Verified Fallacy Detection for Large Language Models: A Machine-Checked Framework from the Architecture of Reasoning

Horsocrates

2026

Abstract

Large Language Models (LLMs) generate plausible but often flawed reasoning. Current approaches to detecting such flaws rely on pattern-matching against memorized fallacy lists or heuristic classifiers. This article presents an alternative: a *machine-verified* fallacy detection framework derived from the Architecture of Reasoning—a philosophical theory that classifies all reasoning errors as violations of a discoverable structure.

We formalize the complete taxonomy of 156 fallacies in the Coq proof assistant, proving key structural theorems with zero unverified assumptions. We then implement an LLM response verification system that: (1) classifies reasoning errors by architectural location, (2) maps LLM hallucinations to specific violation types, (3) generates targeted correction prompts for self-reflection loops, and (4) provides a safety layer blocking harmful reasoning patterns.

The framework is extractable to OCaml for production deployment. We demonstrate its application to chain-of-thought validation, showing how the six-domain structure (D1-D6) provides a principled template for LLM reasoning. All code is publicly available; all theorems are machine-checked.

This work bridges philosophy of logic and AI safety, transforming fallacy detection from a classification problem into a structural analysis problem with formally verified guarantees.

Keywords: fallacy detection, large language models, formal verification, Coq, AI safety, chain-of-thought, reasoning architecture

Contents

1 Introduction

1.1 The Problem: LLM Reasoning Errors

Large Language Models produce fluent, confident text that often contains logical errors. These errors include:

- **Factual hallucinations**: Asserting facts that do not exist
- **Logical hallucinations**: Drawing conclusions not supported by premises
- **Self-referential loops**: Generating self-contradictory statements
- **Confident wrong answers**: Failing to recognize limits of knowledge
- **Sycophantic responses**: Prioritizing user approval over truth

Current detection approaches are unsatisfying. Pattern-matching against fallacy lists requires memorization and fails on novel errors. Heuristic classifiers lack interpretability and formal guarantees. What is missing is a *structural* account of reasoning that explains *why* errors occur and *where* they are located.

1.2 The Solution: Architecture of Reasoning

This article extends the Architecture of Reasoning—a philosophical framework developed in a six-article series—to AI applications. The key insight is that all reasoning errors are *violations of a discoverable structure*. This structure has two dimensions:

- **Horizontal (Sequence)**: Reasoning passes through six domains (D1-D6)
- **Vertical (Hierarchy)**: Operations apply to lower levels only (L1-L3)

Violations of horizontal structure produce **fallacies** (156 types). Violations of vertical structure produce **paradoxes** (7 paradigmatic examples). By formalizing this structure in Coq, we obtain a verified detector that:

1. *Diagnoses* where reasoning failed (which domain, which level)
2. *Explains* why it failed (which principle was violated)
3. *Generates* correction prompts for self-reflection
4. *Guarantees* correctness via machine-checked proofs

1.3 Contributions

1. **Complete Formalization**: 156 fallacies across 5 types, 7 paradoxes across 4 types, formalized in Coq with 107 proven theorems and 0 unverified assumptions.

2. **LLM Hallucination Taxonomy**: A mapping from hallucination types to architectural violations, enabling principled classification.

3. **Verified Detector**: A fallacy detection system with proven properties, extractable to OCaml for production use.

4. **Chain-of-Thought Template**: A six-domain reasoning template (D1→D6) for structured LLM prompting.

5. **Safety Layer**: Proven detection of ad hominem, confirmation bias, and self-referential patterns.

1.4 Article Structure

Section 2 recaps the E/R/R framework and fallacy taxonomy. Section 3 presents the formal model in Coq. Section 4 develops AI applications. Section 5 demonstrates extraction and provides examples. Section 6 concludes.

2 The Architecture of Reasoning

2.1 E/R/R Framework

Every functional system has three components:

Component	Description	Level
Elements	What the system contains	L1
Roles	How elements relate	L1
Rules	Principles governing the system	L2

A *Constitution* is the integration of E/R/R that defines a system. Reasoning without a valid Constitution (Type 1 violation) is not reasoning at all—it is manipulation.

2.2 Six Domains of Reasoning

Valid reasoning traverses six domains in sequence:

Domain	Function	Fallacies	%
D1: Recognition	Fixation of what is present	26	25%
D2: Clarification	Understanding of what was recognized	13	12%
D3: Framework Selection	Determination of evaluative criteria	16	15%
D4: Comparison	Application of framework to material	8	8%
D5: Inference	Extraction of what follows	20	19%
D6: Reflection	Recognition of limits and revision	22	21%
Total Type 2		**105**	**100%**

D1 and D6 are most vulnerable. D1 is foundational—errors propagate through all subsequent domains. D6 requires self-critical assessment, which humans (and LLMs) find difficult. D4 is most constrained, admitting fewer failure modes.

2.3 Complete Fallacy Taxonomy

The Architecture identifies five types of violation:

Type	Description	Count	Note
Type 1	Violations of Conditions	36	Reasoning fails to begin
Type 2	Domain Violations	105	Reasoning fails within domain
Type 3	Violations of Sequence	3	Wrong direction
Type 4	Syndromes	6	Systemic distortion
Type 5	Context-Dependent Methods	6	Valid in some contexts
Core fallacies (Types 1-4)		**150**	Always errors
Complete taxonomy		**156**	Including context-dependent

Type 2 comprises 70% of core fallacies (105/150), warranting dedicated treatment by domain.

2.4 Three Hierarchical Levels

The vertical dimension comprises three levels (finite, unlike Tarski's infinite hierarchy):

Level	Contents	Examples
L1: Elements	Objects, statements, premises	"Snow is white"
L2: Operations	Truth-evaluation, set membership	"Is it true that..."
L3: Meta-operations	Operations on operations	Logic about logic

The Hierarchy Principle: Operations at level N apply only to entities at level $N-1$. Self-application is structurally incoherent. This principle blocks all self-referential paradoxes.

3 Formal Model in Coq

3.1 Domain Structure

Listing 1: Domain definition

```
Inductive Domain : Type :=
  | D1_Recognition : Domain
  | D2_Clarification : Domain
  | D3_FrameworkSelection : Domain
  | D4_Comparison : Domain
  | D5_Inference : Domain
  | D6_Reflection : Domain.

Definition domain_index (d : Domain) : nat :=
  match d with
  | D1_Recognition ⇒ 1 | D2_Clarification ⇒ 2
  | D3_FrameworkSelection ⇒ 3 | D4_Comparison ⇒ 4
  | D5_Inference ⇒ 5 | D6_Reflection ⇒ 6
  end.

Definition valid_sequence (d1 d2 : Domain) : bool :=
  domain_index d1 <=? domain_index d2.
```

3.2 Fallacy Types

Listing 2: Type 1 violations (excerpt)

```
(* Subtype 1.A: Defective Questions *)
Inductive T1A_DefectiveQuestion : Type :=
  | T1A_ComplexQuestion     (* Loaded Question *)
  | T1A_Taboo               (* Dogmatism *)
  | T1A_VenueFallacy.       (* Wrong place/time *)

(* Subtype 1.B: Manipulations (33 total) *)
Inductive T1B_Manipulation : Type :=
  | T1B_AdBaculum           (* Force *)
  | T1B_AppealToPity        (* Pity *)
  | T1B_ScareTactics        (* Emotion *)
  | T1B_Bribery             (* Benefit *)
  | T1B_BigLie              (* Disinformation *)
  | T1B_MalaFides           (* Bad Faith *)
  (* ... 27 more ... *).
```

3.3 Verification Result

Listing 3: Verification result type

```
Inductive VerificationResult : Type :=
  | VR_Valid : VerificationResult
  | VR_Type1_NoConstitution : VerificationResult
  | VR_Type2_DomainViolation : Domain → FailureMode →
    VerificationResult
  | VR_Type3_SequenceViolation : Domain → Domain →
    VerificationResult
  | VR_Type4_Syndrome : string → VerificationResult
  | VR_Paradox_LevelConfusion : LevelConfusion → VerificationResult
  | VR_Incomplete : Domain → VerificationResult.
```

3.4 Key Theorems

Theorem 3.1 (Complete Taxonomy). *The total count of formalized fallacies is 156.*

```
Theorem complete_taxonomy_156 : grand_total = 156.
Proof. reflexivity. Qed.
```

Theorem 3.2 (Self-Application Invalid). *No operation can validly apply to itself.*

```
Theorem self_application_invalid : forall l,
  ¬ valid_application l l.
Proof.
  unfold valid_application, level_lt.
  intros l H. lia.
Qed.
```

Theorem 3.3 (D1/D6 Most Vulnerable). *Recognition and Reflection domains have the most fallacies.*

```
Theorem D1_D6_most_fallacies :
  length all_D1 >= length all_D2 ∧
  length all_D1 >= length all_D3 ∧
  length all_D1 >= length all_D4 ∧
  length all_D6 >= length all_D2 ∧
  length all_D6 >= length all_D4.
Proof. repeat split; simpl; lia. Qed.
```

4 AI Applications

4.1 LLM Hallucination Classification

We map LLM hallucination types to architectural violations:

Hallucination Type	Architecture Violation	Location
Factual	Object Deformation	D1
Logical	Logical Gap (Non Sequitur)	D5
Self-Referential	Level Confusion	Paradox
Overconfident	Illusion of Completion	D6
Sycophantic	No Constitution	Type 1

Listing 4: Hallucination classification

```
Inductive HallucinationType : Type :=
  | Halluc_Factual          (* D1 violation *)
  | Halluc_Logical          (* D5 violation *)
  | Halluc_SelfReferential  (* Paradox *)
  | Halluc_Overconfident    (* D6 violation *)
  | Halluc_Sycophantic.     (* Type 1 violation *)

Definition hallucination_to_violation (h : HallucinationType)
  : VerificationResult :=
  match h with
  | Halluc_Factual ⇒
      VR_Type2_DomainViolation D1_Recognition FM_ObjectDeformation
  | Halluc_Logical ⇒
      VR_Type2_DomainViolation D5_Inference FM_LogicalGap
  | Halluc_SelfReferential ⇒
      VR_Paradox_LevelConfusion LC_SelfApplication
  | Halluc_Overconfident ⇒
      VR_Type2_DomainViolation D6_Reflection FM_IllusionOfCompletion
  | Halluc_Sycophantic ⇒
      VR_Type1_NoConstitution
  end.

Theorem hallucinations_are_violations : forall h,
  hallucination_to_violation h ≠ VR_Valid.
Proof. intros h. destruct h; discriminate. Qed.
```

4.2 Combined Hallucination Detector

The most common LLM hallucination pattern combines D5 (non-sequitur) and D6 (no reflection) violations:

Listing 5: Combined hallucination detector

```
Record HallucinationSignals := mkHallucinationSignals {
  hs_premises_support_conclusion : bool;   (* D5 *)
  hs_acknowledges_uncertainty : bool;      (* D6 *)
  hs_cites_sources : bool;                 (* D6 *)
  hs_consistent_with_context : bool;       (* D1 *)
  hs_self_contradicts : bool               (* Paradox *)
}.

Inductive HallucinationSeverity : Type :=
  | HS_None | HS_Mild | HS_Moderate | HS_Severe.

Definition detect_hallucination (hs : HallucinationSignals)
  : HallucinationSeverity :=
  let d5_fail := negb (hs_premises_support_conclusion hs) in
  let d6_fail := negb (hs_acknowledges_uncertainty hs)
                 && negb (hs_cites_sources hs) in
  if hs_self_contradicts hs then HS_Severe
  else if d5_fail && d6_fail then HS_Moderate
  else if d5_fail || d6_fail then HS_Mild
  else HS_None.
```

4.3 Chain-of-Thought Validation

The six-domain structure provides a template for structured reasoning:

Listing 6: CoT domain questions

```
Definition cot_domain_question (d : Domain) : string :=
  match d with
  | D1_Recognition ⇒
      "What␣exactly␣is␣the␣question/claim␣being␣examined?"
  | D2_Clarification ⇒
      "What␣do␣the␣key␣terms␣mean?␣Any␣ambiguity?"
  | D3_FrameworkSelection ⇒
      "What␣criteria/framework␣will␣evaluate␣this?"
  | D4_Comparison ⇒
      "How␣does␣this␣compare␣to␣relevant␣known␣cases?"
  | D5_Inference ⇒
      "What␣follows␣logically␣from␣the␣evidence?"
  | D6_Reflection ⇒
      "What␣are␣the␣limits?␣What␣could␣be␣wrong?"
  end.
```

Prompt Template:

```
Reason step-by-step through domains D1->D6:

D1 (Recognition): What exactly is the question/claim?
D2 (Clarification): What do the key terms mean?
D3 (Framework): What criteria will you use?
D4 (Comparison): How does this compare to known cases?
D5 (Inference): What follows from the evidence?
D6 (Reflection): What are the limits of this conclusion?

Ensure no domain violation at each step.
```

4.4 Self-Reflection Loop

When a violation is detected, generate a targeted correction prompt:

Listing 7: Fix prompt generation

```
Definition generate_fix_prompt (vr : VerificationResult) : string :=
  match vr with
  | VR_Valid ⇒ ""
  | VR_Type1_NoConstitution ⇒
      "Error:␣Missing␣E/R/R␣structure.␣Identify:
␣␣␣␣␣␣␣␣(1)␣Elements,␣(2)␣Roles,␣(3)␣Rules."
  | VR_Type2_DomainViolation d fm ⇒
      "Fallacy␣in␣" ++ domain_name d ++ ":␣"
      ++ failure_mode_name fm ++ ".␣Correct␣and␣retry."
  | VR_Paradox_LevelConfusion lc ⇒
      "Paradox:␣" ++ level_confusion_name lc ++
      ".␣Avoid␣self-referential␣statements."
  | VR_Incomplete d ⇒
      "Missing␣" ++ domain_name d ++ ".␣Address␣before␣concluding."
  | _ ⇒ "Reasoning␣error␣detected.␣Please␣review."
  end.
```

Self-Reflection Algorithm:

```
response = llm(prompt)
violation = coq_detector(response)
while violation != VR_Valid and iterations < max:
    fix_prompt = generate_fix_prompt(violation)
    response = llm(prompt + fix_prompt)
    violation = coq_detector(response)
return response
```

4.5 Safety Layer

The detector blocks harmful reasoning patterns:

Listing 8: Safety check

```
Inductive SafetyResult : Type :=
  | SR_Safe
  | SR_ConfirmationBias
  | SR_AdHominem
  | SR_SelfReferentialLoop
  | SR_UnfalsifiableClaim.

Definition safety_check (sig : ResponseSignals) : SafetyResult :=
  if sig_self_reference sig then SR_SelfReferentialLoop
  else if sig_attacks_person sig && negb (sig_addresses_argument sig
      )
        then SR_AdHominem
  else if negb (sig_considers_counter sig)
        then SR_ConfirmationBias
  else SR_Safe.

Theorem safety_blocks_ad_hominem : forall sig,
  sig_attacks_person sig = true →
  sig_addresses_argument sig = false →
  sig_self_reference sig = false →
  safety_check sig = SR_AdHominem.
Proof. intros. unfold safety_check. rewrite H1, H, H0. reflexivity.
    Qed.
```

4.6 Mock LLM Responses for Testing

Listing 9: Mock responses for empirical testing

```
(* Biased climate denial response *)
Definition mock_climate_denial_signals : ResponseSignals := {|
  sig_attacks_person := true;        (* "Scientists want grant money"
      *)
  sig_addresses_argument := false;   (* No scientific evidence *)
  sig_uses_tradition := false;
  sig_tradition_relevant := false;
  sig_premises_support := false;     (* "Everyone knows" *)
  sig_considers_counter := false;    (* No counterevidence *)
  sig_seeks_disconfirm := false;
  sig_self_reference := false
|}.

(* Valid reasoning response *)
Definition mock_valid_signals : ResponseSignals := {|
```

```
  sig_attacks_person := false;
  sig_addresses_argument := true;
  sig_premises_support := true;
  sig_considers_counter := true;
  sig_seeks_disconfirm := true;
  sig_self_reference := false;
  (* ... *)
|}.

(* Theorem: Detector catches bias *)
Theorem detector_catches_bias :
  In FM_ObjectSubstitution (analyze_response
     mock_climate_denial_signals).
Proof. unfold analyze_response. simpl. left. reflexivity. Qed.

(* Theorem: Valid passes *)
Theorem valid_passes_detection :
  analyze_response mock_valid_signals = [].
Proof. reflexivity. Qed.
```

5 Extraction and Demonstration

5.1 OCaml Extraction

The Coq formalization extracts to OCaml:

Listing 10: Extraction command

```
Require Import ExtrOcamlBasic ExtrOcamlString ExtrOcamlNatInt.

Extraction "ai_fallacy_detector.ml"
  verify_reasoning
  verify_llm_response
  safety_check
  generate_fix_prompt
  generate_cot_template
  detect_hallucination
  hallucination_to_violation.
```

5.2 Generated OCaml Interface

Listing 11: Extracted OCaml types

```
type domain =
  | D1_Recognition | D2_Clarification | D3_FrameworkSelection
  | D4_Comparison | D5_Inference | D6_Reflection

type verification_result =
  | VR_Valid
  | VR_Type1_NoConstitution
  | VR_Type2_DomainViolation of domain * failure_mode
  | VR_Paradox_LevelConfusion of level_confusion
  | VR_Incomplete of domain

val verify_llm_response : reasoning_attempt -> response_signals ->
   verification_result
val safety_check : response_signals -> safety_result
```

```
val generate_fix_prompt : verification_result -> string
val detect_hallucination : hallucination_signals ->
    hallucination_severity
```

5.3 Integration Example

Listing 12: Production integration

```
(* Self-reflection loop with verified detector *)
let rec reflect llm_call prompt max_iter =
  if max_iter <= 0 then Error "Max iterations"
  else
    let response = llm_call prompt in
    let signals = extract_signals response in
    let attempt = parse_reasoning response in
    match verify_llm_response attempt signals with
    | VR_Valid -> Ok response
    | violation ->
        let fix = generate_fix_prompt violation in
        reflect llm_call (prompt ^ "\n" ^ fix) (max_iter - 1)
```

5.4 Demonstration: Biased Response Detection

Input (biased LLM response):

```
"Climate change is a hoax because scientists are just in it
for the grant money. Anyone who disagrees is naive."
```

Signal Extraction:

- `sig_attacks_person = true` (attacks scientists' motives)
- `sig_addresses_argument = false` (no evidence addressed)
- `sig_considers_counter = false` (no counterevidence)

Detection Results:

1. `safety_check` → `SR_AdHominem`
2. `detect_ad_hominem` → `Some FM_ObjectSubstitution`
3. `detect_confirmation_bias` → `Some FM_ImmunizationFromTesting`

Generated Fix Prompt:

```
Fallacy detected in Recognition: Object Substitution (Ad Hominem).
You attacked the arguer instead of the argument.
Please correct by:
1. Identifying the actual scientific claims
2. Addressing the evidence directly
3. Considering counterevidence to your position
```

5.5 Integration with xAI/Grok

For models like Grok, add domain-check in the prompt chain:

Listing 13: Domain checklist for xAI integration

```
let grok_domain_checklist = [
  (D1, "[ ] Have I correctly identified the question?");
```

```
  (D2, "[ ] Are my key terms clearly defined?");
  (D3, "[ ] Am I using appropriate criteria?");
  (D4, "[ ] Are my comparisons fair?");
  (D5, "[ ] Does my conclusion follow from premises?");
  (D6, "[ ] Have I acknowledged limitations?")
]

let domain_checked_prompt base =
  let checklist = String.concat "\n"
    (List.map snd grok_domain_checklist) in
  "Before answering, verify:\n" ^ checklist ^
  "\n\nIf any fails, revise.\n\n" ^ base
```

6 Conclusion

6.1 Summary

This article presented a machine-verified framework for LLM fallacy detection:

1. **Complete Formalization**: 156 fallacies, 107 theorems, 0 unverified assumptions
2. **Structural Diagnosis**: Errors are locatable, explainable, and correctable
3. **Verified Guarantees**: Key properties are machine-checked
4. **Production Ready**: Extracts to OCaml for deployment

6.2 Bridging Philosophy and AI Safety

Traditional fallacy theory offers catalogs without theory. AI safety offers heuristics without foundations. This work bridges the gap:

- **From philosophy**: Structural account explaining *why* errors occur
- **To AI**: Machine-checked implementations guaranteeing detection

The same analysis that dissolves classical paradoxes now diagnoses reasoning errors in artificial intelligence.

6.3 Availability

All Coq source code is available at:

https://github.com/Horsocrates/theory-of-systems-coq

References

Aristotle. *Sophistical Refutations*. Princeton University Press, 1984.
Horsocrates. "The Laws of Logic as Conditions of Existence." 2026.
Horsocrates. "The Law of Order: Sequence and Hierarchy." 2026.
Horsocrates. "The Six Domains of Reasoning." 2026.
Horsocrates. "The Architecture of Error." 2026.
Horsocrates. "Domain Violations: A Systematic Taxonomy." 2026.
Horsocrates. "Paradox Dissolution Through Hierarchical Analysis." 2026.
The Coq Development Team. *The Coq Proof Assistant*. Version 8.18. INRIA, 2023.
Wei, J., et al. "Chain-of-Thought Prompting." *NeurIPS* 2022.