Article

Not peer-reviewed version

Diagonal Arguments and Infinite Dependencies: Analyzing Classical Undecidability and Universality Under Finite Resource Constraints

Rithvik Sreekantham *

Posted Date: 27 October 2025

doi: 10.20944/preprints202510.2040.v1

Keywords: diagonal arguments; finite resource constraints; Cantor's uncountability; Gödel's incompleteness; Turing's halting problem; computational universality; decidability; bounded formal systems; infinitary idealizations; philosophy of mathematics; constructive mathematics; algorithmic enumerability; physical realizability

 doi:10.20944/preprints202510.2040.v1

Article

Diagonal Arguments and Infinite Dependencies: Analyzing Classical Undecidability and Universality Under Finite Resource Constraints

Rithvik Sreekantham [1,2]

[1] Quantum Science and Technology Laboratory, Physical Research Laboratory, Navrangpura, Ahmedabad 380009, India; rithvik_ks@iitgn.ac.in
[2] Indian Institute of Technology Gandhinagar, Palaj, Gandhinagar 382055, India

Abstract

This paper examines classical diagonal-based results (Cantor's uncountability, Gödel's incompleteness, Turing's halting problem, and computational universality) through a finite-resource lens. We analyze the diagonal pattern and its dependence on completed enumerations and on unbounded time, space, and precision, then formalize a finite framework $S(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ with integer bounds on time, memory, numerical precision, and symbolic length, and analyze each result within this framework. Within this setting: (i) finite-decimal reals admit explicit enumeration via constant-time bijections; (ii) for formal systems, when bounds are chosen adequate for the system under study, formulas and proofs are finitely enumerable and provability is decidable (complete within bounds); (iii) for the halting problem, adequacy (time beyond the finite-configuration threshold) yields a definitive HALTS/LOOP decision for every machine-input pair, whereas without adequacy the same procedure provides a sound bounded classification (HALTS/TIMEOUT); and (iv) no machine operating under fixed finite bounds is universal in the classical sense. These results show how classical results depend on infinite idealizations and exhibit different behavior under explicit finite resource constraints.

Keywords: diagonal arguments; finite resource constraints; Cantor's uncountability; Gödel's incom-pleteness; turing's halting problem; computational universality; decidability; bounded formal systems; infinitary idealizations; philosophy of mathematics; constructive mathematics; algorithmic enumer-ability; physical realizability

1. Introduction

The diagonal argument, first introduced by Georg Cantor in 1891 to prove the uncountability of real numbers, has become one of the most influential proof techniques in mathematical logic and computer science [1]. Its power extends far beyond set theory: Gödel employed diagonal methods in his incompleteness theorems [2], and Turing used similar reasoning to establish the undecidability of the halting problem [3], laying the foundation for modern computability theory and what became known as the Church-Turing thesis [4].

This paper examines a fundamental question: **How do these foundational results behave when analyzed within finite resource constraints rather than assuming completed infinite processes?** Classical computability theory provides a framework where machines operate with unbounded resources, while practical computation necessarily involves finite time, space, and precision constraints. While Meyer has critiqued diagonal arguments from a meta-linguistic perspective [5], our approach analyzes how diagonal constructions that rely on infinite enumeration procedures exhibit different properties under explicit finite resource constraints.

Our investigation is motivated by recent work establishing explicit bijections between finite-decimal real numbers and natural numbers [6]. This research demonstrated that the set of finite-decimal real numbers—precisely those numbers representable in scientific computation—admits complete

 doi:10.20944/preprints202510.2040.v1

enumeration through constructive algorithms with constant-time operations [7]. This concrete result provides a foundation for examining how diagonal methods behave differently in finite versus infinite domains, and how computational domains with explicit resource bounds exhibit distinct mathematical properties.

The constructive mathematics tradition, from Kronecker through Brouwer's intuitionism to Bishop's constructive analysis [8–10], emphasizes that mathematical objects should be explicitly constructible through finite means. This philosophical stance, which has historical roots in various algorithmic mathematical traditions (see Section 9), motivates our analysis. Notably, even Hilbert maintained a finitist stance, declaring that "the infinite is nowhere to be found in reality" and advocating for finitary methods in his consistency program [11].

Structure of This Investigation: We systematically analyze classical diagonal arguments by exposing their infinite dependencies. Section 2 analyzes the common diagonal structure underlying impossibility results, then Section 2.4 demonstrates how Cantor's diagonal argument breaks down when restricted to finite-decimal real numbers. Section 3 examines Gödel's incompleteness theorems, Section 4 investigates Turing's halting problem proof, and Section 5 analyzes classical computational universality claims. Section 6 provides a comprehensive constructive critique establishing criteria for finite mathematical objects. Section 7 develops a unified finite resource framework with explicit computational bounds $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ that fundamentally alters the landscape of mathematical decidability and completeness. Throughout, we distinguish two regimes: for formal systems, we choose bounds *adequate for the system under study*, which yields constructive decidability; for the halting problem, adequacy enables a definitive HALTS/LOOP decision, whereas without adequacy our bounded analysis reports HALTS/TIMEOUT. Section 9 provides historical context for algorithmic and constructive approaches to mathematics. Finally, Section 8 synthesizes these results.

2. The Diagonal Method: Infinite Assumptions and Finite Limitations

The diagonal argument, first introduced by Georg Cantor in 1891 and subsequently employed by Gödel and Turing in their foundational impossibility results, represents one of the most powerful proof techniques in mathematical logic. However, this technique depends fundamentally on infinite mathematical assumptions that become problematic when we adopt a constructive mathematical framework restricted to finite computational resources [1–3].

This section provides a critical examination of the diagonal method's dependence on infinite completeness assumptions and analyzes how this method behaves when subjected to finite constraints. Our analysis reveals that classical impossibility results may be artifacts of infinite idealizations rather than inherent limitations of finite computation.

2.1. The Common Diagonal Structure

The diagonal method exhibits a consistent logical structure across its various applications. Understanding this pattern is crucial for analyzing how finite constraints affect each application.

Definition 1 (Abstract Diagonal Construction)**.** *Let $\mathcal{S}$ be a collection of objects and suppose there exists an enumeration $\{s_1, s_2, s_3, \ldots\}$ of elements in $\mathcal{S}$. A diagonal construction produces an object d such that for each $i \in \mathbb{N}$, the object d differs from s_i in a systematic way that depends on the index i.*

This abstract structure underlies various mathematical applications, with Cantor's argument being the most fundamental instance. The diagonal pattern creates objects that systematically differ from every element in a proposed enumeration, leading to contradictions that appear to establish impossibility results.

2.2. Cantor's Diagonal Argument: The Infinite Foundation

Theorem 1 (Cantor's Diagonal Argument - Classical Version)**.** *The set of infinite binary sequences $\{0,1\}^{\mathbb{N}}$ is uncountable.*

 doi:10.20944/preprints202510.2040.v1

Classical Proof. Assume, for contradiction, that $\{0,1\}^{\mathbb{N}}$ is countable. Then there exists an enumeration:

$$s_1 = (a_{11}, a_{12}, a_{13}, a_{14}, \ldots) \tag{1}$$
$$s_2 = (a_{21}, a_{22}, a_{23}, a_{24}, \ldots) \tag{2}$$
$$s_3 = (a_{31}, a_{32}, a_{33}, a_{34}, \ldots) \tag{3}$$
$$\vdots \tag{4}$$

Construct the diagonal sequence $d = (d_1, d_2, d_3, \ldots)$ where:

$$d_i = \begin{cases} 0 & \text{if } a_{ii} = 1 \\ 1 & \text{if } a_{ii} = 0 \end{cases}$$

By construction, d differs from s_n in position n for every $n \in \mathbb{N}$, so d cannot appear in the enumeration, yielding a contradiction. □

Critical Infinite Dependencies:

1. **Completed infinite sequences**: The argument assumes infinite sequences exist as completed mathematical objects
2. **Infinite enumeration completeness**: The enumeration $\{s_1, s_2, \ldots\}$ is assumed to be a completed infinite collection
3. **Universal quantification over infinity**: The phrase "for every $n \in \mathbb{N}$" appeals to actual infinity
4. **Diagonal construction completeness**: The construction of d requires completing an infinite process

2.3. The General Pattern and Its Infinite Dependencies

The power of the diagonal method lies in its ability to demonstrate that certain mathematical objects cannot be enumerated or systematically listed. However, this power depends critically on several infinite assumptions that become problematic under finite constraints. Unlike foundational critiques that focus on meta-linguistic representability issues with Gödel numbering functions [5], our analysis demonstrates that diagonal arguments fail under finite resource constraints due to their fundamental dependence on completed infinite enumerations:

1. **Infinite enumeration assumption**: The method assumes we can meaningfully discuss "all possible" objects of a given type arranged in an infinite sequence
2. **Completed infinity**: The diagonal construction treats infinite processes as completed mathematical objects rather than ongoing procedures
3. **Universal quantification**: Arguments rely on statements that hold "for all n" where n ranges over infinite sets
4. **Non-constructive existence**: The diagonal object is proven to exist through contradiction rather than explicit construction

2.4. Failure of Cantor's Argument for Finite Decimals

When restricted to finite-decimal numbers, Cantor's diagonal argument breaks down completely:

Proposition 1 (Finite-Decimal Real Numbers are Enumerable)**.** *The set of finite-decimal real numbers $\mathbb{R}_{\text{finite-decimal}} = \{r \in \mathbb{R} : r \text{ has terminating decimal representation}\}$ is countably infinite and admits explicit enumeration with a bijection to natural numbers.*

Proof Outline. The complete bijection construction is established in [6,7]. We summarize the key components:

 doi:10.20944/preprints202510.2040.v1

Canonical Representation: Every finite-decimal real number r admits a unique canonical representation $(\text{sign}, N_1, N_2, N_3) \in \{-1, +1\} \times \mathbb{N}_0^3$ where the canonical form eliminates representational ambiguity:

- No trailing zeros: N_3 contains no trailing zeros unless $N_3 = 0$
- Pure integer constraint: If $N_3 = 0$, then $N_2 = 0$
- Unique representation: 3.14 and 3.1400 both map to $(+1, 3, 0, 14)$

Enumeration Strategy: Order 4-tuples by information complexity $K = N_1 + N_2 + N_3$, with lexicographic ordering within each complexity level. The complexity-level cardinalities are:

$$C(K) = \begin{cases} 1 & \text{if } K = 0 \\ K(K+1) + 2 & \text{if } K > 0 \end{cases}$$

Bijection Properties:

1. **Injectivity**: Distinct finite-decimal reals have distinct canonical 4-tuples
2. **Surjectivity**: Every natural number maps to a unique 4-tuple through systematic enumeration
3. **Constant-Time**: Both $f(r)$ and $f^{-1}(n)$ execute in O(1) using closed-form formulas

Therefore, $\mathbb{R}_{\text{finite-decimal}}$ is countably infinite with explicit algorithmic enumeration [6,7]. □

Corollary 1 (Diagonal Construction Failure)**.** *When applied to finite-decimal real numbers, Cantor's diagonal argument fails because:*

1. *The diagonal construction produces d where d_i differs from the i-th digit of r_i*
2. ***Case 1**: If $d \in \mathbb{R}_{\text{finite-decimal}}$, then $d = r_k$ for some k (bijection completeness)*
3. *This requires $d_k \neq (r_k)_k$, but $d = r_k$ implies $d_k = (r_k)_k$ (contradiction)*
4. ***Case 2**: If $d \notin \mathbb{R}_{\text{finite-decimal}}$, then d lies outside our domain*

The diagonal method therefore either yields internal contradictions or produces numbers beyond the finite-decimal domain, demonstrating that uncountability depends on infinite decimal representations.

Constructive Significance: This explicit enumeration algorithm demonstrates that the set of numbers appearing in scientific computation and digital representation—precisely the finite-decimal real numbers—admits complete algorithmic enumeration. Unlike classical diagonal arguments that appeal to completed infinities, our bijection provides constructive proof that computational real numbers form a countable set [6,7], with direct implications for numerical analysis and scientific computing where all calculations necessarily involve finite-precision arithmetic.

3. Classical Gödel Construction and Its Infinite Dependencies

The previous section demonstrated how Cantor's diagonal argument depends fundamentally on infinite decimal expansions. We now examine Gödel's incompleteness theorems, which employ a remarkably similar diagonal construction in the domain of formal logic. Gödel's proof, developed in 1931, predates Turing's 1936 work and provides the logical foundation for understanding how diagonal arguments operate in formal systems.

Theorem 2 (Gödel's First Incompleteness Theorem - Classical Form)**.** *In any consistent formal system $\mathcal{F}$ capable of expressing basic arithmetic, there exists a statement G such that neither G nor $\neg G$ is provable in $\mathcal{F}$.*

Explicit Diagonal Construction in Gödel's Proof:

The diagonal structure in Gödel's incompleteness theorem becomes clear when we represent it as an infinite table, exactly parallel to Cantor's diagonal argument [2,12,13].

Step 1: The Diagonal Table Setup

Consider an infinite table where:

 doi:10.20944/preprints202510.2040.v1

- Rows represent all possible formulas $\phi_1, \phi_2, \phi_3, \ldots$ (enumerated by Gödel numbers)
- Columns represent all possible formulas $\psi_1, \psi_2, \psi_3, \ldots$
- Entry (i, j) contains: "Does ϕ_i prove ψ_j?"

	ψ_1	ψ_2	ψ_3	$\cdots$
ϕ_1	$\phi_1 \vdash \psi_1?$	$\phi_1 \vdash \psi_2?$	$\phi_1 \vdash \psi_3?$	$\cdots$
ϕ_2	$\phi_2 \vdash \psi_1?$	$\phi_2 \vdash \psi_2?$	$\phi_2 \vdash \psi_3?$	$\cdots$
ϕ_3	$\phi_3 \vdash \psi_1?$	$\phi_3 \vdash \psi_2?$	$\phi_3 \vdash \psi_3?$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Step 2: Focus on the Diagonal

The diagonal entries (i, i) have special significance:

- Position $(1, 1)$: Does ϕ_1 prove ϕ_1? (Self-provability)
- Position $(2, 2)$: Does ϕ_2 prove ϕ_2? (Self-provability)
- Position (k, k): Does ϕ_k prove ϕ_k? (Self-provability)

Each diagonal entry represents whether a formula proves itself.

Step 3: The Diagonal Construction

Gödel constructs a new formula G that systematically differs from every diagonal entry:

$$G \equiv \forall k \left[\neg (\phi_k \vdash \phi_k) \right] \tag{5}$$

In words: G says "No formula proves itself."

The Self-Reference Mechanism: Here's the crucial insight - when G is assigned a Gödel number and becomes ϕ_m in our enumeration, it creates a self-referential loop [14,15]:

- G makes a claim about ALL formulas (including itself when it becomes ϕ_m)
- Specifically, G says "ϕ_m does not prove ϕ_m" (among other things)
- But G IS ϕ_m, so G is essentially saying "I do not prove myself"
- This transforms the general statement into a self-referential assertion about its own provability

This is the diagonal move: G contradicts what appears on the diagonal at every position simultaneously.

Step 4: The Diagonal Contradiction

1. **Assumption**: Suppose G is provable
2. **Enumeration**: Then G appears somewhere in our list, say $G = \phi_m$ for some m
3. **Diagonal Analysis**: Consider position (m, m) on the diagonal:
   - G (which is ϕ_m) says: "No formula proves itself"
   - In particular: "ϕ_m does NOT prove ϕ_m"
   - Now, if G is provable, this means the formal system can derive G as a theorem
   - Since $G = \phi_m$, this means the system can derive ϕ_m as a theorem
   - But what does ϕ_m (which IS G) assert? It asserts "ϕ_m is not provable"
   - So we have: The system proves ϕ_m, but ϕ_m itself says "ϕ_m is not provable"
   - This creates a direct contradiction: ϕ_m is simultaneously provable and asserts its own unprovability
4. **Key Insight**: The diagonal entry (m, m) asks "Does ϕ_m prove ϕ_m?" But we must be careful about what this means:
   - The question is NOT whether ϕ_m can prove itself (self-reference)
   - The question IS whether the formal system can prove the statement ϕ_m
   - Since ϕ_m IS the statement G, this becomes: "Can the system prove G?"
   - G asserts "No formula proves itself" - specifically, "ϕ_m is not provable"

 doi:10.20944/preprints202510.2040.v1

The Contradiction Analysis:

- **Case 1 - If G is provable**: The system proves ϕ_m (since $G = \phi_m$), but ϕ_m says "ϕ_m is not provable" - direct contradiction
- **Case 2 - If G is not provable**: Then G is true (since it correctly states that ϕ_m is not provable), giving us a true but unprovable statement

5. **Conclusion**: Since Case 1 leads to contradiction, we must have Case 2: G cannot be provable. Therefore, G is a true statement that the formal system cannot prove, establishing incompleteness.

Step 5: Parallel to Cantor's Argument

Cantor's Diagonal	Gödel's Diagonal
Binary sequences $s_1, s_2, s_3, \ldots$	Formulas $\phi_1, \phi_2, \phi_3, \ldots$
Entry (i, j) = j-th bit of s_i	Entry (i, j) = "$\phi_i \vdash \phi_j$?"
Diagonal: bit i of sequence s_i	Diagonal: "$\phi_i \vdash \phi_i$?" (self-provability)
New sequence differs at position i	G differs from ϕ_i on self-provability
Contradiction: new sequence $\notin$ enumeration	Contradiction: $G \notin$ provable statements

Infinite Dependencies: Gödel's construction requires:

1. **Infinite formula enumeration**: All formulas in the formal system enumerated as $\phi_1, \phi_2, \phi_3, \ldots$ in a completed infinite sequence
2. **Unbounded Gödel numbering**: Effective encoding of formulas as natural numbers without complexity bounds or size restrictions
3. **Infinite provability domain**: Diagonal self-reference operates over the infinite domain of all possible formulas and their provability relationships

4. The Halting Problem: Diagonal Construction and Infinite Dependencies

Turing's 1936 proof of the undecidability of the halting problem employs diagonal argumentation techniques parallel to those developed by Gödel in 1931. While Gödel's construction operates in the domain of formal logical systems, Turing's diagonal argument targets the space of computable functions themselves. This connection reveals the fundamental unity underlying classical impossibility results—all depend on the same infinite diagonal construction pattern we identified in Sections 2 and 3.

4.1. Historical Context: From Turing's Circle-Free Machines to Davis's Halting Problem

Understanding the historical development helps clarify the infinite assumptions underlying both formulations. Turing's seminal 1936 work "On Computable Numbers, with an Application to the Entscheidungsproblem" established fundamental concepts about what he termed "a-machines" [3], while the modern presentation reflects Martin Davis's influential reformulation. The remarkable 1936 emergence of computability theory through independent parallel work by Church [16], Kleene [17], and Post [18] created a unified mathematical framework for understanding effective computation, later formalized as the Church-Turing thesis [4].

Turing's Original Framework (1936): Turing distinguished between two classes of computing machines [3,19]:

- **Circle-free machines**: Those that continue to produce output indefinitely
- **Circular machines**: Those that eventually enter a loop and "never write down more than a finite number of symbols"

Turing's central undecidability result concerned determining whether a machine is circle-free, proved using diagonal argumentation applied to computable sequences.

Davis's Modern Formulation (1958): Martin Davis coined the term "halting problem" and provided its explicit modern formulation in his 1958 book "Computability and Unsolvability" [20] [21].

 doi:10.20944/preprints202510.2040.v1

While the underlying mathematical undecidability result is Turing's, the specific presentation we use today comes from Davis.

Diagonal Structure in Both Formulations: Whether we consider Turing's circle-free problem or Davis's halting problem, both rely fundamentally on diagonal argumentation and infinite assumptions. Our critique applies equally to both historical and modern presentations.

4.2. The Diagonal Construction Explicit

The proof's diagonal nature becomes clear when we consider the enumeration of all Turing machines and construct what we call the *diagonal halting function*.

Step 1: Enumeration. Let $M_0, M_1, M_2, \ldots$ be an enumeration of all Turing machines. We can visualize their halting behavior as an infinite table T where $T[i,j] = 1$ if machine M_i halts on input j, and $T[i,j] = 0$ otherwise:

	0	1	2	3	$\cdots$
M_0	$T[0,0]$	$T[0,1]$	$T[0,2]$	$T[0,3]$	$\cdots$
M_1	$T[1,0]$	$T[1,1]$	$T[1,2]$	$T[1,3]$	$\cdots$
M_2	$T[2,0]$	$T[2,1]$	$T[2,2]$	$T[2,3]$	$\cdots$
M_3	$T[3,0]$	$T[3,1]$	$T[3,2]$	$T[3,3]$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Step 2: The Diagonal Construction. The diagonal argument constructs a specific function that exposes the contradiction:

$$D(i) = 1 - T[i,i]$$

This function D is designed to differ from every row of the table T along the diagonal: for any machine M_k, we have $D(k) \neq T[k,k]$, ensuring that D cannot represent the halting behavior of any machine M_k in our enumeration.

Step 3: The Contradiction. The proof proceeds by contradiction:

Classical Proof - Diagonal Structure Explicit. Assumption: Suppose there exists a Turing machine H that decides the halting problem for all Turing machines.

Construction of the Diagonal Machine: Since H can determine $T[i,i]$ for any i, we can construct a specific machine D that implements the diagonal function:

Algorithm 1 Diagonal Machine D

```
Input: Natural number i
Use machine H to compute T[i,i] (whether M_i halts on input i)
if T[i,i] = 1 then                          ▷ M_i halts on input i
    Enter infinite loop                     ▷ So D does NOT halt on input i
else                                        ▷ T[i,i] = 0, meaning M_i does not halt on input i
    Halt immediately                        ▷ So D DOES halt on input i
```

Key Property: By construction, D computes the diagonal function $D(i) = 1 - T[i,i]$ for every input i.

The Contradiction: Since D is a Turing machine, it must appear somewhere in our enumeration—say, $D = M_k$ for some index k. But this leads to a logical impossibility:

- **Row k represents D's behavior:** Entry $T[k,k]$ should tell us whether D halts on input k
- **Diagonal property forces contradiction:** By definition, $D(k) = 1 - T[k,k]$
  - If $T[k,k] = 1$ (meaning D halts on k), then $D(k) = 1 - 1 = 0$ (meaning D does NOT halt on k)
  - If $T[k,k] = 0$ (meaning D does not halt on k), then $D(k) = 1 - 0 = 1$ (meaning D DOES halt on k)
- **Logical impossibility:** D cannot both halt and not halt on the same input k

 doi:10.20944/preprints202510.2040.v1

Conclusion: This contradiction proves that our assumption was false—no halting-deciding machine H can exist. □

Parallel to Cantor's Argument: This proof structure exactly mirrors Cantor's diagonal argument:

Cantor's Diagonal Argument	Turing's Halting Problem
Enumerate all real numbers	Enumerate all Turing machines
Construct diagonal real number	Construct diagonal halting function
Diagonal differs from every listed real	Diagonal differs from every listed function
Contradiction: diagonal not in list	Contradiction: diagonal machine not decidable
Conclusion: Reals are uncountable	Conclusion: Halting problem undecidable

Both arguments use the same logical structure: assume complete enumeration is possible, construct a diagonal object that systematically differs from every enumerated object, derive contradiction, conclude the enumeration assumption was false.

Essential Infinite Dependencies: The halting problem proof exhibits the same infinite assumptions as Cantor's diagonal argument:

1. **Infinite machine enumeration**: All Turing machines enumerated as $M_1, M_2, M_3, \ldots$ in a completed infinite sequence
2. **Infinite halting matrix completeness**: The theoretical matrix $T[i, j]$ exists as a completed infinite mathematical object
3. **Universal quantification over infinity**: The diagonal function $D(i) = 1 - T[i, i]$ is defined "for every $i \in \mathbb{N}$"
4. **Infinite time allowance**: Machines may run arbitrarily long, with "infinite loops" as completed objects
5. **Unbounded computational resources**: No constraints on memory, program size, or computational complexity

This reveals that the halting problem's undecidability is not an intrinsic property of computation, but rather a consequence of allowing infinite resources and processes.

5. Classical Universality and Its Infinite Dependencies

The diagonal arguments in Cantor's uncountability proof (Section 2), Gödel's incompleteness theorems (Section 3), and Turing's halting problem (Section 4) all share a common requirement: they depend on the assumption that we can enumerate "all possible" objects of a given type. Classical computational universality makes a similar infinite assumption—that universal Turing machines can simulate "any computation." We now examine how this universality claim depends on analogous infinite idealizations, revealing a parallel pattern of dependencies that will prove equally problematic under finite resource constraints.

5.1. Classical Universality Claims and Their Infinite Dependencies

Both Turing's original 1936 universal machine construction and modern abstract universality claims fundamentally depend on infinite resource assumptions. While Turing provided a concrete construction for a specific universal machine, his theoretical framework was built on the same infinite idealizations that characterize modern computability theory [21,22].

5.1.1. Resource Requirements in Turing's Original Universal Machine

Turing's 1936 universal machine construction explicitly relied on infinite resource assumptions as foundational elements of his theoretical model:

1. **Unlimited Tape**: Turing's fundamental assumption was an infinite tape divided into "F-squares" (for computed results) and "E-squares" (for erasable working space) [3]

 doi:10.20944/preprints202510.2040.v1

2. **Unbounded Simulation Time**: No time limits on how long the universal machine could run while simulating another machine
3. **Arbitrary Machine Complexity**: The universal machine must accommodate encodings of machines of unlimited complexity
4. **Infinite Input Domains**: The system must handle inputs of arbitrary length

Under finite resource constraints $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$, Turing's original universal machine operates within modified parameters:

- The tape becomes finite: $|F\text{-squares}| + |E\text{-squares}| \leq S_{\max}$
- Simulation time is bounded: $\text{Time}(\text{simulation}) \leq T_{\max}$
- Machine encodings are length-limited: $|\langle M \rangle| \leq L_{\max}$

These constraints fundamentally alter the scope of simulatable machines and computable functions.

5.1.2. Historical Evolution of Universality Claims

Rather than representing a departure from finite foundations, the evolution of universality claims from Turing's 1936 work to modern recursion theory represents a consistent development within the framework of infinite resource assumptions. Both historical and contemporary formulations share the same fundamental dependencies on infinite idealizations.

Theoretical Systematization (1950s-1960s): Martin Davis's seminal work extended Turing's concrete construction into a systematic theory of universal machines as abstract computational objects [21]. Davis solved fundamental problems posed by McCarthy and Shannon about encoding efficiency, demonstrating how Turing's original infinite resource model could be systematized into broader theoretical frameworks. This development refined rather than transformed the infinite assumptions underlying universality.

Church-Turing Thesis Formalization: The philosophical principle that "anything computable by intuitive means is computable by Turing machine" became formally connected to technical claims about universal simulation. This formalization maintained Turing's original infinite resource assumptions—infinite tape, unbounded time, unlimited complexity—as essential features rather than optional idealizations. [4]

Recursion Theory Development: Post-1950s developments in recursion theory [23] systematized Turing's infinite resource model within abstract mathematical frameworks. Kleene's foundational 1952 work provided the mathematical framework for recursion theory, while subsequent developments made explicit the infinite dependencies that were already present in Turing's original 1936 construction.

This historical analysis reveals that both Turing's original work and subsequent developments consistently depend on infinite resource assumptions. The challenge posed by finite resource constraints applies equally to historical and contemporary formulations of computational universality.

6. A Constructive Critique of Classical Diagonal Arguments

From a constructive mathematical perspective, the classical arguments face several fundamental challenges:

Definition 2 (Constructive Mathematical Object)**.** *A mathematical object is constructive if and only if:*

1. *It can be explicitly constructed through a finite procedure*
2. *Its properties can be verified through finite computational steps*
3. *Its existence does not depend on non-constructive principles (like the axiom of choice or actual infinity)*

Under this definition, neither infinite decimal expansions nor infinite computations qualify as legitimate mathematical objects.

 doi:10.20944/preprints202510.2040.v1

Proposition 2 (Finite Decimal Sufficiency)**.** *Every number that appears in actual mathematical practice or scientific computation is representable as a finite decimal, and such finite-decimal real numbers admit explicit enumeration [6] as demonstrated in Section 2.4.*

Proof. Consider any number used in practical computation:

- Physical measurements have finite precision
- Computational representations use finite precision (IEEE 754, decimal arithmetic)
- Mathematical constants are approximated to finite precision for use
- Even "infinite" computations (like computing π) terminate at finite precision

Therefore, the set of practically meaningful numbers is contained within finite-decimal representations. □

6.1. Precision-Based Diagonal Construction: Logical Incoherence Under Finite Constraints

Central Question: Can diagonal constructions escape finite precision enumerations, regardless of computational resources available?

Definition 3 (Finite Precision Sets)**.** *Let $\mathbb{R}_{p,k}$ denote all numbers with exactly p digits after the radix point (fractional part) and k digits before the radix point (integer part) in a positional number system with radix r, including both positive and negative numbers. The set $\mathbb{R}_{p,k}$ contains exactly $2 \cdot r^{p+k}$ distinct numbers (accounting for positive and negative values) with total positions $n = p + k + 1$ (including one sign position plus $p + k$ digit positions).*

Theorem 3 (Fundamental Logical Barrier to Diagonal Construction)**.** *For any finite precision system $\mathbb{R}_{p,k}$, diagonal construction encounters an insurmountable logical barrier that cannot be overcome by increasing computational resources, memory capacity, or processing time.*

The Fundamental Barrier: *For positions $i > p + k + 1$ (where $p + k + 1$ is the total number of positions including sign), the diagonal instruction "$x_i \neq (d_i)_i$" becomes mathematically undefined since numbers in $\mathbb{R}_{p,k}$ possess no i-th position beyond position $p + k + 1$. This is not a computational limitation but a logical impossibility inherent to finite precision systems.*

Proof. Let $E_{p,k} = \{d_1, d_2, \ldots, d_{2 \cdot r^{p+k}}\}$ be a complete enumeration of $\mathbb{R}_{p,k}$.

Diagonal Construction Analysis:

- **Positions $i \leq p + k + 1$:** Construction proceeds as expected. Each d_i possesses an i-th position (sign or digit), enabling the definition $x_i \neq (d_i)_i$.
- **Positions $i > p + k + 1$: Insurmountable logical barrier.** Numbers in $\mathbb{R}_{p,k}$ mathematically lack any i-th position for $i > p + k + 1$. The diagonal instruction "$x_i \neq (d_i)_i$" becomes logically undefined—not merely computationally difficult, but mathematically meaningless.

Critical Insight: This barrier is absolute and cannot be circumvented by:

- Increasing computational power or memory
- Extending processing time or storage capacity
- Developing more sophisticated algorithms
- Applying any finite computational resources whatsoever

The limitation arises from the mathematical structure of finite precision systems themselves, not from computational constraints. No finite precision system $\mathbb{R}_{p,k}$, regardless of the values of p and k, can support diagonal construction beyond its intrinsic position boundary $p + k + 1$. □

Example 1 (Concrete Case: $\mathbb{R}_{2,1}$ in Decimal)**.** *Consider $\mathbb{R}_{2,1}$ with $p = 2$ fractional digits and $k = 1$ integer digit, containing $2 \cdot 10^{2+1} = 2000$ numbers:*

$$\{-9.99, -9.98, \ldots, -0.01, 0.00, 0.01, \ldots, 9.99\}$$

 doi:10.20944/preprints202510.2040.v1

Each number has exactly $p + k + 1 = 2 + 1 + 1 = 4$ positions total (sign + 3 digits). For diagonal construction, we examine the i-th position of the i-th number in our enumeration.

Diagonal Construction:

- *Position 1:* $d_1 = +0.00 \rightarrow (d_1)_1 = +$ *(sign)* $\rightarrow$ *Set* $x_1 = -$ ✓
- *Position 2:* $d_2 = +0.01 \rightarrow (d_2)_2 = 0$ *(integer digit)* $\rightarrow$ *Set* $x_2 = 1$ ✓
- *Position 3:* $d_3 = +0.02 \rightarrow (d_3)_3 = 0$ *(1st fractional digit)* $\rightarrow$ *Set* $x_3 = 1$ ✓
- *Position 4:* $d_4 = +0.03 \rightarrow (d_4)_4 = 2$ *(2nd fractional digit)* $\rightarrow$ *Set* $x_4 = 3$ ✓
- *Position 5:* $d_5 = +0.04 \rightarrow (d_5)_5 =$? ***Undefined! No 5th position exists.***

Breakdown Analysis: *Numbers in $\mathbb{R}_{2,1}$ possess exactly $p + k + 1 = 2 + 1 + 1 = 4$ total positions (sign plus digit positions) by mathematical definition. The diagonal construction becomes logically undefined at position 5, not because of insufficient computational resources, but because the mathematical objects themselves lack the required structural properties. Even with unlimited computational power, infinite memory, and arbitrary processing time, the construction cannot proceed beyond position 4—the barrier is mathematical, not computational.*

Remark 1 (Finite vs. Infinite Precision: The Fundamental Distinction)**.** *The distinction between finite and infinite precision systems reveals why classical diagonal arguments succeed in one domain but fail completely in the other:*

- ***Infinite precision systems:*** *Diagonal construction remains mathematically coherent indefinitely because there is always a "next digit position" available, enabling systematic escape from any proposed enumeration*
- ***Finite precision systems:*** *Diagonal construction encounters an absolute mathematical boundary at position $p + k + 1$, beyond which the required positions (sign or digit) simply do not exist. For any complete enumeration of all $2 \cdot r^{p+k}$ numbers in $\mathbb{R}_{p,k}$ (including both positive and negative values), the diagonal construction cannot produce a number with $(p + k + 1)$-position precision that differs from the enumeration—the mathematical structure of finite systems inherently prevents it.*

6.2. Implications for Classical Impossibility Results

Universal Failure of Diagonal Construction on Finite Enumerations: The breakdown of diagonal argumentation extends beyond finite precision real numbers to encompass all classical impossibility results. Whether we enumerate finite precision real numbers (our demonstration), finite sets of Gödel formulas under resource bounds $L_{\max}$, or finite sets of Turing machines under computational limits $(T_{\max}, S_{\max}, L_{\max})$, the same fundamental barrier emerges: *diagonal construction requires more objects than any finite enumeration contains*. For any finite enumeration of N objects, diagonal construction can proceed through at most N positions before exhausting the available objects to diagonalize against. This reveals that Gödel's incompleteness and Turing's undecidability encounter the identical structural limitation demonstrated for Cantor's uncountability—all diagonal arguments fail when applied to genuinely finite domains, regardless of the specific mathematical objects being enumerated.

The failure of diagonal construction in finite precision systems established in this section has profound implications for the classical impossibility theorems analyzed in Sections 3 and 4.

Corollary 2 (Collapse of Classical Impossibility Results Under Finite Constraints)**.** *Since Gödel's incompleteness theorems and Turing's undecidability of the halting problem both depend fundamentally on diagonal argumentation (as demonstrated in Sections 3 and 4), and diagonal construction encounters insurmountable logical barriers in finite precision systems (Theorem 3), the failure of diagonal construction in finite settings implies:*

1. ***Gödel's Incompleteness Disappears****: In formal systems with bounded formula length $L_{\max}$ and finite proof complexity, the diagonal formula G cannot be constructed outside the finite domain of expressible statements. Incompleteness transforms into finite completeness.*

 doi:10.20944/preprints202510.2040.v1

2. ***Turing's Undecidability Dissolves***: *For finite sets of Turing machines operating under resource bounds* $(T_{\max}, S_{\max}, L_{\max})$*, the halting problem becomes decidable through exhaustive enumeration of the finite machine-input space.*
3. ***Cantor's Uncountability Fails***: *As established in Section 2.4, finite-decimal real numbers admit explicit enumeration, contradicting classical uncountability results.*

The Fundamental Insight: Classical impossibility theorems are artifacts of infinite idealization rather than inherent mathematical limitations. When mathematical objects are constrained to finite, constructible domains—as required by any mechanically implementable system—the diagonal method loses its paradoxical power and impossibility results transform into finite decidability.

From Non-Constructive to Constructive Proof: The analysis in Sections 2–6 provides a *non-constructive* demonstration that classical diagonal arguments fail under finite constraints. However, to establish the positive claims about finite decidability and logical consistency, we require explicit constructive procedures and algorithms.

The subsequent framework will provide these constructive proofs by developing explicit finite resource models where:

- Every mathematical object admits algorithmic construction and verification
- All decision problems become computationally tractable through systematic enumeration
- Classical impossibility results are replaced by finite constructive procedures
- The relationship between logical systems and computational realizability is made precise

7. Finite Resource Framework for Mathematical Logic

Sections 2–5 have demonstrated that Cantor's diagonal argument, Gödel's incompleteness theorems, Turing's halting problem, and classical universality all depend on infinite idealizations. These results suggest the need for a unified finite resource framework that makes these dependencies explicit and shows systematically how classical impossibility results are transformed under constructive resource constraints.

7.1. Historical Foundations: From Leibniz's Dream to Hilbert's Program

The quest to mechanize mathematical reasoning represents one of the most persistent and ambitious projects in the history of logic and mathematics. This enterprise, spanning over three centuries, reveals a fundamental tension between the infinite aspirations of pure mathematics and the finite constraints of mechanical implementation.

Leibniz's Characteristica Universalis (1670s-1680s): Gottfried Wilhelm Leibniz envisioned the ultimate mechanization of human reasoning through his *characteristica universalis*—a universal symbolic language combined with a *calculus ratiocinator* for mechanical inference. In Leibniz's prophetic words: "Through this calculus, it is always possible to terminate that part of a controversy that can be determined from the data, by simply taking a pen, so that it will suffice for two debaters to say to each other: *Calculemus!* [Let us calculate!]" [24]. Crucially, Leibniz insisted that reasoning be reduced to purely **mechanical symbol manipulation**: "All truths can be demonstrated *solo calculo*, or solely by the manipulation of characters according to a certain form, without any labor of the imagination or effort of the mind, just as occurs in arithmetic and algebra." This mechanical requirement implicitly imposed finite constraints—physical symbols, finite manipulation steps, and bounded computational procedures—though Leibniz never formalized these limitations explicitly.

Symbolic Logic and Formalization (19th-Early 20th Century): The mechanization project gained momentum through systematic developments in symbolic logic. George Boole established algebraic frameworks for logical reasoning [25], while Gottlob Frege revolutionized logic with his *Begriffsschrift* (1879), introducing formal quantification theory and axiomatized predicate logic to reduce arithmetic to purely symbolic, rule-driven procedures [26]. Bertrand Russell and Alfred North Whitehead extended this vision in *Principia Mathematica* (1910-1913), developing comprehensive symbolic frameworks

 doi:10.20944/preprints202510.2040.v1

demonstrating that mathematics could be constructed from logic using explicit rules and mechanical proof procedures [27].

The Formalist Program (1890s-1930s): David Hilbert's program represented the most systematic attempt to realize Leibniz's vision through modern mathematical rigor. Hilbert's formalism sought to establish the consistency and completeness of mathematical systems using exclusively **finitary methods**—proof procedures that avoid infinite sets, non-constructive existence proofs, and other infinitary reasoning techniques [11].

The Computational Revolution (1930s): The emergence of computability theory through the independent work of Church [16], Kleene [17], Post [18], and Turing [3] transformed Leibniz's and Hilbert's intuitions into precise mathematical frameworks. Turing's analysis of mechanical computation provided the missing link: **logical systems derive their bounds from the requirements of mechanical implementation**, not from abstract mathematical constraints [4]. Hilbert and Ackermann's systematic development of theoretical logic provided the formal framework that enabled this mathematical mechanization [28].This historical progression from Leibniz's symbolic calculus through Hilbert's finitary methods to Turing's computational theory is comprehensively documented in Davis's authoritative account [29].

Gödel(1931) and Turing (1936): Gödel's incompleteness theorems [2] and Turing's undecidability of the halting problem [3] appeared to establish fundamental limitations in the mechanization project: that infinite logical systems cannot be mechanically decided or verified for consistency. These results were interpreted as demonstrating inherent barriers to complete formalization of mathematics and computation. Both Gödel's and Turing's diagonalizations operated over infinite domains of formulas, proofs, and computational resources, though the role of these infinite dependencies was not systematically examined at the time. Our analysis demonstrates (Corollary 2) that when these diagonal constructions are constrained to finite precision domains, they fail to produce valid counterexamples, revealing that classical impossibility results depend critically on infinite idealizations rather than representing fundamental mathematical limitations.

The Key Insight: The historical progression from Leibniz through Hilbert to Turing reveals that **logical systems inherit resource bounds from their computational realizability**. Pure mathematical logic, considered as abstract Platonic objects, admits infinite formulas, unbounded proofs, and unlimited quantification. However, the moment we require that logic be *mechanically implementable*—whether through Leibniz's symbolic calculus, Hilbert's finitary methods, or Turing's computational procedures—finite resource constraints become inevitable.

This historical analysis establishes the philosophical foundation for our finite resource framework: bounds are not arbitrary limitations imposed on logic, but necessary consequences of the requirement that mathematical reasoning be mechanically executable and practically implementable.

7.2. The Finite Resource Model

We now formalize the unified framework that applies to both computational and logical systems:

Definition 4 (Finite Resource System)**.** *A finite resource system $\mathcal{S}$ operates under explicit bounds $(T_{\max}, S_{\max}, P_{\max}, L_{\max}) \in \mathbb{N}^4$ where:*

Resource Parameters:

- $T_{\max} \in \mathbb{N}$*: Maximum computational steps, proof verification operations, or logical derivations*
- $S_{\max} \in \mathbb{N}$*: Maximum total memory capacity (in bits) available for all computational aspects*
- $P_{\max} \in \mathbb{N}$*: Maximum total digit positions in positional number representations*
- $L_{\max} \in \mathbb{N}$*: Maximum string length constraint for symbolic objects (formulas, programs, statements) imposed by memory storage limitations*

System Specification: *The system $\mathcal{S} = \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ consists of:*

1. ***Finite Turing Machines**:*

$$\mathcal{M}_L = \{M : |\langle M \rangle| \leq L_{\max}\}$$

 doi:10.20944/preprints202510.2040.v1

where $\langle M \rangle$ denotes the encoding of machine M as a string over a finite alphabet Σ. When executed within the resource-bounded system, any machine $M \in \mathcal{M}_L$ either halts with output using $\leq T_{\max}$ steps and $\leq S_{\max}$ memory, or terminates with "resource exceeded" when bounds are reached. The cardinality is bounded by $|\mathcal{M}_L| \leq |\Sigma|^{L_{\max}} < \infty$.

2. ***Bounded Formal Systems**:*

$$\mathcal{F}_{T,S,L,P} = \{\mathcal{F} : length(\phi) \leq L_{\max} \text{ for all } \phi \in \mathcal{F}, precision(\mathcal{F}) \leq P_{\max}\}$$

where every formula has length $\leq L_{\max}$ symbols, all proofs require $\leq T_{\max}$ verification steps, proof storage uses $\leq S_{\max}$ memory, and numerical constants have precision $\leq P_{\max}$.

3. ***Finite Precision Numbers**:*

$$\mathbb{R}_{P_{\max}} = \{r \in \mathbb{R} : r \text{ has at most } P_{\max} \text{ total digit positions}\}$$

representing all numbers expressible within $P_{\max}$ digit positions, where $P_{\max} = p + k$ with p fractional digits and k integer digits in the earlier notation. For example, with $r = 2$ (binary) and $P_{\max} = 5$, we have exactly $2 \cdot 2^5 = 64$ possible numbers (including positive and negative); with $r = 10$ (decimal) and $P_{\max} = 3$, we have exactly $2 \cdot 10^3 = 2000$ possible numbers (including positive and negative).

***Unified String Length Constraint Justification:** The parameter $L_{\max}$ applies uniformly to both machine encodings and logical formulas because both are stored as finite strings over the same underlying alphabet Σ within the shared memory capacity $S_{\max}$. In any finite computational system, machine programs $\langle M \rangle$ and logical formulas ϕ are both represented as strings Σ^* stored in the same finite memory $S_{\max}$, creating direct memory competition where longer programs reduce available space for formulas and vice versa, expressed by the constraint $|\langle M \rangle| \cdot \log_2(|\Sigma|) + |\phi| \cdot \log_2(|\Sigma|) \leq S_{\max}$. The maximum storable string length is determined by memory constraints rather than conceptual distinctions between "programs" and "formulas," reflecting the implementation reality that in any concrete system—whether theorem provers or programming environments—the same string length limits apply to both program text and formula text due to shared memory architecture. Therefore, $L_{\max}$ represents the fundamental constraint on symbolic complexity imposed by finite memory, applicable to any string-based mathematical object regardless of its semantic interpretation.*

***Fundamental Constraint:** Every component of $\mathcal{S}$ contains finitely many objects, bounded by memory constraints:*

$$|Formulas_L| \leq N_{formulas} \leq \frac{S_{\max}}{L_{\max} \cdot \log_2(|\Sigma|)} < \infty \tag{6}$$

$$|Proofs_{T,S,L}| \leq N_{proofs} \leq \left(\frac{S_{\max}}{L_{\max} \cdot \log_2(|\Sigma|)}\right)^{T_{\max}} < \infty \tag{7}$$

$$|\mathbb{R}_{P_{\max}}| = 2 \cdot r^{P_{\max}} = 2 \cdot r^{p+k} < \infty \tag{8}$$

$$|\mathcal{M}_L| \leq |\Sigma|^{L_{\max}} < \infty \tag{9}$$

where Σ denotes the finite alphabet, $N_{formulas}$ is the number of formulas stored simultaneously, and N_{proofs} is the number of proofs stored simultaneously.

***Physical Foundations of Resource Bounds:** The fundamental constraints $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ emerge from physical limits governing any mechanically implementable computational system.*

***Temporal Bound from Landauer's Principle:** The maximum number of computational operations $T_{\max}$ is fundamentally limited by Landauer's principle [30], which establishes that any irreversible computational operation (such as erasing a bit) must dissipate at least $k_B T \ln(2)$ joules of energy, where k_B is Boltzmann's constant and T is the ambient temperature. For any finite computational system with total available energy E and operating temperature T, this yields: $T_{\max} \leq \frac{E}{k_B T \ln(2)}$.*

***Spatial Memory Bound from Bekenstein Limit:** The maximum information storage capacity $S_{\max}$ is constrained by the Bekenstein bound [31], which establishes that any finite region of space with radius R and*

 doi:10.20944/preprints202510.2040.v1

total energy E can store at most $S_{\max} \leq \frac{2\pi RE}{\hbar c}$ bits of information, where $\hbar$ is the reduced Planck constant and c is the speed of light.

Derived Bounds through Memory Partitioning: *Given the fundamental physical limits on $T_{\max}$ and $S_{\max}$, the constraints on program length and numerical precision follow from memory partitioning requirements. The total available memory $S_{\max}$ must be divided among: (1) program storage requiring $L_{\max} \cdot \log_2(|\Sigma|)$ bits per program, (2) numerical data storage requiring $P_{\max} \cdot \log_2(|\Sigma|)$ bits per number, and (3) working memory for computation. This yields the constraint: $L_{\max} + P_{\max} \leq \frac{S_{\max}}{\log_2(|\Sigma|)}$, establishing that program complexity and numerical precision are fundamentally bounded by available physical storage capacity.*

7.3. Universality Under Finite Resource Constraints

Theorem 4 (Infinite Dependencies of Classical Universality)**.** *Classical Turing machine universality cannot be formulated without the infinite assumptions $(T_{\max}, S_{\max}, P_{\max}, L_{\max}) = (\infty, \infty, \infty, \infty)$, where $T_{\max}$ represents unlimited time, $S_{\max}$ unlimited space, $P_{\max}$ unlimited precision, and $L_{\max}$ unlimited program complexity.*

Proof. From Definition 4, classical universality operates outside any bounded system $\mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$.

Let $\mathcal{U}$ denote classical universality: $\forall M, \forall w \in \Sigma^*, \exists U : U(\langle M \rangle, w) = M(w)$.

This requires:

$$|\mathcal{M}_L| = \infty \quad \text{(infinite program space)} \tag{10}$$

$$|\Sigma^*| = \infty \quad \text{(infinite input domains)} \tag{11}$$

$$\text{Time}(U(\langle M \rangle, w)) \leq \infty \quad \text{(unbounded simulation time)} \tag{12}$$

$$|\langle M \rangle| \leq \infty \quad \text{(arbitrarily complex encodings)} \tag{13}$$

For any finite constraint $(T_{\max}, S_{\max}, P_{\max}, L_{\max}) \neq (\infty, \infty, \infty, \infty)$, the universality claim $\mathcal{U}$ requires modification to account for the bounded computational domain $\mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$. □

Corollary 3 (Impossibility of Finite Universality)**.** *$\forall U \in \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ with $(T_{\max}, S_{\max}, P_{\max}, L_{\max}) \in \mathbb{N}^4$: U is not universal.*

Proof. We proceed by contradiction. Assume $U \in \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ is universal, meaning:

$$\forall M, \forall w \in \Sigma^* : U(\langle M \rangle, w) = M(w)$$

From Definition 4, U must satisfy:

- $|\langle U \rangle| \leq L_{\max}$ (bounded program encoding)
- $\text{Time}(U, x) \leq T_{\max}$ for all inputs x (bounded execution time)
- $\text{Space}(U, x) \leq S_{\max}$ for all inputs x (bounded memory usage)

Construction of Counterexample Machines:

Case 1: Time Bound Violation. Define machine M_T with encoding $\langle M_T \rangle$ such that $|\langle M_T \rangle| \leq L_{\max}$ and:

Algorithm 2 Machine $M_T(w)$

```
counter ← 0
while counter < T_max + 1 do
    counter ← counter + 1
return 1
```

By construction: $\text{Time}(M_T, w) = T_{\max} + 1 > T_{\max}$ for any input w.

 doi:10.20944/preprints202510.2040.v1

If U is universal, then $U(\langle M_T \rangle, w) = M_T(w) = 1$. However, this requires:

$$\text{Time}(U, (\langle M_T \rangle, w)) \geq T_{\max} + 1 > T_{\max}$$

contradicting the constraint that $U \in \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$.

Case 2: Space Bound Violation. Define machine M_S with encoding $\langle M_S \rangle$ such that $|\langle M_S \rangle| \leq L_{\max}$ and:

Algorithm 3 Machine $M_S(w)$

for $i = 1$ **to** $S_{\max} + 1$ **do**
  Write symbol '1' to tape cell i
return 1

By construction: $\text{Space}(M_S, w) = S_{\max} + 1 > S_{\max}$ for any input w.

If U is universal, then $U(\langle M_S \rangle, w) = M_S(w) = 1$. However, simulating M_S requires:

$$\text{Space}(U, (\langle M_S \rangle, w)) \geq S_{\max} + 1 > S_{\max}$$

contradicting the space constraint on U.

Case 3: Encoding Length Bound Violation. Consider any machine M_L with $|\langle M_L \rangle| = L_{\max} + 1$. By Definition 4:

$$M_L \notin \mathcal{M}_L = \{M : |\langle M \rangle| \leq L_{\max}\}$$

Since $U \in \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$, the input $\langle M_L \rangle$ cannot be stored within U's memory constraints, making $U(\langle M_L \rangle, w)$ undefined.

Conclusion: In each case, U fails to simulate machines that violate at least one resource bound. Therefore, U can only simulate machines in the bounded set:

$$\mathcal{M}_L^{\text{sim}} = \{M : |\langle M \rangle| \leq L_{\max} \text{ and } \forall w : \text{Time}(M, w) \leq T_{\max}, \text{Space}(M, w) \leq S_{\max}\}$$

Since $\mathcal{M}_L^{\text{sim}} \subset \mathcal{M}_{\text{all}}$ and $\mathcal{M}_L^{\text{sim}} \neq \mathcal{M}_{\text{all}}$ (the set of all Turing machines), U is not universal. □

7.4. Gödel's Incompleteness Under Finite Resource Constraints

Gödel's incompleteness theorems, as detailed in Section 3, employ a diagonal construction remarkably parallel to Cantor's argument. The classical proof constructs a formula G that systematically differs from every diagonal entry in an infinite table of formula provability relationships, ultimately creating a self-referential statement asserting its own unprovability. However, as established in Corollary 2, diagonal constructions encounter insurmountable logical barriers in finite precision systems. These results depend critically on infinite assumptions—unlimited formula complexity, unbounded proof length, and infinite enumeration of mathematical statements. We now examine how Gödel's diagonal construction transforms when subjected to explicit finite resource bounds $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$.

Scope conditioned by non-universality: As established earlier in Section 7.3, no machine within any bounded system $\mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ is universal. This matters for the Gödel analysis because it fixes our interpretive scope: we are not attempting to mechanize an *unbounded* logical universe, but rather to analyze a *sufficiently expressive yet non-universal* formal system whose objects (formulas, proofs, encodings, and numeric constants) satisfy the resource bounds from Definition 4. Within this scope, we choose parameters $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ *adequate for the specific system under study* so that full enumeration of expressible formulas and verifiable proofs is mechanically realizable inside the bounds.

Theorem 5 (Finite Formal System Completeness). *For any bounded formal system $\mathcal{F}_{T,S,L,P} \subset \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$:*

1. *The set of expressible formulas has finite cardinality bounded by equation (6)*

 doi:10.20944/preprints202510.2040.v1

2. *The set of constructible proofs has finite cardinality bounded by equation (7)*
3. *All provability questions become decidable through exhaustive enumeration*
4. *Classical diagonal constructions fail due to completeness of finite formula enumeration*

Proof. Step 1: Finite Formula Space Cardinality

From Definition 4 and equation (6), the bounded formal system $\mathcal{F}_{T,S,L,P}$ contains at most:

$$|\text{Formulas}_L| \leq |\Sigma|^{L_{\max}} < \infty \tag{14}$$

where Σ is the finite logical alphabet and every formula $\phi \in \mathcal{F}_{T,S,L,P}$ satisfies $\text{length}(\phi) \leq L_{\max}$.

Step 2: Finite Proof Space Cardinality

From equation (7), the space of all possible proofs within the system is bounded:

$$|\text{Proofs}_{T,S,L}| \leq (|\text{Formulas}_L|)^{T_{\max}} < \infty \tag{15}$$

where each proof π requires at most $T_{\max}$ verification steps and uses at most $S_{\max}$ memory for storage and verification. The exponential bound arises because a proof is a sequence of at most $T_{\max}$ formulas (proof steps), where each formula comes from the finite set bounded by $|\text{Formulas}_L| \leq |\Sigma|^{L_{\max}}$ from equation (12), giving at most $(|\text{Formulas}_L|)^{T_{\max}}$ possible proof sequences.

Step 3: Constructive Completeness in Finite Formal Systems

Having established in Corollary 2 that diagonal constructions fail in finite systems—because they cannot produce objects outside finite enumerations when all objects are bounded by finite resource constraints—we now demonstrate constructively how finite formal systems achieve complete decidability. Rather than attempting to construct undecidable statements, we show explicitly how every expressible statement becomes decidable through finite enumeration.

Constructive Enumeration of Finite Formal Systems:

For any bounded formal system $\mathcal{F}_{T,S,L,P}$, we can explicitly construct:

Complete Formula Enumeration:

Algorithm 4 Enumerate All Formulas in $\mathcal{F}_{T,S,L,P}$

```
L ← ∅                                              ▷ Complete formula list
for ℓ = 1 to L_max do
    for each string s ∈ Σ^ℓ do
        if s is a well-formed formula then
            L ← L ∪ {s}
return L                                           ▷ Complete enumeration of all expressible formulas
```

Result: $|\mathcal{L}| \leq |\Sigma|^{L_{\max}}$ contains every formula expressible in the system.

Complete Proof Enumeration:

Algorithm 5 Enumerate All Proofs in $\mathcal{F}_{T,S,L,P}$

```
P ← ∅                                              ▷ Complete proof list
for t = 1 to T_max do
    for each sequence π = (φ_1, ..., φ_t) with φ_i ∈ L do
        if π is a valid proof sequence using ≤ S_max memory then
            P ← P ∪ {π}
return P                                           ▷ Complete enumeration of all possible proofs
```

Result: $|\mathcal{P}| \leq (|\mathcal{L}|)^{T_{\max}}$ contains every proof constructible in the system.

Constructive Decidability for Every Statement:

With complete enumerations $\mathcal{L}$ and $\mathcal{P}$, every formula's provability becomes constructively decidable:

 doi:10.20944/preprints202510.2040.v1

Algorithm 6 Constructive Decision Procedure for Any Formula

Input: Formula $\phi \in \mathcal{L}$
for each proof $\pi \in \mathcal{P}$ **do**
  if π proves ϕ **then**
    return (PROVABLE, π) ▷ Constructive proof witness
return NOT PROVABLE ▷ Definitive negative result

Constructive Properties:

1. **Termination**: All algorithms terminate in finite time
2. **Completeness**: Every expressible formula receives a definitive answer
3. **Witnessing**: Provable statements come with explicit proof witnesses
4. **Decidability**: No statement remains undecidable within the system

Comparison with Classical Gödel Construction:

Classical Gödel construction attempts to create formula $G \equiv \neg\text{Provable}(\ulcorner G \urcorner)$ that falls outside any enumeration. However, as established in Corollary 2, diagonal constructions encounter insurmountable logical barriers in finite precision systems because they fundamentally depend on the ability to construct objects that systematically differ from every element in a proposed enumeration—a procedure that becomes logically impossible when the enumeration is genuinely finite and complete.

Constructive Resolution:

- **If** $\text{length}(G) \leq L_{\max}$: Then $G \in \mathcal{L}$ and Algorithm 6 decides its provability constructively. The diagonal construction cannot produce a formula "outside" the enumeration $\mathcal{L}$ because $\mathcal{L}$ contains all expressible formulas by construction.
- **If** $\text{length}(G) > L_{\max}$: Then $G \notin \mathcal{F}_{T,S,L,P}$ and is irrelevant to the bounded system. The attempted diagonal construction exceeds the system's expressive capacity, rendering it meaningless within the finite framework.

No Paradox Emerges: The finite enumeration is genuinely complete for all expressible statements, eliminating the basis for self-referential undecidability.

Constructive Completeness Result:

Proposition 3 (Finite System Constructive Completeness)**.** *For any bounded formal system $\mathcal{F}_{T,S,L,P}$:*

1. *Every expressible formula is constructively enumerable*
2. *Every possible proof is constructively enumerable*
3. *Every provability question is constructively decidable*
4. *The system is complete and consistent within its bounds*

The Constructive Result: Within bounded formal systems $\mathcal{F}_{T,S,L,P}$, the systematic enumeration approach provides complete decidability for all expressible statements. The resource constraints that initially appear limiting actually enable constructive completeness through finite enumeration methods. □

7.5. The Halting Problem Under Finite Resource Constraints

Turing's halting problem, analyzed in Section 4, employs diagonal construction remarkably parallel to both Cantor's diagonal argument and Gödel's incompleteness theorem. The classical proof constructs a diagonal machine D that systematically differs from every machine in an infinite enumeration along the diagonal of the halting matrix, ultimately creating a self-referential computation that leads to logical contradiction. However, our analysis in Corollary 2 demonstrates that such diagonal escape mechanisms fail completely when operating within bounded computational domains. In finite systems, every computational object resides within a complete enumeration, eliminating the foundational requirement for diagonal constructions to produce objects that lie outside the enumeration—a process that becomes untenable under finite precision constraints.

 doi:10.20944/preprints202510.2040.v1

We now demonstrate constructively how Turing's diagonal construction transforms when subjected to explicit finite resource bounds $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$, revealing that for *adequately bounded domains* (as motivated by non-universality in Section 7.3) the halting problem admits a definitive HALTS/LOOP decision for every machine-input pair, while for *arbitrary* analyses it yields bounded classification with a TIMEOUT outcome when adequacy is not assumed.

Theorem 6 (Finite Halting Problem Decidability)**.** *For any bounded computational system $\mathcal{M}_L \subset \mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ operating under finite resource constraints:*

1. *The set of finite Turing machines has finite cardinality*
2. *The set of possible inputs has finite cardinality*
3. *All halting queries become decidable through exhaustive verification*
4. *Classical diagonal constructions fail due to completeness of finite machine enumeration*

Proof. Step 1: Finite Machine Space Cardinality

From Definition 4, the bounded computational system $\mathcal{M}_L$ contains machines subject to explicit constraints:

$$|\mathcal{M}_L| \leq |\Sigma|^{L_{\max}} < \infty \tag{16}$$

where every machine $M \in \mathcal{M}_L$ satisfies:

- $|\langle M \rangle| \leq L_{\max}$ (bounded program encoding length)
- $\text{Time}(M, w) \leq T_{\max}$ for all inputs w (bounded execution time)
- $\text{Space}(M, w) \leq S_{\max}$ for all inputs w (bounded memory usage)

Step 2: Finite Input Space Cardinality

Input strings are constrained by available memory for input storage and processing:

$$|\text{Inputs}_S| \leq |\Sigma|^{S_{\max}/\log_2(|\Sigma|)} < \infty \tag{17}$$

where the bound reflects that each input string of length ℓ requires $\ell \cdot \log_2(|\Sigma|)$ bits of storage, so the maximum input length is $\ell_{\max} = S_{\max}/\log_2(|\Sigma|)$, giving at most $|\Sigma|^{\ell_{\max}}$ possible input strings.

Step 3: Finite Configuration Space and Loop-or-Halt

For any fixed $(S_{\max}, L_{\max})$, the number of instantaneous configurations of a machine $M \in \mathcal{M}_L$ on an input $w \in \text{Inputs}_S$ is finite (finite control, bounded head positions within the active tape region, and tape contents over at most $S_{\max}$ cells). Therefore any run either halts within the configuration bound or revisits a configuration and enters a loop.

Formal adequacy lemma and corollary.

Lemma 1 (Finite configuration bound)**.** *For any $M \in \mathcal{M}_L$ and $w \in \text{Inputs}_S$, the set of instantaneous configurations of the run of M on w under bounds $(S_{\max}, L_{\max})$ is finite. In particular, if Q is the finite state set and Σ the tape alphabet, then for some constant $c > 0$ (accounting for heads/tracks),*

$$N_{conf}(M, w) \;\leq\; c \cdot |Q| \cdot (S_{\max} + 1) \cdot |\Sigma|^{S_{\max}} < \infty.$$

Consequently, any run of length $> N_{conf}(M, w)$ repeats a configuration and hence is ultimately periodic (loops).

Proof sketch. An instantaneous configuration is determined by: the control state in Q, the head position within the active tape region (bounded by $S_{\max}$ cells), and the tape contents over at most $S_{\max}$ cells from the finite alphabet Σ (plus a constant factor for multi-head/track variants). The configuration graph is therefore finite. By the pigeonhole principle, more than $N_{\text{conf}}(M, w)$ steps forces a repeated configuration, yielding a loop. □

Step 4: Finite Halting Matrix Completeness

 doi:10.20944/preprints202510.2040.v1

The classical infinite halting matrix $T[i,j]$ becomes a finite matrix $T_{\text{finite}}[i,j]$ with dimensions bounded by:

$$|\text{Rows}| = |\mathcal{M}_L| \leq |\Sigma|^{L_{\max}} < \infty \tag{18}$$

$$|\text{Columns}| = |\text{Inputs}_S| \leq |\Sigma|^{S_{\max}/\log_2(|\Sigma|)} < \infty \tag{19}$$

$$|\text{Total Entries}| = |\text{Rows}| \times |\text{Columns}| < \infty \tag{20}$$

Each entry $T_{\text{finite}}[i,j]$ can be computed definitively once $T_{\max}$ is chosen to exceed the configuration bound for the domain: classify HALTS (with trace) if halting occurs before the bound; otherwise classify LOOP (with witness of repeated configuration). For analyses over arbitrary machines where such adequacy is not fixed, the TIMEOUT classification from earlier remains appropriate.

Step 4: Constructive Decidability Through Complete Enumeration

Having established finite cardinalities for both machines and inputs, we can construct explicit algorithms that decide the halting problem completely within the bounded domain:

Complete Machine-Input Enumeration:

Algorithm 7 Enumerate All Machine-Input Pairs in $\mathcal{M}_L$

```
P ← ∅                                              ▷ Complete machine-input pair list
for ℓ = 1 to L_max do
    for each string s ∈ Σ^ℓ do
        if s encodes a valid Turing machine M then
            for each input w ∈ Inputs_S do
                P ← P ∪ {(M, w)}
return P                                           ▷ Complete enumeration of computational domain
```

Constructive Halting/Loop Decision for Every Machine-Input Pair (Adequate Bounds):

Algorithm 8 Constructive Halting Problem Decision Procedure

```
Input: Machine M ∈ M_L, input w ∈ Inputs_S
Initialize simulation counter: t ← 0
Initialize memory tracker: s ← 0
while t ≤ T_max and s ≤ S_max do
    Execute one step of M on input w
    t ← t + 1, update s based on memory usage
    if M halts then
        return (HALTS, steps = t, output)
return (LOOP, witness = first repeated configuration)
```

Global Halting Matrix Construction:

Algorithm 9 Construct Complete Finite Halting Matrix

```
Initialize matrix T_finite[1..||M_L||, 1..|Inputs_S|]
for each machine M_i ∈ M_L do
    for each input w_j ∈ Inputs_S do
        (result, details) ← Algorithm 8(M_i, w_j)
        if result = HALTS then
            T_finite[i, j] ← 1
        else
            T_finite[i, j] ← 0
return T_finite                                    ▷ Complete decidable halting matrix
```

Constructive Properties:

 doi:10.20944/preprints202510.2040.v1

1. **Termination**: All algorithms terminate in finite time with specific bounds:
   - Algorithm 7: $O(|\Sigma|^{L_{\max}} \times |\text{Inputs}_S|)$
   - Algorithm 8: $O(T_{\max})$ per machine-input pair
   - Algorithm 9: $O(|\mathcal{M}_L| \times |\text{Inputs}_S| \times T_{\max})$
2. **Complete Enumeration**: Every machine-input pair in the bounded domain $\mathcal{M}_L \times \text{Inputs}_S$ is systematically processed
3. **Definitive Classification (Adequate Bounds):** For adequately bounded domains, every machine-input combination receives one of two definitive outcomes: HALTS (with trace) or LOOP (with repeated-configuration witness)
4. **Bounded Analysis (Arbitrary Machines):** Without adequacy assumptions over arbitrary machines, the procedure yields HALTS or TIMEOUT, providing a bounded classification rather than universal decidability

Step 5: Diagonal Construction Failure and Constructive Resolution

Classical diagonal construction attempts to create machine D with $D(i) = 1 - T[i,i]$ that escapes any enumeration. However, this fails in finite systems: if $|\langle D \rangle| \leq L_{\max}$, then $D \in \mathcal{M}_L$ and cannot escape the complete finite enumeration; if $|\langle D \rangle| > L_{\max}$, then D exceeds system capacity and is irrelevant to the bounded framework.

Proposition 4 (Finite System Halting Classification under Adequate Bounds)**.** *In any adequately bounded domain $\mathcal{M}_L \times Inputs_S$, where $T_{\max}$ exceeds the configuration bound for all pairs, every machine-input pair receives a definitive classification through complete enumeration (Algorithms 7–9): either HALTS (with finite execution trace) or LOOP (with repeated-configuration witness).*

Remark 2. *For analyses over arbitrary machines without an adequacy assumption, the same procedure yields a bounded classification: HALTS or TIMEOUT (resource exhaustion).*

Theorem 7 (Decidability under adequate bounds)**.** *If $T_{\max} \geq \max_{(M,w) \in \mathcal{M}_L \times Inputs_S} N_{conf}(M,w)$, then the halting problem on the bounded domain $\mathcal{M}_L \times Inputs_S$ is decidable: there exists a procedure that, for every (M,w), outputs either HALTS or LOOP.*

Proof. Simulate (M,w) for at most $N_{\text{conf}}(M,w)$ steps while recording seen configurations (as in Algorithm 8). If M halts within that bound, output HALTS together with a finite execution trace; otherwise, by Lemma 1, a configuration repeats and we output LOOP together with a repeated-configuration witness. The adequacy assumption $T_{\max} \geq \max N_{\text{conf}}$ ensures the simulation never exceeds the time bound, so no TIMEOUT outcome arises in this regime. □

 doi:10.20944/preprints202510.2040.v1

Classical Infinite Case	Finite Resource Case
Infinite machine enumeration	Finite: $\lvert\mathcal{M}_L\rvert \leq \lvert\Sigma\rvert^{L_{\max}} < \infty$
Infinite input domain	Bounded: $\lvert\text{Inputs}_S\rvert \leq \lvert\Sigma\rvert^{S_{\max}/\log_2(\lvert\Sigma\rvert)} < \infty$
Unbounded execution time	Time limit: $\text{Time}(M,w) \leq T_{\max}$ (choose $T_{\max}$ to exceed configuration bound for adequacy)
Unlimited memory	Memory bound: $\text{Space}(M,w) \leq S_{\max}$
"Does not halt" = infinite claim	Adequate bounds: HALTS/LOOP via configuration bound; otherwise: HALTS/TIMEOUT (bounded observation)
Diagonal escapes enumeration	Diagonal contained in complete enumeration
Undecidable halting problem	Decidable as HALTS/LOOP under adequate bounds; bounded classification (HALTS/TIMEOUT) otherwise

Key Distinction - Two Interpretations of TIMEOUT:

1. **For Bounded Computational Domains (Adequate Bounds):** When $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$ are chosen to exceed the configuration bound for the domain, there is no TIMEOUT outcome: every computation is classified as HALTS (with finite trace) or LOOP (with repeated-configuration witness).
2. **For Arbitrary Computational Problems:** When analyzing arbitrary Turing machines without an adequacy assumption, TIMEOUT represents resource exhaustion rather than claims about infinite behavior. This provides bounded classification without solving the universal halting problem.

This distinction reveals that finite resource frameworks can achieve genuine decidability for appropriately bounded domains while providing useful classification for arbitrary computational problems.

□

8. Conclusions

This investigation has systematically examined the infinite foundations underlying classical impossibility results in mathematics and computer science. Our analysis demonstrates that fundamental theorems—including Cantor's uncountability proof, Gödel's incompleteness theorems, and Turing's halting problem—depend critically on infinite mathematical assumptions that become problematic when subjected to constructive finite resource constraints.

Cantor's Diagonal Argument and Real Number Uncountability: Section 2 revealed that Cantor's diagonal argument depends fundamentally on infinite decimal expansions and completed infinite enumerations. However, Section 2.4 demonstrated that when restricted to finite-decimal real numbers—precisely those appearing in scientific computation—the canonical bijection framework provides explicit constant-time enumeration algorithms. This shows that uncountability depends on infinite decimal representations rather than fundamental mathematical structure, with all computationally meaningful real numbers being countably enumerable.

Gödel's Incompleteness and Mathematical Logic: Section 3 exposed how Gödel's incompleteness theorems employ diagonal constructions that depend critically on infinite formula enumeration and unbounded proof search procedures. Conversely, Section 7.4 showed that under finite resource bounds $(T_{\max}, S_{\max}, P_{\max}, L_{\max})$, formal systems become finite structures where consistency, completeness, and decidability become tractable computational problems. Using explicit bounds on formulas (Equa-

 doi:10.20944/preprints202510.2040.v1

tion 6) and proofs (Equations 7 and 15), mathematical logic becomes decidable through constructive enumeration procedures (Algorithms 4, 5, 6).

Turing's Halting Problem and Computational Decidability: Section 4 demonstrated that Turing's halting problem proof relies on the same infinite diagonal construction pattern, requiring unlimited machine enumeration and unbounded execution time. However, Section 7.5 established that when restricted to finite resource systems $\mathcal{S}(T_{\max}, S_{\max}, P_{\max}, L_{\max})$, the halting problem becomes a finite verification task. With explicit bounds on machines (Equation 9), exhaustive enumeration becomes feasible using Algorithms 7, 8, and 9. In adequately bounded domains—where $T_{\max}$ meets or exceeds the configuration bound—every machine–input pair is decidable as HALTS or LOOP (no TIME-OUT). Outside adequacy, the same constructive procedure yields a sound bounded classification (HALTS/TIMEOUT).

Universal Computation and Resource Dependencies: Section 5 traced how classical Turing machine universality claims evolved from Turing's original resource-aware formulation to modern infinite assumptions. Our rigorous analysis in Section 7.3 showed that any purportedly universal machine U attempting to simulate machines exceeding resource bounds $(T_{\max}, S_{\max}, L_{\max})$ must itself violate these constraints, creating fundamental contradictions. This eliminates classical universality under realistic computational limits, revealing that infinite resource assumptions are essential for traditional universality claims.

Constructive Mathematical Foundation: Section 6 provided a comprehensive constructive critique, establishing through Corollary 2 that classical impossibility results collapse when infinite assumptions are replaced with finite constructive procedures. This analysis reveals the philosophical tension between Platonic mathematical idealism and algorithmic constructive traditions.

Unified Finite Resource Framework: Section 7 developed a systematic finite resource model using Definition 4, with explicit bounds on formulas (Equation 6), proofs (Equations 7 and 15), real numbers (Equation 8), and machines (Equation 9). For formal systems, we choose bounds adequate for the system under study, yielding constructive decidability via Algorithms 4, 5, and 6. For the halting analysis, adequacy yields HALTS/LOOP decidability; otherwise, the same procedure provides a sound bounded classification (HALTS/TIMEOUT).

Fundamental Implications: Our analysis reveals that the scope and applicability of classical impossibility theorems—undecidability, incompleteness, and uncomputability—may be more limited than traditionally assumed. These results depend critically on infinite constructions and may not directly apply to finite computational contexts. When mathematical frameworks are restricted to constructive finite procedures, we observe that:

1. Computationally realizable real numbers within finite precision constraints admit explicit enumeration
2. Formal systems operating under bounded resources admit systematic procedures; with bounds chosen adequate for the system, they are decidable (complete within bounds)
3. Bounded halting analysis: under adequate bounds, every pair is decidable as HALTS/LOOP; otherwise, HALTS/TIMEOUT provides a sound bounded classification
4. Universal computation requires infinite resources, suggesting natural limits to universality claims in finite settings
5. Mathematical verification and search become tractable within appropriately bounded domains

These observations suggest that theoretical frameworks relying on infinite constructions may not fully capture the behavior of computational systems operating under practical resource limitations. Constructive mathematics, emphasizing algorithmic procedures over infinite abstractions, offers complementary insights for understanding computational and logical phenomena within realistic constraints, with adequacy clarifying when full decidability is achievable versus when bounded classification is the right notion.

The broader implications merit careful consideration: classical impossibility theorems, while mathematically rigorous within their infinite frameworks, may have limited applicability to finite

 doi:10.20944/preprints202510.2040.v1

computational domains. This perspective resonates with constructive mathematical traditions and contemporary computational practice, suggesting that finite constructive approaches provide valuable alternative viewpoints for analyzing the mathematical foundations of computation and logic within practical constraints.

9. Historical Context: Algorithmic and Constructive Mathematical Traditions

The tension between algorithmic constructivism and abstract idealism in mathematics has deep historical roots across multiple traditions. Early algorithmic approaches emphasized constructive procedures over abstract existence claims: the *Śulbasūtras* (c. 800-350 BCE) by Baudhāyana, Āpastamba, and Kātyāyana provided geometric construction procedures using rope algorithms [32,33]; Āryabhaṭa's *Āryabhaṭīya* (499 CE) systematized place-value arithmetic and algorithmic procedures [34]; Brahmagupta's *Brahmasphutasiddhanta* (628 CE) established finite algebraic algorithms while formalizing zero [35]; and the Kerala School (14th-16th centuries) developed algorithmic methods for infinite series using *yāvat-tāvat* ("as much as needed"), treating infinite processes as ongoing procedures rather than completed totalities [36,37]. Western constructive traditions developed in parallel: Leibniz's *characteristica universalis* (1670s-1680s) envisioned mechanical reasoning through symbolic manipulation, rejecting infinite wholes as "fictitious totalities" [24]; Kronecker's dictum "God made the integers, all else is the work of man" epitomized skepticism toward non-constructive proofs [8]; and this continued through Brouwer's intuitionism [9] to Bishop's modern constructive analysis [10], extended by Bridges and Richman [38]. Notably, Hilbert maintained a finitist stance, declaring "the infinite is nowhere to be found in reality," advocating for finitary methods in his consistency program [11]. The modern acceptance of completed infinities in set theory represents a departure from these constructive traditions: Platonic idealism, positing mathematical objects as eternal forms existing independently of construction, influenced Cantor's transfinite set theory [39], treating infinite sets as completed totalities rather than ongoing processes. This historical tension remains relevant to modern computational practice, where all actual computations terminate within finite time and space using finite-precision arithmetic. When mathematical reasoning must be mechanically executable—whether through ancient algorithmic procedures or modern computer systems—finite resource constraints become fundamental features rather than mere practical limitations. The constructive mathematical traditions, spanning multiple cultures and centuries, provide philosophical grounding for examining whether infinite idealizations represent intrinsic mathematical limitations or artifacts of foundational choices, supporting the technical results established in earlier sections that classical impossibility theorems depending on infinite assumptions may transform into decidability results within genuinely finite computational domains.

References

1. G. Cantor, "Über eine elementare Frage der Mannigfaltigkeitslehre," *Jahresbericht der Deutschen Mathematiker-Vereinigung*, vol. 1, pp. 75-78, 1891.
2. K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, pp. 173-198, 1931.
3. A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, ser. 2, vol. 42, pp. 230-265, 1936.
4. B. J. Copeland, "The Church-Turing Thesis," *Stanford Encyclopedia of Philosophy*, 2002, revised 2017. Available at: `https://plato.stanford.edu/entries/church-turing/`.
5. J. Meyer, "The Diagonal Lemma: Flaws in the Foundations," *Personal Communication/Web Resource*, Available at: `https://jamesrmeyer.com/ffgit/diagonal_lemma`, 2025.
6. S. K. Rithvik, "A Canonical Bijection Between Finite-Decimal Real Numbers and Natural Numbers with Constant-Time Enumeration Formulas," *arXiv preprint arXiv:2508.10750*, 2025. Available at: `https://arxiv.org/abs/2508.10750`.
7. S. K. Rithvik, "Canonical Bijection between Finite-Decimal Real Numbers and Natural Numbers: Implementation," GitHub Repository, 2025. Available at: https://github.com/rithvik1122/canonical-bijection-finite-decimals.

 doi:10.20944/preprints202510.2040.v1

8. Leopold Kronecker, quoted in Heinrich Weber, "Leopold Kronecker," *Mathematische Annalen*, vol. 43, pp. 1-25, 1893; see also Harold M. Edwards, "Kronecker's Views on the Foundations of Mathematics," in *The History of Modern Mathematics*, vol. 1, Academic Press, 1989, pp. 67-77.
9. L. E. J. Brouwer, "Intuitionism and Formalism," *Bulletin of the American Mathematical Society*, vol. 20, no. 2, pp. 81-96, 1913; see also Arend Heyting, *Intuitionism: An Introduction*, 3rd edition, North-Holland, Amsterdam, 1971.
10. E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.
11. D. Hilbert, "On the Infinite," *Mathematische Annalen*, vol. 95, pp. 161-190, 1925; English translation in P. Benacerraf and H. Putnam (eds.), *Philosophy of Mathematics: Selected Readings*, Prentice-Hall, 1964, pp. 134-151.
12. R. M. Smullyan, *Gödel's Incompleteness Theorems*, Oxford University Press, New York, 1992.
13. T. Franzén, *Gödel's Theorem: An Incomplete Guide to Its Use and Abuse*, A K Peters, Wellesley, MA, 2005.
14. G. S. Boolos, *The Logic of Provability*, Cambridge University Press, Cambridge, 1993.
15. J. van Heijenoort, editor, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, Harvard University Press, Cambridge, MA, 1967.
16. A. Church, "An Unsolvable Problem of Elementary Number Theory," *American Journal of Mathematics*, vol. 58, no. 2, pp. 345-363, 1936.
17. S. C. Kleene, "General recursive functions of natural numbers," *Mathematische Annalen*, vol. 112, no. 1, pp. 727-742, 1936.
18. E. L. Post, "Finite combinatory processes—formulation 1," *The Journal of Symbolic Logic*, vol. 1, no. 3, pp. 103-105, 1936.
19. A. M. Turing, *Collected Works of A.M. Turing: Mechanical Intelligence*, edited by D. C. Ince, North-Holland, Amsterdam, 1992.
20. J. D. Hamkins and T. Nenu, "Did Turing prove the undecidability of the halting problem?" *arXiv preprint arXiv:2407.00680*, 2024. Available at: `https://arxiv.org/abs/2407.00680`.
21. M. Davis, *Computability and Unsolvability*, McGraw-Hill, New York, 1958.
22. C. E. Shannon, "A universal Turing machine with two internal states," in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds., Annals of Mathematics Studies, no. 34, pp. 157–165, Princeton University Press, 1956.
23. S. C. Kleene, *Introduction to Metamathematics*, D. Van Nostrand Company, Inc., New York, 1952.
24. G. W. Leibniz, *De Arte Combinatoria* (1666) and *Characteristica Universalis* fragments (c. 1679-1690), in *Philosophical Essays*, translated by Roger Ariew and Daniel Garber, Hackett Publishing, 1989; see also Lenzen, Wolfgang, "Leibniz's Logic and the 'Universal Characteristic,'" *Topoi*, vol. 9, no. 2, pp. 129-137, 1990.
25. G. Boole, "An Investigation of the Laws of Thought," Walton and Maberly, London, 1854.
26. G. Frege, *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, Halle a.S., Verlag von Louis Nebert, 1879.
27. B. Russell and A. N. Whitehead, *Principia Mathematica*, 3 volumes, Cambridge University Press, Cambridge, 1910–1913.
28. D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik*, Springer, Berlin, 1928.
29. M. Davis, *The Universal Computer: The Road from Leibniz to Turing*, 3rd edition, CRC Press, Boca Raton, 2018.
30. R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183-191, 1961.
31. J. D. Bekenstein, "Universal upper bound on the entropy-to-energy ratio for bounded systems," *Physical Review D*, vol. 23, no. 2, pp. 287-298, 1981.
32. Baudhāyana, *Baudhāyana Śulbasūtra*, c. 800-600 BCE; edited by Willem Caland, *The Śrautasūtra of Baudhāyana*, Calcutta: Asiatic Society of Bengal, 1913; see also complete translation in S. N. Sen and A. K. Bag, *The Śulbasūtras of Baudhāyana, Āpastamba, Kātyāyana and Mānava*, Indian National Science Academy, New Delhi, 1983.
33. S. N. Sen and A. K. Bag, *The Śulbasūtras of Baudhāyana, Āpastamba, Kātyāyana and Mānava: with text, English translation and commentary*, Indian National Science Academy, New Delhi, 1983; see also A. Seidenberg, "The Ritual Origin of Geometry," *Archive for History of Exact Sciences*, vol. 1, no. 5, pp. 488-527, 1962.
34. Āryabhaṭa, *Āryabhaṭīya*, c. 499 CE; English translation by Walter Eugene Clark, *The Āryabhaṭīya of Āryabhaṭa: An Ancient Indian Work on Mathematics and Astronomy*, University of Chicago Press, 1930; see also Takao Hayashi, "Āryabhaṭa's Rule and Table for Sine-Differences," *Historia Mathematica*, vol. 24, no. 4, pp. 396-406, 1997.
35. Brahmagupta, *Brahmasphutasiddhanta*, 628 CE; English translation excerpts in H. T. Colebrooke, *Algebra, with Arithmetic and Mensuration, from the Sanscrit of Brahmegupta and Bhascara*, London: John Murray, 1817; see

 doi:10.20944/preprints202510.2040.v1

also David Pingree, "Brahmagupta, Balabhadra, and the *Brahmasphutasiddhanta*," *Journal for the History of Astronomy*, vol. 1, no. 1, pp. 67-72, 1970.

36. K. V. Sarma, *A History of the Kerala School of Hindu Astronomy*, Hoshiarpur: Vishveshvaranand Vedic Research Institute, 1972; David Bressoud, "Was Calculus Invented in India?" *The College Mathematics Journal*, vol. 33, no. 1, pp. 2-13, 2002.
37. Madhava of Sangamagrama, *Infinite Series Expansions for Trigonometric Functions*, c. 1400, as documented in later Kerala School texts; see C. T. Rajagopal and M. S. Rangachari, "On an untapped source of medieval Keralese mathematics," *Archive for History of Exact Sciences*, vol. 18, no. 2, pp. 89-102, 1978.
38. D. Bridges and F. Richman, *Varieties of Constructive Mathematics*, Cambridge University Press, 1987.
39. G. Cantor, *Contributions to the Founding of the Theory of Transfinite Numbers*, translated by P. E. B. Jourdain, Open Court, 1915.