Infinite Series Binary Tree (ISBT) in Functional Spaces (FS) (ISBT-FS)

Author: José Manuel Briceño Mendoza,

Juan José Briceño Mendoza,

Andrés de Jesús Araujo Briceño

Contact: josemanuelbric@gmail.com

Date: December 2025

Area: Mathematics - Number Theory

Abstract
This work introduces a new mathematical structure called the Infinite Series Binary Tree (ISBT), defined within a framework of Functional Spaces (FS). The system uses a root or "pivot" node N(x) that projects a recursive bifurcated structure based on the behavior of an input function. A projection algorithm is defined, mapping elements of a series to a path (Path in {0, 1}) in the tree.

1. Fundamental Definitions

1.1 **Node (X):** A Node (x) is defined as a unit of information containing a specific numerical set S = {n1, n2, ..., nk}.

1.2 **Geometric Pivot Node N(X):** Denoted as N(X), it acts as the initial generator of the system. It is the point of origin in the geometric space from which the initial numerical set is injected to start the recursion.

1.3 **Empty Nodes (~):** The set of Empty Nodes = (~) is defined as those Nodes where their [numerical set] is empty.

1.4 **Infinite Series Binary Tree (ISBT):** A Binary Tree in this context is a topological structure that bifurcates into two branches (child nodes) recursively towards infinity. This expansion is only interrupted by the appearance of Empty Nodes (~), which act as termination points or boundaries of the tree.

1.5 **Functional Spaces:** The Functional Space is defined as the mathematical environment where transformation functions are introduced. In this space, functions are not limited to processing values, but operate on the projection to generate the complete structure of the Infinite Series Binary Tree.

1.6 **The Projection Algorithm:** For each node, traversing from the projection pivot down the levels to infinity, the function must return a tuple that determines the system's evolution:
`f(x) -> (v, p)`
Where:

•`v`: Is the new transformed value (the state of the next node).
•`p`: Is the path discriminator or Path, such that `(p in {L, R})`, where L=0 and R=1.

2. Non-Conventional Notation for Operational Clarity
A shift symbolism is introduced to describe the dynamics of numerical sets within Functional Spaces:
2.1 **The Shift Operator (>m):**
- `>`: Symbol representing the shift or jump operator for the numerical set.
- `m`: Represents the quantity or magnitude of the jump applied to the set's elements.

2.2 **Node Definition: ({C}{V}):** The structure of a node in transition is defined by the combination of its domain and its jump:
- `{C}`: The base Numerical Set (e.g., N for Naturals, Z for Integers).
- `{V}`: The Jump Value or magnitude of displacement applied to the set.
Example:
If we define a node as (N6), under this notation it is broken down as:
`(N6) = {6, 12, 18, 24, ...}`
Where the set of Natural numbers has been transformed by a factor or jump of magnitude 6.

2.3 **Example of Projection Notation: (>9N6):** This expression combines displacement and location:
1. `>9`: A displacement of magnitude 9 is applied to the incoming numerical set.
2. `N6`: The result is projected onto level 6 of the tree: `(9, 15, 21, ...)`.

3. Functional Space Indexers
To allow access and manipulation of data within the ISBT-FS structure, indexing operators are defined to uniquely locate elements in the space:

3.1 **Operational Syntax:**
The alternative operator `||` is used to denote equivalence between the

global structure and the local reference (this.) of its Functional Space:
- **Level Access:** Tree(n) || this.Level(n). Locates the nodes at depth n.
- **Path Access:** Tree(n){Path} || this.Level(n){Path}. Level Access + Access by Path of a specific Level. Takes the parts following the bit path (L/R).
- **Element Access:** Tree(n){Path}[i] || this.Level(n){Path}[i]. Level Access + Path Access + Element Access. Locates the i-th element of the infinite series within a specific node.

4. Example in Pseudocode:

```
enum Path { L = 0, R = 1 }

tuple<int, Path> evenDiv2Oddplus1(int x) {

    if (x % 2 == 0)

        return tuple(x / 2, Path.R);

    else

        return tuple(x + 1, Path.L);

}

EspacioFuncional a = new EspacioFuncional(parEntre2ImparMas1);

a.projection(N1);

/*
                    N(N1)
                    /    \
              L(>1N2)    R(>2N2)
               /    \    /    \
          (~) (>2N2) (>1N2) (~)
*/
```

5. Conclusion

The ISBT-FS model allows visualizing the dynamics of infinite series not as critical lines, but as branched structures in a functional space. The shift notation (>mNk) facilitates understanding how complex numerical sets are transformed and positioned geometrically according to binary recursion rules.

The Demonstration of the Collatz Conjecture using Infinite Series Binary Trees (ISBT) in Functional Spaces (FS)

Author: José Manuel Briceño Mendoza, Juan José Briceño Mendoza, Andrés de Jesús Araujo Briceño
Contact: josemanuelbric@gmail.com
Date: December 2025

Area: Mathematics - Number Theory

Abstract
This work presents the demonstration of the Collatz Conjecture using the Infinite Series Binary Tree in Functional Spaces (ISBT-FS) structure. By projecting the Collatz function onto a binary decision tree, the mechanics by which numerical sets shift through recursion levels are demonstrated, establishing absolute convergence towards the (4, 2, 1) cycle through jump operators and set displacements that guarantee the collapse of every trajectory into the pivot node.

2. Formalization in ISBT-FS

We define the solution as a projection that generates a level structure where each node N represents a state of the transformed numerical set:
- **Level 0 (Pivot):** (n1)
- **Level 1:** [L(>2n6), R(n1)]
- **Level 2:** {L[(~), (>1n3)], R[(>2n6), (n1)]}
- **Level 3:** L{[(~), (~)], [(>2n18), (>2n3)]}, R{[(~), (>1n3)], [(>2n6), (n1)]}

3. Solution

3.1 **Solution for the Even Numbers Subset:**
For elements processed by the right branch (R), which fulfills this Level Symmetry Rule:
`this.Level(Current){R} = this.Level(Current - 1)`
This indicates that the successive division by 2 in the ISBT-FS simply shifts the node's position towards levels of lesser complexity without altering the base structure of the set.

3.2 **Solution for the Odd Numbers Subset:**
For elements processed by the Left branch (L), it fulfills this Level Symmetry Rule:
`this.Level(Current){L} = this.Level(Current + 1){RL}`

3.3 **Rule of Convergence to the Projection Node N(X):**
As a consequence of rules 3.1 and 3.2, the system generates a unidirectional flow:
- The process continues recursively until reaching Level 1, where all elements inevitably pass to the Projection Pivot Node N(X) (the fundamental state 1).

4: View of the ISBT-FS Projection

```
                              N(n1)
                         [L(>2n6), R(n1)]
                  {L[(~), (>1n3)], R[(>2n6), (n1)]}
     L{[(~), (~)], [(>2n18), (>2n3)]}, R{[(~), (>1n3)], [(>2n6), (n1)]}
 L{{{[(~) (~)][(~) (~)]}{[(~) (>1n9)][(>14n18) (>1n3)]}}}, R{{Level4-1}}
```

5: Conclusion.

The ISBT-FS model demonstrates that, although the tree structure expands infinitely, the dynamic flow of numerical sets is strictly contractive towards the system's core. Convergence is valid because the functional symmetry rules (3.1 and 3.2) force every element, regardless of its initial magnitude or branch depth, to execute a finite level shift until collapsing into the Pivot Node N(1). Thus, the fractal's growth does not represent dispersion, but the map of finite paths that guarantee the fundamental state is the sole universal attractor of the functional space.