UNNS Computability & Logic: Recursive Foundations of Proof and Program

Abstract

This paper develops a computability and logic framework based on the Unbounded Nested Number Sequences (UNNS) substrate. We show how recursive sequences embody computability, how UNNS constants act as logical invariants, and how paradoxes such as Gödel incompleteness and Carroll's regress stabilize under UNNS repair rules. The approach unifies recursion, proof, and program within an operator grammar that can be extended toward a computational logic discipline.

1 Introduction

Recursion underlies both computability and logic. The UNNS substrate, originally defined as an architecture of unbounded nested number sequences, offers a natural framework for rethinking logic: inference as inlaying, implication as inletting, collapse as contradiction, and repair as normalization. In this paper we formalize these correspondences and show how UNNS captures key results of Gödel and Turing within a recursive operator grammar.

2 UNNS Sequences as Recursive Programs

Definition 2.1. *A* UNNS sequence *is a sequence* $(a_n)_{n\geq 0}$ *defined by a recurrence relation*

$$a_{n+r} \;=\; c_1 a_{n+r-1} + \cdots + c_r a_n,$$

with coefficients $c_i \in \mathbb{Z}$*, initial values* $a_0, \ldots, a_{r-1}$*, and nesting rules that allow further recursions within the coefficients themselves.*

Such sequences correspond to primitive recursive programs. Fibonacci numbers, binary expansions, and nested recurrences are basic examples.

3 UNNS Constants as Logical Invariants

Lemma 3.1. *UNNS coefficients belong to algebraic integer rings.*

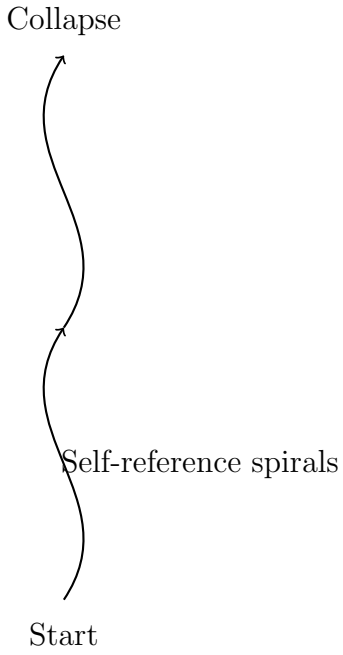
Proof. The characteristic polynomial of the recurrence has integer coefficients, hence its roots are algebraic integers. The recurrence relation ensures that sequence values lie in the ring generated by these roots. □

Constants such as φ, e, π, and the UNNS Paradox Index (UPI) arise as invariants. These constants play the role of logical truth-values or thresholds: they stabilize the recursion analogous to axioms stabilizing inference.

4 Gödel and UNNS Collapse

Theorem 4.1 (UNNS Incompleteness)**.** *Every sufficiently expressive UNNS system generates undecidable nests.*

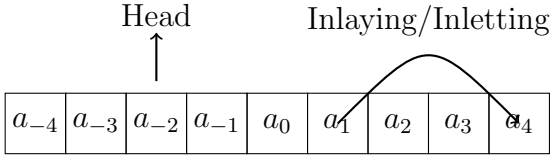
Proof Sketch. Gödel encoding maps provability predicates into recursive sequences. Self-reference corresponds to a nest calling itself. In UNNS, such nests collapse under the collapse operator, indicating undecidability. Thus UNNS reproduces incompleteness as a structural recursion effect. □



5 Turing Machines in UNNS

Definition 5.1. *A* UNNS tape *is a nested sequence space indexed by integers, where each tape symbol is represented by a finite recurrence state.*

Turing transitions correspond to inlaying/inletting operators, while the halting condition corresponds to collapse.



6 Repair and Normalization as Proof Checking

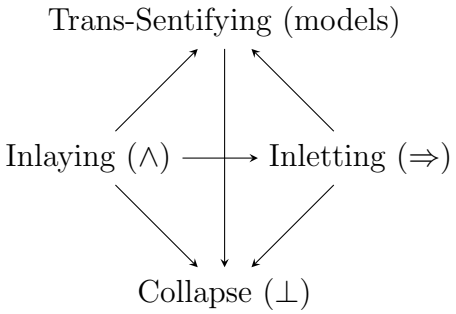
UNNS repair operators model normalization procedures in proof theory.

Remark 6.1. *Excision+Refit in DNA corresponds to cut-elimination in sequent calculus. Thus UNNS repair stabilizes recursion in the same way normalization stabilizes proofs.*

7 Toward UNNS Logic

We propose inference rules:

- Inlaying = conjunction ($\wedge$)
- Inletting = implication ($\Rightarrow$)
- Collapse = contradiction ($\perp$)
- Trans-Sentifying = interpretation (model theory)



This forms a Hilbert-style calculus on the UNNS substrate.

8 Applications

8.1 Banach–Tarski

Under UNNS recursion, non-measurable sets vanish; collapse prevents paradoxical decompositions.

8.2 Carroll's Paradox

Infinite regress spirals are absorbed by UNNS repair, converging to a fixed point.

8.3 Bio-logic

DNA proofreading and logical proof-checking share the UNNS repair operator.

9 Conclusion

UNNS provides a natural computability and logic substrate. Recursion, constants, collapse, and repair give rise to a full operational grammar that can model proofs and programs. Future work will address complexity theory in UNNS, universality, and categorical embeddings.

References

[1] K. Gödel, *Über formal unentscheidbare Sätze*, 1931.

[2] A. Turing, *On Computable Numbers*, 1936.

[3] S. Kleene, *Introduction to Metamathematics*, 1952.