Title: Formalization of Integer Arithmetic via Recursive Function Systems

Abstract: This document introduces a novel system for representing integers and fundamental arithmetic operations through a structured set of recursive functions. It defines positive and negative integer functions, their composition, and extensions to real numbers, while expressing operations such as addition, subtraction, multiplication, and division via formal recursive structures.

---

1. Definitions

1.1 Positive Integer Function (PIF)

Definition: Let $n \in \mathbb{Z}^+$. Then: $\mathrm{PIF}(n)(\mathrm{args} = 1) := n$ This function returns the integer n when called with a base argument of 1.

Diagram 1: Recursive Structure of Positive Integer Function (PIF)\ *Each triangle node represents a PIF application, where each input of 1 recursively feeds into the next, forming the positive integers incrementally.*

1.2 Negative Integer Function (NIF)

Definition: Let $k \in \mathbb{Z}^+$. Then the NIF is recursively defined as:

$$ \text{NIF}(1)(\text{args} = 1) := 0 \ \text{NIF}(2)(\text{args} = 1) := -1 \ \text{NIF}(k)(\text{args} = 1) := \text{NIF}(k-1)(\text{args} = 1) - 1 $$

Diagram 2: Recursive Structure of Negative Integer Function (NIF)\ *Each step subtracts 1 recursively from the previous value, generating negative integers downward from 0.*

1.3 Integer Domains

- $\mathbb{Z}^+$: Positive integers
- $\mathbb{Z}^-$: Negative integers
- $\mathbb{R}^+$: Positive real numbers
- $\mathbb{R}^-$: Negative real numbers

---

2. Arithmetic Construction

2.1 Addition

For two positive integers $a, b \in \mathbb{Z}^+$:

$$ +(a, b) := \text{PIF}(b)(\text{args}[1] = \text{PIF}(2)(\text{args}[1] = \text{PIF}(a)(\text{args} = 1), \text{args}[2] = 1)) $$

This recursive structure results in $a + b$.

Diagram 3: Nested Structure of Positive Integer Addition\ *Depicts the construction of addition using recursive application of PIF and intermediary argument passing.*

Diagram 5: Recursive Multiplication by Repeated Addition\ *Illustrates multiplication as a layered recursive addition process.*

2.4 Division

Given integers a, k , division is defined as:

$$ a = b \cdot k + c \quad \text{with remainder } c $$

Where b is the quotient and c is the remainder.

Diagram 6: Division with Quotient and Remainder\ *Breaks down division into a multiplication and remainder structure.*

---

3. Extensions

3.1 Squared Functions

- Positive Real Number Function (PRNF):

$$ \text{PRNF}(n) := \text{PIF}(n)(\text{args} = \text{PIF}(n)(\text{args} = 1)) $$

- Negative Real Number Function (NRNF):

$$ \text{NRNF}(k) := \text{NIF}(k)(\text{args} = \text{NIF}(k)(\text{args} = 1)) $$

3.2 Real Numbers

Constructed by composing integer functions recursively:

$$ \text{Real}(k) := \text{PRNF}(\text{NRNF}(k)) $$

---

4. Advanced Constructs

4.1 Rational Numbers

Represented as ordered pairs of integers (a, b) where $b \neq 0$

$$ \text{Rational}(a, b) := a \div b = \frac{a}{b} $$

Rational functions are implemented using the division structure and generalized to recursive numerator/denominator pairs.

4.2 Irrational Numbers

Approximated using recursive infinite processes, e.g.,

$$ \sqrt{2} \approx f(n) = 1 + \frac{1}{2 + \frac{1}{2 + \dots}} \quad (n \text{ terms}) $$

Recursive function constructs can represent converging infinite processes.

4.3 Complex Numbers

Defined using pairs of real recursive functions:

$$ \text{Complex}(a, b) := \text{Real}(a) + i\cdot\text{Real}(b) $$

Functions over complex numbers can be constructed via recursion over each real part.

4.4 Algebraic Expressions

Recursive expressions can represent polynomial and algebraic operations:

$$ P(x) = \sum_{i=0}^{n} a_i x^i \Rightarrow P(x) := \text{RecursiveSum}(i = 0 \text{ to } n, a_i * x^i) $$

Each term x^i can be defined through repeated multiplication.

4.5 Calculus Foundations

Recursive difference approximations can define limits and derivatives:

$$ \text{Limit}(f, x, h) := \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} $$

With f built using PIF or composed functions, we can derive basic calculus operations.

---

5. Conclusion

This system encodes arithmetic operations and numerical values in a recursive function framework, offering a visual and symbolic method for exploring elementary and advanced number structures. Its structure allows extension to rational, irrational, complex, and calculus-based mathematics, providing a potential foundation for symbolic logic engines or computational number theory.

Note: Further work may involve formal computability theory, symbolic logic foundations, or implementation in functional programming systems such as Haskell or Lisp.