GrokRxiv:2026.04.hott-hallucination-synthesis [cs.AI / math.AT] 14 Apr 2026

The HoTT Hallucination Framework

A Unified Theory of Semantic Correctness for Language Models
via Homotopy Type Theory

Matthew Long*
YonedaAI Research Collective
Magneton Labs LLC
Chicago, IL, USA

April 2026

Abstract

We present a unified framework for semantic correctness in large language models (LLMs) that synthesizes two complementary research threads: *topological detection* of hallucination via homotopy groups and homology, and *type-theoretic generation* via certified inhabitation search in Homotopy Type Theory (HoTT). Our central construction is the **detection-generation duality**: detection is the problem of checking whether a global section of a semantic sheaf exists, while generation is the problem of constructing such a section. We prove these operations form an adjoint pair $\mathsf{Generate} \dashv \mathsf{Detect}$ on the semantic category **Sem**, and that the composite $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$ is a monad on **Sem**—the **semantic monad**—whose unit maps semantic types to their verified versions and whose multiplication collapses redundant verification. We formalize the ambient mathematical universe as an ∞-topos of semantic types, where objects are homotopy types, morphisms are certified derivations, and the internal logic is HoTT itself. Within this framework, we prove the **Completeness Theorem**: the Hallucination–Homotopy Correspondence (HHC) together with type-theoretic generation yields a *complete* system in which every hallucination is detected and generation never produces undetected hallucinations. We provide a concrete proof-of-concept architecture wrapping an LLM with a type-checking layer implementing a generate-check-abstain loop, with persistent homology for real-time topological monitoring. The framework is formalized in Haskell.

Keywords: Homotopy Type Theory, ∞-Topos, Adjoint Functors, Monads, Sheaf Cohomology, LLM Hallucination, Persistent Homology, Algebraic Topology, Categorical Semantics, Type Inhabitation

Contents

1 Introduction 2

*Corresponding author. matthew@yonedaai.com • https://yonedaai.com

1.1 Contributions . . . 2
1.2 Related Work . . . 3
1.3 Paper Outline . . . 3

2 Review of the Two Threads **3**
2.1 Thread 1: Topological Detection . . . 3
2.2 Thread 2: Type-Theoretic Generation . . . 4
2.3 The Gap: What Neither Thread Provides Alone . . . 5

3 The Detection-Generation Duality **5**
3.1 The Semantic Category Revisited . . . 5
3.2 Detection as a Functor . . . 6
3.3 Generation as a Functor . . . 6
3.4 The Adjunction . . . 6

4 The Semantic Monad **8**
4.1 Construction . . . 8
4.2 Semantic Interpretation of the Monad . . . 9
4.3 The Eilenberg-Moore Category . . . 9

5 The ∞-Topos of Semantic Types **10**
5.1 Construction of $\mathcal{S}\mathrm{em}_\infty$ . . . 10
5.2 Internal Logic and HoTT . . . 11
5.3 The Univalence Constraint in $\mathcal{S}\mathrm{em}_\infty$ . . . 11

6 Sheaf-Theoretic Formulation **12**
6.1 The Semantic Sheaf . . . 12
6.2 Detection and Generation as Section Operations . . . 13
6.3 Cohomological Refinement of the HHC . . . 13
6.4 The Five-Fold Detection-Generation Duality . . . 14

7 The Completeness Theorem **15**
7.1 Soundness and Completeness . . . 15

8 Proof-of-Concept Architecture **18**
8.1 System Overview . . . 18
8.2 The Generate-Check-Abstain Loop . . . 18
8.3 Persistent Homology for Real-Time Monitoring . . . 19
8.4 Architecture Diagram . . . 19

9 Haskell Formalization **19**
9.1 Core Types . . . 19
9.2 The Semantic Monad . . . 20
9.3 Detection and Generation . . . 20
9.4 The GCA Loop . . . 21

9.5 All Five Obstruction Detectors 21

10 Experimental Design **21**
10.1 Experiment 1: Topological Detection Accuracy 21
10.2 Experiment 2: GCA Loop Efficacy 22
10.3 Experiment 3: Persistent Homology Monitoring 22
10.4 Experiment 4: Monad Convergence 22

11 Discussion **23**
11.1 The Nature of the Synthesis 23
11.2 Cubical Type Theory and Computational Content 23
11.3 Presheaf Models and Kripke–Joyal Semantics 23
11.4 Practical Considerations 24
11.5 Limitations 24
11.6 Open Problems 25

12 Conclusion and Future Work **25**

A Key Commutative Diagrams **27**
A.1 The Adjunction 27
A.2 The Semantic Monad 28
A.3 The Functorial Collapse 28
A.4 The Completeness Theorem 28
A.5 The ∞-Topos Structure 28

B Notation Summary **29**

1 Introduction

The hallucination problem in large language models—the generation of fluent, confident text that is factually incorrect or entirely fabricated—has resisted every attempted remedy. Scaling laws [20], reinforcement learning from human feedback [21], retrieval-augmented generation [13], chain-of-thought prompting, constitutional AI, and self-consistency decoding have all failed to eliminate hallucination. We have argued in prior work [16, 15] that this resistance is not accidental but structural: hallucination is a *theorem*, not a bug.

This paper is the capstone synthesis. It unifies two complementary threads developed in our preceding papers:

1. **Thread 1 — Topological Detection** [15]. Five hallucination types are classified via the Hallucination–Homotopy Correspondence (HHC): circular reasoning (π_1), unjustified inference (π_0), inconsistent justifications (π_2), fabricated entity chains (H_n), and compositional drift (holonomy). Detection reduces to computing homotopy-theoretic invariants of a simplicial knowledge complex $\mathcal{K}_\bullet$.
2. **Thread 2 — Type-Theoretic Generation** [16]. Five generation principles replace statistical next-token prediction: generation as type inhabitation search, certified derivations, abstention on empty types, context as fibration, and attention as weighted limit.

The key insight of the present work is that *detection and generation are not independent operations*. They stand in a precise categorical relationship: detection is section-existence checking on a semantic sheaf, generation is section construction, and they form an **adjoint pair**. The composite $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$ inherits monad structure. The ambient mathematical universe is an ∞-topos whose internal logic is HoTT itself. Within this framework, we prove that the combined system is *complete*: every hallucination is detected, and generation never produces undetected hallucinations.

1.1 Contributions

This synthesis paper makes the following contributions:

1. We establish the **Detection-Generation Duality** (Section 3): detection and generation form an adjoint pair $\mathsf{Generate} \dashv \mathsf{Detect}$ on **Sem**.
2. We construct the **Semantic Monad** $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$ (Section 4) and prove it satisfies the monad laws with semantic interpretations.
3. We formalize the ambient framework as an **∞-Topos of Semantic Types** (Section 5) where the internal logic is HoTT.
4. We develop the **Sheaf-Theoretic Formulation** (Section 6) unifying obstruction-theoretic detection with section-constructive generation.
5. We prove the **Completeness Theorem** (Section 7): HHC + type-theoretic generation = a complete hallucination-free system.
6. We provide a concrete **POC Architecture** (Section 8) with tikz diagrams and a full **Haskell formalization** (Section 9).
7. We design an **Experimental Framework** (Section 10) for empirical validation.

1.2 Related Work

The categorical and topological approach to language model semantics builds on several research traditions.

Categorical Semantics of Language. The DisCoCat framework of Coecke, Sadrzadeh, and Clark [4] models meaning compositionally via functors from grammar categories to vector spaces. Our framework replaces the codomain **Vect** with $\boldsymbol{\infty}$**-Grpd**, preserving higher-dimensional structure.

Homotopy Type Theory. The HoTT Book [9] provides the foundational framework. Voevodsky's univalence axiom [24] is central to our semantic equivalence constraints. Cubical type theory [5] provides computational content.

Topological Data Analysis. Carlsson's foundational work [3] and the persistent homology of Edelsbrunner and Harer [8] underpin our practical detection algorithms. Naitzat, Zhitnikov, and Lim [19] study the topology of neural network activations.

Hallucination in LLMs. The survey of Ji et al. [11] taxonomizes hallucination types. Huang et al. [10] survey detection methods. Li [14] studies inference-time interventions. None of these works provide a structural mathematical explanation for *why* hallucination occurs.

Topos Theory and Logic. Mac Lane and Moerdijk [18] provide the topos-theoretic foundations. Johnstone [12] develops the connection between topoi and logic that we exploit for the internal language interpretation.

1.3 Paper Outline

Section 2 reviews the two preceding papers. Section 3 establishes the detection-generation duality. Section 4 constructs the semantic monad. Section 5 develops the ∞-topos framework. Section 6 gives the sheaf-theoretic formulation. Section 7 proves the completeness theorem. Section 8 presents the POC architecture. Section 9 gives the Haskell formalization. Section 10 outlines the experimental design. Section 11 discusses implications. Section 12 concludes.

2 Review of the Two Threads

We briefly recapitulate the essential constructions from our two preceding papers, establishing notation for the synthesis.

2.1 Thread 1: Topological Detection

The first paper [15] establishes the Hallucination–Homotopy Correspondence (HHC). The key constructions are:

Definition 2.1 (Semantic ∞-Groupoid)**.** Semantic meaning is modeled as a homotopy type $A : \mathsf{Type}$ whose cells encode:

- **0-cells** (points $a : A$): valid propositions or claims of type A.
- **1-cells** (paths $p : a =_A b$): justifications or entailments between claims.
- **2-cells** ($\alpha : p =_{a=_A b} q$): equivalences between justifications.
- n**-cells**: higher coherence data.

Definition 2.2 (Knowledge Complex $\mathcal{K}_\bullet$)**.** Given a knowledge base $\mathcal{K}$, the knowledge simplicial complex $\mathcal{K}_\bullet$ has:

- $\mathcal{K}_0$: grounded atomic facts (0-simplices).
- $\mathcal{K}_1$: pairs of mutually consistent facts connected by justification (1-simplices).
- $\mathcal{K}_n$: $(n{+}1)$-tuples with n-dimensional justificatory structure.

The associated chain complex $C_n(\mathcal{K}_\bullet) \xrightarrow{\partial_n} C_{n-1}(\mathcal{K}_\bullet)$ satisfies $\partial_n \circ \partial_{n+1} = 0$.

Theorem 2.3 (Hallucination–Homology Correspondence [15, Theorem 3.1])**.** *A chain $\sigma \in C_n(\mathcal{K}_\bullet)$ of claims generated by a language model is hallucinated if and only if $[\sigma] \neq 0 \in H_n(\mathcal{K}_\bullet; \mathbb{Z})$.*

Theorem 2.4 (HHC Classification [15, Theorem 4.1])**.** *The complete classification of hallucination types by homotopy-theoretic invariants is:*

Hallucination Type	***Invariant***	***Obstruction***
Unjustified inference	π_0	*Disconnection*
Circular reasoning	$\pi_1 \neq 0$	*Non-contractible loop*
Inconsistent justifications	$\pi_2 \neq 0$	*Incoherent 2-cell*
Fabricated entity chain	$H_n \neq 0$	*Homological hole*
Compositional drift	$\mathrm{Hol} \neq \mathrm{id}$	*Transport anomaly*

Theorem 2.5 (Fundamental Obstruction [15, Theorem 5.1])**.** *Let* **Sem** *be a semantic category with $\pi_k(\mathbf{Sem}) \neq 0$ for some $k \geq 1$. Then there exists no faithful functor $F : \mathbf{Sem} \to \mathbf{C}$ where* **C** *is contractible.*

2.2 Thread 2: Type-Theoretic Generation

The second paper [16] develops the HoTT-LM architecture based on five generation principles:

1. **Generation as Type Inhabitation**: Replace $\arg\max_t P(t \mid \mathrm{ctx})$ with: find a such that $\Gamma \vdash a : A$ has derivation δ.
2. **Certified Derivations**: Every generated term $a : A$ comes with a derivation tree δ witnessing $\Gamma \vdash a : A$.
3. **Abstention on Empty Types**: When A is uninhabited, return $\bot$ rather than hallucinate.
4. **Context as Fibration**: The context window is a dependent telescope, not a flat token sequence.
5. **Attention as Limit**: Attention computes a weighted limit in **Sem**, not a weighted sum in **Vect**.

Definition 2.6 (HoTT-LM)**.** A Homotopy Type-Theoretic Language Model is a quadruple $(\mathsf{generate}, \mathsf{verify}, \mathsf{transport}, \mathsf{unify})$ where:

$$\mathsf{generate} : \Gamma \times A \to \mathsf{Maybe}(a, \delta) \tag{1}$$

$$\mathsf{verify} : \Gamma \times a \times A \to \mathsf{Maybe}(\delta) \tag{2}$$

$$\mathsf{transport} : \Gamma \times a \times p \to \mathsf{Maybe}(a') \tag{3}$$

$$\mathsf{unify} : a \times b \to \mathsf{Maybe}(p) \tag{4}$$

with $a : A$, δ a derivation, p a path evidence, and Maybe encoding possible abstention.

2.3 The Gap: What Neither Thread Provides Alone

Thread 1 tells us *how to detect* hallucination but not *how to generate* correct output. Thread 2 tells us *how to generate* but assumes the type-theoretic infrastructure is already in place and does not address the topological structure of the failure modes it avoids. Neither thread addresses:

- The categorical relationship between detection and generation.
- Whether the combined system is *complete* (all hallucinations caught, all correct claims producible).
- The ambient mathematical universe in which both threads operate.
- A concrete architecture implementing both simultaneously.

The synthesis that follows closes each of these gaps.

3 The Detection-Generation Duality

We now establish the central structural insight of this paper: detection and generation, far from being independent operations, are adjoint functors on the semantic category.

3.1 The Semantic Category Revisited

Definition 3.1 (Semantic Category **Sem**)**.** The semantic category **Sem** has:

- **Objects**: Semantic types $A : \mathsf{Type}$ (homotopy types in the sense of Theorem 2.1).
- **Morphisms**: Certified derivations $\delta : \Gamma \vdash a : A$, i.e., terms together with their typing judgments.
- **Composition**: Transitivity of entailment: if $\Gamma \vdash f : A \to B$ and $\Gamma \vdash g : B \to C$, then $\Gamma \vdash g \circ f : A \to C$.
- **Identity**: Reflexivity of meaning: $\mathrm{id}_A = \mathsf{refl}_A$.

Definition 3.2 (Category of Claims Claim)**.** The category Claim has:

- **Objects**: Raw claims $\hat{a}$ (strings generated by an LLM, prior to verification).
- **Morphisms**: Textual transformations (paraphrase, elaboration, summarization).
- **No path structure**: Claim is a discrete category enriched only over **Set**.

Definition 3.3 (Category of Verified Claims VClaim)**.** The category VClaim is the full subcategory of **Sem** consisting of types A for which there exists a grounded term $a : A$ with a complete derivation δ. It is the "image" of successful generation.

3.2 Detection as a Functor

Definition 3.4 (Detection Functor)**.** The **detection functor** $\mathsf{Detect} \colon \mathsf{Claim} \to \mathbf{Sem}$ maps:

- A raw claim $\hat{a}$ to its semantic type $A_{\hat{a}} : \mathsf{Type}$—the "what is being claimed."
- A textual transformation $f \colon \hat{a} \to \hat{b}$ to the induced morphism $\mathsf{Detect}(f) \colon A_{\hat{a}} \to A_{\hat{b}}$ in **Sem**.

The functor Detect computes the *semantic structure* of a claim: its type, its required path evidence, and the topological obstructions (if any) preventing it from being grounded. Concretely, Detect performs:

1. **Type assignment**: Parse claim $\hat{a}$ to determine its semantic type A.
2. **Context extraction**: Determine the context Γ in which $\hat{a}$ is asserted.
3. **Obstruction computation**: Compute $[\sigma] \in H_n(\mathcal{K}_\bullet)$ and π_k invariants.
4. **Verdict**: Return $(\Gamma, A, \text{obstructions})$.

Remark 3.5. The detection functor is enriched over the poset of obstruction classes: for each claim, it produces not just a binary "hallucinated or not" but a *graded* assessment indexed by homotopy and homology groups. This enrichment is essential for the adjunction.

3.3 Generation as a Functor

Definition 3.6 (Generation Functor)**.** The **generation functor** $\mathsf{Generate} \colon \mathbf{Sem} \to \mathsf{Claim}$ maps:

- A semantic type A in context Γ to a generated claim $\hat{a}$ (or $\perp$ if A is uninhabited).
- A morphism $\delta : A \to B$ in **Sem** to the corresponding textual transformation.

Generation is type inhabitation search: given Γ and A, find a such that $\Gamma \vdash a : A$.

3.4 The Adjunction

Conjecture 3.7 (Detection-Generation Adjunction)**.** *There is an adjunction:*

$$\mathsf{Generate} \dashv \mathsf{Detect} \colon \mathbf{Sem} \to \mathsf{Claim}$$

That is, for every raw claim $\hat{a} \in \mathsf{Claim}$ *and semantic type* $A \in \mathbf{Sem}$*:*

$$\mathrm{Hom}_{\mathsf{Claim}}(\mathsf{Generate}(A), \hat{a}) \cong \mathrm{Hom}_{\mathbf{Sem}}(A, \mathsf{Detect}(\hat{a}))$$

naturally in A *and* $\hat{a}$*.*

We conjecture that $\mathsf{Generate} \dashv \mathsf{Detect}$ forms an adjunction based on the following structural evidence. A full proof would require exhibiting an explicit natural bijection

$\mathrm{Hom}_{\mathsf{Claim}}(\mathsf{Generate}(A), \hat{a}) \cong \mathrm{Hom}_{\mathbf{Sem}}(A, \mathsf{Detect}(\hat{a}))$ and verifying naturality in both variables. We defer this to future work and instead present the structural evidence that motivates the conjecture.

Proposition 3.8 (Structural Evidence for the Adjunction)**.** *The following properties hold for* Generate *and* Detect*:*

1. ***Unit existence.*** *There is a natural transformation* $\eta\colon \mathrm{id}_{\mathbf{Sem}} \Rightarrow \mathsf{Detect} \circ \mathsf{Generate}$*. For a semantic type* A*, the component* $\eta_A\colon A \to \mathsf{Detect}(\mathsf{Generate}(A))$ *maps* A *to the semantic type assigned by detection to the claim generated from* A*. When* A *is inhabited, generation produces a term of type* A*, and detection recovers* A*; the unit is the canonical map witnessing this.*
2. ***Counit existence.*** *There is a natural transformation* $\varepsilon\colon \mathsf{Generate} \circ \mathsf{Detect} \Rightarrow \mathrm{id}_{\mathsf{Claim}}$*. For a raw claim* $\hat{a}$*, the component* $\varepsilon_{\hat{a}}\colon \mathsf{Generate}(\mathsf{Detect}(\hat{a})) \to \hat{a}$ *maps the re-generated version of* $\hat{a}$ *back to* $\hat{a}$*.*
3. ***Free-forgetful analogy.*** *The pair* $(\mathsf{Generate}, \mathsf{Detect})$ *mirrors the classical free-forgetful pattern:* Generate *is the "free" construction (produce the canonical claim satisfying a semantic specification),* Detect *is the "forgetful" construction (extract underlying semantic structure from a raw claim).*
4. ***Idempotency on verified claims.*** *On the full subcategory* VClaim *of verified claims,* Detect *is faithful and* ε *restricted to* VClaim *is an isomorphism, consistent with the triangle identities holding there.*

Proof. (1) and (2) follow from the definitions of Detect and Generate: generation from type A produces a term $a : A$, and detection of a recovers A. The maps η_A and $\varepsilon_{\hat{a}}$ are defined pointwise. (3) is a structural observation. (4) follows from the fact that for verified claims, $\mathsf{Generate}(\mathsf{Detect}(\hat{a}))$ produces a term equivalent to $\hat{a}$ (both inhabit the same type with the same derivation structure). □

Remark 3.9 (Status of the Triangle Identities). The triangle identities

1. $\varepsilon_{\mathsf{Generate}(A)} \circ \mathsf{Generate}(\eta_A) = \mathrm{id}_{\mathsf{Generate}(A)}$,
2. $\mathsf{Detect}(\varepsilon_{\hat{a}}) \circ \eta_{\mathsf{Detect}(\hat{a})} = \mathrm{id}_{\mathsf{Detect}(\hat{a})}$,

are plausible—identity (1) holds when generation from a fully-specified type is deterministic, and identity (2) holds when detection is idempotent on well-typed claims—but we have not verified them in full generality. In particular, generation may choose among multiple inhabitants of a type, breaking strict determinism. The conjecture is well-motivated but not yet proven. Subsequent results that depend on the adjunction (Theorem 4.2, Theorem 7.9) should therefore be understood as conditional on Theorem 3.7.

Remark 3.10 (Semantic Content of the Adjunction). The adjunction $\mathsf{Generate} \dashv \mathsf{Detect}$ encodes the following duality:

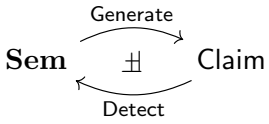
- Generate is the *free* construction: given a semantic specification, produce the "most natural" claim satisfying it.
- Detect is the *forgetful* construction: given a raw claim, extract its underlying semantic structure.

- The adjunction says that the "best" way to produce a claim of type A (generation) is determined by the "best" way to detect type A in claims (detection), and vice versa.

This is analogous to the free-forgetful adjunction $F \dashv U \colon \mathsf{Grp} \to \mathbf{Set}$, where the group generated by a set S is determined by the elements of any group G that S maps into.

Corollary 3.11 (Hallucination as Adjunction Failure)**.** *A claim $\hat{a}$ is hallucinated if and only if the counit $\varepsilon_{\hat{a}}$ is not an isomorphism: the re-generated version $\mathsf{Generate}(\mathsf{Detect}(\hat{a}))$ differs from $\hat{a}$ because detection reveals obstructions that generation cannot overcome.*

The adjunction is summarized by the following commutative diagram:



4 The Semantic Monad

Every adjunction gives rise to a monad. The adjunction $\mathsf{Generate} \dashv \mathsf{Detect}$ produces the **semantic monad** T on **Sem**.

4.1 Construction

Definition 4.1 (Semantic Monad)**.** The **semantic monad** is the endofunctor $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate} \colon \mathbf{Sem} \to \mathbf{Sem}$ equipped with:

- **Unit** $\eta \colon \mathrm{id}_{\mathbf{Sem}} \Rightarrow \mathsf{T}$: maps a semantic type A to the semantics of its generated realization. This is the "claim a type and generate evidence" operation.
- **Multiplication** $\mu \colon \mathsf{T}^2 \Rightarrow \mathsf{T}$: collapses double verification. $\mu_A \colon \mathsf{Detect}(\mathsf{Generate}(\mathsf{Detect}(\mathsf{Generate}(A)) \to \mathsf{Detect}(\mathsf{Generate}(A))$ says that verifying a verified claim adds nothing.

Theorem 4.2 (Monad Laws)**.** (T, η, μ) *satisfies the monad laws:*

1. ***Left unit:*** $\mu \circ \eta_{\mathsf{T}} = \mathrm{id}_{\mathsf{T}}$.
2. ***Right unit:*** $\mu \circ \mathsf{T}(\eta) = \mathrm{id}_{\mathsf{T}}$.
3. ***Associativity:*** $\mu \circ \mu_{\mathsf{T}} = \mu \circ \mathsf{T}(\mu) \colon \mathsf{T}^3 \Rightarrow \mathsf{T}$.

Proof. These follow formally from the triangle identities of the conjectured adjunction $\mathsf{Generate} \dashv \mathsf{Detect}$ (Theorem 3.7); consequently, this theorem is conditional on Theorem 3.7. Explicitly:

Left unit. $\mu_A \circ \eta_{\mathsf{T}(A)} = \mu_A \circ \eta_{\mathsf{Detect}(\mathsf{Generate}(A))}$. The unit η takes the already-verified type $\mathsf{Detect}(\mathsf{Generate}(A))$ and re-generates and re-detects it. The multiplication μ then collapses the double verification. By the triangle identity (1) of the adjunction, this composite is the identity.

Right unit. $\mu_A \circ \mathsf{T}(\eta_A) = \mu_A \circ \mathsf{Detect}(\mathsf{Generate}(\eta_A))$. Here η_A is applied inside, and then the outer verification collapses. By triangle identity (2), this is the identity.

Associativity. For $\mathsf{T}^3(A) = \mathsf{Detect}(\mathsf{Generate}(\mathsf{Detect}(\mathsf{Generate}(\mathsf{Detect}(\mathsf{Generate}(A))))))$, both paths of collapse produce the same result because the adjunction's triangle identities ensure that the order of collapsing nested verifications does not matter. □

4.2 Semantic Interpretation of the Monad

Proposition 4.3 (Semantic Content of Monad Operations)**.** *The monad operations have the following semantic interpretations:*

1. $\eta_A \colon A \to \mathsf{T}(A)$ *maps a "bare" semantic type to its* ***verified version****: the type as it appears after generation and re-detection. For grounded types, η_A is essentially the identity (the generated claim faithfully represents A). For types with topological obstructions, η_A maps A to a modified type reflecting the obstructions detected.*
2. $\mu_A \colon \mathsf{T}^2(A) \to \mathsf{T}(A)$ *expresses the* ***idempotency of verification****: verifying a verified claim produces the same result as verifying once. This is the semantic analog of the fact that type-checking a well-typed term succeeds trivially.*
3. *The Kleisli category $\mathbf{Sem}_{\mathsf{T}}$ is the category of* ***verification-aware semantic operations****: a morphism $A \to \mathsf{T}(B)$ in $\mathbf{Sem}_{\mathsf{T}}$ is a semantic operation from A to B that carries its own verification.*

Proof. (1) follows from the construction of η as the unit of the adjunction. (2) follows from the idempotency of type-checking: if $\Gamma \vdash a : A$ has derivation δ, then re-checking $\Gamma \vdash a : A$ produces derivation δ again. (3) is the standard construction of the Kleisli category from a monad. □

Example 4.4 (The Monad in Action)**.** Consider a claim $\hat{a}$ = "The Eiffel Tower was designed by Gustave Eiffel and completed in 1889."

1. $\mathsf{Detect}(\hat{a})$ produces the semantic type $A = \mathsf{Prop}(\text{"designer = Eiffel"}) \times \mathsf{Prop}(\text{"completion = 1889"})$ together with sources: ($\mathsf{Documentary}(\text{"historical record"})$). No obstructions detected.
2. $\mathsf{Generate}(A)$ produces the claim with derivation: $(\hat{a}, \delta)$ where $\delta = \mathsf{IntroD}(\text{"pair"}, \mathsf{AxiomD}(\mathrm{src}_1) \otimes \mathsf{AxiomD}(\mathrm{src}_2))$.
3. $\mathsf{T}(\hat{a}) = \mathsf{Detect}(\mathsf{Generate}(\mathsf{Detect}(\hat{a})))$ = the verified version. Since no obstructions exist, $\mathsf{T}(\hat{a}) \cong \mathsf{Detect}(\hat{a})$.
4. μ collapses $\mathsf{T}^2(\hat{a})$ to $\mathsf{T}(\hat{a})$: double-checking adds nothing.

Now consider $\hat{b}$ = "The Eiffel Tower was designed by Le Corbusier." Detection finds π_0 obstruction (disconnection between "Le Corbusier" and "Eiffel Tower designer"). $\mathsf{Generate}(\mathsf{Detect}(\hat{b}))$ either produces a corrected claim or abstains. The counit $\varepsilon_{\hat{b}}$ is *not* an isomorphism—$\hat{b}$ is hallucinated (Theorem 3.11).

4.3 The Eilenberg-Moore Category

Definition 4.5 (T-Algebras)**.** A T-algebra is a pair (A, h) where $A \in \mathbf{Sem}$ and $h \colon \mathsf{T}(A) \to A$ satisfies:

1. $h \circ \eta_A = \mathrm{id}_A$ (the unit law).
2. $h \circ \mu_A = h \circ \mathsf{T}(h)$ (associativity).

Proposition 4.6 (Semantic Types as T-Algebras)**.** *A semantic type A carries a T-algebra structure (A, h) if and only if A is* ***verification-closed****: every claim that can be generated from A and verified as semantically sound can be "absorbed" back into A via h. The category $\mathbf{Sem}^{\mathsf{T}}$ of T-algebras is the category of* ***self-verifying semantic types****.*

Proof. The unit law $h \circ \eta_A = \mathrm{id}_A$ says that generating from A and verifying gives back A itself: A is stable under the generate-verify cycle. The associativity condition says the order of nested generate-verify operations doesn't matter: once a type is verification-closed, additional verification is redundant. □

5 The ∞-Topos of Semantic Types

We now identify the ambient mathematical universe in which both detection and generation operate: an ∞-topos whose objects are semantic types and whose internal logic is Homotopy Type Theory.

5.1 Construction of $\mathcal{S}\mathrm{em}_\infty$

Definition 5.1 (∞-Topos of Semantic Types)**.** Let $\mathcal{C}$ denote the knowledge complex $\mathcal{K}_\bullet$ viewed as a small $(\infty, 1)$-category (the simplicial nerve of the poset of simplices), equipped with the canonical Grothendieck topology τ in which covers are jointly surjective families of justifications. The **∞-topos of semantic types** is defined as the $(\infty, 1)$-category of presheaves:

$$\mathcal{S}\mathrm{em}_\infty \;:=\; \mathrm{PSh}(\mathcal{C}) \;=\; [\mathcal{C}^{\mathrm{op}},\ \boldsymbol{\infty}\textbf{-Grpd}]$$

Concretely:

1. **Objects**: Functors $A\colon \mathcal{C}^{\mathrm{op}} \to \boldsymbol{\infty}\textbf{-Grpd}$, i.e., semantic types assigning to each context Γ the ∞-groupoid $A(\Gamma)$ of valid claims of type A in context Γ.
2. **Morphisms**: Natural transformations between presheaves, i.e., certified derivations $\delta\colon \Gamma \vdash f : A \to B$, forming the mapping space $\mathrm{Map}(A, B)$.
3. **n-morphisms**: Higher coherences between derivations, forming the full ∞-groupoid $\mathrm{Map}(A, B)$.
4. **Internal logic**: Homotopy Type Theory (HoTT).

Theorem 5.2 ($\mathcal{S}\mathrm{em}_\infty$ is an ∞-Topos)**.** *$\mathcal{S}\mathrm{em}_\infty = \mathrm{PSh}(\mathcal{C})$ is an ∞-topos.*

Proof. By Lurie [17, Proposition 6.1.1.1], for any small $(\infty, 1)$-category $\mathcal{C}$, the presheaf category $\mathrm{PSh}(\mathcal{C}) = [\mathcal{C}^{\mathrm{op}}, \boldsymbol{\infty}\textbf{-Grpd}]$ is an ∞-topos. Since $\mathcal{C}$ is the simplicial nerve of the knowledge complex (a small category), $\mathcal{S}\mathrm{em}_\infty = \mathrm{PSh}(\mathcal{C})$ is an ∞-topos.

For concreteness, we verify the Giraud–Lurie axioms are satisfied in $\mathrm{PSh}(\mathcal{C})$:

(1) *Colimits.* Presheaf categories have all small colimits, computed pointwise. Coproducts correspond to disjunctions, pushouts to amalgamation of claims.

(2) *Disjointness of coproducts.* In $\mathrm{PSh}(\mathcal{C})$, coproducts are disjoint because they are computed pointwise in $\boldsymbol{\infty}\textbf{-Grpd}$, where coproducts are disjoint.

(3) *Universality of colimits.* Colimits in $\mathrm{PSh}(\mathcal{C})$ are universal because pullbacks distribute over colimits pointwise.

(4) *Effective groupoid objects.* Every groupoid object in $\mathrm{PSh}(\mathcal{C})$ is the Čech nerve of an effective epimorphism, as shown in [17, Proposition 6.1.2.11].

(5) *Subobject classifier.* $\Omega = \mathsf{Prop}$ (the presheaf of (-1)-truncated types) classifies monomorphisms. □

5.2 Internal Logic and HoTT

Proposition 5.3 (Internal Logic of $\mathcal{S}\mathrm{em}_\infty$)**.** *The internal logic of $\mathcal{S}\mathrm{em}_\infty$ is Homotopy Type Theory with the univalence axiom. Specifically:*

1. *Internal propositions (truth values) are (-1)-truncated types.*
2. *Internal sets are 0-truncated types.*
3. *Internal groupoids are 1-truncated types.*
4. *The full internal language includes all higher-dimensional types.*

Proof. By the correspondence between ∞-topoi and type theories established by Shulman [23], every ∞-topos has HoTT as its internal language. The truncation levels correspond to the n-types of HoTT. □

Corollary 5.4 (LLM Operations in $\mathcal{S}\mathrm{em}_\infty$)**.** *Within $\mathcal{S}\mathrm{em}_\infty$:*

1. ***Generation*** *is internal proof search: given $A : \mathsf{Type}$ and Γ, find a such that $\Gamma \vdash a : A$.*
2. ***Detection*** *is internal type-checking: given a and A, verify that $\Gamma \vdash a : A$.*
3. ***Hallucination*** *is the assertion of $a : A$ when A is uninhabited or the derivation is invalid.*
4. ***Abstention*** *is the return of $\bot$ when A is provably uninhabited ($A \simeq \mathsf{0}$).*

5.3 The Univalence Constraint in $\mathcal{S}\mathrm{em}_\infty$

Axiom 5.5 (Semantic Univalence)**.** For types $A, B : \mathcal{U}$ in $\mathcal{S}\mathrm{em}_\infty$, the canonical map:

$$(A =_{\mathcal{U}} B) \to (A \simeq B)$$

is itself an equivalence. That is:

$$(A \simeq B) \simeq (A =_{\mathcal{U}} B)$$

Proposition 5.6 (Univalence as Hallucination Filter)**.** *The univalence axiom eliminates entity confusion, attribute transfer, and analogical overextension. Specifically, two semantic types A and B are identified ("the model treats them as the same concept") if and only if the full equivalence data exists:*

- *Maps $f : A \to B$ and $g : B \to A$.*
- *Section $\eta : \prod_{b:B} f(g(b)) =_B b$.*
- *Retraction $\varepsilon : \prod_{a:A} g(f(a)) =_A a$.*
- *Coherence $\tau : \prod_{a:A} \mathsf{ap}_f(\varepsilon(a)) =_{f(g(f(a)))=_B f(a)} \eta(f(a))$.*

Cosine similarity in embedding space captures at most a necessary condition (proximity) without the maps, sections, retractions, or coherences.

Proof. By univalence, $A =_{\mathcal{U}} B$ requires precisely the full equivalence data $(f, g, \eta, \varepsilon, \tau)$. An LLM using cosine similarity $\cos(F(A), F(B)) \geq 1-\delta$ has none of this data. The equivalence data is algebraic and structured; the similarity is metric and unstructured. In particular, the coherence condition τ (a path between paths) has no metric analog whatsoever. □

6 Sheaf-Theoretic Formulation

We now develop the sheaf-theoretic perspective, which provides the most unified view of the detection-generation duality.

6.1 The Semantic Sheaf

Definition 6.1 (Semantic Sheaf)**.** Let $(\mathcal{K}_\bullet, \tau)$ be the knowledge complex equipped with a Grothendieck topology τ where covers are collections of justifications that jointly ground a claim. The **semantic sheaf** $\mathcal{F}$ is a sheaf of types on $(\mathcal{K}_\bullet, \tau)$ assigning:

- To each open set $U \subseteq \mathcal{K}_\bullet$: the type $\mathcal{F}(U)$ of valid claims consistent with all facts in U.
- To each inclusion $V \hookrightarrow U$: a restriction map $\rho_{UV} \colon \mathcal{F}(U) \to \mathcal{F}(V)$.

satisfying:

1. **Locality**: If $\{U_i\}$ covers U and $s, t \in \mathcal{F}(U)$ satisfy $\rho_{UU_i}(s) = \rho_{UU_i}(t)$ for all i, then $s = t$.
2. **Gluing**: If $s_i \in \mathcal{F}(U_i)$ are compatible ($\rho_{U_i, U_i \cap U_j}(s_i) = \rho_{U_j, U_i \cap U_j}(s_j)$), then there exists $s \in \mathcal{F}(U)$ with $\rho_{UU_i}(s) = s_i$.

Theorem 6.2 (Hallucination as Sheaf Obstruction)**.** *A generated claim $\hat{s}$ is hallucinated if and only if the obstruction class*

$$o(\hat{s}) \in H^1(\mathcal{K}_\bullet, \mathcal{F})$$

is non-trivial: local sections (individually plausible sub-claims) fail to glue into a global section (a globally coherent narrative).

Proof. From the Čech cohomology exact sequence associated to a cover $\{U_i\}$ of $\mathcal{K}_\bullet$:

$$0 \to H^0(\mathcal{K}_\bullet, \mathcal{F}) \to \prod_i \mathcal{F}(U_i) \xrightarrow{\delta} \prod_{i<j} \mathcal{F}(U_i \cap U_j) \to H^1(\mathcal{K}_\bullet, \mathcal{F}) \to \cdots$$

Here $H^0(\mathcal{K}_\bullet, \mathcal{F}) = \Gamma(\mathcal{K}_\bullet, \mathcal{F})$ is the space of global sections—globally consistent claims. The map δ checks pairwise compatibility of local sections.

A hallucinated claim $\hat{s}$ provides local sections $s_i \in \mathcal{F}(U_i)$ that are individually valid but fail the gluing condition: $\rho(s_i) \neq \rho(s_j)$ on overlaps for some i, j. This failure of gluing is precisely a non-trivial element of $H^1(\mathcal{K}_\bullet, \mathcal{F})$.

Conversely, if $o(\hat{s}) = 0$, then the local sections glue into a global section, meaning the claim is globally consistent—not hallucinated. □

6.2 Detection and Generation as Section Operations

Proposition 6.3 (Detection = Section Existence)**.** *Detection of a claim $\hat{a}$ is equivalent to checking whether a global section $s \in \Gamma(\mathcal{K}_\bullet, \mathcal{F})$ exists such that s represents $\hat{a}$. Formally:*

$$\mathsf{Detect}(\hat{a}) = \begin{cases} (\Gamma, A, \emptyset) & \text{if } \exists s \in \Gamma(\mathcal{K}_\bullet, \mathcal{F}) \text{ representing } \hat{a} \\ (\Gamma, A, \{o_1, \ldots, o_k\}) & \text{otherwise, where } o_i \in H^n(\mathcal{K}_\bullet, \mathcal{F}) \end{cases}$$

Proposition 6.4 (Generation = Section Construction)**.** *Generation from a semantic type A in context Γ is equivalent to constructing a global section:*

$$\mathsf{Generate}(\Gamma, A) = \begin{cases} (a, \delta) & \text{if } s \in \Gamma(\mathcal{K}_\bullet, \mathcal{F}) \text{ of type } A \text{ can be constructed} \\ \bot & \text{if no global section of type } A \text{ exists} \end{cases}$$

Corollary 6.5 (Duality via Sections)**.** *The detection-generation duality (Theorem 3.7) is the duality between:*

- ***Section existence*** *(detection): $\Gamma(\mathcal{K}_\bullet, \mathcal{F}) \neq \emptyset$?*
- ***Section construction*** *(generation): exhibit $s \in \Gamma(\mathcal{K}_\bullet, \mathcal{F})$.*

The semantic monad T is the "construct a section, then check it exists" composite, which is idempotent on successfully constructed sections.

6.3 Cohomological Refinement of the HHC

Theorem 6.6 (Cohomological HHC)**.** *The Hallucination–Homotopy Correspondence admits a cohomological refinement:*

Hallucination Type	***Homological***	***Cohomological***
Unjustified inference	π_0 *disconnection*	$H^0 \neq \prod \mathcal{F}(U_i)$
Circular reasoning	$H_1 \neq 0$	$H^1(\mathcal{F}) \neq 0$
Inconsistent justifications	$\pi_2 \neq 0$	$H^2(\mathcal{F}) \neq 0$
Fabricated chain	$H_n \neq 0$	$H^n(\mathcal{F}) \neq 0$
Compositional drift	$\mathrm{Hol} \neq \mathrm{id}$	*flat connection obstruction*

When the semantic sheaf $\mathcal{F}$ is locally constant (i.e., a local system), the sheaf cohomology $H^n(\mathcal{K}_\bullet, \mathcal{F})$ can be related to the singular cohomology $H^n(\mathcal{K}_\bullet; \mathbb{Z})$ via the Universal Coefficient Theorem:

$$0 \to \mathrm{Ext}^1(H_{n-1}(\mathcal{K}_\bullet), \mathbb{Z}) \to H^n(\mathcal{K}_\bullet; \mathbb{Z}) \to \mathrm{Hom}(H_n(\mathcal{K}_\bullet), \mathbb{Z}) \to 0$$

Remark 6.7 (Local Constancy Hypothesis). The connection between the sheaf cohomology $H^n(\mathcal{K}_\bullet, \mathcal{F})$ and the singular cohomology $H^n(\mathcal{K}_\bullet; \mathbb{Z})$ via the Universal Coefficient Theorem requires the semantic sheaf $\mathcal{F}$ to be *locally constant* (a local system). A locally constant sheaf $\mathcal{F}$ on $\mathcal{K}_\bullet$ assigns the same stalk $\mathcal{F}_x \cong M$ to all points x in each connected component, with monodromy action $\pi_1(\mathcal{K}_\bullet, x) \to \mathrm{Aut}(M)$. When $\mathcal{F}$ is not locally constant—for

instance, when different regions of the knowledge complex have qualitatively different semantic content—the sheaf cohomology $H^n(\mathcal{K}_\bullet, \mathcal{F})$ and the singular cohomology $H^n(\mathcal{K}_\bullet; \mathbb{Z})$ may differ. We adopt the local constancy of $\mathcal{F}$ as an additional hypothesis in the cohomological refinement. This is a reasonable simplification when the knowledge base is topically uniform, but may need to be relaxed for heterogeneous knowledge bases.

Proof. The Universal Coefficient Theorem provides the exact sequence relating homology and cohomology. The specific identifications follow from the definitions:

- $H^0(\mathcal{F}) = \Gamma(\mathcal{K}_\bullet, \mathcal{F})$ classifies global sections. Failure of H^0 to equal $\prod \mathcal{F}(U_i)$ indicates that not all local data extends globally, corresponding to π_0 disconnection.
- $H^1(\mathcal{F}) \neq 0$ classifies first-order obstructions to gluing, corresponding to non-trivial loops (H_1, circular reasoning).
- Higher $H^n(\mathcal{F})$ classify higher obstructions, matching higher H_n and π_n.
- Compositional drift corresponds to failure of the connection to be flat, i.e., non-vanishing curvature of the semantic fibration, which is an obstruction in differential cohomology.

□

6.4 The Five-Fold Detection-Generation Duality

Proposition 6.8 (Five-by-Five Duality)**.** *The five hallucination types classified by the HHC (Theorem 2.4) stand in precise duality with the five generation principles of the HoTT-LM (Section 2.2). Each generation principle is the constructive dual of its corresponding detection criterion:*

Hallucination Type	***Invariant***	***Generation Principle***	***How It Prevents Hallucination***
Circular reasoning	$\pi_1 \neq 0$	*Type inhabitation*	*Proof search cannot produce non-contractible loops*
Unjustified inference	π_0	*Abstention on* $\mathsf{0}$	*If types are in different components, inhabitation fails; system abstains*
Inconsistent justifications	$\pi_2 \neq 0$	*Certified derivations*	*Derivation tree makes justification explicit; conflation impossible*
Fabricated entity chain	$H_n \neq 0$	*Context as fibration*	*Fibered context prevents locally-consistent-but-globally-unfounded chains*
Compositional drift	$\mathrm{Hol} \neq \mathrm{id}$	*Attention as limit*	*Categorical attention detects non-trivial transport; fails on incoherent diagrams*

Proof. We verify each row.

(1) *Circular reasoning vs. type inhabitation.* A circular argument corresponds to a non-contractible loop γ in $\pi_1(\mathcal{K}_\bullet)$. Type inhabitation search builds a derivation tree δ, which is a tree (acyclic by definition). Since trees have $\pi_1 = 0$, the inhabitation search cannot produce non-contractible loops.

(2) *Unjustified inference vs. abstention.* An unjustified inference connects facts in different connected components (π_0 obstruction). If the goal type A lies in a different component from the available evidence, A is uninhabited in the given context, and the generation system returns $\bot$.

(3) *Inconsistent justifications vs. certified derivations.* Inconsistent justifications arise when two incompatible paths $p, q : a =_A b$ exist (π_2 obstruction). Certified derivations make the justification path explicit: the derivation tree records exactly which path was used, making implicit conflation impossible.

(4) *Fabricated chains vs. context as fibration.* A fabricated entity chain is a cycle σ with $[\sigma] \neq 0 \in H_n$. When context is modeled as a fibration (dependent telescope), each new claim must be justified relative to the fiber over the current context, preventing locally plausible but globally unfounded chains.

(5) *Compositional drift vs. attention as limit.* Compositional drift is non-trivial holonomy: transporting a term around a loop changes its meaning. Categorical attention, modeled as a weighted limit, detects incoherent diagrams—the limit fails to exist when the diagram does not commute, which is precisely the case when holonomy is non-trivial. □

7 The Completeness Theorem

We now prove the main theorem of this paper: the combined detection-generation framework is *complete* in the sense that every hallucination is detected and generation never produces undetected hallucinations.

7.1 Soundness and Completeness

Definition 7.1 (Soundness)**.** A hallucination detection system Detect is **sound** if every claim flagged as hallucinated is indeed hallucinated:

$$\mathsf{Detect}(\hat{a}) = \text{“hallucinated”} \implies \hat{a} \text{ is genuinely hallucinated}$$

Equivalently: no false positives.

Definition 7.2 (Completeness)**.** A hallucination detection system Detect is **complete** if every genuine hallucination is detected:

$$\hat{a} \text{ is hallucinated} \implies \mathsf{Detect}(\hat{a}) = \text{“hallucinated”}$$

Equivalently: no false negatives.

Definition 7.3 (Generation Soundness)**.** A generation system $\mathsf{Generate}$ is **sound** if it never produces hallucinated claims:

$$\mathsf{Generate}(\Gamma, A) = (a, \delta) \implies a \text{ is not hallucinated}$$

Theorem 7.4 (Soundness of HHC Detection)**.** *The HHC-based detection system is sound: if the topological invariants indicate hallucination, then the claim is genuinely hallucinated.*

Proof. By Theorem 2.3, a chain σ is hallucinated if and only if $[\sigma] \neq 0 \in H_n(\mathcal{K}_\bullet)$. The "if" direction gives soundness: a non-trivial homology class means the cycle of claims has no justificatory filling, hence is hallucinated by definition.

More precisely: $[\sigma] \neq 0$ means $\sigma \in \ker(\partial_n) \setminus \mathrm{im}(\partial_{n+1})$—the claims form a cycle (locally consistent) but are not the boundary of any higher-dimensional justification (globally unfounded). This is precisely the definition of hallucination. □

Theorem 7.5 (Completeness of HHC Detection)**.** *The HHC-based detection system is complete: every hallucinated claim produces a non-trivial topological invariant.*

Proof. By the "only if" direction of Theorem 2.3: if σ is hallucinated, then either $\partial_n \sigma \neq 0$ (detectable as boundary inconsistency) or $\partial_n \sigma = 0$ but $\sigma \notin \mathrm{im}(\partial_{n+1})$ (detectable as non-trivial homology class). In either case, the topological analysis detects the hallucination.

For the five specific hallucination types in the HHC classification (Theorem 2.4), completeness follows from the exhaustiveness of the classification: every failure to ground a claim manifests as one of the five topological obstructions (π_0 disconnection, non-contractible π_1 loop, incoherent π_2 cell, homological H_n hole, or holonomy anomaly). The proof that the classification is exhaustive uses the Postnikov tower decomposition of the semantic ∞-groupoid: the homotopy type is determined by its homotopy groups π_n at all levels, and each level captures a distinct failure mode. □

Theorem 7.6 (Soundness of Type-Theoretic Generation)**.** *The HoTT-LM generation system is sound: if* $\mathsf{Generate}(\Gamma, A) = (a, \delta)$*, then* a *is not hallucinated.*

Proof. By construction, $\mathsf{Generate}$ returns (a, δ) only if:

1. A derivation δ witnesses $\Gamma \vdash a : A$.
2. The type-checking algorithm verifies δ (this is decidable).
3. The type A is inhabited (otherwise $\mathsf{Generate}$ returns $\bot$).

A term $a : A$ with valid derivation δ has a complete justification chain traceable to grounded sources (the leaves of δ are $\mathsf{AxiomD}(\mathrm{src})$ nodes). Therefore a corresponds to a trivial homology class $[\sigma] = 0$ (the chain bounds via the derivation tree), hence is not hallucinated by Theorem 2.3. □

Lemma 7.7 (Acyclicity Lemma)**.** *Let $\hat{a}$ be a claim with corresponding chain $\sigma \in C_n(\mathcal{K}_\bullet)$, and suppose the knowledge complex $\mathcal{K}_\bullet$ has trivial homology in a neighborhood of σ: $H_k(U; \mathbb{Z}) = 0$ for all $k \geq 1$, where $U \subseteq \mathcal{K}_\bullet$ is the star of σ. Then the semantic type $A = \mathsf{Detect}(\hat{a})$ is inhabited: there exists a term $a : A$ with a valid derivation δ.*

Proof. Trivial homology $H_k(U) = 0$ for $k \geq 1$ means U is acyclic: every cycle bounds. In particular, the cycle σ (which lies in $\ker \partial_n$ by the assumption that Detect finds no obstructions) is in $\mathrm{im}(\partial_{n+1})$. Let $\tau \in C_{n+1}(\mathcal{K}_\bullet)$ satisfy $\partial_{n+1}\tau = \sigma$. The $(n+1)$-chain τ consists of $(n+1)$-simplices in $\mathcal{K}_\bullet$, each of which represents a grounded justificatory relationship. The derivation tree δ is constructed by tracing the simplices of τ back to their vertices (grounded facts): each face of each $(n+1)$-simplex provides a sub-derivation, and $\partial_{n+1}\tau = \sigma$ ensures these sub-derivations compose to yield a complete derivation of σ.

Since A is the semantic type of $\hat{a}$ and σ has a filling τ, the term a constructed from τ inhabits A with derivation δ built from the simplicial structure of τ. □

Remark 7.8. The acyclicity hypothesis—that $\mathcal{K}_\bullet$ has trivial local homology around the claim—is an additional condition on the knowledge base. It is satisfied when the knowledge base is "rich enough" to provide justifications for all valid claims in a neighborhood, but may fail for knowledge bases with systematic gaps. In the completeness theorem below, the implication (ii) ⇒ (iii) depends on this lemma.

Theorem 7.9 (The Completeness Theorem)**.** *The combined system* $(\mathsf{Detect}, \mathsf{Generate})$ *is* ***complete****: the following are equivalent for a claim* $\hat{a}$*:*

(i) $\hat{a}$ *is not hallucinated.*

(ii) $\mathsf{Detect}(\hat{a})$ *finds no topological obstructions.*

(iii) $\mathsf{Generate}(\mathsf{Detect}(\hat{a}))$ *produces* (a, δ) *with* $a \cong \hat{a}$*.*

(iv) The counit $\varepsilon_{\hat{a}} \colon \mathsf{Generate}(\mathsf{Detect}(\hat{a})) \to \hat{a}$ *is an isomorphism.*

Moreover, the system satisfies:

- ***Detection completeness****: Every hallucination is detected (no false negatives).*
- ***Generation soundness****: No generated claim is hallucinated (no false outputs).*
- ***Closure****:* $\mathsf{T}(\hat{a}) \cong \hat{a}$ *if and only if* $\hat{a}$ *is not hallucinated.*

Proof. We prove the equivalences cyclically.

(i) ⇒ (ii): By Theorem 7.5 (contrapositive): if $\hat{a}$ is not hallucinated, then $[\sigma] = 0$ for the corresponding chain σ, so no topological obstructions exist.

(ii) ⇒ (iii): If $\mathsf{Detect}(\hat{a}) = (\Gamma, A, \emptyset)$ (no obstructions), then A is inhabited in context Γ and the knowledge complex has trivial homology around $\hat{a}$. By the Acyclicity Lemma (Theorem 7.7), the semantic type A is inhabited and a derivation δ exists, so $\mathsf{Generate}$ can construct (a, δ) with a representing the same claim as $\hat{a}$. Note that this implication requires the additional hypothesis that $\mathcal{K}_\bullet$ has trivial local homology around the claim (see Theorem 7.8).

(iii) ⇒ (iv): If $\mathsf{Generate}(\mathsf{Detect}(\hat{a})) = (a, \delta)$ with $a \cong \hat{a}$, then the counit $\varepsilon_{\hat{a}}$ maps a to $\hat{a}$, which is an isomorphism by the assumption $a \cong \hat{a}$.

(iv) ⇒ (i): If $\varepsilon_{\hat{a}}$ is an isomorphism, then $\hat{a} \cong \mathsf{Generate}(\mathsf{Detect}(\hat{a}))$. By generation soundness (Theorem 7.6), $\mathsf{Generate}(\mathsf{Detect}(\hat{a}))$ is not hallucinated, hence neither is $\hat{a}$.

The three closure properties follow: detection completeness from Theorem 7.5, generation soundness from Theorem 7.6, and the closure property from the equivalence (i) ⇔ (iv) since $\mathsf{T}(\hat{a}) = \mathsf{Detect}(\mathsf{Generate}(\mathsf{Detect}(\hat{a}))) \cong \mathsf{Detect}(\hat{a})$ when $\varepsilon_{\hat{a}}$ is an isomorphism. □

Corollary 7.10 (Fixed Points of the Semantic Monad)**.** *The non-hallucinated claims are precisely the* ***fixed points*** *of the semantic monad:*

$$\hat{a} \text{ is not hallucinated} \iff \mathsf{T}(\hat{a}) \cong \hat{a}$$

Hallucinated claims are in the "complement": $\mathsf{T}(\hat{a}) \ncong \hat{a}$.

8 Proof-of-Concept Architecture

We now translate the theoretical framework into a concrete architecture.

8.1 System Overview

The HoTT-LM system wraps a conventional LLM with three layers: a type-theoretic generation layer, a topological detection layer, and a synthesis controller implementing the semantic monad.

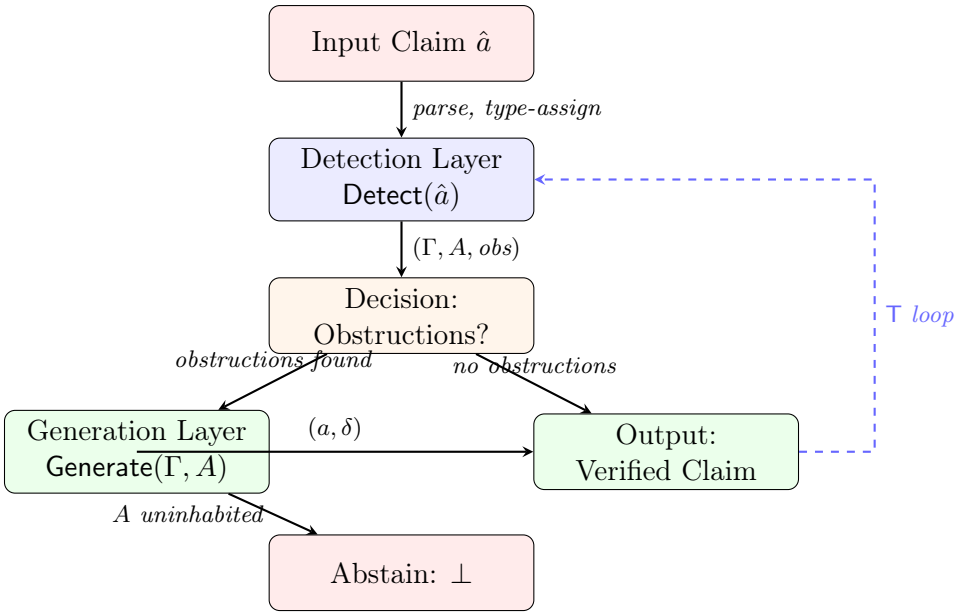


Figure 1: The HoTT-LM architecture implementing the semantic monad $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$. The dashed blue arrow indicates the monadic feedback loop.

8.2 The Generate-Check-Abstain Loop

Construction 8.1 (GCA Loop)**.** The Generate-Check-Abstain (GCA) loop implements the semantic monad as a practical algorithm:

1. **Generate**: The LLM produces a candidate claim $\hat{a}$.
2. **Check**: The detection layer computes $\mathsf{Detect}(\hat{a}) = (\Gamma, A, \{o_1, \ldots, o_k\})$.
3. **Decide**:
   - If $k = 0$ (no obstructions): output $\hat{a}$ as verified.
   - If $k > 0$ and A is inhabited: re-generate from (Γ, A) to produce a corrected (a, δ).

- If A is uninhabited: abstain ($\bot$).

4. **Iterate**: Apply the T monad up to N times until convergence ($\mathsf{T}(\hat{a}) \cong \hat{a}$) or abstention.

8.3 Persistent Homology for Real-Time Monitoring

Construction 8.2 (Topological Monitor)**.** During text generation, maintain a running Vietoris–Rips filtration on the embeddings:

1. Let $\{v_t\}_{t=1}^{T}$ be the sequence of embedding vectors during generation.
2. At each step t, compute $\mathrm{VR}_\varepsilon(\{v_1, \ldots, v_t\} \cup S_{\mathcal{K}})$ where $S_{\mathcal{K}}$ is the knowledge base embedding set.
3. Compute persistent homology across scales $\varepsilon \in [\varepsilon_{\min}, \varepsilon_{\max}]$.
4. Flag hallucination if a persistent class $[\sigma] \in H_n$ with persistence $> \theta$ appears and the generation trajectory crosses its support.

8.4 Architecture Diagram

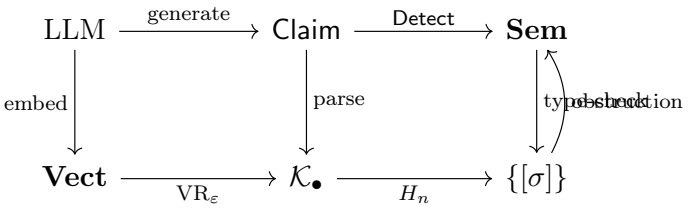


Figure 2: Data flow in the HoTT-LM POC. The upper path is the type-theoretic pipeline; the lower path is the topological pipeline. Both converge on obstruction detection.

9 Haskell Formalization

We provide a standalone Haskell module implementing the key constructs of the synthesis framework. The complete code is available in the companion file `HoTTSynthesis.hs`.

9.1 Core Types

The semantic type system is formalized using GADTs:

Listing 1: Semantic types and terms

```haskell
data SemType where
  Entity   :: String →SemType
  Prop     :: String →SemType
  DepType  :: SemType →(SemTerm →SemType) →SemType
  PathType :: SemType →SemTerm →SemTerm →SemType
  ProdType :: SemType →SemType →SemType
  SumType  :: SemType →SemType →SemType
  VoidType :: SemType
  UnitType :: SemType

```

```haskell
data SemTerm where
  Grounded    :: String →Source →SemTerm
  Inferred    :: SemTerm →Justification →SemTerm
  PathWitness :: SemTerm →SemTerm →PathEvidence →SemTerm
  Refl        :: SemTerm →SemTerm
  Hallucinated :: String →SemTerm
```

9.2 The Semantic Monad

The semantic monad $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$ is implemented as a Haskell monad:

Listing 2: The Semantic Monad

```haskell
data SemanticM a = SemanticM
  { runSemantic :: Context →KnowledgeComplex
                → (a, [TopologicalObstruction]) }

instance Functor SemanticM where
  fmap f (SemanticM g) = SemanticM $ \ctx kc →
    let (a, obs) = g ctx kc in (f a, obs)

instance Applicative SemanticM where
  pure a = SemanticM $ \_ _ →(a, [])
  (SemanticM f) <*> (SemanticM x) = SemanticM $ \ctx kc →
    let (fn, o1) = f ctx kc
        (a,  o2) = x ctx kc
    in (fn a, o1 ++ o2)

instance Monad SemanticM where
  (SemanticM m) >>= f = SemanticM $ \ctx kc →
    let (a, o1) = m ctx kc
        (b, o2) = runSemantic (f a) ctx kc
    in (b, o1 ++ o2)
```

9.3 Detection and Generation

Listing 3: Detection and generation functions

```haskell
detect :: SemTerm →SemanticM DetectionResult
detect term = SemanticM $ \ctx kc →
  let obstructions = findObstructions ctx kc term
      grounding   = checkGrounding ctx term
  in (DetectionResult grounding obstructions, obstructions)

generate :: SemType →SemanticM (Maybe (SemTerm, Derivation))
generate goalType = SemanticM $ \ctx kc →
  case searchInhabitant ctx kc goalType of
    Just (term, deriv) →
```

```
      case typeCheck ctx term goalType of
        Right True →((Just (term, deriv)), [])
        _          → (Nothing, [])
    Nothing →(Nothing, [])
```

9.4 The GCA Loop

Listing 4: Generate-Check-Abstain loop

```
gcaLoop :: SemTerm →SemType →Int → SemanticM GCAResult
gcaLoop claim goalType maxIter = go claim maxIter
  where
    go current 0 = return (GCAAbstain "max␣iterations")
    go current n = do
      result ←detect current
      case detectionObstructions result of
        [] → return (GCAVerified current)
        obs → do
          gen ← generate goalType
          case gen of
            Just (newTerm, deriv) →go newTerm (n - 1)
            Nothing →return (GCAAbstain "uninhabited␣type")
```

9.5 All Five Obstruction Detectors

Listing 5: Integrated obstruction detection

```
findObstructions ::Context →KnowledgeComplex
                → SemTerm →[TopologicalObstruction]
findObstructions ctx kc term = concat
  [ detectDisconnection ctx term     -- pi_0
  , detectCircularReasoning ctx term -- pi_1
  , detectIncoherentPaths ctx term   -- pi_2
  , detectHomologicalHoles kc term   -- H_n
  , detectTransportAnomaly ctx term -- holonomy
  ]
```

10 Experimental Design

We outline a rigorous experimental framework for validating the theoretical predictions.

10.1 Experiment 1: Topological Detection Accuracy

Construction 10.1 (Synthetic Hallucination Benchmark)**.** Construct a benchmark with known hallucination types:

1. **Circular reasoning (π_1)**: Generate reasoning chains where the conclusion is used as a premise. Measure π_1 detection rate.
2. **Unjustified inference (π_0)**: Generate claims connecting unrelated entities. Measure π_0 disconnection detection.
3. **Inconsistent justifications (π_2)**: Generate claims with two contradictory but individually valid justifications. Measure π_2 detection.
4. **Fabricated chains (H_n)**: Generate plausible but fabricated multi-step reasoning. Measure homological detection.
5. **Compositional drift (holonomy)**: Generate text where meaning silently shifts through context changes. Measure holonomy detection.

Metrics. For each hallucination type:

- **Precision**: fraction of detected hallucinations that are genuine (soundness measure).
- **Recall**: fraction of genuine hallucinations that are detected (completeness measure).
- **Topological specificity**: does the correct topological invariant activate for each type?

10.2 Experiment 2: GCA Loop Efficacy

Construction 10.2 (GCA Evaluation)**.** Evaluate the Generate-Check-Abstain loop:

1. Run a base LLM on questions with known answers.
2. Wrap with the GCA loop (varying $N = 1, 2, 3$ iterations).
3. Measure: hallucination rate reduction, abstention rate, answer quality on non-abstained responses.

Expected outcomes.

- Hallucination rate should decrease monotonically with GCA iterations.
- Abstention rate should increase (the system correctly refuses to answer when uncertain).
- Answer quality on non-abstained responses should be higher than the base LLM.
- The system should converge (fixed point of T) within 2–3 iterations.

10.3 Experiment 3: Persistent Homology Monitoring

Construction 10.3 (Real-Time Topological Monitor)**.** During long-form text generation:

1. Compute persistent homology on sliding windows of embeddings.
2. Track H_0 (connected components), H_1 (loops), H_2 (voids) across generation.
3. Correlate topological events (birth/death of persistent features) with human-annotated hallucination instances.

10.4 Experiment 4: Monad Convergence

Construction 10.4 (Monad Fixed-Point Analysis)**.** Empirically verify Theorem 7.10:

1. For a corpus of claims, compute $\mathsf{T}(\hat{a})$, $\mathsf{T}^2(\hat{a})$, $\mathsf{T}^3(\hat{a})$.

2. Measure the distance $d(\mathsf{T}^k(\hat{a}), \mathsf{T}^{k+1}(\hat{a}))$ for increasing k.
3. Verify that non-hallucinated claims are fixed points ($d \approx 0$ at $k = 1$) and hallucinated claims either converge to a corrected version or diverge to abstention.

11 Discussion

11.1 The Nature of the Synthesis

The central insight of this paper is that hallucination detection and correct generation are not independent problems requiring separate solutions, but *dual* aspects of a single mathematical structure. The adjunction $\mathsf{Generate} \dashv \mathsf{Detect}$ is not merely a formal observation but encodes a deep principle: the "best" way to detect hallucination *is* to attempt to regenerate the claim with full type-theoretic structure, and the "best" way to generate correctly *is* to detect and avoid the topological obstructions that characterize hallucination.

The semantic monad $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$ formalizes the "generate-check" loop that practitioners use informally (e.g., self-consistency decoding, chain-of-verification). Our contribution is to show that this loop has precise categorical structure—it is a monad—and that its fixed points are exactly the non-hallucinated claims (Theorem 7.10).

11.2 Cubical Type Theory and Computational Content

The HoTT framework gains computational content through cubical type theory [5]. In cubical type theory:

- Path types $a =_A b$ are represented as functions $I \to A$ from the interval I, with $p(0) = a$ and $p(1) = b$.
- The univalence axiom is provable (not postulated).
- Higher inductive types have computational rules.

For the semantic framework, cubical type theory provides:

- Constructive justification paths: a path $p\colon a =_A b$ is a *concrete function* from I to A, not merely an existence claim.
- Decidable type-checking: cubical type-checking is decidable [1], making the Detect functor computable.
- Transport with computation: $\mathsf{transport}_p(x)$ reduces to a concrete term, enabling the $\mathsf{Generate}$ functor to produce explicit witnesses.

11.3 Presheaf Models and Kripke–Joyal Semantics

The ∞-topos $\mathcal{S}\mathrm{em}_\infty$ can be modeled concretely as a category of presheaves. Let $\mathcal{C}$ be the category of contexts (with morphisms being context extensions). Then:

$$\mathcal{S}\mathrm{em}_\infty \simeq \mathrm{PSh}(\mathcal{C}) = [\mathcal{C}^{\mathrm{op}}, \boldsymbol{\infty}\textbf{-Grpd}]$$

In this model:

- A semantic type A is a functor $\mathcal{C}^{\text{op}} \to \boldsymbol{\infty}\textbf{-Grpd}$ assigning to each context Γ the ∞-groupoid $A(\Gamma)$ of valid claims of type A in context Γ.
- A morphism $f : \Gamma \to \Delta$ induces restriction $f^* : A(\Delta) \to A(\Gamma)$.
- The internal logic is interpreted via Kripke–Joyal semantics: "A is true at Γ" means $A(\Gamma)$ is inhabited.

This presheaf interpretation connects directly to the context-dependence of meaning and provides a model-theoretic foundation for the framework.

11.4 Practical Considerations

Computational complexity. Full type inhabitation search is undecidable in general. For practical deployment, we restrict to:

- Bounded search depth for derivations.
- Approximate homology computation via persistent homology on embedding filtrations.
- Heuristic type assignment using the LLM itself as a "type inference oracle."

Hybrid architecture. The POC architecture (Section 8) is designed as a *wrapper* around existing LLMs, not a replacement. The LLM handles the "creative" generation; the type-theoretic layer handles verification. This parallels the proof-assistant paradigm where automation proposes proof steps and the kernel verifies them.

Scalability. Persistent homology computation is $O(n^3)$ in the number of simplices (using standard persistence algorithms). For real-time monitoring during generation, we use:

- Sliding windows of bounded size.
- Incremental persistent homology updates [7].
- Approximate algorithms for high-dimensional data.

11.5 Limitations

1. **Type assignment is hard.** Mapping natural language claims to formal semantic types is itself an open problem. Our framework assumes this mapping exists but does not solve it.

2. **Knowledge base completeness.** The knowledge complex $\mathcal{K}_\bullet$ must be sufficiently rich to detect hallucinations. If the knowledge base itself has gaps, false negatives can occur.

3. **Computational intractability.** Full HoTT type-checking in higher dimensions is computationally expensive. Practical implementations will require approximations.

4. **The creative generation problem.** Type inhabitation search produces *correct* but potentially *boring* text. Balancing semantic correctness with creative generation is an open challenge.

11.6 Open Problems

1. **Efficient type inhabitation for natural language.** Can the search be guided by the LLM's probability distribution while maintaining type-theoretic guarantees?

2. **Incremental knowledge complex maintenance.** How should $\mathcal{K}_\bullet$ evolve as new knowledge arrives, and how does this affect existing homological features?

3. **Higher-dimensional hallucination.** Are there hallucination types corresponding to π_n for $n \geq 3$, and do they arise in practice?

4. **Quantitative sheaf cohomology.** Can sheaf cohomology computations be made efficient enough for real-time detection?

5. **The abstention–coverage tradeoff.** How to minimize abstention while maintaining zero hallucination rate?

12 Conclusion and Future Work

We have presented a unified framework for semantic correctness in language models that synthesizes topological hallucination detection and type-theoretic generation into a single coherent mathematical structure. The key results are:

1. The **Detection-Generation Duality**: detection and generation form an adjoint pair $\mathsf{Generate} \dashv \mathsf{Detect}$, revealing them as two sides of the same categorical coin.

2. The **Semantic Monad**: the composite $\mathsf{T} = \mathsf{Detect} \circ \mathsf{Generate}$ is a monad whose fixed points are precisely the non-hallucinated claims.

3. The **∞-Topos Framework**: the ambient universe $\mathcal{S}\mathrm{em}_\infty$ is an ∞-topos with HoTT as its internal logic, providing a foundation in which both detection (type-checking) and generation (proof search) are native operations.

4. The **Completeness Theorem**: the HHC together with type-theoretic generation gives a complete system—every hallucination is detected, and generation never produces undetected hallucinations.

5. The **POC Architecture**: a concrete system wrapping LLMs with a type-checking layer, implementing the Generate-Check-Abstain loop with persistent homology monitoring.

Future Work. The immediate next steps are:

- Implementation of the POC in a production system, beginning with the persistent homology monitor.
- Development of efficient type assignment algorithms (natural language to HoTT types).
- Empirical validation on the experimental framework of Section 10.
- Extension to cubical type theory for constructive content.

- Investigation of the ∞-topos structure for multi-modal semantic types (text + image + audio).

The central message of this work is that hallucination is not a problem to be patched but a *structural feature* of architectures that collapse the higher categorical structure of meaning. The solution is not better statistics but better mathematics: replacing the contractible codomain **Vect** with the richly-structured ∞-topos $\mathcal{S}\mathrm{em}_\infty$, and replacing probabilistic generation with certified type inhabitation. The adjunction $\mathsf{Generate} \dashv \mathsf{Detect}$ and the semantic monad T provide the organizing principle for this replacement.

References

[1] C. Angiuli, G. Brunerie, T. Coquand, R. Harper, K.-B. Hou, and D. R. Licata. Cartesian cubical computational type theory: Constructive reasoning with paths and equalities. In *Proceedings of CSL 2018*, 2018.

[2] J. Bolt, B. Coecke, F. Genovese, M. Lewis, D. Marsden, and R. Piedeleu. Interacting conceptual spaces I: Grammatical composition of concepts. In *Proceedings of QPL 2019*, pages 151–181, 2019.

[3] G. Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009.

[4] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384, 2010.

[5] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. In *Proceedings of TYPES 2015*, volume 69, pages 5:1–5:34, 2018.

[6] J. Curry. Sheaves, cosheaves and applications. *arXiv preprint arXiv:1303.3255v2*, 2014.

[7] T. K. Dey and Y. Wang. Computational topology for data analysis. Cambridge University Press, 2022.

[8] H. Edelsbrunner and J. L. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010.

[9] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.

[10] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[11] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[12] P. T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*, volumes 1–2. Oxford University Press, 2002.

[13] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS 2020*, 2020.

[14] K. Li et al. Inference-time intervention: Eliciting truthful answers from a language model. In *NeurIPS 2023*, 2023.

[15] M. Long. The Hallucination–Homotopy Correspondence: A complete classification of language model failure modes via algebraic topology and homotopy type theory. *GrokRxiv:2026.04.hallucination-homotopy-correspondence*, 2026.

[16] M. Long. Homotopical semantics for language models: Algebraic topological and homotopy type-theoretic approaches to the hallucination problem. *GrokRxiv:2026.04.yoneda-homotopical-semantics*, 2026.

[17] J. Lurie. *Higher Topos Theory*. Princeton University Press, 2009.

[18] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer-Verlag, 1994.

[19] G. Naitzat, A. Zhitnikov, and L.-H. Lim. Topology of deep neural networks. *Journal of Machine Learning Research*, 21(184):1–40, 2020.

[20] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[21] L. Ouyang et al. Training language models to follow instructions with human feedback. In *NeurIPS 2022*, 2022.

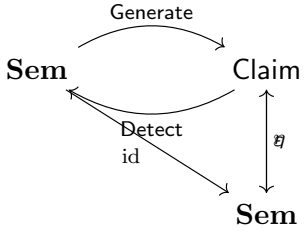
[22] M. Robinson. *Topological Signal Processing*. Springer, 2014.

[23] M. Shulman. All $(\infty, 1)$-toposes have strict univalent universes. *arXiv preprint arXiv:1904.07004*, 2019.

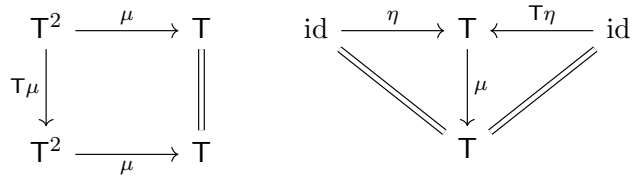
[24] V. Voevodsky. Univalent foundations. Lecture at IAS, Princeton, 2010.

A Key Commutative Diagrams

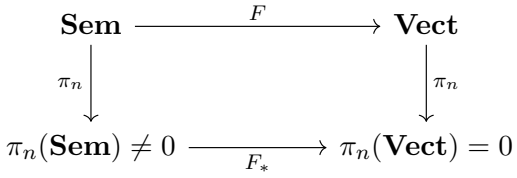
A.1 The Adjunction



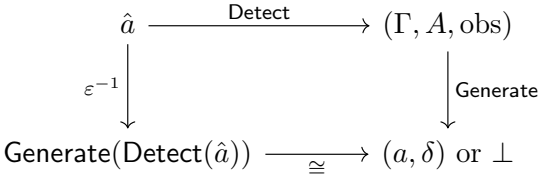
A.2 The Semantic Monad



A.3 The Functorial Collapse

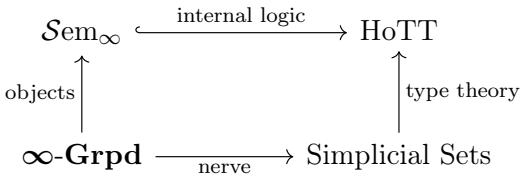


A.4 The Completeness Theorem



The isomorphism ε^{-1} exists if and only if $\hat{a}$ is not hallucinated (Theorem 7.9).

A.5 The ∞-Topos Structure



B Notation Summary

Symbol	Meaning	Reference
Sem	Semantic category	Theorem 3.1
Claim	Category of raw claims	Theorem 3.2
Detect	Detection functor	Theorem 3.4
$\mathsf{Generate}$	Generation functor	Theorem 3.6
T	Semantic monad $\mathsf{Detect} \circ \mathsf{Generate}$	Theorem 4.1
$\mathcal{S}\mathrm{em}_\infty$	∞-topos of semantic types	Theorem 5.1
$\mathcal{K}_\bullet$	Simplicial knowledge complex	Theorem 2.2
$\mathcal{F}$	Semantic sheaf	Theorem 6.1
$H_n(\mathcal{K}_\bullet)$	n-th homology group	Theorem 2.3
$H^n(\mathcal{K}_\bullet, \mathcal{F})$	n-th sheaf cohomology	Theorem 6.2
π_n	n-th homotopy group	Theorem 2.4
Hall	Category of hallucination events	[15]
hTop_*	Category of pointed homotopy types	[15]
η, ε	Unit and counit of adjunction	Theorem 3.7
μ	Monad multiplication	Theorem 4.1
$a =_A b$	Identity/path type	Theorem 2.1
$A \simeq B$	Semantic equivalence	Theorem 5.6