary 7.4 (Practical Usage)**.** *The apparent paradox of information loss in practical calculations is resolved by recognizing that we must explicitly work in $\mathbb{CU}$ when we want to preserve information through zero multiplication. The unstated assumption of working in $\mathbb{Z}$ is what leads to information loss in practical applications.*

At the end of such diligence all of the apples are our reward.

8 Implementations and Applications

The structure of complete numbers $\mathbb{CU}$ provides both theoretical insights and practical implications. We explore these through several key areas where the preservation of information under zero multiplication becomes significant.

8.1 Theoretical Implications

The construction of $\mathbb{CU}$ reveals several important theoretical insights:

Theorem 8.1 (Information Conservation)**.** *For any sequence of operations in $\mathbb{CU}$ involving multiplication by zero, the original values can be recovered through the inverse mapping established by the Zero-Connector Property.*

Proof. Given any $cu \in \mathbb{CU}$ where $cu = a + bi + c_r u + d_i u$:

1. Under multiplication by zero: $cu \times 0 = 0 + 0i + au + (bi)u$
2. The original values a and b are preserved in the absorbed components
3. These can be recovered through the Zero-Connector mapping: $(au + (bi)u)/0_z = a + bi$

□

8.2 Structural Properties

The complete number system $\mathbb{CU}$ exhibits several notable structural properties:

Proposition 8.2 (Absorption Chain Property)**.** *In $\mathbb{CU}$, repeated zero multiplications reduce to familiar arithmetic operations composed in $\mathbb{U}$, making absorption chains computationally tractable. Specifically, for any complete number $cu = a + bi + c_r u + d_i u$:*

$$\begin{aligned} cu \times 0 &= 0 + 0i + (a + bi)u \\ (cu \times 0) \times 0 &= 0 + 0i + 0u + ((a + bi)u)u \\ ((cu \times 0) \times 0) \times 0 &= 0 + 0i + 0u + (0u + ((a + bi)u)u)u \end{aligned}$$

where each step follows standard arithmetic rules composed with operations in $\mathbb{U}$. The original value $a + bi$ remains recoverable through successive applications of the Zero-Connector Property, with each level of absorption corresponding to a straightforward reversal of these familiar operations.

Remark 8.3. It reduces seemingly complex chains of zero multiplication to simple compositions of standard operations within $\mathbb{U}$. Each absorption level maintains a clear relationship to familiar arithmetic, making the tracking of information natural despite the multiple layers of zero multiplication.

Theorem 8.4 (Structure Preservation)**.** *The algebraic structure of* $\mathbb{CU}$ *preserves:*

1. *Ring properties under standard operations*
2. *Information content under zero multiplication*
3. *Compatibility with existing number systems through the Zero-Connector Property*

8.3 Application to Mathematical Systems

The framework of complete numbers has natural applications in several mathematical contexts:

Example 8.5 (Linear Transformations). Consider a linear transformation $T : V \to W$ between vector spaces. When $\ker(T) \neq \{0\}$, the complete number structure allows tracking of vectors mapped to zero while preserving their original components:

$$T(v) = 0 \implies v_u = v \times (0u \to 0_z) \tag{23}$$

where v_u maintains the information about the original vector v.

Example 8.6 (Polynomial Systems). In polynomial rings, roots with multiplicity create information loss similar to zero multiplication. The complete number structure provides a framework for tracking this information:

$$(x - a)^n = 0 \implies x = a + ku \text{ where } k \text{ tracks multiplicity} \tag{24}$$

8.4 Extensions and Future Directions

The structure of complete numbers suggests several promising directions for future research:

1. Extension to other algebraic structures where information preservation under special elements is desired
2. Development of generalized absorption chains for tracking multiple transformations
3. Investigation of complete number analogs for other fields and rings
4. Study of topology induced by complete number operations

These directions suggest that complete numbers may provide insights beyond their original motivation in zero multiplication, potentially offering new perspectives on information preservation in various mathematical contexts.

9 Computational Realization

The theoretical framework of complete numbers can be practically implemented in computational systems. We present a Python implementation that demonstrates both the mathematical structure and its practical applications.

9.1 Implementation Structure

The `CompleteNumber` class implements the full structure of $\mathbb{CU}$, maintaining separate components for real, imaginary, and absorbed values. The implementation preserves all essential properties while providing practical methods for computation.

```python
# # Complex Number Patterns
# z = 3 + 4j                # Create complex number
# z = complex(3, 4)         # Alternative construction
# z.real * 0  -> 0.0        # Real component times zero
# z.imag * 0  -> 0.0        # Imaginary component times zero
# z * 0       -> 0j         # Whole complex number times zero

# cu = 3 + 4j                # Create complete number
# cu = CompleteNumber(3, 4) # Alternative construction
# cu.real * 0  -> 3u         # Real component times zero
# cu.imag * 0  -> 4uj        # Imaginary component times zero
# cu * 0      -> 3u + 4uj    # Whole complete number times zero

from typing import Union, Tuple
import numbers
import cmath

class CompleteComponent:
    """Drop-in replacement for float that maintains absorption into U"""
    def __init__(self, value, component_type):
        self._value = float(value)
        self._type = component_type    # 'real' or 'imag'

    def __mul__(self, other):
        if other == 0:
            if self._type == 'real':
                # Real component times zero -> just the absorbed value
                return f"{self._value}u"
            else:  # imaginary
                # Imaginary component times zero -> just the absorbed value
                return f"{self._value}uj"
        else:
            # Non-zero multiplication - return just the float value
            return self._value * other

    def __truediv__(self, other):
        """
        Component division should behave like regular float division,
        raising ZeroDivisionError when dividing by zero
        """
        if other == 0:
            raise ZeroDivisionError("float division by zero")
        return self._value / other

    def __rtruediv__(self, other):
        if self._value == 0:
            raise ZeroDivisionError("float division by zero")
        return other / self._value

    def __rmul__(self, other):
        return self.__mul__(other)

    def __rtruediv__(self, other):
        """Implement reverse division"""
        # This handles cases like 1 / CompleteNumber
        return NotImplemented  # For now, we'll implement if needed

    def __repr__(self):
        return str(self._value)

    def __float__(self):
        return self._value

class CompleteNumber:
    def __init__(self, real, imag, u_real=0, u_imag=0):
        self._real = float(real)
        self._imag = float(imag)
```

```
        self._u_real = float(u_real)
        self._u_imag = float(u_imag)

    @property
    def real(self):
        return CompleteComponent(self._real, 'real')

    @property
    def imag(self):
        return CompleteComponent(self._imag, 'imag')

    def __mul__(self, other):
        if isinstance(other, (int, float)):
            if other == 0:
                # Whole number times zero -> both components go to U
                return CompleteNumber(0, 0, self._real, self._imag)
            else:
                # Regular multiplication
                return CompleteNumber(self._real * other, self._imag * other)
        return NotImplemented

    def __truediv__(self, other):
        """
        Complete number division. Raises ZeroDivisionError if any non-absorbed
        components would be divided by zero.
        """
        if isinstance(other, (int, float)):
            if other == 0:
                # If there are any non-absorbed components, division by zero is
    undefined
                if self._real != 0 or self._imag != 0:
                    raise ZeroDivisionError("division by zero")
                # Only absorbed components present
                return CompleteNumber(
                    self._u_real,
                    self._u_imag
                )
            else:
                # Regular division
                return CompleteNumber(
                    self._real / other,
                    self._imag / other,
                    self._u_real / other,
                    self._u_imag / other
                )
        return NotImplemented

    def __rtruediv__(self, other):
        """Implement reverse division"""
        if self._value == 0:
            raise ZeroDivisionError("division by zero")
        return other / self._value

    def __rmul__(self, other):
        return self.__mul__(other)

    def __repr__(self):
        parts = []
        if self._real != 0 or (self._imag == 0 and self._u_real == 0 and self._u_imag
    == 0):
            parts.append(str(self._real))
        if self._imag != 0:
            parts.append(f"{self._imag}j")
        if self._u_real != 0:
            parts.append(f"{self._u_real}u")
        if self._u_imag != 0:
            parts.append(f"{self._u_imag}uj")

```

```
        return " + ".join(parts).replace(" + -", " - ")

def test_basic_multiplication():
    """
    Test both complete number and complex number behaviors with multiplication.
    """
    print("\nBasic Multiplication Tests:")

    # Create a complete number
    cu = CompleteNumber(3, 4)  # 3 + 4j
    print(f"Initial number: {cu}")

    # Test 1: Whole number multiplication by 0
    result1 = cu * 0
    print(f"\nCase 1 - Whole number * 0:")
    print(f"({cu}) * 0 = {result1}")  # Should be 3u + 4uj

    # Test 2: Real component multiplication by 0
    result2 = cu.real * 0
    print(f"\nCase 2 - Real component * 0:")
    print(f"({cu}).real * 0 = {result2}")  # Should be 3u + 4j

    # Test 3: Imaginary component multiplication by 0
    result3 = cu.imag * 0
    print(f"\nCase 3 - Imaginary component * 0:")
    print(f"({cu}).imag * 0 = {result3}")  # Should be 3 + 4uj

    # Test 4: Verify normal multiplication still works
    print(f"\nVerification of normal multiplication:")
    print(f"Initial number: {cu}")
    result4 = cu * 1
    print(f"({cu}) * 1 = {result4}")  # Should be 3 + 4j
    result5 = cu.real * 1
    print(f"({cu}).real * 1 = {result5}")  # Should be 3
    result6 = cu.imag * 1
    print(f"({cu}).imag * 1 = {result6}")  # Should be 4

    return result1, result2, result3, result4, result5, result6

def test_basic_division():
    """
    Test division behavior, particularly division by zero cases.
    """
    print("\nBasic Division Tests:")

    # Create a complete number
    cu = CompleteNumber(3, 4)  # 3 + 4j
    print(f"Initial number: {cu}")

    # Test 1: Whole number after absorption divided by 0
    cu_absorbed = cu * 0  # First absorb into U
    try:
        result1 = cu_absorbed / 0
        print(f"\nCase 1 - Complete absorbed number / 0:")
        print(f"({cu_absorbed}) / 0 = {result1}")
    except ZeroDivisionError as e:
        print(f"({cu_absorbed}) / 0 -> ZeroDivisionError: {e}")

    # Test 2: Real component division by 0 (should raise exception)
    print(f"\nCase 2 - Real component / 0:")
    try:
        result2 = cu.real / 0
        print(f"({cu}).real / 0 = {result2}")
    except ZeroDivisionError as e:
        print(f"({cu}).real / 0 -> ZeroDivisionError: {e}")

    # Test 3: Imaginary component division by 0 (should raise exception)
    print(f"\nCase 3 - Imaginary component / 0:")
```

```
    try:
        result3 = cu.imag / 0
        print(f"({cu}).imag / 0 = {result3}")
    except ZeroDivisionError as e:
        print(f"({cu}).imag / 0 -> ZeroDivisionError: {e}")

    # Test 4: Pure absorbed components divided by 0
    cu_absorbed_only = CompleteNumber(0, 0, 3, 4)  # 3u + 4uj
    result4 = cu_absorbed_only / 0
    print(f"\nCase 4 - Pure absorbed components / 0:")
    print(f"({cu_absorbed_only}) / 0 = {result4}")  # Should recover original values

    return cu_absorbed, result4

if __name__ == "__main__":
    results = test_basic_multiplication()

    results = test_basic_division()
```

Listing 1: Complete Number Class Implementation

9.2 Practical Examples

We demonstrate the practical application of complete numbers using the apple counting example from our theoretical discussion.

```
def demonstrate_vanishing_summation():
    """
    Demonstrates how multiplication by zero represents the vanishing of the summation
    structure itself, not the removal of quantities. The complete number system
    preserves this structural information that would otherwise be lost.
    """
    print("\nVanishing Summation Structure Demonstration:")

    # Create complete numbers for our quantities
    box_quantity = CompleteNumber(5, 0)    # 5 as a quantity
    crate_quantity = CompleteNumber(3, 0)  # 3 as a quantity
    print(f"Initial quantities:")
    print(f"Box:   {box_quantity}")
    print(f"Crate: {crate_quantity}")

    # Demonstrate multiplication by zero (vanishing of summation structure)
    box_vanished = box_quantity * 0
    crate_vanished = crate_quantity * 0
    print(f"\nAfter multiplication by zero (vanishing of summation structure):")
    print(f"Box:   {box_vanished}")    # Should be 5u - preserving the vanished
    structure
    print(f"Crate: {crate_vanished}")  # Should be 3u - preserving the vanished
    structure

    # Add the vanished quantities (combining our structural information)
    total_vanished = CompleteNumber(
        0, 0,
        box_vanished._u_real + crate_vanished._u_real,
        box_vanished._u_imag + crate_vanished._u_imag
    )
    print(f"\nCombined vanished structures:")
    print(f"Total: {total_vanished}")  # Should be 8u

    # Recover the original total through division by zero
    try:
        total = total_vanished / 0
        print(f"\nRecovered quantity from vanished structure:")
        print(f"Total: {total}")  # Should be 8
    except ZeroDivisionError as e:
```

```
        print(f"Error recovering total: {e}")

    return box_vanished, crate_vanished, total_vanished

def test_complete_number_system():
    """
    Comprehensive test suite for the complete number system implementation.
    """
    # First run multiplication tests
    print("\n=== Complete Number System Tests ===")
    results_mult = test_basic_multiplication()
    results_div = test_basic_division()
    results_vanishing = demonstrate_vanishing_summation()

if __name__ == "__main__":
    test_complete_number_system()
```

Listing 2: Apple Counting with Complete Numbers

9.3 Output Analysis

The implementation demonstrates several key features of complete numbers:

1. **Information Preservation**: The original quantities (5 and 3) are preserved even after multiplication by zero.
2. **Algebraic Consistency**: The absorbed values maintain the expected algebraic properties, allowing operations like addition to be performed on the absorbed quantities.
3. **Recovery**: The original values can be recovered at any point using the `recover()` method.
4. **Zero-Connector Property**: The implementation maintains consistency with standard arithmetic through the Zero-Connector Property.

The output of this implementation exactly matches our theoretical predictions:

$$\begin{aligned} b_c \times 0u &= 0 + 5u \\ c_c \times 0u &= 0 + 3u \\ (5u + 3u)/0_z &= 8 \end{aligned}$$

This computational realization demonstrates that complete numbers are not merely a theoretical construct but can be practically implemented and used in computational systems where information preservation under zero multiplication is required.

9.4 Example Output

The implementation produces the following output, demonstrating the key features of complete numbers:

```
=== Complete Number System Tests ===

Basic Multiplication Tests:
Initial number: 3.0 + 4.0j

Case 1 - Whole number * 0:
(3.0 + 4.0j) * 0 = 3.0u + 4.0uj

Case 2 - Real component * 0:
(3.0 + 4.0j).real * 0 = 3.0u

Case 3 - Imaginary component * 0:
```

```
(3.0 + 4.0j).imag * 0 = 4.0uj

Verification of normal multiplication:
Initial number: 3.0 + 4.0j
(3.0 + 4.0j) * 1 = 3.0 + 4.0j
(3.0 + 4.0j).real * 1 = 3.0
(3.0 + 4.0j).imag * 1 = 4.0

Basic Division Tests:
Initial number: 3.0 + 4.0j

Case 1 - Complete absorbed number / 0:
(3.0u + 4.0uj) / 0 = 3.0 + 4.0j

Case 2 - Real component / 0:
(3.0 + 4.0j).real / 0 -> ZeroDivisionError: float division by zero

Case 3 - Imaginary component / 0:
(3.0 + 4.0j).imag / 0 -> ZeroDivisionError: float division by zero

Case 4 - Pure absorbed components / 0:
(3.0u + 4.0uj) / 0 = 3.0 + 4.0j

Vanishing Summation Structure Demonstration:
Initial quantities:
Box:   5.0
Crate: 3.0

After multiplication by zero (vanishing of summation structure):
Box:   5.0u
Crate: 3.0u

Combined vanished structures:
Total: 8.0u

Recovered quantity from vanished structure:
Total: 8.0
```

This output demonstrates several key features of the complete number system:

1. **Multiplication Behavior**: - Complete numbers preserve structural information when the summation operation vanishes - Individual components (real and imaginary) maintain their values in absorbed form - Normal multiplication operations remain unaffected

2. **Division Properties**: - Division by zero is well-defined for absorbed components - Regular division behaves as expected for non-absorbed components - Division by zero properly recovers original values from absorbed structures

3. **Vanishing Summation Demonstration**: - Shows how multiplication by zero represents the vanishing of the summation structure - Demonstrates preservation of structural information through the vanishing process - Illustrates recovery of original quantities from vanished structures - Proves the system's utility in tracking structural information through zero multiplication

The output verifies that our implementation correctly maintains the theoretical properties established in the paper while providing a practical computational framework for working with complete numbers. In particular, it shows how the system preserves the structural information that would otherwise be lost when the summation operation vanishes through multiplication by zero.

A Deconstruction of Multiplication by Zero

To provide deeper insight into the structural nature of multiplication by zero, we present a detailed deconstruction of multiplication as a series of sums. This analysis reinforces why our construction of $\mathbb{U}$ preserves essential structural information.

A.1 Expansion of Multiplication as a Series of Sums

Consider the expression $(a \times b)$ where $a, b \in \mathbb{Z}$. We can define multiplication as the number a repeatedly summed b times.

$$ab = \sum_{0}^{b} a, \quad ab \in \mathbb{Z} \tag{25}$$

Let's create a series of equations from the equation $(3 \times 3) = (3 \times 3)$ and apply the expansion. We will count down the a and b terms until they both reach 0 to produce a series of commutative equations.

$$\begin{aligned}(3 \times 3) + 0 &= (3 \times 3)\\ (3 \times 2) + 0 &= (2 \times 3)\\ (3 \times 1) + 0 &= (1 \times 3)\\ (3 \times 0) + 0 &= (0 \times 3)\end{aligned}$$

We can see an algebraic difference between the two expressions $a \times 0$ and $0 \times b$, if we consider the sum expansion for each of these equations.

$$\begin{aligned}(3 + 3 + 3) + 0 &= (3 + 3 + 3)\\ (3 + 3) + 0 &= (2 + 2 + 2)\\ (3) + 0 &= (1 + 1 + 1)\\ () + 0 &= (0 + 0 + 0)\end{aligned}$$

For $a \times 0$ we are left with 0 instances of a, which is the expression $()$. We are forced to sum an extra 0. However, for $0 \times b$ we are left with b instances of 0. The recursion on summing a simply vanishes because there is nothing to execute, or rather is performed 0 times.

A.2 The Series of Sums Produces a Singleton Set

To formulate this concept more precisely, we will convert our equations to set notation and consider the singleton set that results from the expansion of each.

$$\{ab\} = \{\sum_{0}^{b} a\}, \quad \{ab\} \subseteq \mathbb{Z} \wedge b \neq 0 \tag{26}$$

If we look at the same commutative equations that result from decaying the a and b term to 0 respectively, we seem to be left with a strange contradiction.

$$\begin{aligned}\{3 + 3 + 3\} &= \{3 + 3 + 3\}\\ \{3 + 3\} &= \{2 + 2 + 2\}\\ \{3\} &= \{1 + 1 + 1\}\\ \{\} &= \{0 + 0 + 0\}\\ \{\} &\neq \{0\}\end{aligned}$$

This is the most literal interpretation of multiplication, and this is what we mean by vanishing. When b instances of summing becomes 0, the term a vanishes to the empty set $\{\}$. We can fix our series of sums using the **Additive Identity Property** of integers, and it will then produce a consistent result.

$$\{ab + 0\} = \{0 + \sum_{0}^{b} a\}, \quad \{ab + 0\} \subseteq \mathbb{Z} \tag{27}$$

Here are the new expansions for our series of commutative equations.

$$\begin{aligned}\{3+3+3+0\} &= \{3+3+3\}\\ \{3+3+0\} &= \{2+2+2\}\\ \{3+0\} &= \{1+1+1\}\\ \{+0\} &= \{0+0+0\}\\ \{0\} &= \{0\}\end{aligned}$$

We are attempting to add 0 to nothing because $\{\}$ is the set which is produced from the sum implied by $a \times 0$. What we have conceptually done is the union of the two sets. Can this be justified? The goal is not to answer this, but track the information that has vanished.

$$\begin{aligned}\{\} \cup \{0\} &= \{0+0+0\}\\ \{0\} &= \{0\}\end{aligned}$$

Remark A.1. This deconstruction illuminates why the set of vanished numbers $\mathbb{U}$ is fundamentally concerned with preserving structural information rather than tracking physical quantities. The vanishing of the summation structure itself, rather than the elimination of values, is what our Zero-Connector Property and subsequent constructions capture.