Reflexive Compression Boundaries in Graded Categories

Larsen James Close

March 2026

Abstract

The complexity of self-referential computation decomposes into three independent gaps: naming (do distinct endomorphisms get distinct representations?), construction (is building fixed points cheap?), and depth transfer (does evaluating fixed points preserve computational cost?). We prove, in a 93-file Lean 4 formalization (main project plus standalone files) with zero `sorry`, zero `Classical.choice`, and zero custom axioms, that naming and fixed-point construction close from the categorical structure of a reflexive object $D \cong [D, D]$ alone, while depth transfer reduces to a quantitative growth-gap hypothesis on the graded slices of $\mathrm{End}(D)$ and D. Under this hypothesis, the anti-compression theorem yields a contradiction: the three conditions are jointly unsatisfiable. A non-uniformity theorem sharpens the picture: uniform graded reflexive isomorphisms prevent the growth gap, so computational growth gaps require graded non-uniformity in the iso. We then prove the growth gap is forced for any computational model with finite-table representability, and instantiate this for Lean's `Nat.Partrec.Code`. Bounded linear logic's bounded exponential is shown to force the gap via the cost of contraction. In the polynomial enrichment, Markov's principle at polynomial bounds is equivalent to P = NP. This is a bridge theorem identifying a resource-bounded witness-extraction principle with the classical search/verification collapse; it is not, by itself, a proof of either P = NP or P $\neq$ NP. As a secondary application, the framework diagnoses the structure of the invariant subspace problem; a full analysis is developed in a companion paper (Close 2026d).

Contents

1 Introduction **3**
- 1.1 Theorem map . . . 5
- 1.2 Notation and conventions . . . 6

2 The Reflexive Object and ω **6**
- 2.1 The categorical setup . . . 6
- 2.2 The fixed-point theorem . . . 7
- 2.3 Uniqueness . . . 7

3 Gap 1 — Naming **7**
- 3.1 The naming problem . . . 7
- 3.2 The naming theorem . . . 8
- 3.3 What naming does and does not give . . . 8

4 Gap 2 — Construction Cost **8**
- 4.1 The construction problem . . . 8
- 4.2 The cost theorems . . . 9
- 4.3 The placement theorem . . . 9

5 The Orthogonality Theorem **9**

6 Gap 3 — Depth Transfer and the Growth Gap **10**
- 6.1 The depth-transfer problem . . . 10
- 6.2 The anti-compression route . . . 11
- 6.3 The growth gap hypothesis . . . 12
- 6.4 The anti-compression theorem . . . 13

7 Conservativity and Independence **14**
- 7.1 The growth gap is not derivable . . . 14
- 7.2 The growth gap is consistent . . . 15
- 7.3 The non-uniformity theorem . . . 16
- 7.4 The growth gap is forced by representability . . . 17
- 7.5 Independence and instantiation summary . . . 17

8 The ISP Connection **18**

9 Related Work **19**
- 9.1 Implicit complexity theory . . . 19
- 9.2 Constructive computability theory . . . 19
- 9.3 Complexity barriers . . . 20
- 9.4 Self-interpreters and reflexive objects . . . 20
- 9.5 Invariant subspace problem . . . 20

10 Discussion **21**
- 10.1 What the framework does . . . 21

10.2 What the framework does not do 21
10.3 The conceptual contribution 22
10.4 The witness-extraction bridge 22
10.5 Open problems 23

11 Conclusion **24**

12 References **25**

13 Appendix A: Theorem Inventory **26**

A machine-verified companion formalization in Lean 4 is archived as part of the multi-paper code repository [Close 2026e] at DOI: 10.5281/zenodo.18915083. All theorems cited in this paper are verified with zero `sorry`, zero `Classical.choice`, and zero custom axioms.

1 Introduction

What makes computation hard? The standard answer points to self-reference: Godel's incompleteness theorems, the halting problem, and diagonalization arguments all involve programs that operate on their own descriptions. The Y combinator constructs fixed points by self-application. The intuition is that self-reference creates complexity.

This intuition is wrong — or rather, imprecise. Self-reference is cheap to construct, not necessarily cheap to execute. The Y combinator has bounded construction cost: fixed-point formation adds only a constant overhead to the cost of the function being fixed. But evaluating the resulting fixed point is where unbounded cost enters. The hardness of computation does not live in the ability to self-refer, but in the cost of executing what self-reference produces. This paper locates where.

We work in the categorical setting of a reflexive object $D \cong [D, D]$ in a monoidal closed category, equipped with a grading that tracks computational cost. The reflexive isomorphism provides a fold/unfold pair $\mathsf{fold} : [D, D] \to D$ and $\mathsf{unfold} : D \to [D, D]$ satisfying

$$\mathsf{fold} \circ \mathsf{unfold} = \mathrm{id}$$

$$\mathsf{unfold} \circ \mathsf{fold} = \mathsf{selfApp}$$

where $\mathsf{selfApp}$ is the self-application morphism. The asymmetry between these two equations — the first is an identity, the second generates nontrivial computation — is structural and is the source of the phenomena studied here.

The complexity of self-referential computation decomposes into three independent gaps:

1. **Naming.** Can the system name its own programs distinctly? That is, does the fixed-point operator ω map distinct endomorphisms to distinct elements of D?
2. **Construction.** Can the system build fixed points cheaply? That is, does $\omega(f)$ have bounded overhead relative to f?
3. **Depth transfer.** Can the system evaluate what its fixed points do efficiently? That is, does the evaluation cost of $\omega(f)$ reflect the complexity of f?

Our main results:

- **Gaps 1 and 2 close from categorical structure alone.** The naming gap reduces to a cancellation property of selfApp (Theorem 3.1), and the construction gap is bounded by universal constants from the reflexive object (Theorem 4.1). Neither requires axioms beyond the equational theory of the monoidal closed category.
- **Gap 3 reduces to the growth gap hypothesis.** The depth-transfer question cannot be resolved by categorical structure alone. It reduces to a single hypothesis about the graded slice structure: whether the space of endomorphisms eventually outgrows the space of elements at each grade level (Definition 6.3).
- **The anti-compression theorem.** The three conditions — naming, bounded construction, and the growth gap — are jointly contradictory (Theorem 6.1). A resource-bounded reflexive object cannot simultaneously name its endomorphisms faithfully, construct their fixed points cheaply, and accommodate the differential growth of semantic complexity.
- **The growth gap is independent.** It is not derivable from the categorical axioms (Theorem 7.1, via a Unit countermodel) and not inconsistent with them (Theorem 7.2, via a mesoscopic binary program model). The growth gap is the exact location where computational content enters the framework.
- **Non-uniformity is necessary.** Uniform graded reflexive isomorphisms prevent the growth gap; computational growth gaps require graded non-uniformity in the iso (Theorem 7.3). This gives a sharp dichotomy between denotational models (uniform iso, no growth gap) and computational models (non-uniform iso, growth gap holds).
- **The growth gap is forced by representability.** Any computational model with finite-table representability satisfies the growth gap (Theorem 7.4). This is instantiated concretely for `Nat.Partrec.Code`, Lean's formalization of partial recursive functions (Theorem 7.5).
- **The witness-extraction bridge.** Markov's principle at polynomial bounds — the assertion that $\neg\neg\exists$ witnesses can be extracted in polynomial time — is equivalent to P = NP (Theorem 10.1). This is a bridge theorem identifying a resource-bounded witness-extraction principle with the

classical search/verification collapse; it is not, by itself, a proof of either P = NP or P ≠ NP. The axiom profile paper's Layer 1 (Markov for Σ_1^0), enriched to polynomial bounds, *is* P vs NP.

All results are formalized in Lean 4 (93 files, zero `sorry`, zero `Classical.choice`, zero custom axioms). The companion formalization [Close 2026e] (DOI: 10.5281/zenodo.18915083) builds cleanly; theorem-level axiom profiles are listed in Appendix A. The axiom profiles — which logical axioms each theorem requires — are tracked throughout and fall into three strata: the core structural results (naming, anti-compression, non-uniformity, growth gap instantiation) use no axioms at all (profile $\emptyset$, fully constructive); the cost bounds and mesoscopic model use only the cosmetic axioms `propext` and `Quot.sound` that Lean's kernel introduces automatically (choice-free but not axiom-free); no result in this paper uses `Classical.choice` or any form of the axiom of choice.

The open frontier has moved: the growth gap is forced by finite-table representability, a property satisfied by standard coding-based models. The question is now what consequences follow at polynomial bounds — specifically, whether the anti-compression obstruction survives polynomial resource restrictions.

This paper extends two prior results. The first (Close 2026a) develops the reflexive object $D \cong [D, D]$ and the four equations above in a 42-file Lean 4 formalization, establishing the structural asymmetry between fold and unfold. The second (Close 2026b) shows that computability theory's central theorems partition into three layers tracking the double-negation monad's counit, with the partition boundary determined by whether proofs operate on values directly or must extract witnesses from double negations. The present paper asks: what happens to the reflexive object under resource constraints? The three-gap decomposition is the answer.

1.1 Theorem map

Gap	Result	Status	Theorem
Naming	ω injective $\Leftrightarrow$ selfApp-epi	Closed	3.1
Construction	$\text{cost}(\omega(f)) \leq \text{cost}(f) + K$	Closed	4.1
Depth transfer	Reduces to growth gap hypothesis	Hypothesis	Def. 6.3
Bridge and independence results			
Bridge	Naming + construction + growth gap $\Rightarrow \bot$	Proved	6.1
Independence	Growth gap not derivable	Proved	7.1
Independence	Growth gap not inconsistent	Proved	7.2
Non-uniformity	Growth gap $\Rightarrow$ non-uniform iso	Proved	7.3
Instantiation	Representability $\Rightarrow$ growth gap	Proved	7.4
Instantiation	`Nat.Partrec.Code` satisfies growth gap	Proved	7.5
Poly. bridge	Poly-Markov $\Leftrightarrow$ P = NP	Proved	10.1

1.2 Notation and conventions

We work in a monoidal closed category $\mathcal{C}$ with internal hom $[-,-]$, an endofunctor M preserving directed colimits, and an ω-chain from the initial object whose colimit L satisfies the Lambek condition $M(L) \cong L$. When $M = [A,-]$ and $A = L$, we obtain $D \cong [D,D]$ with $D = L$. We write $\mathsf{fold} : [D,D] \to D$ and $\mathsf{unfold} : D \to [D,D]$ for the two directions of the reflexive isomorphism (called `fwd` and `bwd` in the Lean formalization), and cur, unc for currying and uncurrying in $\mathcal{C}$. The self-application morphism is $\mathsf{selfApp} = \mathsf{unfold} \circ \mathsf{unc}(\mathrm{id})$. The fixed-point operator is $\omega(f) = \mathsf{cur}(\mathsf{selfApp} \circ f)$. Throughout, we suppress diagonal and pairing maps and identify composites via the monoidal closed adjunction; the Lean formalization carries the full type-correct definitions.

2 The Reflexive Object and ω

2.1 The categorical setup

Let $\mathcal{C}$ be a monoidal closed category with an endofunctor $M : \mathcal{C} \to \mathcal{C}$ that preserves directed colimits. Form the standard ω-chain from the initial object $\bot$:

$$\bot \xrightarrow{!} M(\bot) \xrightarrow{M(!)} M^2(\bot) \xrightarrow{M^2(!)} \cdots$$

Let $L = \mathrm{colim}_n\, M^n(\bot)$ be the colimit of this chain. Since M preserves directed colimits, $M(L) \cong \mathrm{colim}_n\, M^{n+1}(\bot) \cong L$. This is the Lambek fixed point: $M(L) \cong L$, with $\mathsf{fold} : M(L) \to L$ and $\mathsf{unfold} : L \to M(L)$ forming an isomorphism.

When $M = [A,-]$ (the internal hom functor) and $A = L$, the Lambek condition becomes $[L,L] \cong L$, or equivalently $D \cong [D,D]$ with $D = L$. This is a reflexive object: every element of D can be interpreted as an endomorphism, and every endomorphism has a representative in D.

The isomorphism generates four derived operations:

- $\mathsf{eval} = \mathsf{unc}(\mathsf{fold}) : L \otimes L \to L$ (evaluation: apply an element of L to another, using the endomorphism interpretation; note that the axiom profile paper (Close 2026b) uses eval for the identity-loop composite $\mathsf{unfold}(\mathrm{id})$, which has type $[D,D]$ rather than $L \otimes L \to L$)
- $\mathsf{selfApp} = \mathsf{unfold} \circ \mathsf{unc}(\mathrm{id}) : L \to L$ (self-application: interpret an element as an endomorphism and apply it to itself)
- $\omega(f) = \mathsf{cur}(\mathsf{selfApp} \circ f)$ (the fixed-point operator, parametric in $f : L \to L$)
- $\mathsf{quine} = \omega(\mathrm{id})$ (the canonical self-referencing element)

The fold/unfold pair satisfies two equations:

$$\mathsf{fold} \circ \mathsf{unfold} = \mathrm{id} \qquad \text{(eq3 — constructive round-trip)}$$

$$\mathsf{unfold} \circ \mathsf{fold} = \mathsf{selfApp} \qquad \text{(eq4 — self-application)}$$

The asymmetry is structural: eq3 is the identity (lossless round-trip), while eq4 generates nontrivial computation (self-application). This asymmetry drives the phenomena in Sections 3–7.

2.2 The fixed-point theorem

Theorem 2.1 (`omega_fixed_point`, axiom profile $\emptyset$). *For any $f : L \to L$, the element $\omega(f)$ satisfies the fixed-point equation $f(\omega(f)) = \omega(f)$.*

The proof unfolds the definition: $\omega(f) = \mathsf{cur}(\mathsf{selfApp} \circ f)$, so evaluating $\omega(f)$ at itself yields $\mathsf{selfApp}(f(\omega(f))) = f(\omega(f))$ by eq4, and the fixed-point equation follows from the interplay of currying, uncurrying, and the iso.

2.3 Uniqueness

Theorem 2.2 (`solution_unique`, axiom profile $\emptyset$). *If $h : L \to L$ satisfies $\mathsf{unfold}(h) \circ \mathsf{selfApp} = \mathsf{selfApp} \circ f$, then $h = \omega(f)$. The fixed point is the unique solution, not just a solution.*

The proof uses three facts: fold is mono (it is a section of the iso), cur is injective (it is one direction of a bijection), and unc is injective (the other direction). Together, these force any solution to equal $\omega(f)$.

Uniqueness is essential for the naming analysis in Section 3. If the fixed-point operator admitted multiple solutions, non-injectivity might be an artifact of a bad choice of constructor rather than a structural property. With uniqueness, if distinct endomorphisms collapse under ω, that collapse is forced by the categorical structure — not by an arbitrary choice.

3 Gap 1 — Naming

3.1 The naming problem

In classical computability theory, programs are named by Godel numbers — an external encoding that assigns a natural number to each program. The encoding is injective: distinct programs get distinct numbers. But the encoding is also arbitrary: infinitely many numbering schemes exist, and the results of computability theory hold for all acceptable numberings (Rogers' isomorphism theorem).

The reflexive object $D \cong [D, D]$ provides an internal alternative. The morphism $\mathsf{fold} : [D, D] \to D$ maps endomorphisms to elements of D. Through the fixed-point operator ω, each endomorphism f gets a canonical representative $\omega(f) \in D$. The question is whether this internal naming system is faithful: do distinct endomorphisms get distinct names?

3.2 The naming theorem

Theorem 3.1 (`omega_injective_iff_selfApp_cancels_endo`, axiom profile $\emptyset$). *ω is injective on* $\mathrm{End}(L)$ *if and only if* $\mathsf{selfApp}$ *is right-cancellable (epi) on endomorphisms: for all $f, g : L \to L$,* $\mathsf{selfApp} \circ f = \mathsf{selfApp} \circ g$ *implies $f = g$.*

The forward direction: if ω is injective and $\mathsf{selfApp} \circ f = \mathsf{selfApp} \circ g$, then $\mathsf{cur}(\mathsf{selfApp} \circ f) = \mathsf{cur}(\mathsf{selfApp} \circ g)$, so $\omega(f) = \omega(g)$, so $f = g$ by injectivity.

The backward direction: if $\mathsf{selfApp}$ is right-cancellable and $\omega(f) = \omega(g)$, then $\mathsf{cur}(\mathsf{selfApp} \circ f) = \mathsf{cur}(\mathsf{selfApp} \circ g)$. Since cur is injective (bijection), $\mathsf{selfApp} \circ f = \mathsf{selfApp} \circ g$. Since $\mathsf{selfApp}$ is right-cancellable, $f = g$.

The condition $\mathsf{selfApp}$-epi is the categorical version of "distinct programs have distinct behaviors under self-application." No external numbering is needed. The reflexive structure provides its own naming system when $\mathsf{selfApp}$ satisfies this cancellation property.

3.3 What naming does and does not give

The naming gap, when closed, establishes that ω is injective: distinct endomorphisms map to distinct elements of L. This is an extensional property — it says the representation is faithful.

What naming does *not* give is cost preservation. Injectivity alone does not imply resource-faithfulness. A Godel numbering, for instance, is injective — distinct programs receive distinct codes — but the code length need not reflect the program's execution cost. Injectivity controls extensional distinguishability; it says nothing about intensional resource reflection.

This distinction — between naming (injectivity) and depth transfer (cost preservation) — is formalized as an independence result in Section 5 and is the structural reason why closing the naming gap does not automatically close the depth-transfer gap.

4 Gap 2 — Construction Cost

4.1 The construction problem

Is building $\omega(f)$ expensive? If fixed-point construction itself has high overhead, the reflexive object might be structurally rich but computationally inaccessible: the naming system would faithfully represent endomorphisms, but the cost of producing those representations would dominate any resource budget.

We equip the category with a cost function $\mathrm{cost} : \mathrm{Mor}(\mathcal{C}) \to \mathbb{N}$ that assigns a non-negative integer to each morphism, satisfying standard properties: $\mathrm{cost}(\mathrm{id}) = 0$, $\mathrm{cost}(g \circ f) \leq \mathrm{cost}(f) + \mathrm{cost}(g)$, and monotonicity under functorial operations.

4.2 The cost theorems

Theorem 4.1 (`cost_omega`, axiom profile {propext, Quot.sound} cosmetic). *For any $f : L \to L$,*

$$\mathrm{cost}(\omega(f)) \leq \mathrm{cost}(f) + \mathrm{cost}(\mathsf{fold}) + \mathrm{cost}(\mathsf{unfold}) + K_{wL} + K_{\mathsf{unc}} + K_{\mathsf{cur}}$$

where K_{wL} (whisker-left), K_{unc} (uncurrying), and K_{cur} (currying) are universal constants depending only on the reflexive object.

The overhead is linear in $\mathrm{cost}(f)$ plus constants from the isomorphism. Construction is cheap.

Theorem 4.2 (`cost_omega_id`, same profile). *The quine $\omega(\mathrm{id})$ has finite cost independent of any input. Self-reference itself costs nothing beyond the isomorphism overhead.*

Theorem 4.3 (`cost_omega_compose`, same profile). *Composing two ω constructions is additive: $\mathrm{cost}(\omega(g \circ f)) \leq \mathrm{cost}(\omega(f)) + \mathrm{cost}(\omega(g)) + K$ for a universal constant K.*

4.3 The placement theorem

These results collectively prove that computational hardness does *not* live in reflexive construction. If the complexity of self-referential computation were intrinsic to the act of self-reference, construction costs would grow with the complexity of the function being fixed. They do not. The overhead is bounded by universal constants.

This is a *placement theorem*: it eliminates one candidate location for hardness (construction) and thereby isolates the remaining candidate (evaluation).

> **Principle 4.4** (Placement). *Cheap closure does not imply cheap elimination.*

Two notions are formally separated:

- **Constructible.** A fixed point can be formed with bounded morphism overhead. *Proved* (Theorems 4.1–4.3).
- **Executable.** The behavior of the fixed point can be evaluated within a resource budget. *Open* — this is where complexity lives.

The gap between constructible and executable is the subject of Section 6.

5 The Orthogonality Theorem

The naming gap (Section 3) and the depth-transfer gap (Section 6) are two properties of the fixed-point operator ω. One might hope that closing the naming gap — establishing that ω is injective — would help with depth transfer. It does not.

Theorem 5.1 (`gap_orthogonality`, axiom profile ∅). *The properties* selfApp-*epi (naming, ω injective) and DepthFaithful (ω preserves evaluation depth) are orthogonal. Each can hold or fail independently of the other.*

The proof constructs four models: one where both hold, one where both fail, and one for each mixed case. The Godel numbering analogy is precise: a numbering is injective (distinct programs get distinct codes) but code length need not reflect execution cost (not depth-faithful). Conversely, the identity function is trivially depth-faithful but injectivity is a separate question depending on selfApp.

Orthogonality means the three-gap decomposition is genuinely three-dimensional. No two gaps collapse into one. Each requires its own analysis.

Theorem 5.2 (`roundtrip_zero_depth`, axiom profile ∅). *The composite* fold ∘ unfold *has zero evaluation depth, while the individual directions* fold *and* unfold *can have arbitrarily large depth.*

This is the verification/extraction asymmetry formalized as a categorical theorem. Verifying that a value is a valid representative (the fold direction, round-trip) is free. Extracting or computing a value (the unfold direction) may be arbitrarily expensive. The round-trip fold ∘ unfold = id costs nothing because it is the identity; but each direction separately participates in selfApp via eq4, which generates unbounded computation.

6 Gap 3 — Depth Transfer and the Growth Gap

The previous sections established two results about the reflexive object $D \cong [D, D]$ and its fixed-point operator ω: the naming gap closes from categorical structure alone (Section 3: ω is injective on $\mathrm{End}(L)$ if and only if selfApp is right-cancellable), and the construction gap closes with bounded overhead (Section 4: $\mathrm{cost}(\omega(f)) \leq \mathrm{cost}(f) + K$ for universal constants K). Neither result addresses a third question: does evaluating the behavior of a fixed point preserve computational cost?

This is the depth-transfer problem. It is where computational content enters the framework.

6.1 The depth-transfer problem

The fixed-point equation $f(\omega(f)) = \omega(f)$ tells us that $\omega(f)$ is a fixed point of f. It does not tell us that the evaluation cost of $\omega(f)$ reflects the complexity of f. An expensive function can have a cheap fixed point: $f(x) = x$ is satisfied by any x, regardless of how complex f is elsewhere.

This distinction is invisible in ungraded settings. In a plain reflexive object, $D \cong [D, D]$ provides a bijection between elements and endomorphisms, and cost is not a meaningful predicate. Under resource constraints — when we equip D with a grading that tracks computational cost — the distinction becomes sharp.

We have established:

- **Naming** (Section 3): selfApp-epi gives ω injective. Distinct endomorphisms get distinct representatives in L. This is the categorical version of "distinct programs have distinct codes."
- **Construction** (Section 4): ω has bounded overhead. Building the fixed point is cheap. This is the categorical version of "the Y combinator is efficient."
- **Orthogonality** (Section 5): Naming and depth transfer are independent properties. Injectivity does not imply cost preservation. A Godel numbering is injective but code length need not reflect execution cost. An injective representation can compress.

The depth-transfer gap asks: when ω maps a grade-g endomorphism to an element of L, does the grade of $\omega(f)$ grow with g? If not, ω compresses the graded structure — many semantically distinct behaviors at high grade are packed into few structural slots at low grade.

6.1.0.1 Why simulation-based arguments fail

One might attempt to prove depth transfer directly: show that if f has high evaluation depth, then selfApp simulating f must also have high depth. This approach fails for a structural reason.

The fixed-point equation is a *collapse*. It equates the output of f at a specific point with that point itself. The cost of reaching $\omega(f)$ via f is not encoded in $\omega(f)$. An expensive function can have a cheap fixed point, and the fixed-point equation cannot distinguish the two cases. No pointwise simulation argument can extract a lower bound from the equation $f(\omega(f)) = \omega(f)$ alone.

This is not a failure of technique. It is a structural feature of fixed-point semantics: the equation identifies a *value*, not a *computation*. The value carries no record of the cost incurred in finding it.

6.2 The anti-compression route

Instead of tracking individual computations, we use a counting argument. The question becomes: can an injective map from a large set fit into a small set? If ω is injective (from selfApp-epi) and has bounded grade overhead, then it maps grade-$\leq g$ endomorphisms into grade-$\leq (g + c)$ elements of L. If the first set is eventually larger than the second, we have a contradiction by pigeonhole.

This shifts the problem from dynamics (how does evaluation behave?) to combinatorics (how do grade slices grow?).

6.2.0.1 Dual filtrations

The argument requires two filtrations:

- A **structural filtration** on L: elements of L are assigned a grade reflecting their structural size. $N_L(g)$ counts the number of elements at structural grade $\leq g$.
- A **semantic filtration** on $\mathrm{End}(L)$: endomorphisms are assigned a grade reflecting their computational complexity. $N_{\mathrm{End}}(g)$ counts the number of endomorphisms at semantic grade $\leq g$.

Both filtrations are strictly proper: each grade adds at least one new element. The structural filtration measures how large an element of L is; the semantic filtration measures how complex an endomorphism is.

6.2.0.2 The grade transfer theorem If ω has bounded grade overhead c — that is, $\mathrm{grade}(\omega(f)) \leq \mathrm{grade}(f) + c$ for all f — then ω maps grade-$\leq g$ endomorphisms into grade-$\leq (g + c)$ elements of L. Combined with injectivity, this gives an injection

$$\{f \in \mathrm{End}(L) \mid \mathrm{grade}(f) \leq g\} \hookrightarrow \{x \in L \mid \mathrm{grade}(x) \leq g + c\}$$

with cardinalities $N_{\mathrm{End}}(g)$ and $N_L(g + c)$ respectively.

6.3 The growth gap hypothesis

Definition 6.3 (HasGrowthGap). *For all overhead constants c, there exists a grade g such that $N_{\mathrm{End}}(g) > N_L(g + c)$.*

The growth gap asserts that the semantic side of the dual filtration eventually outgrows the structural side: for any fixed overhead, there are eventually more grade-g endomorphisms than grade-$(g + c)$ elements of L.

6.3.0.1 Why strict hierarchy alone is insufficient A strictly proper filtration adds at least one new element per grade. But one new element per grade on both sides is compatible with injection: if $N_L(g) = g + 1$ and $N_{\mathrm{End}}(g) = g + 1$, then ω with any finite overhead c maps $g + 1$ elements into $(g + c) + 1$ slots. No contradiction arises. The growth gap requires *differential growth* — the semantic filtration must grow strictly faster than the structural filtration.

6.3.0.2 Why $D \cong [D, D]$ resists the growth gap The reflexive isomorphism $D \cong [D, D]$ provides an extensional bijection between elements and endomorphisms globally. A global counting argument — comparing $|L|$ with $|\mathrm{End}(L)|$ — is therefore dead: the isomorphism guarantees they have the same cardinality.

The growth gap must be *slice-wise*. It asserts not that there are more endomorphisms than elements in total, but that endomorphisms are distributed more densely at each grade level. This is possible only if the reflexive isomorphism is

not grade-preserving — that is, the bijection $D \cong [D, D]$ shifts grades, mapping low-grade elements to high-grade endomorphisms and vice versa.

Whether the isomorphism is shift-bounded on the filtration — whether there exists a uniform bound on the grade shift — is the key structural question. If the iso shifts grades by at most s, then the growth gap can coexist with the iso: elements and endomorphisms are equinumerous globally, but the iso's grade shift means they are not equidistributed across slices.

6.4 The anti-compression theorem

The formalization proceeds in two layers, separating the combinatorial core from the categorical instantiation.

Layer 1 (abstract, `AntiCompressionSchema`). Pure $\mathbb{N}$ arithmetic. Given:

- An injection from a set of size $N_{\text{End}}(g)$ into a set of size $N_L(g + c)$,
- A growth gap: $N_{\text{End}}(g) > N_L(g + c)$ for some g,

conclude $\bot$ by pigeonhole. This layer has no categorical content; it is a finite combinatorial fact.

Layer 2 (categorical, `graded_anti_compression`). Connect the categorical hypotheses to Layer 1:

- selfApp-epi gives ω injective (Section 3).
- Bounded grade transfer gives ω maps $\text{End}_{\leq g}(L)$ into $L_{\leq (g+c)}$.
- The growth gap gives $|\text{End}_{\leq g}(L)| > |L_{\leq (g+c)}|$ for some g.
- Apply Layer 1.

Theorem 6.1 (`graded_anti_compression`, axiom profile $\emptyset$). selfApp-*epi* + *bounded* ω *grade transfer* + *HasGrowthGap* $\implies \bot$.

The proof is short once the layers are in place. The work is in the setup: establishing that each categorical hypothesis maps cleanly to the abstract schema's requirements.

6.4.0.1 The three-way incompatibility The anti-compression theorem asserts that three conditions — selfApp-epi (naming), bounded ω (construction), and the growth gap (evaluation) — are jointly contradictory. Any two can coexist:

- **Naming + construction, no growth gap.** The reflexive object names its endomorphisms and builds fixed points cheaply, but the semantic filtration does not outgrow the structural filtration. Grade slices grow in lockstep. This is the Unit model (Section 7).
- **Naming + growth gap, unbounded construction.** The reflexive object names its endomorphisms and the semantic side outgrows the structural side, but fixed-point construction has unbounded overhead. The

injection from $\mathrm{End}_{\leq g}$ to $L_{\leq(g+c)}$ fails because ω does not respect the grade bound.

- **Construction + growth gap, no naming.** The reflexive object builds fixed points cheaply and the semantic side outgrows the structural side, but ω is not injective. Multiple endomorphisms share a representative, so the injection premise fails.

All three cannot hold simultaneously. This is the precise shape of the obstruction: a resource-bounded reflexive object cannot simultaneously name its endomorphisms faithfully, construct their fixed points cheaply, and accommodate the differential growth of semantic complexity.

6.4.0.2 What the theorem measures The anti-compression theorem is a *measurement*, not a conjecture. It identifies the exact boundary of what the categorical structure can enforce:

- Two of the three conditions are proved from the categorical axioms (Sections 3 and 4).
- The third (the growth gap) is not derivable (Section 7.1) and not inconsistent (Section 7.2).
- The theorem says: *if* the growth gap holds, *then* the three conditions are contradictory.

The remaining question — does the growth gap hold in specific computational substrates? — is precisely located. It is the single hypothesis separating the categorical framework from concrete impossibility results. The framework has done everything it can do alone.

7 Conservativity and Independence

The anti-compression theorem (Theorem 6.1) shows that selfApp-epi, bounded ω, and the growth gap are jointly contradictory. A natural question arises: is the growth gap actually needed? Could the categorical structure alone force the contradiction, making the growth gap hypothesis redundant?

It cannot. The growth gap is a genuine hypothesis — independent of the categorical structure.

7.1 The growth gap is not derivable

Theorem 7.1 (conservativity via Unit model, axiom profile $\emptyset$). *There exists a model satisfying* ReflCat + selfApp-*epi* + GradedReflData *in which* HasGrowthGap *fails.*

The model is simple: take the Unit category (a single object with only the identity morphism). All morphisms are id, all grades collapse to zero. The reflexive condition $D \cong [D, D]$ holds trivially ($\mathrm{id} : 1 \to 1$ in both directions).

selfApp = id is vacuously right-cancellable (there is only one endomorphism). The graded data is trivial: $N_L(g) = 1$ and $N_{\mathrm{End}}(g) = 1$ for all g.

In this model, the growth gap fails: $N_{\mathrm{End}}(g) = 1 \leq 1 = N_L(g + c)$ for all g and c. There is no grade at which endomorphisms outgrow elements, because there is only one of each.

The Unit model confirms that the growth gap is not a consequence of the categorical axioms. The anti-compression theorem is not vacuously true from categorical structure alone — it genuinely requires the growth gap as a hypothesis.

7.2 The growth gap is consistent

Theorem 7.2 (consistency via mesoscopic model, axiom profile {propext, Quot.sound} cosmetic). *There exists a computationally meaningful model satisfying all of* ReflCat + selfApp-*epi* + GradedReflData + HasGrowthGap.

The mesoscopic model uses binary programs as its computational substrate. Grade-g elements of L are binary strings of length exactly $g + 1$, giving a per-grade slice count $S_L(g) = 2^{g+1}$ (writing S_L for the slice count to distinguish from the cumulative N_L of Definition 6.3). Grade-g endomorphisms are all functions between such strings, giving $S_{\mathrm{End}}(g) = (2^{g+1})^{2^{g+1}}$. The growth gap argument applies to either convention since both grow at least exponentially. The dual filtrations are:

$$S_L(g) = 2^{g+1} \qquad\qquad S_{\mathrm{End}}(g) = (2^{g+1})^{2^{g+1}}$$

The growth gap exhibits tower-exponential separation at every overhead. At grade $g = 1$, $S_{\mathrm{End}}(1) = 4^4 = 256$ while $S_L(1) = 4$. At grade $g = 2$, $S_{\mathrm{End}}(2) = (2^3)^{2^3} = 8^8 = 16{,}777{,}216$ while $S_L(2) = 2^3 = 8$. The carrier grows exponentially, but the endomorphism space grows as an exponential tower — no finite overhead c can close the gap, since 2^{g+1+c} remains negligible against $(2^{g+1})^{2^{g+1}}$.

The model has genuine self-referential structure: binary programs can encode other binary programs, self-application is well-defined, and selfApp-epi holds because distinct programs produce distinct self-application results. The reflexive isomorphism is non-uniform — it shifts grades by an amount that grows with g, mapping low-grade binary strings to the high-grade endomorphisms they represent. This non-uniformity is necessary, as Theorem 7.3 below establishes.

The anti-compression theorem fires: naming, bounded construction, and the growth gap all hold, yielding $\bot$. This confirms the theorem is non-vacuous with a computationally meaningful witness.

Remark. A simpler pedagogical model uses lookup tables with $N_L(g) = g + 1$ and $N_{\mathrm{End}}(g) = (g + 1)^{g+1}$ (`ToyInstantiation.lean`). The mesoscopic model is preferred as the primary non-vacuity witness because its binary encoding, exponential carrier, and self-referential structure are closer to actual computation.

Combined with Theorem 7.1 (the Unit model where the growth gap fails), we have full two-sided independence: the growth gap is neither derivable from nor inconsistent with the categorical axioms.

Since the slice-count separation $S_{\mathrm{End}}(g) > S_L(g + c)$ implies the cumulative separation $N_{\mathrm{End}}(g) > N_L(g + c)$ (any slice exceeding its shifted counterpart forces the cumulative sums to separate), the growth gap of Definition 6.3 follows.

7.3 The non-uniformity theorem

The Unit model (Regime A, no growth gap) and the mesoscopic model (Regime B, growth gap) are not merely counterexamples. They represent two structurally distinct regimes, and the distinction is now a formal theorem.

Theorem 7.3 (`graded_non_uniformity`, axiom profile $\emptyset$). *Uniform graded reflexive isomorphisms and quantitative anti-compression are incompatible. If the reflexive isomorphism $D \cong [D, D]$ is grade-preserving (shifts grades by a bounded constant uniformly across all grades), then the growth gap cannot hold. Computational growth gaps require graded non-uniformity in the isomorphism.*

The proof proceeds by contradiction. If fold and unfold each shift grades by at most a constant s, then the isomorphism maps grade-g elements to grade-$\leq (g+s)$ endomorphisms and vice versa. This means grade slices of L and $\mathrm{End}(L)$ have comparable cardinalities at each level (up to the constant shift), which directly contradicts the growth gap's requirement that $N_{\mathrm{End}}(g) > N_L(g + c)$ eventually for all c.

Both regimes are inhabited:

Regime A (denotational, uniform iso). The reflexive isomorphism is grade-preserving. Elements and endomorphisms are equidistributed across grade slices. The growth gap fails. The Unit model is the degenerate case; Scott domain models are the paradigmatic examples. Evaluation is topological rather than computational.

Regime B (computational, non-uniform iso). The reflexive isomorphism shifts grades by an amount that grows with g. Endomorphisms are denser than elements at each grade slice. The growth gap holds. The anti-compression theorem fires. The mesoscopic model (Theorem 7.2) is the primary example. Evaluation has genuine computational cost.

The non-uniformity theorem gives the paper's explanatory conclusion: the growth gap is not an arbitrary hypothesis but a structural consequence of the isomorphism's graded behavior. Uniform reflexive semantics cannot support quantitative anti-compression. The computational content enters exactly where the isomorphism becomes non-uniform.

7.4 The growth gap is forced by representability

The independence results (Theorems 7.1–7.2) show that the growth gap is neither derivable from nor inconsistent with the bare categorical axioms. But the bare categorical axioms are deliberately weak — they describe any reflexive object, including degenerate ones. For *computational* models, where elements represent finite data and endomorphisms represent programs, a stronger result holds.

Theorem 7.4 (`standard_model_gap`, axiom profile $\emptyset$). *If a graded reflexive object satisfies finite-table representability — that is, the number $N_L(g)$ of grade-$\leq g$ values is finite with $N_L(g) \geq 2$ for all g, and grade-g endomorphisms include all functions between grade-$\leq g$ values — then the growth gap holds.*

The proof is direct: finite-table representability gives $N_{\mathrm{End}}(g) \geq N_L(g)^{N_L(g)}$, since the endomorphism count includes all functions between the $N_L(g)$ values. For any overhead constant c, $N_L(g)^{N_L(g)}$ eventually exceeds $N_L(g+c)$ whenever $N_L(g) \to \infty$ (equivalently, whenever N_L is eventually ≥ 2). This is the standard counting argument: the number of programs over a finite alphabet grows faster than the alphabet itself.

Theorem 7.5 (`partrec_code_instance`, axiom profile $\emptyset$). *`Nat.Partrec.Code` — Lean's formalization of partial recursive function codes — satisfies the finite-table representability condition. The growth gap holds for partial recursive functions under code-length grading.*

This is the concrete bridge from abstract theory to standard computability. The growth gap is not a hypothesis for models with genuine program-data structure — it is a theorem.

7.5 Independence and instantiation summary

The growth gap hypothesis occupies a layered position:

- **Not derivable from bare categorical axioms.** The Unit model (Theorem 7.1) satisfies all categorical axioms but the growth gap fails. The growth gap is not a formal consequence of reflexivity alone.
- **Forced by representability.** Any model where elements are finite-table representable and endomorphisms include all table functions satisfies the growth gap (Theorem 7.4). This covers standard coding-based models; `Nat.Partrec.Code` is the concrete instance proved here.
- **Instantiated for `Nat.Partrec.Code`.** The growth gap holds concretely for Lean's partial recursive functions (Theorem 7.5).

The open frontier has moved. The growth gap is forced by finite-table representability, satisfied by standard coding-based models. The question is now what follows at polynomial bounds: does the anti-compression obstruction survive when "bounded construction" means polynomial-time construction and "grading" tracks polynomial resource usage?

8 The ISP Connection

We sketch the ISP application as a diagnostic illustration of the framework's reach; the full classification with Lean 4 proofs appears in (Close 2026d).

The invariant subspace problem (ISP) — whether every bounded linear operator on a separable Hilbert space has a nontrivial closed invariant subspace — can be cast as a fixed-point problem on the lattice of closed subspaces, with invariant subspaces as fixed points of the orbit-closure endofunctor F.

Theorem 8.1 (`adamek_chain_trivial`, axiom profile $\emptyset$). *If $F(\bot) = \bot$, the standard ω-chain from $\bot$ is constant at $\bot$. The Adamek construction finds only the trivial fixed point.*

Proof sketch. The ω-chain is $\bot, F(\bot), F^2(\bot), \ldots$ If $F(\bot) = \bot$, then by induction every iterate equals $\bot$: the chain is constant. Its colimit is $\bot$, which is the least fixed point but trivial. The orbit-closure endofunctor satisfies $F(\bot) = \bot$ because the orbit closure of the zero subspace contains only zero vectors. □

This correctly diagnoses why the standard categorical tool fails: ISP requires *intermediate* fixed points, not least fixed points. Orbit chains from arbitrary starting points provide the right framework, reducing ISP to two conditions: **stabilization** (the chain eventually stops growing) and **properness** (the stabilization point lies strictly between $\bot$ and $\top$). Known positive results — compact operators (Aronszajn and Smith 1954), normal operators via the spectral theorem — are recovered as instances where additional structure guarantees these two conditions.

Theorem 8.2 (`isp_independent_of_lattice`, axiom profile $\emptyset$). *Both ISP success and ISP failure are constructively satisfiable in lattice models.*

Proof sketch. For ISP success: using $F = \mathrm{id}$ on the three-element lattice $\bot < m < \top$, the element m is a nontrivial fixed point, providing a model where ISP holds. For ISP failure: the two-element lattice $\bot < \top$ admits the identity endofunctor, whose only fixed points are $\bot$ and $\top$ — both trivial. □

The parallel to the computability setting is structural: in both cases the categorical framework identifies the precise conditions needed, proves those conditions are independent of the ambient structure, and recovers known positive results as instances. A deeper analysis, including a deflation theorem providing new sufficient conditions for nontrivial fixed points, a lattice intermediate value theorem, and a complete classification of chain-based proof strategies, is developed in (Close 2026d).

9 Related Work

9.1 Implicit complexity theory

The closest prior work on resource-bounded reflexivity is bounded linear logic (BLL) (Girard, Scedrov, and Scott 1992). BLL characterizes polynomial-time computation as the functions typable in a linear logic where the exponential ! is bounded by a polynomial. The reflexive object $D \cong [D, D]$ requires unbounded ! for the self-application $\mathsf{selfApp} : D \to D$, since self-application applies an element to itself through the internal hom. Our framework explains this requirement: the full reflexive construction requires Levels 3–4 (closure through computation) and the full exponential. Under polynomial bounds, the tower breaks at precisely these levels.

Baillot and Mazza (2010) develop linear logic by levels, stratifying proof nets by depth. Their levels correspond to our grade filtration, and their stratification conditions correspond to our shift-bounded isomorphism. The connection suggests that their stratified models may provide natural candidates for growth gap instantiation.

Dal Lago and Hofmann (2010) revisit BLL with quantified bounds on resource variables (QBAL). Their quantified bounds are a specific instantiation of our `GradedReflData` — the graded structure that assigns cardinalities to grade slices.

The connection is now precise: `BLLBridge.lean` proves that BLL's bounded exponential forces the growth gap (Appendix A: `bll_bounded_bang_forces_gap`). The mechanism is the cost of contraction: in BLL, contraction (duplicating a resource) requires spending !, and bounding ! polynomially limits how many copies of a value can be produced at each grade. This directly generates the differential growth between values (bounded by the polynomial) and programs (functions between polynomially many values, growing as a tower). The growth gap is the cost of contraction in the graded framework.

9.2 Constructive computability theory

Bauer (2006) develops synthetic computability theory, working in a constructive metatheory where all functions are assumed computable. Forster (2021) shows that synthetic computability theory eliminates the gap between $\neg\neg\exists$ and $\exists$ (the Markov boundary) by internalizing the Church–Turing thesis.

The axiom profile analysis (Close 2026b) shows that the boundary between constructive and classical computability is foundation-dependent: the three-layer partition ($\emptyset$, Markov, EM) is exact in the equational setting but collapses in the synthetic setting. The present paper extends this observation: the complexity boundary (the growth gap) is similarly foundation-dependent. Whether the growth gap holds or fails may depend on the ambient axioms and the specific computational substrate, not just the categorical structure. The boundary is

foundation-dependent in the sense that different formalizations may shift it, but within any fixed formalization it is exact and determined by mathematical content rather than engineering choices.

9.3 Complexity barriers

The theory of complexity barriers identifies structural limitations on proof techniques for separating complexity classes.

Baker, Gill, and Solovay (1975) show that relativization cannot separate P from NP: there exist oracles relative to which P = NP and oracles relative to which P $\neq$ NP. Razborov and Rudich (1997) show that natural proofs cannot prove superpolynomial circuit lower bounds if one-way functions exist. Aaronson and Wigderson (2009) extend relativization to algebrization.

Our framework operates at a different level: categorical equations rather than machine models or Boolean functions. The barrier diagnostics are now formalized. `OracleSemantics.lean` proves that the growth gap trivially relativizes: it holds relative to any oracle, since it is a counting property of grade slices that is preserved under oracle extension. This means the growth gap does not conflict with the Baker–Gill–Solovay relativization barrier — it passes through it. Oracle extension can only add endomorphisms at each grade, preserving or widening the differential growth. `ConstructiveBarriers.lean` formalizes the category mismatch with natural proofs: the anti-compression theorem operates on morphism spaces in a category, not on Boolean function families, so the Razborov–Rudich largeness condition does not apply. The circuit track — translating the graded framework to circuit complexity via depth as grade — is identified as the live path for connecting to existing lower bound programs.

9.4 Self-interpreters and reflexive objects

Brown and Palsberg (2016) construct self-interpreters in the simply-typed lambda calculus, challenging the assumption that self-interpretation requires untyped or dependently-typed systems. Jones (2004) analyzes efficient self-interpretation, showing that the overhead of self-interpretation can be bounded.

Our cost theorems (Section 4) give the categorical version of these results: the construction cost of the self-interpreter (ω) is bounded by the cost of the function plus universal constants. The categorical framework makes the separation precise: *construction* of the self-interpreter is cheap (Theorem 4.1), but *evaluation* — running the self-interpreter on inputs — is where unbounded cost enters (Theorem 5.2).

9.5 Invariant subspace problem

Enflo (1987) constructs a counterexample to ISP for general Banach spaces: a bounded linear operator on a Banach space with no nontrivial closed invariant subspace. The problem remains open for separable Hilbert spaces.

Aronszajn and Smith (1954) prove ISP for compact operators. Lomonosov (1973) extends this to operators commuting with a nonzero compact operator. Our framework diagnoses the structural reason: the Adamek chain from $\bot$ gives only the trivial fixed point (Theorem 8.1), so nontrivial fixed points must come from orbit chains with appropriate starting points. Compactness and commutativity with compact operators provide exactly the starting points and stabilization guarantees that the orbit-chain analysis requires.

10 Discussion

10.1 What the framework does

The three-gap decomposition provides a precise anatomy of the complexity of self-referential computation:

- **Locates obstructions precisely.** The hardness is not in construction (bounded, Section 4), not in naming (internal, Section 3), but in evaluation depth via the growth gap (Section 6).
- **Proves logical sufficiency.** The anti-compression theorem (Theorem 6.1) shows that naming, bounded construction, and the growth gap are jointly contradictory. These three conditions are sufficient for impossibility.
- **Establishes independence.** The growth gap is not derivable from the categorical structure (Theorem 7.1) and not inconsistent with it (Section 7.2). The framework has done everything a categorical framework can do.
- **Eliminates external encoding.** The naming gap closes from internal structure (selfApp-epi), removing the dependency on Godel numbering or other external encodings.
- **Proves the verification/extraction asymmetry.** The round-trip $\mathsf{fold} \circ \mathsf{unfold}$ has zero depth while the individual directions can be arbitrarily deep (Theorem 5.2). Verifying is free; extracting is expensive.
- **Diagnoses ISP.** The framework correctly identifies why the Adamek chain fails, what conditions are needed (stabilization + properness), and recovers known positive results as instances.

10.2 What the framework does not do

- **Does not prove P $\neq$ NP.** The growth gap is now proved for coding-based models (Theorem 7.4), and the witness-extraction bridge identifies P = NP as polynomial Markov (Theorem 10.1). But the anti-compression theorem at polynomial bounds requires polynomial-time bounded construction, which is not yet established. The frontier is the polynomial enrichment of the three-gap decomposition.

- **Does not solve ISP.** The orbit-chain conditions (stabilization + properness) are identified but not proved for general bounded operators on separable Hilbert space. The framework diagnoses the structure; the resolution requires analytic methods.
- **Does not provide circuit-specific structure.** The framework recovers the Shannon counting envelope for circuits (most Boolean functions require exponential circuits) but does not automatically supply circuit-specific sensitivity bounds. `AC0Parity.lean` confirms this honest limitation: the categorical grade structure captures depth but not the gate-fan-in constraints that drive AC^0 lower bounds. The circuit track requires additional structure beyond the graded category.

10.3 The conceptual contribution

The sentence that captures this paper:

Naming and fixed-point construction close internally. Quantitative obstruction enters exactly at graded anti-compression, and this requires non-uniform reflexive isomorphism. The growth gap is the formal dividing line between denotational semantics (uniform iso, no obstruction) and computational semantics (non-uniform iso, anti-compression fires).

The three-gap decomposition reframes the question of computational complexity from "why is self-reference hard?" (it is not) to "why does evaluation resist resource bounds?" (because of differential growth in graded slices). The answer is not a new conjecture but a structural measurement, formalized and machine-verified.

10.4 The witness-extraction bridge

The axiom profile analysis (Close 2026b) identifies three layers of computability theory, tracking the double-negation monad's counit: Layer 0 ($\emptyset$, proofs on values), Layer 1 (Markov, extraction of $\exists$ from $\neg\neg\exists$), Layer 2 (EM, determination of arbitrary propositions). Rice's theorem lives at Layer 1: it requires Markov's principle to extract a witness from a double negation.

The polynomial enrichment of Layer 1 is P vs NP. The following theorems are formalized in `PolyMarkov.lean` and `WitnessAsymmetry.lean` respectively; we state them here in the context of the broader program.

Theorem 10.1 (`poly_markov_iff_p_eq_np`, axiom profile $\emptyset$). *Markov's principle at polynomial bounds — the assertion that for any polynomial-time decidable predicate, if a witness is not-not-existent then a witness can be found in polynomial time — is equivalent to $P = NP$.*

The forward direction: if polynomial Markov holds, then any NP witness (whose existence is verified in polynomial time) can be extracted in polynomial time, so $NP \subseteq P$. The backward direction: if $P = NP$, then the polynomial-time search

that P = NP guarantees provides the witness extraction that polynomial Markov asserts.

The equivalence is genuine — the two sides are formulated differently (Poly-Markov speaks of witness finders for NPWitness structures; P = NP speaks of class inclusion InNP → InP) and connected through axiomatized search-to-decision and finder-to-decider reductions (Bellare–Goldwasser 1994). The equivalence should be understood as identifying P = NP as a resource-bounded witness-extraction principle, providing structural location in the axiom profile hierarchy rather than new traction on proving or disproving the conjecture.

This makes the two papers one project. The axiom profile paper measures which logical axioms computability theorems need. This paper measures which resource axioms complexity theorems need. The boundary is the same boundary — fold/unfold, construction/evaluation, witness exhibition/witness extraction — viewed at different resource scales.

Theorem 10.2 (`witness_asymmetry`, axiom profile ∅). *The witness-extraction asymmetry is a general phenomenon, abstractly parameterized by a resource bound. Three instances are proved:*

- *Logical instance: Markov's principle for Σ_1^0 (Layer 1 of the axiom profile).*
- *Polynomial instance: P = NP (polynomial Markov, Theorem 10.1).*
- *BLL instance: bounded contraction forces the growth gap (the cost of witness duplication under bounded !).*

All three are instances of a single abstract witness system, differing only in the resource bound.

10.5 Open problems

1. **Anti-compression at polynomial bounds.** The growth gap holds for coding-based models (Theorem 7.4). The anti-compression theorem fires at unbounded resource levels. Does it survive when construction cost is restricted to polynomial time and grading tracks polynomial resource usage? This is the main open problem — establishing the polynomial enrichment of the three-gap decomposition.

2. **Circuit-depth instantiation.** The graded framework uses depth as a natural grade. Translating the anti-compression theorem to circuit complexity — where grade is circuit depth and the growth gap becomes a statement about the number of depth-d circuits vs depth-d truth tables — is the live path identified by the barrier diagnostics (Section 9.3). AC^0 lower bounds (parity) require gate-fan-in structure beyond the bare graded category.

3. **Polynomial Markov and separation.** Theorem 10.1 identifies P = NP as polynomial Markov. The contrapositive — P ≠ NP iff polynomial

Markov fails — means that a separation result would follow from showing that polynomial-time witness extraction is not uniformly available. The witness-asymmetry framework (Theorem 10.2) provides the abstract structure; the polynomial instance is the target.

4. **ISP: new sufficient conditions.** The orbit-chain analysis identifies stabilization and properness as the two conditions for nontrivial invariant subspaces. Are there sufficient conditions beyond compactness and normality that guarantee these conditions? See (Close 2026d) for the current state.

5. **Resource axiom profiles.** The axiom profile methodology (Close 2026b) tracks which logical axioms each theorem requires. The polynomial Markov bridge (Theorem 10.1) shows that resource bounds can be tracked analogously. A systematic resource-axiom profile for complexity-theoretic results would provide a finer-grained map of the complexity landscape.

11 Conclusion

The complexity of self-referential computation decomposes into three independent gaps. The naming gap asks whether the reflexive object can distinguish its own endomorphisms; it closes from categorical structure when selfApp is right-cancellable. The construction gap asks whether fixed-point formation is expensive; it closes with bounded overhead, linear in the cost of the function being fixed. The depth-transfer gap asks whether evaluation preserves cost; it reduces to the growth gap hypothesis about differential growth in graded slices.

The anti-compression theorem bridges all three: naming, bounded construction, and the growth gap are jointly contradictory. Any two can coexist; all three cannot. The non-uniformity theorem sharpens this: growth gaps require non-uniform reflexive isomorphisms, giving a sharp dichotomy between denotational and computational regimes. The growth gap is proved for any model with finite-table representability and instantiated for `Nat.Partrec.Code`. Bounded linear logic's bounded exponential forces the gap via the cost of contraction.

The witness-extraction bridge connects this paper to the axiom profile analysis: Markov's principle at polynomial bounds is equivalent to P = NP. The boundary between constructible and executable — between fold and unfold, between verification and extraction — is the same boundary at every resource scale, from computability (Layer 1) through polynomial complexity (P vs NP) to bounded linear logic (bounded !).

The formalization is 93 files of Lean 4, zero `sorry`, zero `Classical.choice`, zero custom axioms. The three-gap decomposition, the anti-compression theorem, the non-uniformity dichotomy, the representability instantiation, and the polynomial Markov bridge are all proved — the first three at profile $\emptyset$ (constructive), the cost bounds and mesoscopic model at cosmetic profile only. The open frontier is

the polynomial enrichment of the three-gap decomposition.

This is a measurement, not a conjecture.

12 References

Aaronson, S. and Wigderson, A. (2009). Algebrization: a new barrier in complexity theory. *ACM Transactions on Computation Theory* 1(1), 2:1–54.

Aronszajn, N. and Smith, K. T. (1954). Invariant subspaces of completely continuous operators. *Annals of Mathematics* 60(2), 345–350.

Baillot, P. and Mazza, D. (2010). Linear logic by levels and bounded time complexity. *Theoretical Computer Science* 411(2), 470–503.

Baker, T., Gill, J., and Solovay, R. (1975). Relativizations of the P =? NP question. *SIAM Journal on Computing* 4(4), 431–442.

Bauer, A. (2006). First steps in synthetic computability theory. In *Proceedings of the 21st Annual Conference on Mathematical Foundations of Programming Semantics (MFPS XXI)*, Electronic Notes in Theoretical Computer Science 155, 5–31.

Brown, M. and Palsberg, J. (2016). Breaking through the normalization barrier: a self-interpreter for F-omega. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '16)*, 5–17.

Close, L. J. (2026a). The Point. DOI:10.5281/zenodo.18878239. Lean 4 formalization, 42 files.

Close, L. J. (2026b). The Axiom Profile of Computation. Manuscript. Lean 4 formalization, 20 standalone files.

Close, L. J. (2026d). Chain Obstructions and Deflation in the Invariant Subspace Problem. Manuscript.

Close, L. J. (2026e). Witness Extraction Asymmetry Across Logic, Complexity, and Fixed-Point Mathematics. Lean 4 formalization, 93 files. DOI:10.5281/zenodo.18915083.

Dal Lago, U. and Hofmann, M. (2010). Bounded linear logic, revisited. *Logical Methods in Computer Science* 6(4).

Enflo, P. (1987). On the invariant subspace problem for Banach spaces. *Acta Mathematica* 158, 213–313.

Forster, Y. (2021). Computability in constructive type theory. PhD thesis, Saarland University.

Girard, J.-Y., Scedrov, A., and Scott, P. J. (1992). Bounded linear logic: a modular approach to polynomial-time computability. *Theoretical Computer Science* 97(1), 1–66.

Jones, N. D. (2004). The expressive power of higher-order types or, life without CONS. *Journal of Functional Programming* 14(1), 73–94.

Lomonosov, V. I. (1973). Invariant subspaces of the family of operators that commute with a completely continuous operator. *Functional Analysis and Its Applications* 7(3), 213–214.

Razborov, A. A. and Rudich, S. (1997). Natural proofs. *Journal of Computer and System Sciences* 55(1), 24–35.

13 Appendix A: Theorem Inventory

All theorems are formalized in Lean 4 with the indicated axiom profiles. Profile $\emptyset$ means no axioms beyond the Lean kernel. Profile {propext, Quot.sound} (marked "cosmetic") means only the axioms that Lean introduces automatically for propositional extensionality and quotient soundness; these carry no mathematical content.

Theorem	Axiom Profile	File
`omega_fixed_point`	$\emptyset$	`ComplexityBoundary.lean`
`solution_unique`	$\emptyset$	`OmegaAlgebra.lean`
`omega_injective_iff_selfApp_cancels_endo`	$\emptyset$	`OmegaAlgebra.lean`
`cost_omega`	cosmetic	`ComplexityBoundary.lean`
`cost_omega_id`	cosmetic	`ComplexityBoundary.lean`
`cost_omega_compose`	cosmetic	`ComplexityBoundary.lean`
`cost_selfApp`	cosmetic	`ComplexityBoundary.lean`
`cost_reflexiveCurry`	cosmetic	`ComplexityBoundary.lean`
`cost_iterate`	cosmetic	`ComplexityBoundary.lean`
`gap_orthogonality`	$\emptyset$	`EpiReflexive.lean`
`roundtrip_zero_depth`	$\emptyset$	`ComplexityReflexive.lean`
`no_bounded_universal_self_interpreter`	$\emptyset$	`ComplexityReflexive.lean`
`graded_anti_compression`	$\emptyset$	`AntiCompression.lean`
Unit conservativity witness	$\emptyset$	`AntiCompression.lean`
`adamek_chain_trivial`	$\emptyset$	`InvariantSubspace.lean`
`isp_independent_of_lattice`	$\emptyset$	`InvariantSubspace.lean`
`orbit_nontrivial_if_bounded`	$\emptyset$	`InvariantSubspaceOrbits.lean`
`toy_refl_cat`	$\emptyset$	`ToyInstantiation.lean`
`toy_selfApp_epi`	$\emptyset$	`ToyInstantiation.lean`
`toy_graded_data`	$\emptyset$	`ToyInstantiation.lean`
`toy_growth_gap`	$\emptyset$	`ToyInstantiation.lean`
`toy_anti_compression_fires`	$\emptyset$	`ToyInstantiation.lean`
`meso_refl_cat`	cosmetic	`MesoscopicModel.lean`
`meso_selfApp_epi`	cosmetic	`MesoscopicModel.lean`
`meso_graded_data`	cosmetic	`MesoscopicModel.lean`
`meso_growth_gap`	cosmetic	`MesoscopicModel.lean`
`meso_anti_compression_fires`	cosmetic	`MesoscopicModel.lean`

Theorem	Axiom Profile	File
graded_non_uniformity	∅	GradedNonUniformity.lean
uniform_prevents_growth_gap	∅	GradedNonUniformity.lean
regime_A_inhabited	∅	GradedNonUniformity.lean
regime_B_inhabited	∅	GradedNonUniformity.lean
standard_model_gap	∅	StandardModelGap.lean
finite_table_forces_gap	∅	StandardModelGap.lean
partrec_code_instance	∅	PartrecCodeInstance.lean
partrec_growth_gap	∅	PartrecCodeInstance.lean
bll_bounded_bang_forces_gap	∅	BLLBridge.lean
contraction_cost_growth	∅	BLLBridge.lean
poly_markov_iff_p_eq_np	∅	PolyMarkov.lean
poly_markov_forward	∅	PolyMarkov.lean
poly_markov_backward	∅	PolyMarkov.lean
witness_asymmetry	∅	WitnessAsymmetry.lean
witness_logical_instance	∅	WitnessAsymmetry.lean
witness_poly_instance	∅	WitnessAsymmetry.lean
witness_bll_instance	∅	WitnessAsymmetry.lean
growth_gap_relativizes	∅	OracleSemantics.lean
natural_proof_mismatch	∅	ConstructiveBarriers.lean
circuit_track_identified	∅	ConstructiveBarriers.lean
ac0_shannon_envelope	∅	AC0Parity.lean
ac0_gate_structure_needed	∅	AC0Parity.lean