

Bounded Recursion and Topological Discretization: An Isomorphism Between Continuous Streams and Finite Lists

Jiaxuan Yang

Monash University

February 2026

Abstract

In Domain Theory and Computable Analysis, real numbers are typically modeled as infinite streams of digits (or Cauchy sequences). We investigate the topological consequences of imposing a strict **Resource Bound** (or "Gas Limit") on the evaluation of such continuous types. We define a *Bounded Recursive Operator* acting on a Hilbert space of functions. We conjecture that under a finite "propagation cost" constraint, the domain of definable continuous functions becomes isomorphic to a discrete simplicial complex (a finite graph structure), effectively reducing the uncountable type \mathbb{R} to a discrete inductive type.

1 Introduction: The Cost of Continuity

In Constructive Mathematics, a real number $x \in \mathbb{R}$ is not a static value but a process (a Turing machine or Lambda term) that produces digits. Let $S = \text{Stream}(\Sigma)$ be the domain of infinite streams over an alphabet Σ .

A fundamental problem in Exact Real Arithmetic is the **Unbounded Recursion** problem: determining the equality of two computable reals is undecidable because it may require inspecting infinite digits.

We propose a regularization based on **Step-Indexing**. We introduce a "Cost Function" \mathcal{C} analogous to a metric derivative. If the cost of computing the next approximation step exceeds a global bound Ω , the computation must halt (or approximate).

2 Formal Setup: Operator Theoretic Semantics

Let \mathcal{F} be a function space (e.g., L^2). Consider a recursive operator $T : \mathcal{F} \rightarrow \mathcal{F}$ defined by a fixed-point equation $f = T(f)$. In Lambda Calculus, this is expressed via the Y-combinator:

$$f = \mathbf{Y}(\lambda g. \dots) \tag{1}$$

2.1 The Propagation Bound Constraint

We define the **Computational Propagation Speed** as the rate at which information (dependencies) spreads through the domain during recursion. Let \mathcal{L} be the "Lie Derivative" of the computation, representing the cost of state updates. We impose:

$$\sup \frac{\|\mathcal{L}_T f\|}{\|f\|} \leq C_{max} \quad (2)$$

where C_{max} is the maximum "gas limit" per reduction step.

3 The Discretization Conjecture

Standard Domain Theory (Scott) posits that continuous domains are limits of finite posets. We argue a stronger result under resource bounds.

Conjecture 1 (Type Isomorphism). *Let $\mathcal{D}_{C_{max}}$ be the subspace of continuous functions definable under the resource bound C_{max} . There exists a discrete Graph Laplacian operator L_G on a finite lattice $G = (V, E)$ such that:*

$$\mathcal{D}_{C_{max}} \cong \text{List}(\mathbb{C}) \quad (\text{Modulo Sampling}) \quad (3)$$

*This implies that a resource-bounded continuous stream is **isomorphic** to a finite list.*

4 Algorithmic Interpretation (Lisp/Scheme)

Consider a continuous signal represented as a lazy stream. In standard analysis, we assume we can '(stream-cdr)' infinitely.

Listing 1: The Continuous Ideal (Unbounded)

```
;; A standard definition of a continuous process
(define (continuous-evolution state)
  (stream-cons
    state
    (continuous-evolution (apply-operator state))))
;; This allows infinite recursion depth.
```

Under our constraint, the evaluator monitors the "computational cost" (norm of the derivative). When the cost exceeds C_{max} , the stream is forced to terminate (or loop), effectively collapsing the type from Stream to List.

Listing 2: The Bounded Reality (Discretized)

```
(define (bounded-evolution state gas)
  (let ((cost (measure-complexity state)))
    (if (> cost gas)
        '() ;; "Collapse" to Nil (End of List)
```

```
(cons
  state
  (bounded-evolution (next state) gas))))
;; The result is a finite List, not an infinite Stream.
```

5 Conclusion

We suggest that "Continuity" in computable analysis is an artifact of assuming unbounded resources. Once a strict complexity bound (spectral cutoff) is introduced to the fixed-point operator, the underlying topological space spontaneously discretizes. The *Manifold* becomes a *Graph*; the *Stream* becomes a *List*.