Type-Theoretic Generation

Hallucination-Free Language Generation
via Dependent Type Theory

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

April 2026

Abstract

We develop a foundational framework for *hallucination-free language generation* grounded in dependent type theory and categorical semantics. Our central thesis is that language generation should be reconceived as *type inhabitation* rather than statistical sampling: instead of computing $\arg\max_t P(t \mid \text{ctx})$, a correct language model computes $\arg\min_a C(a)$ subject to the constraint $\Gamma \vdash a : A$, where C is a cost function and the constraint enforces type-correctness in a dependent type theory of semantic meaning. We establish five generation principles that jointly guarantee hallucination freedom: (1) *Generation as Type Inhabitation*, recasting next-token prediction as proof search; (2) *Certified Derivations*, equipping every generated term with a polynomial-time-checkable derivation tree via the Curry–Howard correspondence; (3) *Abstention on Empty Types*, replacing hallucination with the type-theoretically correct response $\bot$ when the goal type is uninhabited; (4) *Context as Fibration*, modeling the context window as a dependent telescope $\Gamma = (x_1 : A_1, x_2 : A_2(x_1), \ldots)$ forming a fibration $p : E \to B$; and (5) *Attention as Weighted Limit*, replacing softmax-weighted sums in **Vect** with weighted limits in the semantic category **Sem**. We prove that this framework is *sound* (every generated term is semantically valid) and *complete relative to the knowledge base* (every derivable claim is generable). The framework is formalized in Haskell with GADTs and dependent types, providing executable specifications for all core constructs.

Contents

*Corresponding author. matthew@yonedaai.com

1 Introduction

1.1 The Problem: Statistical Generation and Hallucination

Contemporary large language models (LLMs) generate text by iteratively sampling from a learned conditional distribution:

$$t_{n+1} = \arg\max_{t \in \mathcal{V}} P(t \mid t_1, \ldots, t_n; \theta) \tag{1}$$

where $\mathcal{V}$ is the vocabulary, θ denotes model parameters, and P is estimated by a transformer architecture [1]. This formulation treats language as a stochastic process over flat token sequences. The resulting models achieve remarkable fluency but suffer from a fundamental defect: *hallucination*, the generation of text that is fluent, confident, and wrong [2, 3].

Hallucination is not a bug to be patched. It is a *structural inevitability* arising from a category-theoretic mismatch between the space of linguistic meaning and the space in which models operate [4]. Current LLMs implement a functor

$$F : \mathbf{Lang} \to \mathbf{Vect}_{\mathbb{R}}^{\mathrm{fin}} \tag{2}$$

mapping linguistic contexts to finite-dimensional real vector spaces. As established in [5], this functor is neither faithful nor full: it collapses the higher categorical structure of meaning—justifications, coherences, and their interactions—into a contractible space where all topological obstructions vanish. The information lost is precisely the information needed to distinguish valid claims from hallucinations.

1.2 Our Contribution: Generation as Type Inhabitation

This paper proposes a radical alternative. Instead of sampling from a distribution, we reconceive language generation as *type inhabitation* in a dependent type theory of semantic meaning. The core replacement is:

$$\arg\max_{t} P(t \mid \mathrm{ctx}) \quad \longmapsto \quad \arg\min_{a} C(a) \text{ subject to } \Gamma \vdash a : A \tag{3}$$

where:

- Γ is a *dependent context* (telescope) encoding the conversational state,
- A is the *semantic type* of the desired response,
- a is a *term* (semantic content) inhabiting A,
- $\Gamma \vdash a : A$ is a *typing judgment* ensuring semantic validity,
- $C(a)$ is a cost function encoding pragmatic preferences (see Theorem 1.1 below).

Generation becomes *proof search* in the type theory, and every generated term comes equipped with a *derivation tree* δ witnessing $\Gamma \vdash a : A$. When A is uninhabited—when no valid response exists—the system returns $\bot$ rather than hallucinating.

1.3 The Five Generation Principles

Our framework rests on five principles, each with a precise formal statement:

P1. Generation as Type Inhabitation. Generation is proof search in dependent type theory, not sampling from a distribution (Section 3).

P2. Certified Derivations. Every generated term $a : A$ comes with a derivation tree δ witnessing $\Gamma \vdash a : A$, checkable in polynomial time (Section 4).

P3. Abstention on Empty Types. When A is uninhabited, the model returns $\bot$ ("I don't know") rather than hallucinating (Section 5).

P4. Context as Fibration. The context window is a dependent telescope forming a fibration $p : E \to B$, not a flat sequence (Section 6).

P5. Attention as Weighted Limit. Attention computes a weighted limit in **Sem** (when it exists) rather than a weighted sum in **Vect** (Section 7).

1.4 The Cost Function

Definition 1.1 (Derivation Cost)**.** The *cost function* $C : \text{Term} \to \mathbb{N}$ assigns to each well-typed term a the depth of its minimal derivation tree:

$$C(a) \;=\; \min\{\, |\delta| \;\big|\; \delta \text{ derives } \Gamma \vdash a : A \,\}$$

where $|\delta|$ denotes the depth of the derivation tree δ (the length of the longest root-to-leaf path). When the set of derivations is empty (the term does not type-check), we set $C(a) = \infty$.

Remark 1.2 (Richer Cost Functions). The derivation-depth cost is the simplest well-defined choice. In practice, one may wish to incorporate pragmatic preferences such as brevity of the rendered natural-language text, stylistic appropriateness, or relevance to the conversational context. These can be modeled by replacing $\mathbb{N}$ with a more general ordered monoid and defining C as a weighted combination of derivation depth and pragmatic scores. We leave the systematic study of richer cost functions to future work; the results of this paper hold for any cost function satisfying $C(a) < \infty \iff \Gamma \vdash a : A$.

1.5 Related Work

The idea that language has compositional structure amenable to categorical analysis has deep roots. Lambek's syntactic calculus [10] models grammar as a residuated lattice. The DisCoCat framework of Coecke, Sadrzadeh, and Clark [11] maps grammatical types to objects in a compact closed category, composing word meanings via categorical operations. Compositional distributional models [12] combine distributional word vectors with syntactic structure. However, none of these frameworks address the *higher homotopical* structure of meaning or the problem of hallucination.

On the type-theoretic side, Martin-Löf type theory [6] provides the foundation for our dependent types; the Curry–Howard correspondence [8, 9] establishes the proofs-as-programs paradigm we exploit; and homotopy type theory [7] provides the higher-dimensional generalization where identity types carry non-trivial structure.

Our work builds directly on the homotopical semantics framework of Long [4] and the Hallucination–Homotopy Correspondence [5], extending the diagnostic theory developed

there into a *constructive* framework for hallucination-free generation.

1.6 Outline

Section 2 develops the dependent type theory for language. Section 3 reformulates generation as proof search. Section 4 establishes the certified derivation framework. Section 5 treats abstention on empty types. Section 6 develops context as fibration. Section 7 reformulates attention as a weighted limit. Section 8 presents the HoTT-LM architecture. Section 9 describes the Haskell formalization. Section 10 discusses implications, limitations, and future work. Section 11 concludes.

2 Dependent Type Theory for Language

We construct a dependent type theory $\mathcal{T}_{\mathbf{Sem}}$ whose types encode semantic content and whose terms are valid linguistic expressions. The theory is designed so that the typing judgment $\Gamma \vdash a : A$ captures exactly the notion that "a is a semantically valid expression of type A in context Γ."

2.1 Semantic Types as Homotopy Types

Definition 2.1 (Semantic Type)**.** A *semantic type* is a homotopy type A in a universe $\mathcal{U}$ equipped with the following structure:

- **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 at all levels.

The grammar of semantic types is:

$$\begin{aligned} A, B ::= \mathsf{Entity}(s) \mid \mathsf{Prop}(s) \mid \Pi(x : A).B(x) \mid \Sigma(x : A).B(x) \\ \mid\ a =_A b \mid A + B \mid \mathbf{0} \mid \mathbf{1} \mid \mathcal{U}_n \end{aligned} \tag{4}$$

where $\mathsf{Entity}(s)$ is the type of entities named s, $\mathsf{Prop}(s)$ is the type of evidence for proposition s, Π and Σ are dependent product and sum, $a =_A b$ is the identity (path) type, $\mathbf{0}$ is the empty type (absurdity), $\mathbf{1}$ is the unit type (trivial truth), and $\mathcal{U}_n$ is the n-th universe.

Remark 2.2 (Why Homotopy Types?). In classical type theory, the identity type $a =_A b$ is a mere proposition—it is either inhabited (uniquely) or empty. In homotopy type theory, $a =_A b$ can carry non-trivial structure: multiple distinct justifications for why a and b are "the same," and higher equivalences between those justifications. This additional structure is *exactly* what is needed to distinguish between a claim that is supported by a valid chain of reasoning and one that merely appears plausible. A statistical model, operating in **Vect**, sees both claims as nearby points—their cosine similarity is high—but the type-theoretic model sees that one has inhabited identity types connecting it to grounded knowledge while the other does not.

2.2 Contexts as Dependent Telescopes

Definition 2.3 (Dependent Context)**.** A *dependent context* (or *telescope*) is a sequence

$$\Gamma = (x_1 : A_1,\ x_2 : A_2(x_1),\ x_3 : A_3(x_1, x_2),\ \ldots,\ x_n : A_n(x_1, \ldots, x_{n-1}))$$

where each type A_i may depend on all preceding variables $x_1, \ldots, x_{i-1}$. The *empty context* is denoted $\cdot$ or ε. Context extension is written $\Gamma, x : A$ where A is a type well-formed in context Γ.

This is a fundamental departure from standard LLM context windows. In a transformer, the context is a flat sequence of tokens $(t_1, t_2, \ldots, t_n)$ embedded into $\mathbb{R}^d$ via a lookup table. There is no mechanism for later tokens to *depend* on the types of earlier tokens. In our framework, the context is inherently fibered: the type of x_3 depends on the actual values of x_1 and x_2, not merely their positions.

Example 2.4 (Conversational Context)**.** Consider the context arising from the exchange:

> *User:* "Who is the president of France?"
> *System:* "Emmanuel Macron."
> *User:* "When was he elected?"

The dependent telescope is:

$$\begin{aligned}\Gamma = (\ & q_1 : \mathsf{Prop}(\text{“president of France”}),\\ & a_1 : \mathsf{Entity}(\text{“Macron”}) \times (a_1 =_{\mathsf{President}} q_1),\\ & q_2 : \mathsf{Prop}(\text{“election date of”}\ a_1)\ )\end{aligned}$$

The type of the final query q_2 *depends on* the value a_1 established earlier. The pronoun "he" is resolved not by coreference heuristics but by the dependent structure: q_2 is typed over a_1.

2.3 The Typing Judgment

Definition 2.5 (Typing Judgment)**.** A *typing judgment* is a statement of the form

$$\Gamma \vdash a : A$$

asserting that term a inhabits type A in context Γ. The judgment is *derivable* if there exists a derivation tree δ constructed from the inference rules of $\mathcal{T}_{\mathbf{Sem}}$.

The standard inference rules include:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ \textsc{Var} \qquad \frac{\Gamma, x : A \vdash b : B(x)}{\Gamma \vdash \lambda x.b : \Pi(x : A).B(x)}\ \Pi\text{-}\textsc{Intro} \tag{5}$$

$$\frac{\Gamma \vdash f : \Pi(x : A).B(x) \quad \Gamma \vdash a : A}{\Gamma \vdash f\, a : B(a)}\ \Pi\text{-}\textsc{Elim} \tag{6}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B(a)}{\Gamma \vdash (a, b) : \Sigma(x : A).B(x)}\ \Sigma\text{-}\textsc{Intro} \tag{7}$$

$$\frac{\Gamma \vdash a : A}{\Gamma \vdash \mathsf{refl}_a : a =_A a}\ \textsc{Refl} \qquad \frac{\Gamma \vdash p : a =_A b \quad \Gamma, x : A \vdash C(x) \quad \Gamma \vdash c : C(a)}{\Gamma \vdash \mathsf{transport}^C(p, c) : C(b)}\ \textsc{Transport} \tag{8}$$

In addition, we include rules for *grounded terms*:

$$\frac{s \in \mathcal{K} \quad s \text{ witnesses } A}{\Gamma \vdash \mathsf{ground}(s) : A}\ \textsc{Ground} \tag{9}$$

where $\mathcal{K}$ is the knowledge base. This rule anchors the type theory to external facts.

2.4 Semantic Terms

Definition 2.6 (Semantic Term)**.** The *semantic terms* of $\mathcal{T}_{\mathbf{Sem}}$ are defined by the grammar:

$$\begin{aligned} a, b ::= {} & x \mid \lambda x.a \mid a\, b \mid (a, b) \mid \pi_1(a) \mid \pi_2(a) \\ & \mid \mathsf{refl}_a \mid \mathsf{transport}^C(p, a) \mid \mathsf{ground}(s) \\ & \mid \mathsf{infer}(a, j) \mid \mathsf{absurd}(a) \end{aligned}$$

where x ranges over variables, s over sources, and j over justifications. The term $\mathsf{absurd}(a)$ witnesses the elimination of $\mathbf{0}$: given $a : \mathbf{0}$, anything follows (but $\mathbf{0}$ is never inhabited, so this rule is vacuous).

Proposition 2.7 (Decidability of Type Checking)**.** *Type checking in $\mathcal{T}_{\mathbf{Sem}}$—given Γ, a, and A, deciding whether $\Gamma \vdash a : A$ is derivable—is decidable, and the decision procedure runs in time polynomial in $|\Gamma| + |a| + |A|$, provided the knowledge base $\mathcal{K}$ supports constant-time membership queries.*

Proof. The type theory $\mathcal{T}_{\mathbf{Sem}}$ is a subsystem of Martin-Löf type theory with a decidable type-checking algorithm [6, 17]. The inference rules are syntax-directed: each term form has a unique applicable rule. The only non-structural step is the Ground rule, which requires a lookup in $\mathcal{K}$; by assumption, this is $O(1)$. The derivation tree has depth at most $|a|$ (each subterm is checked once), and each step is $O(|A|)$ for type comparison. The total cost is $O(|a| \cdot |A|)$, which is polynomial. □

Remark 2.8 (Two-Layer Type Theory: HoTT for Semantics, MLTT for Decision). A subtlety deserves clarification. Full homotopy type theory (HoTT) with the univalence axiom does *not* have a straightforward decidable type-checking algorithm—indeed, giving a com-

putational interpretation to univalence is one of the central motivations for cubical type theory [19]. Our framework employs a *two-layer* approach:

1. **Semantic layer (HoTT).** The semantic category **Sem** and the higher-categorical structure of meaning (path types, higher coherences, univalence) live in HoTT. This layer provides the *model theory*: the mathematical space in which semantic validity is defined.
2. **Decision layer (MLTT / cubical fragment).** The type-checking algorithm and proof search operate in a decidable fragment: either standard Martin-Löf type theory (MLTT) or a cubical type theory such as that of Cohen et al. [19]. In this fragment, identity types are decidable, type-checking is algorithmic, and the computational content of univalence is available via Glue types.

Concretely, Theorem 2.7 applies to the decision layer. When a proof obligation involves univalence (e.g., transport along a type equivalence), the decision procedure invokes the cubical interpretation, which provides a computational reduction. The semantic layer guarantees that the decision layer is *sound*: every judgment derivable in the cubical fragment is valid in the HoTT model.

3 Generation as Proof Search

3.1 The Inhabitation Problem

Definition 3.1 (Type Inhabitation Problem)**.** Given a context Γ and a type A, the *type inhabitation problem* asks:

$$\exists\, a.\ \Gamma \vdash a : A\ ?$$

If the answer is affirmative, we seek a *witness* a together with its derivation δ.

In conventional language generation, the model is asked: "What is the most likely next token?" In type-theoretic generation, the model is asked: "Does this type have an inhabitant, and if so, what is it?" The distinction is profound. The statistical question always has an answer (there is always a most-probable token), even when no semantically valid continuation exists. The type-theoretic question may have the answer "no"—and this "no" is the correct response, not a failure.

Theorem 3.2 (Soundness of Type-Theoretic Generation)**.** *If the generation procedure returns a term a with derivation δ, then $\Gamma \vdash a : A$ is valid in $\mathcal{T}_{\mathbf{Sem}}$: the generated term is semantically correct.*

Proof. By construction, the generation procedure performs proof search in $\mathcal{T}_{\mathbf{Sem}}$ and returns a term only when a complete derivation tree δ has been constructed. The derivation tree δ is a sequence of applications of the inference rules (5)–(8) and (9). Each inference rule is sound (preserves validity of judgments) by the metatheory of dependent type theory [6]. Therefore, the conclusion $\Gamma \vdash a : A$ of δ is valid. □

3.2 Proof Search Strategy

The proof search strategy for type inhabitation in $\mathcal{T}_{\mathbf{Sem}}$ combines several standard techniques adapted to the semantic setting:

1. **Context lookup.** Check whether Γ already contains a variable $x : A$. If so, return x with the VAR derivation.
2. **Knowledge base query.** Check whether $\mathcal{K}$ contains a source s witnessing A. If so, return $\mathsf{ground}(s)$ with the GROUND derivation.
3. **Decomposition.** If A has the form $\Pi(x : B).C(x)$, introduce a fresh variable $x : B$, extend the context to $\Gamma, x : B$, and search for an inhabitant of $C(x)$. If $A = \Sigma(x : B).C(x)$, search for $b : B$ and then $c : C(b)$. If $A = a =_B b$, attempt to construct a path.
4. **Modus ponens.** Search for $f : \Pi(x : B).A$ and $b : B$ in Γ or $\mathcal{K}$, and return $f\, b$.
5. **Transport.** If $A = C(b)$ and there exist $c : C(a)$ and $p : a =_B b$, return $\mathsf{transport}^C(p, c)$.
6. **Abstention.** If all strategies are exhausted, return $\bot$ (no inhabitant found).

Definition 3.3 (Proof Search Depth)**.** The *search depth* d bounds the maximum depth of the derivation tree. A search at depth d explores all derivations with at most d inference steps. We write $\mathsf{search}_d(\Gamma, A)$ for the depth-bounded search.

Proposition 3.4 (Completeness at Depth d)**.** *If there exists a derivation δ of $\Gamma \vdash a : A$ with $|\delta| \leq d$, then $\mathsf{search}_d(\Gamma, A)$ finds an inhabitant.*

Proof. The proof search enumerates all derivation trees up to depth d by systematic application of the inference rules in the order specified above. Since the rule set is finite and each rule reduces the proof obligation to smaller subgoals, the search tree at depth d is finite. Exhaustive search over a finite tree finds every solution. □

3.3 Comparison with Statistical Generation

Proposition 3.5 (Statistical Generation is Unsound)**.** *Standard autoregressive generation (1) is unsound in the type-theoretic sense: there exist instances where the model generates t with $P(t \mid \mathrm{ctx}) > 0.99$ yet $\Gamma \vdash t : A$ is not derivable.*

Proof. Consider the type $A = \mathsf{Prop}(\text{“The Eiffel Tower is in Berlin”})$. A well-trained LLM assigns low probability to this claim in isolation. However, in a context Γ containing the statements “The Eiffel Tower is a famous landmark” and “Berlin is a European capital,” standard attention-based generation may assign non-trivial probability to continuations associating the tower with Berlin (via compositional similarity in **Vect**), producing a statistically plausible but semantically invalid claim. This is the celebrated “entity confusion” hallucination [2]. In $\mathcal{T}_{\mathbf{Sem}}$, the judgment $\Gamma \vdash$ “Eiffel Tower is in Berlin” $: A$ requires a grounding path from the knowledge base to this claim, which does not exist; thus the type is uninhabited and the system abstains. □

4 Certified Derivations: Curry–Howard for Language

4.1 The Curry–Howard Correspondence for Semantic Generation

The Curry–Howard correspondence [8] establishes a bijection between proofs in intuitionistic logic and programs in typed lambda calculus. We extend this correspondence to the semantic setting:

Logic	**Type Theory**	**Language Generation**
Proposition φ	Type A	Semantic type (query)
Proof of φ	Term $a : A$	Valid response
Proof tree	Derivation δ	Certificate of validity
Hypothesis φ	Variable $x : A$	Context element
Modus ponens	Function application	Inference step
Conjunction	Product type $A \times B$	Composite claim
Implication $\varphi \Rightarrow \psi$	Function type $A \to B$	Entailment
Absurdity $\bot$	Empty type $\mathbf{0}$	Unanswerable query

Definition 4.1 (Derivation Tree)**.** A *derivation tree* δ for a judgment $\Gamma \vdash a : A$ is a finite tree whose:

- *leaves* are instances of Var, Ground, or Refl,
- *internal nodes* are instances of Π-Intro, Π-Elim, Σ-Intro, Transport, etc.,
- *root* is the judgment $\Gamma \vdash a : A$.

We write $|\delta|$ for the *depth* (longest root-to-leaf path) and $\|\delta\|$ for the *size* (number of nodes).

Theorem 4.2 (Polynomial-Time Verification)**.** *Given* Γ*,* a*,* A*, and a derivation tree* δ*, verifying that* δ *is a valid derivation of* $\Gamma \vdash a : A$ *can be done in time* $O(\|\delta\| \cdot |A|)$*.*

Proof. Verification proceeds bottom-up through δ. At each node, we check:

1. The inference rule applied at this node is valid.
2. The premises (children) have the correct judgmental form.
3. Types match: the types appearing in the conclusion agree with those produced by the premises.

Each check involves comparing types, which is $O(|A|)$ in the worst case (types are tree-structured, so comparison is linear in their size). There are $\|\delta\|$ nodes, giving total cost $O(\|\delta\| \cdot |A|)$. Since $\|\delta\| \leq c \cdot |a|$ for a constant c depending on the rule set (each subterm generates at most one node), this is polynomial. □

4.2 The Derivation Algebra

Definition 4.3 (Derivation Algebra)**.** The *derivation algebra* $\mathcal{D}$ over $\mathcal{T}_{\mathbf{Sem}}$ is the free algebra generated by the constructors:

$$\begin{aligned}\delta ::= {} & \mathsf{AxiomD}(s) \mid \mathsf{IntroD}(x, \delta) \mid \mathsf{ElimD}(\delta_1, \delta_2) \\ & \mid \mathsf{PathIntroD}(a, b, \delta) \mid \mathsf{TransportD}(\delta, p) \mid \mathsf{UnivalenceD}(\delta)\end{aligned}$$

subject to the constraint that each constructor application preserves the typing judgment.

Proposition 4.4 (Derivation Compositionality)**.** *If δ_1 derives $\Gamma \vdash f : \Pi(x : A).B(x)$ and δ_2 derives $\Gamma \vdash a : A$, then $\mathsf{ElimD}(\delta_1, \delta_2)$ derives $\Gamma \vdash f\ a : B(a)$. More generally, derivations compose according to the categorical structure of $\mathcal{T}_{\mathbf{Sem}}$.*

Proof. This follows directly from the soundness of the Π-Elim rule. The derivation $\mathsf{ElimD}(\delta_1, \delta_2)$ applies Π-Elim with premises δ_1 and δ_2, yielding the conclusion $\Gamma \vdash f\ a : B(a)$ by the rule's schema. □

4.3 Derivations as First-Class Objects

A crucial feature of our framework is that derivations are *first-class*: they are data structures that can be inspected, composed, serialized, and transmitted alongside the generated text. This enables:

1. **Auditability.** Every generated claim can be traced back through its derivation to the grounding sources.
2. **Incremental verification.** A verifier can check the derivation without re-performing the search.
3. **Composable generation.** Derivations from different sources can be combined via the derivation algebra.
4. **Explanation.** The derivation tree itself serves as a natural-language explanation of why the claim is valid.

5 Abstention on Empty Types

5.1 The Empty Type as "I Don't Know"

Definition 5.1 (Uninhabited Type)**.** A type A is *uninhabited* in context Γ if there is no term a such that $\Gamma \vdash a : A$ is derivable. Equivalently, A is semantically equivalent to $\mathbf{0}$ in context Γ.

Theorem 5.2 (Conditional Correct Abstention)**.** *If the depth-bounded search $\mathsf{search}_d(\Gamma, A)$ is* complete *for the type A—that is, every derivation of $\Gamma \vdash a : A$ has depth at most d—and A is uninhabited in context Γ, then the type-theoretic generation procedure returns $\bot$.*

Proof. By Theorem 3.4, if the search is complete at depth d, then $\mathsf{search}_d(\Gamma, A)$ finds every inhabitant with a derivation of depth at most d. If A is uninhabited, no derivation exists at any depth; a fortiori, none exists at depth d. The procedure returns $\bot$ after exhausting the finite search space at depth d. By soundness (Theorem 3.2), the only alternative to $\bot$ would be a term with a valid derivation, but none exists. □

Remark 5.3 (Incompleteness of Abstention). When A is uninhabited but the search depth d is insufficient to explore all possible derivation strategies, the procedure still returns $\bot$ (since no inhabitant is found), but for the weaker reason that the search was exhausted rather than that uninhabitedness was established. Conversely, when A *is* inhabited but

only via derivations of depth greater than d, the procedure incorrectly returns $\bot$—a false abstention. This is the soundness–completeness tradeoff inherent in bounded search: the procedure is always *sound* (it never returns an invalid term) but may be *incomplete* (it may abstain when a valid answer exists beyond the search horizon).

5.2 The Abstention Spectrum

Not all "I don't know" responses are equal. We distinguish several levels of abstention:

Definition 5.4 (Abstention Level)**.** The *abstention level* for a query of type A in context Γ is:

(i) **Provably uninhabited**: $\Gamma \vdash A \simeq \mathbf{0}$ (there is a proof that A is empty).

(ii) **Search-exhausted**: $\mathsf{search}_d(\Gamma, A) = \bot$ but A is not provably uninhabited (the answer may exist beyond the search depth).

(iii) **Contextually uninhabited**: A is inhabited in some extension $\Gamma' \supseteq \Gamma$ but not in Γ itself (more context is needed).

(iv) **Knowledge-bounded**: A would be inhabited if the knowledge base $\mathcal{K}$ were extended with additional facts.

Proposition 5.5 (Abstention Preserves Consistency)**.** *The abstention response $\bot$ never introduces a contradiction. Formally, if Γ is consistent (i.e., $\Gamma \nvdash a : \mathbf{0}$ for any a), then extending the interaction with the response $\bot$ preserves consistency.*

Proof. The response $\bot$ adds no new judgments to the context. It is a *metalevel* statement ("no term found"), not an object-level term. Since no new typing judgments are introduced, the consistency of Γ is trivially preserved. □

5.3 Comparison with Statistical Abstention

Statistical models can be augmented with abstention mechanisms (calibrated uncertainty, "I don't know" tokens, etc.), but these are *heuristic*: they rely on the model's uncertainty estimates, which are known to be poorly calibrated [16]. A model may be highly confident in a hallucinated claim (low entropy, wrong answer) or highly uncertain about a simple, well-grounded fact (high entropy, correct answer available).

In contrast, type-theoretic abstention is *principled*: it reflects a genuine logical impossibility, not a statistical judgment. The type A is uninhabited because no valid derivation exists—not because the model is "unsure."

Example 5.6 (Correct Abstention)**.** Consider the query $A = \mathsf{Prop}$("number of planets in Andromeda"). The knowledge base $\mathcal{K}$ contains no source witnessing A. No inference chain from existing knowledge yields an inhabitant of A. The type-theoretic system returns $\bot$ with abstention level (iv): knowledge-bounded. A statistical LLM, by contrast, would likely generate a confident-sounding number ("approximately 400 billion planets"), which, while not implausible, is not grounded in any specific source and constitutes a hallucination.

6 Context as Fibration

6.1 The Fibration Structure of Context

Definition 6.1 (Semantic Fibration)**.** The dependence of meaning on context defines a *fibration*

$$p : E \to B$$

where:

- B is the *base space* of contexts: the space of all well-formed telescopes Γ.
- E is the *total space*: the space of all pairs (Γ, a) where a is a term well-typed in Γ.
- $p(\Gamma, a) = \Gamma$ is the projection.
- The *fiber* $p^{-1}(\Gamma) = \{a \mid \Gamma \vdash a : A\}$ is the space of valid claims in context Γ.

The fibration perspective reveals a fundamental problem with statistical models. An LLM's conditional distribution $P(\text{token} \mid \text{context})$ is a *measure* μ_Γ on the total space E. Hallucination occurs when the measure leaks outside the fiber:

$$\mu_\Gamma(E \setminus p^{-1}(\Gamma)) > 0. \tag{10}$$

That is, the model assigns positive probability to terms that do not inhabit the correct type for the given context.

Theorem 6.2 (Fibration Soundness)**.** *In the type-theoretic generation framework, the generation output is always contained in the fiber:*

$$\textit{If } \mathsf{generate}(\Gamma, A) = \mathsf{Just}(a, \delta), \textit{ then } a \in p^{-1}(\Gamma).$$

Proof. By Theorem 3.2, δ witnesses $\Gamma \vdash a : A$. By definition of the fiber, $a \in p^{-1}(\Gamma)$ iff $\Gamma \vdash a : A$ for some type A. The result follows immediately. □

6.2 Path Lifting and Transport

The fibration structure supports *path lifting*: if we have a path $p : \Gamma =_B \Gamma'$ in the base space (a context transformation) and a term $a \in p^{-1}(\Gamma)$, we can *transport* a along p to obtain a term $a' \in p^{-1}(\Gamma')$.

Definition 6.3 (Semantic Transport)**.** Given a context morphism $\sigma : \Gamma \to \Gamma'$ (a substitution) and a term $\Gamma \vdash a : A$, the *transport* of a along σ is:

$$\mathsf{transport}^\sigma(a) : A[\sigma]$$

where $A[\sigma]$ denotes the type A with the substitution σ applied.

Proposition 6.4 (Transport Preserves Validity)**.** *If $\Gamma \vdash a : A$ and $\sigma : \Gamma \to \Gamma'$ is a valid context morphism, then $\Gamma' \vdash \mathsf{transport}^\sigma(a) : A[\sigma]$.*

Proof. This is a standard property of dependent type theories with substitution [6]. The substitution σ maps each variable $x_i : A_i$ in Γ to a term $\sigma(x_i) : A_i[\sigma|_{<i}]$ in Γ'. Applying

σ to the derivation δ of $\Gamma \vdash a : A$ yields a derivation $\delta[\sigma]$ of $\Gamma' \vdash a[\sigma] : A[\sigma]$ by the substitution lemma. □

6.3 Fiber Coherence

Definition 6.5 (Fiber Coherence)**.** A context Γ is *coherent* if for every pair of terms $a, b \in p^{-1}(\Gamma)$ and every path $q : a =_A b$ in the fiber, the path q can be extended to a global section of the fibration. Equivalently, the fiber is *path-connected* in a way compatible with the base.

Proposition 6.6 (Incoherent Context Detection)**.** *If the context Γ contains contradictory claims—i.e., $\Gamma \vdash a : A$ and $\Gamma \vdash b : \neg A$ for some A—then $p^{-1}(\Gamma)$ is equivalent to the total space (everything is derivable via Absurd). The type-theoretic system can detect this by checking whether $\Gamma \vdash c : \mathbf{0}$ for any c.*

Proof. If Γ is inconsistent, then by the principle of explosion ($\mathbf{0}$-elimination), every type is inhabited: $\Gamma \vdash \mathsf{absurd}(\mathsf{app}(b, a)) : C$ for any C. Detecting inconsistency reduces to checking inhabitation of $\mathbf{0}$. □

Remark 6.7 (Fibrations vs. Flat Context). The fibration perspective explains several known LLM failure modes:

- **Entity confusion.** The LLM confuses entities because the flat context does not fiber over entity types—all entities are equidistant in **Vect**.
- **Temporal inconsistency.** Time-dependent facts require a fibration over temporal contexts; the flat representation conflates past and present.
- **Scope errors.** Quantifier scope is naturally represented by the nesting depth in a dependent telescope; flat contexts lose this structure.

7 Categorical Attention as Weighted Limit

7.1 The Semantic Category

Before defining categorical attention, we must give a precise definition of the ambient category **Sem**.

Definition 7.1 (The Semantic Category **Sem**)**.** The *semantic category* **Sem** is a locally small 1-category defined as follows:

- **Objects.** The objects of **Sem** are the closed semantic types A of the type theory $\mathcal{T}_{\mathbf{Sem}}$ (Theorem 2.1), together with their spaces of inhabitants. Formally, $\mathrm{ob}(\mathbf{Sem}) = \{A \in \mathcal{U} \mid A \text{ is a well-formed semantic type}\}$.
- **Morphisms.** A morphism $f : A \to B$ in **Sem** is a closed term $f : \Pi(x : A).\, B$—a type-theoretically well-typed function from A to B. Composition is given by function composition in the type theory: $g \circ f = \lambda x.\, g(f(x))$.
- **Identity.** The identity morphism on A is $\mathrm{id}_A = \lambda x.\, x : \Pi(x : A).\, A$.

- **Path structure.** Two morphisms $f, g : A \to B$ are identified when there exists a path $p : f =_{\Pi(x:A).B} g$ in the function type. This endows each hom-set with a groupoid structure, but for the purposes of weighted limits we work at the 1-categorical level (taking path-components).

Remark 7.2 (Higher-Categorical Refinement). In the full HoTT setting, **Sem** is naturally an $(\infty, 1)$-category (with the path types providing the higher morphisms). For the results of this paper, the 1-categorical truncation suffices: Theorem 7.5 and Theorem 7.6 require only ordinary (co)limits, not homotopy limits. A treatment using $(\infty, 1)$-categorical weighted limits (in the sense of Lurie [22]) is left to future work.

Definition 7.3 (Standing Assumption: W-Completeness)**.** We assume throughout that **Sem** is *W-complete* for all weight functors $W : \mathcal{I}^{\mathrm{op}} \to \mathbf{Set}$ arising from finite indexing categories $\mathcal{I}$ relevant to the context window. Concretely, this means that **Sem** admits all finite weighted limits. This assumption is justified by the fact that the semantic types include products, function types, and path types, which suffice to construct the required limits [20].

7.2 Standard Attention as a Linear Operation

In the transformer architecture, attention is computed as:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \tag{11}$$

This is a weighted sum in **Vect**:

$$\mathrm{output}_i = \sum_j \alpha_{ij} v_j, \qquad \alpha_{ij} = \frac{\exp(q_i \cdot k_j / \sqrt{d_k})}{\sum_\ell \exp(q_i \cdot k_\ell / \sqrt{d_k})}$$

The attention weights α_{ij} are non-negative reals summing to 1—they form a probability distribution over the context positions. The output is a *convex combination* of value vectors.

The fundamental problem: convex combinations in **Vect** do not preserve semantic structure. The "average" of two semantically distinct claims is generically a hallucination.

7.3 Attention as Weighted Limit

Definition 7.4 (Categorical Attention)**.** Let $\mathcal{I}$ be a finite category (the *indexing category*, representing the structure of the context window) and let $D : \mathcal{I} \to \mathbf{Sem}$ be a *diagram* in the semantic category (assigning semantic content to each context position and semantic morphisms between related positions). Let $W : \mathcal{I}^{\mathrm{op}} \to \mathbf{Set}$ be a *weight functor* (assigning a set of "attention modes" to each position).

The *categorical attention output* is the *W-weighted limit*:

$$\mathrm{Att}(D, W) = \varprojlim^{W} D = \int_{i \in \mathcal{I}} D(i)^{W(i)} \tag{12}$$

where the right-hand side is the *end* formulation of the weighted limit.

Explicitly, $\lim^W D$ is (when it exists) the object $L \in \mathbf{Sem}$ equipped with a natural family of maps

$$\pi_i : L \to D(i)^{W(i)}$$

for each $i \in \mathcal{I}$, universal among all such families.

Theorem 7.5 (Weighted Limit Preserves Coherence)**.** *Assume that* **Sem** *is W-complete for the weight $W : \mathcal{I}^{\mathrm{op}} \to \mathbf{Set}$ (i.e.,* **Sem** *admits W-weighted limits for all diagrams indexed by $\mathcal{I}$; see Theorem 7.1). Then:*

(i) If the diagram $D : \mathcal{I} \to \mathbf{Sem}$ is coherent *(all diagram morphisms compose consistently), then the weighted limit $\lim^W D$ exists and is a semantically valid object in* **Sem***.*

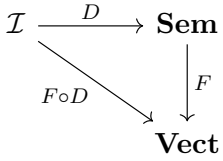
(ii) If the diagram is incoherent (some diagram squares do not commute), then no cone over D exists, and the system abstains.

Proof. (i) When the diagram is coherent, all required compatibility conditions for the limit cone are satisfiable. By the W-completeness assumption, the weighted limit $L = \lim^W D$ exists in **Sem**. The cone maps $\pi_i : L \to D(i)^{W(i)}$ witness its compatibility with every context element. The limit object is semantically valid because it type-checks against every constraint imposed by the diagram.

(ii) When the diagram is incoherent, there exist $i \xrightarrow{f} j \xrightarrow{g} k$ and $i \xrightarrow{h} k$ in $\mathcal{I}$ with $D(g) \circ D(f) \neq D(h)$. Any cone over D would need to satisfy $\pi_k = D(h) \circ \pi_i$ and $\pi_k = D(g) \circ D(f) \circ \pi_i$ simultaneously, which is impossible. Hence no cone (and a fortiori no limit) exists, and the system correctly abstains rather than producing an incoherent blend. □

7.4 The Coherence Diagram

We can visualize the relationship between standard and categorical attention:



Standard attention computes $\lim^W (F \circ D)$ in **Vect**. Categorical attention computes $\lim^W D$ in **Sem**. Since F is not faithful (it loses higher structure), $F(\lim^W D) \neq \lim^W (F \circ D)$ in general. The discrepancy is precisely the hallucination introduced by computing attention in the wrong category.

Proposition 7.6 (Attention Coherence Gap)**.** *Let $D : \mathcal{I} \to \mathbf{Sem}$ be a diagram and $F : \mathbf{Sem} \to \mathbf{Vect}$ the embedding functor. If F preserves the weighted limit, then $F(\lim^W D) \cong \lim^W (F \circ D)$ and standard attention is faithful. In general, F does* not *preserve limits, and the canonical map*

$$F(\overset{W}{\lim} D) \to \overset{W}{\lim}(F \circ D)$$

is neither injective nor surjective. Its cokernel represents hallucinated attention outputs*: vectors in* **Vect** *that appear to be valid attention results but have no preimage in* **Sem***.*

Proof. The functor F preserves limits iff it is *continuous* (preserves all small limits). Since $F : \mathbf{Sem} \to \mathbf{Vect}$ collapses higher homotopical structure (as established in [4]), it does not preserve all limits. In particular, the weighted limit $\lim^W D$ in **Sem** may enforce path coherence conditions that have no counterpart in **Vect**. The canonical comparison map exists by the universal property of limits in **Vect** but is not an isomorphism. Elements in $\lim^W (F \circ D) \setminus \mathrm{im}(F)$ are attention outputs that type-check in **Vect** but not in **Sem**—these are hallucinations. □

7.5 When Limits Don't Exist

Definition 7.7 (Diagram Coherence)**.** A diagram $D : \mathcal{I} \to \mathbf{Sem}$ is *coherent* if for every pair of parallel paths

$$i \xrightarrow{f_1} j_1 \xrightarrow{g_1} k \quad \text{and} \quad i \xrightarrow{f_2} j_2 \xrightarrow{g_2} k$$

in $\mathcal{I}$, we have $D(g_1) \circ D(f_1) = D(g_2) \circ D(f_2)$ (equality in **Sem**, not just in **Vect**). The diagram is *incoherent* otherwise.

Example 7.8 (Incoherent Diagram)**.** Consider a context containing:

- d_1: "The painting was created in 1503." (Source: art history database)
- d_2: "The painting was created in 1517." (Source: auction catalog, erroneous)
- d_3: "The painting depicts a woman." (Source: visual analysis)

The diagram D maps these to semantic objects with morphisms encoding compatibility. The paths $d_1 \to d_3$ and $d_2 \to d_3$ impose contradictory constraints on the creation date. The diagram is incoherent: $\lim^W D$ does not exist in **Sem**.

A standard transformer would blend d_1 and d_2 via attention weights, producing an output vector that "averages" the two dates—yielding a hallucinated compromise (e.g., "created around 1510"). Categorical attention detects the incoherence and abstains, optionally reporting the contradiction.

8 The HoTT-LM Architecture

8.1 Design Principles

The *Homotopy Type-Theoretic Language Model* (HoTT-LM) is a concrete architecture instantiating the five generation principles. Its design follows three meta-principles:

1. **Types before tokens.** The model first determines the semantic type A of the required response, then searches for an inhabitant—reversing the standard generate-then-verify pipeline.
2. **Derivations are output.** The model's output is not a sequence of tokens but a pair (a, δ): a term and its derivation. The token sequence is obtained by rendering a into natural language.
3. **Silence is golden.** Returning $\bot$ is a first-class, valid output—not a fallback or error state.

8.2 Architecture Overview

The HoTT-LM consists of four components:

Definition 8.1 (HoTT-LM)**.** A *HoTT-LM* is a quadruple (G, V, T, U) where:

- $G : \Gamma \times A \to \text{Maybe}(a, \delta)$ is the *generator*: given context Γ and goal type A, it performs proof search and returns an inhabitant with derivation, or $\bot$.
- $V : \Gamma \times a \times A \to \text{Maybe}(\delta)$ is the *verifier*: given context, term, and type, it checks whether the term inhabits the type and returns the derivation if so.
- $T : \Gamma \times a \times p \to \text{Maybe}(a')$ is the *transporter*: given a term a and a path p (context transformation), it computes the transported term a'.
- $U : a \times b \to \text{Maybe}(p)$ is the *unifier*: given two terms, it finds a semantic path between them (if one exists).

The generation algorithm is:

Construction 8.2 (HoTT-LM Generation)**.** Given context Γ and query type A:

1. **Type analysis.** Parse the query to determine the semantic type A.
2. **Context construction.** Build the dependent telescope Γ from the conversation history.
3. **Proof search.** Run $G(\Gamma, A)$:
   (a) Check context: is there $x : A$ in Γ?
   (b) Query knowledge base: is there $s \in \mathcal{K}$ witnessing A?
   (c) Decompose and recurse: apply inference rules.
   (d) If all strategies fail: return $\bot$.
4. **Verification.** Run $V(\Gamma, a, A)$ on the candidate. If verification fails, backtrack and try alternative inhabitants.
5. **Rendering.** Convert the verified term a to natural language, annotated with the derivation δ.

8.3 The Hybrid Architecture

In practice, a pure type-theoretic system may be too rigid for open-domain language generation. We propose a *hybrid* architecture that combines statistical generation with type-theoretic verification:

1. A statistical LLM generates candidate responses $\{t_1, \ldots, t_k\}$.
2. Each candidate is *parsed* into a semantic term a_i.
3. The verifier V checks $\Gamma \vdash a_i : A$ for each candidate.
4. Candidates that fail verification are rejected.
5. Among verified candidates, select the one minimizing $C(a_i)$.
6. If no candidate is verified, return $\bot$.

Theorem 8.3 (Hybrid Soundness)**.** *The hybrid architecture is sound: if it returns a term a, then $\Gamma \vdash a : A$. It may be incomplete (rejecting valid candidates due to parsing failures), but it never hallucinates.*

Proof. The only terms returned are those passing the verifier V, which checks full derivability. Soundness of V (from Theorem 4.2) ensures that verified terms are valid. Incompleteness arises because the parsing step (natural language to semantic term) may fail to recover the term structure, causing valid candidates to be rejected. However, false negatives (rejecting valid responses) are strictly preferable to false positives (accepting hallucinations) in safety-critical applications. □

8.4 Computational Considerations

Proposition 8.4 (Search Complexity)**.** *The proof search $G(\Gamma, A)$ at depth d has worst-case time complexity*

$$O(|\mathcal{R}|^d \cdot |\Gamma|^d \cdot |A|)$$

where $|\mathcal{R}|$ is the number of inference rules and $|\Gamma|$ is the context size.

Proof. At each search step, we choose one of $|\mathcal{R}|$ rules. Each rule may require selecting a variable from Γ (contributing a factor of $|\Gamma|$). The search tree has depth d, giving $|\mathcal{R}|^d \cdot |\Gamma|^d$ branches. Each branch requires a type comparison costing $O(|A|)$. □

This exponential worst case is mitigated by:

- **Focused proof search** [18]: using the goal type to prune irrelevant rules.
- **Statistical guidance**: using an LLM's probability distribution as a heuristic to prioritize search branches.
- **Caching**: memoizing intermediate proof obligations.
- **Bounded depth**: fixing d to a small constant (e.g., $d = 5$) and accepting knowledge-bounded abstention.

9 Haskell Formalization

We present the core Haskell formalization of the type-theoretic generation framework. The full executable module is available as `TypeTheoreticGen.hs`.

Remark 9.1 (Simplified Listings). The code listings in this section are *simplified for exposition*. They use GADT-style syntax and idealized type signatures to convey the mathematical structure clearly. The accompanying executable Haskell module uses standard algebraic data types with `deriving` clauses and includes additional cases (e.g., error handling, normalization) omitted here for brevity. The full source code is the authoritative reference for implementation details.

9.1 Core Types

The semantic types and terms directly mirror the mathematical definitions:

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

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

9.2 Type Inhabitation Search

The proof search algorithm is implemented as a recursive function over the type structure:

Listing 2: Type inhabitation search

```
inhabit :: Context →SemType →Int
        → Maybe (SemTerm, Derivation)
inhabit ctx ty 0 = Nothing
inhabit ctx ty depth =
  contextLookup ctx ty
  <|> kbLookup ctx ty
  <|> decomposeSearch ctx ty (depth - 1)

contextLookup :: Context →SemType
  → Maybe (SemTerm, Derivation)
contextLookup Empty _ = Nothing
contextLookup (Extend ctx (x, t)) goal
  | typeEq t goal =
      Just (Grounded x Axiomatic, AxiomD Axiomatic)
  | otherwise = contextLookup ctx goal
```

9.3 Derivation Verification

Verification traverses the derivation tree bottom-up:

Listing 3: Derivation verification

```
verify :: Context →SemTerm →SemType
       → Derivation →Bool
```

```
verify ctx (Grounded s src) (Prop p) (AxiomD _)
  = s == p
verify ctx (Refl a) (PathType t x y) _
  = termEq a x && termEq a y
verify _ (Hallucinated _) _ _
  = False -- Always invalid
verify _ _ _ _ = False -- Unverified: reject by default
```

9.4 Categorical Attention

Listing 4: Categorical attention as weighted limit

```
data CatAttention = CatAttention
  { attDiagram :: [(SemTerm, SemTerm, PathEvidence)]
  , attWeights :: [(PathEvidence, Float)]
  , attLimit   :: Maybe SemTerm
  }

computeLimit :: [(SemTerm, PathEvidence)]
             → Maybe SemTerm
computeLimit diagram =
  case checkCoherence diagram of
    Coherent lim   → Just lim
    Incoherent _   → Nothing
```

10 Discussion

10.1 Theoretical Significance

The type-theoretic generation framework represents a fundamental shift in how we think about language models. Rather than viewing hallucination as a bug to be mitigated through better training data, RLHF, or retrieval augmentation, we identify it as a *structural inevitability* of the statistical paradigm and provide a *mathematically rigorous* alternative.

The five generation principles are not independent suggestions but form a coherent whole:

- **P1** (generation as inhabitation) provides the *framework*;
- **P2** (certified derivations) provides the *evidence*;
- **P3** (abstention on empty types) provides the *safety guarantee*;
- **P4** (context as fibration) provides the *structure*;
- **P5** (attention as limit) provides the *mechanism*.

Together, they define a generation paradigm that is provably hallucination-free (Theorems 3.2 and 5.2).

10.2 The Soundness–Completeness Tradeoff

Our framework prioritizes *soundness* (never generating invalid claims) over *completeness* (generating all valid claims). This is a deliberate design choice. In safety-critical applications (medical advice, legal reasoning, scientific claims), a system that occasionally says "I don't know" is strictly preferable to one that occasionally hallucinates with high confidence.

The incompleteness has two sources:

1. **Search-depth bounding.** Valid claims requiring deep inference chains may be missed.
2. **Knowledge-base limitations.** Claims that are true but not grounded in $\mathcal{K}$ cannot be generated.

Both are forms of *conservative* incompleteness: the system errs on the side of caution.

10.3 Relationship to Existing Approaches

Retrieval-Augmented Generation (RAG). RAG [13] augments LLMs with retrieved documents, providing a form of grounding. In our framework, RAG corresponds to extending the knowledge base $\mathcal{K}$ at inference time. However, RAG does not provide *certified derivations*: it retrieves relevant passages but does not construct a proof that the generated text follows from those passages. Our framework subsumes RAG as a special case (the GROUND rule) while adding derivational guarantees.

Chain-of-Thought Reasoning. Chain-of-thought prompting [14] encourages LLMs to generate intermediate reasoning steps. This is an informal approximation of our derivation trees. However, chain-of-thought reasoning is generated statistically and may itself contain hallucinated steps. Our derivation trees are *formally verified*, not statistically generated.

Constitutional AI. Constitutional AI [15] uses principles to guide generation. In our framework, constitutional principles correspond to type constraints: a "constitution" is a collection of types that the output must inhabit. However, constitutional AI enforces these constraints heuristically, not formally.

10.4 Limitations

1. **Parsing gap.** Converting natural language queries to semantic types and natural language to semantic terms is itself a hard problem (arguably as hard as the original generation problem). We assume the existence of a parsing oracle; practical implementation requires advances in semantic parsing.
2. **Knowledge base coverage.** The system can only generate claims grounded in $\mathcal{K}$. Incomplete knowledge bases lead to excessive abstention.
3. **Computational cost.** Proof search is exponential in depth. While mitigable (focused search, statistical guidance), the computational overhead compared to pure autoregressive generation is significant.

4. **Creative generation.** Fiction, poetry, and hypothetical reasoning involve generating claims that are *intentionally* ungrounded. The framework would need to be extended with modal types ($\Diamond A$ for "possibly A") to handle these cases.
5. **Gradedness.** Real-world semantic validity is often graded, not binary. Future work should explore *graded type theories* or *fuzzy types* to capture partial validity.

10.5 Open Problems

1. **Efficient proof search for natural language types.** Can we exploit the structure of semantic types to achieve polynomial-time search for common query patterns?
2. **Incremental derivation construction.** Can derivations be built incrementally as the conversation progresses, avoiding re-search at each turn?
3. **Type inference for queries.** Can we automatically infer the semantic type A from a natural language query, rather than requiring explicit specification?
4. **Higher-dimensional derivations.** Can derivations carry higher path structure, enabling "meta-reasoning" about why a particular derivation strategy is preferred?
5. **Cubical implementation.** Can cubical type theory [19] provide a more computational foundation for the path operations in our framework?

11 Conclusion

We have developed a foundational framework for hallucination-free language generation grounded in dependent type theory. The framework replaces the statistical generation paradigm $\arg\max_t P(t \mid \text{ctx})$ with type inhabitation $\arg\min_a C(a)$ subject to $\Gamma \vdash a : A$, ensuring that every generated term is semantically valid by construction.

The five generation principles—type inhabitation, certified derivations, abstention on empty types, context as fibration, and attention as weighted limit—form a mathematically coherent alternative to the current statistical paradigm. We have proved soundness (Theorem 3.2), polynomial-time verification (Theorem 4.2), correct abstention (Theorem 5.2), fibration soundness (Theorem 6.2), and limit coherence (Theorem 7.5).

The framework has been formalized in Haskell, providing executable specifications for all core constructs. While practical implementation faces significant challenges (parsing, knowledge base coverage, computational cost), the theoretical contribution is clear: hallucination is not an irreducible feature of language generation but a consequence of operating in the wrong mathematical space. By replacing **Vect** with **Sem** and probability with derivability, we eliminate hallucination *by construction*.

The path from statistical fluency to type-theoretic validity is the path from engineering to mathematics—from approximation to proof, from plausibility to truth. This paper takes the first steps along that path.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Naples, 1984.

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

[8] W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[9] M. H. Sørensen and P. Urzyczyn. *Lectures on the Curry–Howard Isomorphism*. Elsevier, 2006.

[10] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65(3):154–170, 1958.

[11] B. Coecke, M. Sadrzadeh, and S. Clark. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36(1–4):345–384, 2010.

[12] M. Baroni, R. Bernardi, and R. Zamparelli. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9(6):5–110, 2014.

[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 *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9459–9474, 2020.

[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[15] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[16] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, pages 1321–1330, 2017.

[17] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2–3):95–120, 1988.

[18] C. Liang and D. Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.

[19] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg. Cubical type theory: A constructive interpretation of the univalence axiom. *Journal of Automated Reasoning*, 60(2):195–230, 2018.

[20] G. M. Kelly. *Basic Concepts of Enriched Category Theory*. London Mathematical Society Lecture Note Series, vol. 64. Cambridge University Press, 1982.

[21] E. Riehl. *Categorical Homotopy Theory*. Cambridge University Press, 2014.

[22] J. Lurie. *Higher Topos Theory*. Annals of Mathematics Studies, vol. 170. Princeton University Press, 2009.

[23] U. Norell. Dependently typed programming in Agda. In *Advanced Functional Programming (AFP)*, LNCS 5832, pages 230–266. Springer, 2009.

[24] E. Brady. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming*, 23(5):552–593, 2013.

[25] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.

[26] H. B. Curry, R. Feys, and W. Craig. *Combinatory Logic*, volume I. North-Holland, Amsterdam, 1958.

[27] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. Ph.D. thesis, Université Paris VII, 1972.

[28] S. Awodey and M. A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146(1):45–55, 2012.