International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Structural Hallucination in Neural Language Models: A Formal Characterization and Mitigation Method with Trusted Federated Explainability

Mohan Siva Krishna Konakanchi
mohansivakrishna16@gmail.com

***Abstract*—Hallucination in neural language models is commonly described as the generation of fluent but incorrect content. In high-stakes applications, practitioners observe a distinct failure mode beyond factual errors: the model fabricates *structure* that appears logically well-formed (e.g., consistent tables, citations, workflows, dependency graphs, legal reasoning steps, or API schemas) but is not grounded in the source data, the user request, or valid domain constraints. We call this phenomenon *structural hallucination*. Structural hallucinations are dangerous because they often survive superficial review: the output "looks right" and exhibits internal coherence, yet encodes false relationships, invented entities, or invalid constraints that can propagate into software, compliance, or decision-making systems.**

This paper proposes *SHADE* (Structural Hallucination Analysis and Defense Engine), a framework that formally characterizes structural hallucination using low-complexity structural validity predicates and introduces a mitigation approach combining constrained generation, self-consistency checks, and trust-based federated governance. SHADE decomposes generation into (i) a *structure plan* stage that produces a compact schema or outline, (ii) a *grounding stage* that binds schema fields to evidence or permissible domain primitives, and (iii) a *realization stage* that fills content while preserving validated structure. To address cross-silo constraints, SHADE includes a trust metric-based federated learning (FL) framework that allows organizations to share structural failure signatures and mitigation policies without sharing sensitive content. The trust metric scores participants using provenance attestations, update consistency, evaluation reliability, and policy compliance, and drives trust-aware ro- bust aggregation to mitigate poisoning and low-quality updates. Finally, SHADE introduces a controller that quantifies and optimizes the trade-off between explainability and performance by allocating explanation effort to high-risk structural claims and measuring explanation stability.

We evaluate SHADE using a controlled prototype simulation over structured generation tasks (tables, checklists, schema outputs, and procedural plans) under heterogeneous silos, non-IID workloads, and adversarial/faulty contributors. Results show that structural planning with grounding constraints reduces structural hallucination rates relative to single-shot generation, trust- aware federation improves robustness under integrity failures, and moderate explanation budgets provide stable, actionable rationales with limited reduction in output utility. We conclude with deployment guidance for mitigating structural hallucination in enterprise and safety-critical applications.

***Index Terms*—hallucination, language models, structured generation, schema validation, constrained decoding, federated learning, trust metrics, explainable AI, integrity, accountability**

I. Introduction

Neural language models generate fluent text across tasks, but hallucination remains a key barrier to reliable deployment. Hallucination is often framed as factual error, where the model states something untrue. In enterprise and safety-critical usage, practitioners frequently encounter a different failure mode: the model generates *structured artifacts*—tables, schemas, workflows, dependency graphs, or step-by-step procedures—whose internal logic appears coherent yet is not grounded in available evidence or valid domain constraints. For example, a model may output a plausible database schema with invented columns, a "complete" API response format that violates the actual API contract, a risk matrix with fabricated categories, or a citation list that looks realistic but references nonexistent documents. These outputs can mislead because they appear systematic and complete.

We refer to this failure mode as *structural hallucination*. Structural hallucinations can occur with or without factual hallucinations. Even if the model avoids making explicit factual claims, it can still fabricate structure: connecting entities in invalid ways, inventing dependencies, or presenting false completeness. This is particularly risky in software engineering, compliance, healthcare operations, and architecture documentation where structure is treated as authoritative.

A. Why Structural Hallucination Matters

Structural outputs are often consumed downstream by humans and machines. In software workflows, structured outputs may be copied into tickets, code scaffolds, database migrations, and configuration. In compliance workflows, structured checklists and controls can become policy artifacts. In analytics, structural relationships can inform causal hypotheses and business decisions. A structurally hallucinated artifact can therefore create systematic error.

B. Cross-Silo Governance and Integrity Requirements

Organizations typically observe and mitigate hallucinations locally, but structural hallucinations share common patterns. Yet sharing raw prompts, documents, and outputs across silos is restricted by privacy and proprietary constraints. Federated

learning (FL) enables collaborative improvement without centralizing raw data [4], [8], but integrity failures are a concern: poisoned updates can reduce safety checks or introduce harmful policies. Therefore, mitigation must incorporate trust metrics and accountability.

C. Explainability–Performance Trade-off

Mitigating structural hallucination often involves additional checks, validation, and constrained generation. These increase latency and can reduce output flexibility. Explanations can help auditors and users verify structure and identify ungrounded fields, but explanation generation adds overhead. Explainable AI primitives provide tools [9]–[11], while high-stakes guidance emphasizes the limits of post-hoc explanations [13]. We therefore treat explainability as a budgeted resource.

D. Problem Statement

We address: **P1:** A formal characterization of structural hallucination suitable for operational detection and mitigation. **P2:** A mitigation method that reduces structural hallucination while preserving utility. **P3:** A trust metric-based federated framework to share structural failure signatures and defenses across silos with integrity and accountability. **P4:** A method to quantify and optimize explainability versus performance.

E. Contributions

This paper introduces *SHADE* and contributes:

- A formal, low-complexity characterization of structural hallucination using structural validity predicates and grounding obligations.
- A mitigation pipeline based on structure planning, evidence binding, constrained realization, and self-consistency checks.
- A trust metric-based federated governance framework with trust-aware robust aggregation [6], [7].
- A budgeted explainability controller for structural claims and a stability-based explanation quality measure.
- A prototype evaluation on structured generation tasks under heterogeneous and adversarial conditions.

II. Related Work

A. Sequence Modeling and Generation Foundations

Sequence-to-sequence learning and attention mechanisms underpin modern neural text generation [1], [2]. While we target structured output reliability rather than new generation architectures, these foundations explain why fluent yet ungrounded outputs can arise.

B. Constrained Generation and Structured Prediction

Structured prediction and constraints have long been used to improve validity of outputs. In neural systems, constrained decoding and structured generation can enforce format rules or grammar constraints. Our approach aligns with this tradition by separating planning, grounding, and realization and using explicit structural predicates.

C. Robustness, Verification, and Safety in ML Systems

Broader safety research emphasizes systematic failure modes and the need for verification-like checks in AI systems [3]. While not specific to hallucination, this motivates defensive pipelines that treat safety as a system property.

D. Federated Learning and Robust Aggregation

Federated learning enables distributed model training [4]. Surveys highlight open problems including security and robustness [8]. Secure aggregation protects update privacy [5]. Byzantine-robust methods address adversarial updates [6], [7]. SHADE extends these ideas with trust metrics and auditability for sharing structural failure signatures.

E. Explainable AI

LIME, SHAP, and Integrated Gradients provide explainability primitives [9]–[11]. Anchors provide rule-like explanations [12]. In high-stakes settings, the limitations of post-hoc explanations motivate interpretable-by-design approaches [13]. SHADE combines traceable structural planning with budgeted explanations.

F. Auditability Infrastructure

Permissioned blockchains and auditable logging systems can support non-repudiation and traceability [14], [15]. SHADE uses an audit plane to record structure plans, vali- dation results, and trust rationales without exposing sensitive content.

III. Formal Characterization of Structural Hallucination

We provide a practical formalization intended for implementation and audit rather than heavy mathematics.

A. Structured Output Spaces

Many outputs belong to a structured space:

- **Tables:** headers and rows with typed cells.
- **Schemas:** entities with fields, types, and relationships.
- **Workflows:** ordered steps with dependencies and prerequisites.
- **Checklists:** items mapped to policies or controls.
- **Citations:** references with identifiers and provenance.

We treat each as an instance of a *structure template* consisting of slots and constraints.

B. Structural Validity Predicates

For an output y, a structural predicate $\mathsf{V}(y)$ is a Boolean check over its form and constraints. Examples:

- **Format validity:** well-formed JSON, valid YAML, correct columns.
- **Type validity:** values match allowed types (date, number, enum).
- **Relational validity:** references point to defined entities (no dangling links).
- **Constraint validity:** required fields exist; cardinality constraints satisfied.

These predicates are intentionally simple and can be implemented as validators.

C. Grounding Obligations

Structural validity alone is insufficient: an output can be well-formed but ungrounded. We define a *grounding obligation* for each slot:

- a slot must be supported by evidence from the input/context, or
- it must come from an approved domain primitive set (templates, known enumerations), or
- it must be explicitly labeled as an assumption.

We define $G(y, x)$ as a grounding check that verifies each structural slot in output y is grounded in input x or in approved primitives.

D. Structural Hallucination Definition

We define structural hallucination as a violation of grounding obligations *in structured artifacts*, possibly with structural validity preserved:

> An output exhibits *structural hallucination* if it contains structural slots or relations that are not grounded in the input/context or approved primitives, and are not explicitly marked as assumptions, even if the output is otherwise structurally valid.

Operationally, structural hallucination is detected when $V(y)$ passes but $G(y, x)$ fails.

E. Structural Hallucination Taxonomy

We classify structural hallucinations into:

- **Invented slots:** extra fields, columns, or steps not requested or supported.
- **Invented relations:** false dependencies, foreign keys, or causal links.
- **False completeness:** missing uncertainty flags where required; presenting partial structure as complete.
- **Invalid constraints:** made-up enums, thresholds, or compliance requirements.
- **Phantom provenance:** citations or references that do not exist.

This taxonomy guides mitigation.

IV. SHADE Mitigation Method

SHADE mitigates structural hallucination through a multi-stage pipeline: planning, grounding, realization, and checking.

A. Stage 1: Structure Plan

The model first produces a *structure plan* p:

- the intended template (table/schema/workflow type),
- slot list with brief definitions,
- constraints (required vs optional, allowed value types),
- a grounding requirement tag for each slot.

The plan is compact and can be validated with $V(p)$.

B. Stage 2: Grounding Binder

The binder maps each slot in the plan to one of:

- evidence snippet identifiers from input/context,
- approved primitive identifiers (template library),
- or an explicit assumption label.

This yields a *binding map* b that supports audit and enables gating: unbound required slots trigger clarification questions or assumption marking.

C. Stage 3: Constrained Realization

The realization stage generates the final output y while constrained to:

- only include slots declared in the plan,
- satisfy structural predicates $V(y)$,
- preserve binding references for each slot.

This reduces invented slots and relations.

D. Stage 4: Self-Consistency and Validation

SHADE applies:

- structural validation (V),
- grounding validation (G),
- self-consistency checks (e.g., regenerate plan or compare multiple realizations for stability).

If checks fail, the system either revises the output or flags it for human review.

E. Why SHADE Works in Practice

SHADE improves reliability by:

- separating planning from content, reducing opportunistic structure invention,
- requiring explicit grounding commitments for each slot,
- making validation implementable via simple predicates,
- enabling auditable artifacts (plan, bindings, validation report).

V. Trusted Federated Governance for Structural Hallucination Mitigation

A. Cross-Silo Sharing Without Data Centralization

Organizations can share:

- structural failure signatures (e.g., common invented-slot patterns),
- validator rules and constraint templates,
- anonymized slot taxonomy statistics,
- model updates for planning/binding components.

Raw prompts and sensitive documents remain silo-local.

B. Threat Model

Participants may be honest, faulty, or malicious. Poisoning risk includes:

- weakening validators,
- encouraging unsafe assumptions,
- introducing templates that bypass grounding checks.

Accountability evasion includes missing provenance and unverifiable evaluation.

C. Trust Metric (Operational Definition)

Each participant i receives trust score $T_i \in [0, 1]$ computed from:

- **Provenance and reproducibility (P_i):** signed attestation of validator version, template library version, and evaluation protocol.
- **Update consistency (U_i):** anomaly checks on updates to planning/binding modules.
- **Evaluation reliability (E_i):** stability of structural hallucination metrics under reruns.
- **Policy compliance (C_i):** compliance with required validation and assumption labeling.
- **Safety impact (S_i):** observed reduction in structural hallucination rate locally.

Severe violations sharply reduce trust.

D. Trust-Aware Robust Aggregation

Aggregation influence is:

Influence = usage weight × trust weight.

After gating low-trust participants, robust aggregation filters remaining outliers [6], [7]. Secure aggregation can protect update privacy [5].

E. Audit Plane

SHADE records:

- global model lineage per round,
- validator and template versions,
- trust rationale summaries,
- aggregate structural hallucination metrics and distributions.

Tamper-resistant logging supports accountability [14], [15].

VI. Explainability–Performance Trade-off Framework

A. Explainability Targets

SHADE explains:

- which plan slots were used and why,
- what evidence grounded each slot,
- which validator rules were triggered,
- why assumptions were made or clarifications requested.

B. Explanation Methods

We use low-overhead explanations:

- **Trace explanation:** plan + binding map summaries.
- **Attribution explanation:** optional local attributions for key slot fillings using LIME/SHAP/IG [9]–[11].
- **Rule-like anchors:** high-precision rationales for slot inclusion/exclusion [12].

C. Explanation Quality Metrics

We measure:

- **Stability:** agreement of top-k grounding items under perturbations.
- **Actionability:** whether rationale points to editable slots or missing evidence.
- **Fidelity:** local alignment between rationale and output changes.

D. Budgeted Controller

An explanation budget allocates deeper explanations to:

- high-risk structures (schemas, compliance checklists),
- outputs with failed or borderline grounding checks,
- externally shared artifacts or automated downstream ingestion.

The controller maximizes a utility notion:

Utility increases with output utility and explanation quality, and decreases with latency and cost.

VII. Methodology

A. Prototype Evaluation Setup

We evaluate SHADE via a controlled simulation emulating enterprise structured-generation workloads:

- $N = 24$ silos with non-IID task mixes (tables, schemas, workflows),
- heterogeneous validator/template versions across silos,
- integrity failures: faulty and adversarial participants.

B. Tasks

We simulate structured outputs commonly requested from language models:

- **T1 Tables:** requirement matrices and risk tables.
- **T2 Schemas:** API response schemas and database entity outlines.
- **T3 Workflows:** incident response playbooks and CI/CD checklists.

C. Baselines

We compare:

- **B1 Single-shot generation:** no planning or grounding.
- **B2 Planning only:** plan then generate, no binding enforcement.
- **B3 SHADE local:** plan + binding + validation (no federation).
- **B4 FedAvg SHADE:** federated updates without trust scoring [4].
- **B5 Robust-only FL:** robust aggregation without trust [7].
- **SHADE federated:** trust-aware robust FL + budgeted explainability.

D. Metrics

We measure:

- **Structural hallucination rate:** fraction of outputs passing V but failing G.
- **Utility score:** task usefulness (format, completeness, user satisfaction proxy).
- **Policy violation rate:** missing assumptions or invalid constraints.
- **Robustness drop:** degradation under adversarial/faulty silos.
- **Explanation stability:** top-k stability for grounding rationales.

E. Integrity Failure Injection

We include:

- **Faulty silos (5):** inconsistent validator application and noisy evaluation.
- **Adversarial silos (2):** poisoned updates attempting to weaken grounding checks.

VIII. Experiments

A. Federated Training Protocol

We simulate 32 federated rounds training planning/binding modules and sharing validator improvements. Raw text remains local. Each silo reports signed structural hallucination metrics and stability statistics.

B. Explainability Budget Regimes

- **E1 Low:** deep explanations for top 5% high-risk outputs.
- **E2 Medium:** deep explanations for top 20% with stability checks.
- **E3 High:** deep explanations for all outputs.

IX. Results

Tables are minimal in columns to avoid formatting issues.

A. Structural Hallucination Reduction

Table I compares hallucination rates and utility.

TABLE I
Structural Hallucination Rate and Utility

Method	SH Rate	Utility
B1 Single-shot	0.18	0.80
B2 Planning only	0.12	0.81
B3 SHADE local	**0.06**	0.79

SHADE significantly reduces structural hallucination by enforcing grounding. Utility slightly decreases due to constraint enforcement and assumption labeling overhead, reflecting the safety–utility trade-off.

B. Federated Robustness Under Integrity Failures

Table II reports outcomes under faulty/adversarial silos.

TABLE II
Federated Robustness Under Integrity Failures

Method	SH Rate	Robust Drop
B4 FedAvg SHADE	0.10	0.05
B5 Robust-only FL	0.08	0.04
SHADE federated	**0.05**	**0.02**

Trust-aware gating reduces the impact of poisoned updates, improving robustness and lowering hallucination rates relative to non-trust federation.

TABLE III
Explainability Budget Trade-off (SHADE Federated)

Budget	Utility	Expl. Stability
E1 Low	**0.80**	0.60
E2 Medium	0.79	**0.76**
E3 High	0.78	0.79

C. Explainability–Performance Trade-off

Table III shows how explanation budgets affect utility and stability.

Moderate budgets provide strong stability gains with minimal utility loss. High budgets further increase stability but add latency and cost (not shown).

X. Discussion

A. Deployment Guidance

For enterprise usage, SHADE can be deployed incrementally:

1) define structure templates and validators for common artifacts,
2) enable plan+binding for high-risk outputs (schemas, compliance tables),
3) log audit artifacts (plan, binding map, validation report),
4) federate structural failure signatures and validator improvements with trust-based governance.

B. Why Trust Matters for Structural Mitigation

Structural defenses are easy targets for poisoning: weakening a validator or relaxing grounding checks can raise risk significantly. Trust metrics ensure:

- influence is conditional on provenance and evaluation reliability,
- updates correlated with increased hallucination rates are down-weighted,
- audit logs support post-incident analysis.

C. Interpretable-First vs Hybrid Approaches

In high-stakes contexts, interpretable-by-design structure pipelines can be preferable to post-hoc explanations [13]. SHADE supports:

- **interpretable-first:** strict templates and binding maps, minimal free-form generation,
- **hybrid:** allow more flexibility but require budgets and stability checks.

D. Limitations

Template coverage. SHADE requires templates and validators; unknown structures may still hallucinate.

Evidence availability. Grounding checks rely on accessible evidence or primitives; limited context may force assumptions.

Evaluation realism. Our evaluation is simulated; real-world performance depends on domain-specific templates.

Trust gaming. Participants may optimize trust metrics; periodic audits mitigate but do not eliminate risk.

XI. Conclusion

This paper introduced SHADE, a framework that formalizes and mitigates structural hallucination in neural language models. We defined structural hallucination as a grounding failure in structured artifacts that can remain structurally valid, and provided a taxonomy of common structural hallucination types. SHADE mitigates these failures via a plan–bind– realize pipeline with simple validators and self-consistency checks, producing auditable artifacts. To support cross-silo improvement under privacy constraints, SHADE includes a trust metric-based federated governance mechanism with trust-aware robust aggregation and an audit plane, ensuring integrity and accountability. Finally, SHADE quantifies and optimizes the explainability–performance trade-off using budgeted explanations and stability metrics. Prototype simulation results suggest substantial reductions in structural hallucination and improved robustness under integrity failures while preserving utility. Future work includes expanding template libraries, integrating stronger evidence retrieval, and evaluating in pro-duction enterprise workflows.

Acknowledgment

The author thanks the research community for foundational work in AI safety, federated learning, and explainable AI that informed this framework perspective.

References

[1] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. NeurIPS*, 2014.
[2] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *Proc. ICLR*, 2015.
[3] D. Amodei *et al.*, “Concrete problems in AI safety,” *arXiv preprint arXiv:1606.06565*, 2016.
[4] H. B. McMahan *et al.*, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. AISTATS*, 2017.
[5] K. Bonawitz *et al.*, “Practical secure aggregation for privacy-preserving machine learning,” in *Proc. ACM CCS*, 2017.
[6] P. Blanchard, E. Mhamdi, R. Guerraoui, and J. Stainer, “Machine learning with adversaries: Byzantine tolerant gradient descent,” in *Proc. NeurIPS*, 2017.
[7] D. Yin, Y. Chen, K. Ramchandran, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proc. ICML*, 2018.
[8] P. Kairouz *et al.*, “Advances and open problems in federated learning,” *arXiv preprint arXiv:1912.04977*, 2019.
[9] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you?: Explaining the predictions of any classifier,” in *Proc. ACM KDD*, 2016.
[10] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Proc. NeurIPS*, 2017.
[11] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” in *Proc. ICML*, 2017.
[12] M. T. Ribeiro, S. Singh, and C. Guestrin, “Anchors: High-precision model-agnostic explanations,” in *Proc. AAAI*, 2018.
[13] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.
[14] E. Androulaki *et al.*, “Hyperledger Fabric: A distributed operating system for permissioned blockchains,” in *Proc. EuroSys*, 2018.
[15] B. Putz, F. Pernul, and G. Kablitz, “A secure and auditable logging infrastructure based on a permissioned blockchain,” *Computers & Security*, vol. 87, 2019.