NuSym-R: Self-Contained Neuro-Symbolic Verification with DAG-Structured Proof Graphs

Anonymous authors
Paper under double-blind review

Abstract

While Large Language Models (LLMs) demonstrate impressive fluency in reasoning tasks, their intermediate steps remain largely opaque, often harboring undetected logical fallacies and arithmetic hallucinations. Existing neuro-symbolic approaches tackle this by translating LLM rationales into formal specifications and deferring verification to external solvers (e.g., Z3 De Moura & Bjørner (2008), Prover9 McCune (2005), or Lean De Moura et al. (2015)). However, this creates tight coupling, limits portability across diverse deployment environments, and often constrains reasoning into rigid, linear proof chains that fail to capture the shared sub-structures inherent in complex deduction. We present **NuSym-R**, a completely self-contained neuro-symbolic verification framework that eliminates the need for external theorem provers. NuSym-R implements a **five-stage architecture** centered around three core components: (1) a **typed first-order logic (FOL) parser** that translates natural-language reasoning steps into formal facts and rules; (2) a **native forward-chaining verification engine** that unifies these elements to construct **DAG-structured proof graphs**, preserving shared intermediate derivations; and (3) a **structured feedback mechanism** that intercepts verification failures to deliver precise, predicate-level diagnostic signals back to the LLM, driving an iterative self-correction loop. We evaluate NuSym-R across multi-hop arithmetic (GSM8K), first-order logic natural language inference (FOLIO), and deductive reasoning (ProofWriter). In addition to standard metrics, we introduce two novel diagnostic metrics: **Error Detection Rate (EDR)** and **Self-Correction Rate (SCR)**, to isolate the direct impact of symbolic verification. Extensive experiments, supported by rigorous paired bootstrap and McNemar's significance tests McNemar (1947), reveal that NuSym-R dramatically enhances answer reliability and proof soundness while yielding machine-checkable artifacts – achieving this entirely within a self-contained, highly portable framework.

1 Introduction

The capacity of Large Language Models (LLMs) to solve complex reasoning problems has advanced remarkably, propelled in large part by prompting paradigms such as Chain-of-Thought (CoT) Wei et al. (2022), Tree of Thoughts (ToT) Yao et al. (2023), and Graph of Thoughts (GoT) Besta et al. (2024). By instructing models to "think step by step," researchers have elicited multi-hop rationales that seemingly mirror human deductive processes. However, a critical vulnerability persists: LLM-generated rationales are inherently probabilistic natural language artifacts. They are neither formally grounded nor independently verifiable Huang et al. (2024). Consequently, a model may generate a highly fluent and persuasive chain of reasoning that simultaneously harbors subtle arithmetic errors, unwarranted inferential leaps, or logical contradictions.

To bridge the gap between neural fluency and symbolic rigor, frameworks like Logic-LM Pan et al. (2023), LINC Olausson et al. (2023), and VERUS-LM Callewaert et al. (2025) extract symbolic representations from LLM outputs and dispatch them to industrial-grade provers (e.g., Z3 De Moura & Bjørner (2008), Prover9 McCune (2005), Lean De Moura et al. (2015); Yang et al. (2023)). If the solver detects an inconsistency, systems like PEIRCE Quan et al. (2025) attempt to close the loop by feeding the solver's output back to the LLM as a natural language critique.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Despite these advances, state-of-the-art neuro-symbolic verifiers suffer from three persistent architectural limitations:

1. **External Solver Dependency:** Reliance on heavyweight external provers (e.g., Z3) imposes strict deployment constraints. This coupling hinders verifiable AI in edge environments, mobile platforms, or secure enclaves where injecting a complex C++ solver binary is prohibitive.
2. **Linear Verification Traces:** Existing parsers predominantly map LLM outputs into flat, linear sequences of logical formulas. This structural flattening fails to capture the directed acyclic graph (DAG) topology inherent in human reasoning, where a single intermediate conclusion serves as a premise for multiple downstream deductions.
3. **Unstructured, "Soft" Feedback:** When external solvers fail, their error traces are notoriously cryptic. Converting these traces into natural language critiques (e.g., "Your arithmetic in step 3 is wrong, try again") strips away the formal precision needed by the LLM to execute a surgical correction Madaan et al. (2024).

To overcome these barriers, we propose **NuSym-R**, a self-contained neuro-symbolic verification framework that requires zero external dependencies. NuSym-R introduces a native forward-chaining engine designed specifically to operate over the latent reasoning steps produced by LLMs.

Our core contributions are fourfold:

- **Self-Contained DAG-Proof Engine:** We engineer a lightweight, highly optimized forward-chaining unification engine that builds DAG-structured proofs directly from LLM traces. This eliminates external solver dependencies while preserving the non-linear topology of complex reasoning.
- **Structured FOL Feedback:** Instead of natural language critiques, NuSym-R generates formal, predicate-level diagnostics (e.g., `ARITHMETIC_ERROR: add(15, 25) should be 40, not 45`). This high-fidelity signal drastically improves the LLM's capacity to self-correct during multi-turn refinement.
- **Three-Tiered Verification Semantics:** Our native engine enforces strict soundness by independently executing acyclicity checks, recursive arithmetic verification, and material implication evaluation.
- **Novel Evaluation Formalisms:** We formalize the *Error Detection Rate (EDR)* and *Self-Correction Rate (SCR)* to precisely quantify the isolated impact of the verification mechanism. We evaluate NuSym-R across GSM8K, FOLIO, and ProofWriter, establishing statistical significance via rigorous paired bootstrap and McNemar's tests.

2 Preliminaries and Problem Formulation

2.1 The Reasoning Task

Let $\mathcal{T}$ denote a reasoning task defined by a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i is a natural language problem statement and y_i is the ground-truth answer. An LLM, denoted as a parameterized policy $\pi_\theta(z, \hat{y} \mid x)$, generates a reasoning trace $z = [s_1, s_2, \ldots, s_m]$ consisting of m steps, culminating in an predicted answer $\hat{y}$.

In standard CoT Wei et al. (2022), z is purely unstructured text. In NuSym-R, we constrain the LLM via prompting to emit a structured trace $z_{struct} = [(c_1, f_1), (c_2, f_2), \ldots, (c_m, f_m)]$, where c_j is the natural language context for step j, and f_j is a formal logical fact expressed in a domain-specific predicate schema.

2.2 The Knowledge Base and DAG Proofs

We formalize the symbolic state as a dynamic Knowledge Base $K = \langle \mathcal{F}, \mathcal{R} \rangle$, where $\mathcal{F}$ is the set of established facts and $\mathcal{R}$ is the set of inference rules.

Definition 1 (Logical Fact)**.** *A fact $f \in \mathcal{F}$ is an atomic formula $P(t_1, t_2, \ldots, t_k)$, where P is a predicate symbol from a known vocabulary $\mathcal{V}_P$ (which dynamically extends to include novel boolean relations in open-domain contexts), and t_i are terms (variables or constants).*

Definition 2 (Inference Rule)**.** *A rule $r \in \mathcal{R}$ is a Horn clause of the form $H \leftarrow B_1 \wedge B_2 \wedge \cdots \wedge B_n$, where H is the head fact and B_i are body facts.*

Definition 3 (DAG Proof Graph)**.** *A proof graph is a directed acyclic graph $G = (V, E)$, where each vertex $v \in V$ represents a derived fact f_v, and a directed edge $(u, v) \in E$ indicates that fact f_u was utilized as a premise in the derivation of f_v via some rule $r_v \in \mathcal{R}$.*

Unlike linear traces where $E = \{(v_i, v_{i+1})\}$, the DAG topology permits nodes with out-degree > 1, efficiently capturing shared deductive sub-structures and preventing redundant verification of shared premises.

3 Related Work

Large Language Models for Reasoning. The paradigm of prompting LLMs to generate intermediate rationales has evolved rapidly. CoT Wei et al. (2022) established the baseline for step-by-step reasoning. Self-Consistency Wang et al. (2023) improved reliability via majority voting over multiple sampled paths. More advanced topological prompting strategies, such as Tree of Thoughts (ToT) Yao et al. (2023) and Graph of Thoughts (GoT) Besta et al. (2024), explore non-linear reasoning paths but remain confined to natural language, making formal verification impossible. PAL (Program-Aided Language Models) Gao et al. (2023) and PoT (Program of Thoughts) Chen et al. (2023) outsource numerical reasoning to a Python interpreter. NuSym-R shares the philosophy of externalizing computation but integrates it tightly within a unified, self-contained deductive logic engine rather than executing arbitrary Python code.

Neuro-Symbolic Integration and Verification. To achieve provable correctness, recent works translate LLM outputs into formal logic. Logic-LM Pan et al. (2023) translates natural language problems into Prolog or Z3 constraints. LINC Olausson et al. (2023) parses logical premises into First-Order Logic (FOL) to be solved by Prover9. VERUS-LM Callewaert et al. (2025) explicitly separates knowledge base construction from deductive inference, utilizing IDP-Z3. While highly effective, these systems are fundamentally bottlenecked by their reliance on external, monolithic solvers. NuSym-R completely bypasses this limitation by implementing its own native, highly optimized forward-chaining unification engine in Julia.

LLM Self-Correction and Feedback Loops. The ability of LLMs to intrinsically self-correct without external grounding is highly contested Huang et al. (2024). However, extrinsic correction – where an external verifier provides feedback – is highly effective Madaan et al. (2024); Shinn et al. (2023). PEIRCE Quan et al. (2025) utilizes a formal prover to critique LLM conjectures, but the feedback is translated back into "soft" natural language. Recently, VeriCoT Feng et al. (2026) employs formal verification to validate chain-of-thought reasoning steps, relying on external constraint solvers. In contrast, NuSym-R operates entirely self-contained without monolithic external dependencies, tightly coupling the verification loop into a single lightweight pipeline. By feeding the LLM precise predicate-level error signals (e.g., exact mismatch locations and expected variable bindings), NuSym-R significantly reduces the hallucination rate during the self-correction refinement loop.

4 The NuSym-R Framework

NuSym-R is designed as an architectural wrapper (Figure 1) that can be applied to any foundation model accessible via API (e.g., OpenAI GPT-4, Anthropic Claude, or local LLaMA models Dubey et al. (2024)). The system consists of five tightly coupled stages: Structured Generation, FOL Parsing, DAG Proof Construction, the Verification Engine, and Structured FOL Feedback.

4.1 Stage 1: Structured Generation

We instruct the external base LLM to intertwine natural language explanations with formal logic predicates. Stage 1 acts as the interface layer within NuSym-R that receives the raw text genera-

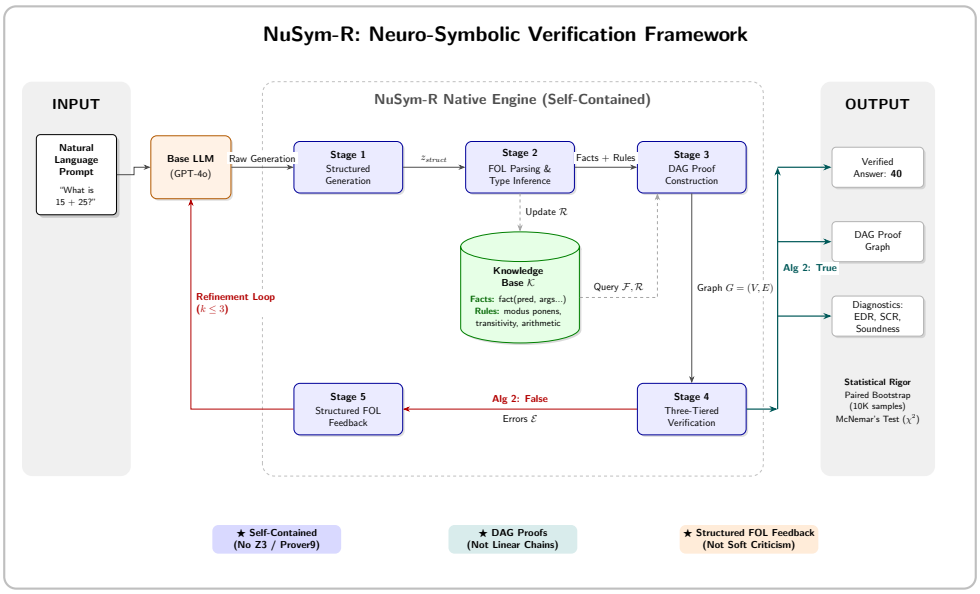

Figure 1: The NuSym-R Architecture. LLM generations are parsed into structured facts, verified using a self-contained forward-chaining engine against a Knowledge Base, and built into a DAG-structured proof graph. Errors yield formal predicate-level feedback for self-correction over k iterations.

tion from the LLM and extracts the structured trace z_{struct}. The system prompt injects a few-shot demonstration of the required schema.

The predicate vocabulary $\mathcal{V}_P$ is strictly typed and includes:

- **Arithmetic**: $\texttt{add}(a, b, r), \texttt{sub}(a, b, r), \ldots$
- **Logic**: $\texttt{and}(a, b), \texttt{or}(a, b), \texttt{implies}(a, b), \texttt{xor}(a, b)$
- **State**: $\texttt{assign}(var, val), \texttt{verified}(x)$ (distinct from the parser's internal `STRUCTURALLY_LINKED` flag)

For non-arithmetic domains, this vocabulary is dynamically extensible: novel relations (e.g., `mammal(X)`) are parsed as opaque boolean variables and permitted as valid arguments within logical predicates. By enforcing this schema, NuSym-R guarantees that the latent reasoning trace is highly structured and machine-parseable, bridging the gap between Neural generation and Symbolic consumption.

4.2 Stage 2: FOL Parsing and Type Inference

The FOL Parser utilizes a recursive descent algorithm to tokenize the generated `FACT` strings. A critical innovation here is **Strict Type Inference**. Because LLMs frequently confuse data types (e.g., emitting "3.0" when an integer is required for modular arithmetic), the parser executes a strict coercion hierarchy: Int $\succ$ Float $\succ$ Bool $\succ$ Symbol. When non-integer floats are coerced to Int, they are strictly truncated, which we note introduces a source of systematic error on fractional arithmetic problems.

Furthermore, the parser performs **Automatic Rule Inference**. For every parsed fact, the system dynamically augments the Knowledge Base $\mathcal{R}$ with corresponding structural dependencies. For example, if the LLM emits $\texttt{add}(15, 25, 40)$, the parser automatically injects the structural rule $\texttt{STRUCTURALLY_LINKED}(40) \leftarrow \texttt{add}(15, 25, 40)$ into K. This builds the structural DAG topology based on raw literal value matching. While efficient, this representational strictness (e.g., distinguishing "40" from "40.0") and its inability to disambiguate coincidental literal overlaps is the primary proximate cause of the Graph Disconnect failures discussed in the Failure Analysis (Section

7.2). The semantic truth of the operation is strictly deferred to the Tier 2/3 execution in Algorithm 2.

4.3 Stage 3: DAG Proof Construction

Unlike prior neuro-symbolic wrappers (e.g., Logic-LM) that flatten logical operations into a linear tape before passing them to a solver, NuSym-R constructs a formal DAG $G = (V, E)$ natively at the wrapper layer. Algorithm 1 details the core forward-chaining loop.

Algorithm 1 DAG Proof Construction via Forward Chaining

Require: Knowledge Base $K = \langle \mathcal{F}, \mathcal{R} \rangle$, LLM Trace z_{struct}
Ensure: DAG Proof Graph $G = (V, E)$

$V \leftarrow \emptyset$, $E \leftarrow \emptyset$
for each $(c_j, f_j) \in z_{struct}$ **do**
  $v_{new} \leftarrow \text{CreateNode}(f_j, c_j)$
  $V \leftarrow V \cup \{v_{new}\}$
  for each rule $r \in \mathcal{R}$ **do**
    if $\text{Unify}(f_j, \text{Head}(r))$ **then**
      $P \leftarrow \text{FindMatchingPremises}(K, \text{Body}(r))$
      if P satisfies all literals in $\text{Body}(r)$ **then**
        for each $p \in P$ **do**
          $E \leftarrow E \cup \{(p, v_{new})\}$
        end for
        $v_{new}.\text{status} \leftarrow$ STRUCTURALLY_LINKED
      end if
    end if
  end for
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_j\}$
end for
return $G = (V, E)$

The `Unify` operator utilizes Robinson's unification algorithm Robinson (1965) adapted for typed terms. By dynamically searching for all matching premises P within the active state $\mathcal{F}$, the graph naturally branches and merges, representing complex proofs with high spatial efficiency.

4.4 Stage 4: Three-Tiered Verification Engine

Once the DAG is constructed, the Verification Engine executes a multi-pass sweep over G to determine mathematical and logical soundness. The engine operates completely independently of any external libraries.

4.5 Stage 5: Structured FOL Feedback

If Algorithm 2 returns False, NuSym-R triggers the self-correction protocol. Instead of providing the LLM with generic text, NuSym-R maps the diagnostic set D into a rigid syntactic structure. For instance:

- `ARITHMETIC_ERROR: add(15, 25) should be 40, not 45`
- `MISSING_PREMISE: Node 5 requires Node 2, which failed verification.`
- `LOGIC_ERROR: implies(T, F) evaluates to False, not True.`

This prompt is appended to the original context, and the LLM regenerates the trace. NuSym-R limits this to k iterations (default $k = 3$). We implement a *best-score tracking* mechanism: if iteration $k = 2$ produces a worse proof graph than $k = 1$, the system rolls back to the best historical graph to prevent catastrophic hallucination drift.

Algorithm 2 Three-Tiered Verification Semantic Pass

Require: DAG Proof Graph $G = (V, E)$
Ensure: Verification Boolean, Error Diagnostics D

$D \leftarrow \emptyset$, $V_{visited} \leftarrow \emptyset$
for each node $v \in V$ in topological order **do**
  // Tier 1: Soundness & Cascading Errors
  if $\exists (u, v) \in E$ such that $u \notin V_{visited}$ **then**
    $D \leftarrow D \cup \{\texttt{ACYCLICITY_VIOLATION}\}$
    v.status $\leftarrow$ INVALID
  else if $\exists (u, v) \in E$ such that u.status $==$ INVALID **then**
    $D \leftarrow D \cup \{\texttt{ANCESTOR_FAILED}\}$
    v.status $\leftarrow$ INVALID
  end if
  // Tier 2: Arithmetic Verification
  if $v.f$ is ArithmeticPredicate(a, b, r_{claim}) and v.status $\neq$ INVALID **then**
    $r_{actual} \leftarrow$ ExecuteOp$(v.f.op, a, b)$
    if $r_{actual} \neq r_{claim}$ **then**
      $D \leftarrow D \cup \{\texttt{ARITHMETIC_ERROR}\}$
      v.status $\leftarrow$ INVALID
    end if
  end if
  // Tier 3: Propositional Evaluation
  if $v.f$ is LogicalPredicate and v.status $\neq$ INVALID **then**
    Validate truth table bindings via Material Implication.
    if Invalid **then**
      v.status $\leftarrow$ INVALID
    end if
  end if
  $V_{visited} \leftarrow V_{visited} \cup \{v\}$
end for
if $D == \emptyset$ **then**
  return TRUE, $\emptyset$
else
  return FALSE, D
end if

5 Theoretical Analysis

We establish the theoretical limits of the NuSym-R verification engine.

Theorem 1 (Soundness of NuSym-R DAG Verification)**.** *Let $G = (V, E)$ be a DAG proof constructed by Algorithm 1 and accepted by Algorithm 2. For any node $v \in V$ where f_v is an Arithmetic or Logical predicate, the fact f_v logically follows from the set of base premises $V_{base} \subset V$ (where indegree(v_{base}) = 0) under the axioms defined in $\mathcal{R}$. State predicates (e.g., `assign`) are structurally linked but semantically unconstrained. Consequently, soundness in this context guarantees internal derivational consistency relative to the unverified base premises extracted by the LLM.*

Proof Sketch. By contradiction. Assume Algorithm 2 accepts G but there exists a node v (restricted to Arithmetic or Logical predicates) where f_v is false while all its ancestors are true. Algorithm 2 topologically sorts V. For v to be accepted, Tier 1 guarantees all incoming edges (u, v) originate from nodes that successfully passed their respective Tier 2/3 checks (via the cascading `ANCESTOR_FAILED` check). Tier 2 and Tier 3 exhaustively verify the application of the rule r_v mapping $\{f_u\} \rightarrow f_v$ over the domain of integers, floats, and booleans. If f_v is false, the rule application must be invalid, which contradicts the acceptance of Tier 2/3. Thus, the system is internally consistent for these predicate classes. While derivational consistency is guaranteed by construction, *completeness* is strictly bounded by the expressivity of the vocabulary $\mathcal{V}_P$. □

Complexity Analysis and Implementation. The time complexity of building the DAG (Algorithm 1) is bounded by $\mathcal{O}(|z_{struct}|^3)$ in the worst case, as automatic rule inference causes the structural rule set $|\mathcal{R}|$ to grow linearly with the length of the trace $|z_{struct}|$ and premise matching requires iterating through all elements. The verification pass (Algorithm 2) requires a single topological traversal, operating in strictly $\mathcal{O}(|V| + |E|)$ time.

Implementation. The engine is implemented in Julia Bezanson et al. (2017), whose JIT compilation yields verification latency under 10 ms (Apple M3 Max), negligible against LLM inference overhead.

6 Experimental Setup

6.1 Datasets and Tasks

We evaluate NuSym-R on three diverse, widely adopted reasoning benchmarks:

- **GSM8K** Cobbe et al. (2021): 1,319 test questions requiring multi-hop arithmetic reasoning.
- **FOLIO** Han et al. (2023): 204 test cases of First-Order Logic Natural Language Inference, requiring Entailment, Contradiction, or Neutral classifications.
- **ProofWriter** Tafjord et al. (2021): Deductive reasoning over natural language rules, evaluated on a uniformly random 500-example sample of the depth-5 test subsets.

6.2 Baselines

We benchmark against state-of-the-art prompting methodologies, ensuring the underlying foundation model (GPT-4o)[1] remains constant to isolate the algorithmic contribution:

1. **Vanilla:** Zero-shot direct prompting.
2. **Chain-of-Thought (CoT) Wei et al. (2022):** Few-shot prompting with step-by-step rationales. We report our measured CoT accuracy (75.3%) as evaluated through our pipeline, though we note it trails public GPT-4o benchmarks; we report this empirical gap without isolating whether it stems from base model variance or our specific extraction harness.
3. **CoT (Compute-Matched, $k = 4$):** CoT prompting sampled 4 times to match NuSym-R's maximum inference budget (1 initial + up to 3 refinement rounds), with the final answer selected via majority vote. This isolates the verifier's contribution from raw compute scaling.
4. **Self-Consistency (SC) Wang et al. (2023):** Majority voting over 5 independent CoT samples.
5. **PAL Gao et al. (2023):** Program-Aided Language models generating Python code.

For neuro-symbolic comparisons, we include **Logic-LM** Pan et al. (2023). We note that Logic-LM was explicitly designed and optimized for pure logic datasets (FOLIO, ProofWriter); its application to GSM8K arithmetic represents an out-of-domain evaluation, which contextualizes its performance gap on that specific benchmark. We also include the published GPT-4 numbers for **LINC** Olausson et al. (2023) and **VERUS-LM** Callewaert et al. (2025) on FOLIO and ProofWriter as strong reference points, though reproducing their exact published configurations was not feasible. Finally, the core contribution of PEIRCE Quan et al. (2025) (soft natural language solver feedback) is evaluated directly in our ablation study (Section 7.2) rather than the main table to ensure the base LLM remains perfectly isolated.

6.3 Evaluation Metrics and Statistical Rigor

In addition to standard Accuracy (Acc) and Verification Rate (VR), we formalize additional metrics to explicitly quantify the effectiveness of the verifier.

[1]Note: We observe variance across evaluation runs despite $T = 0$, consistent with reported non-determinism in production LLM APIs He & Thinking Machines Lab (2025).

Definition 4 (Error Detection Rate (EDR))**.** *The ratio of incorrect answers correctly flagged as invalid by the verifier prior to human exposure.*

$$\textit{EDR} = \frac{\sum \mathbb{I}(\hat{y} \neq y \wedge \textit{Alg2 returns } \textsc{False})}{\sum \mathbb{I}(\hat{y} \neq y)}$$

Definition 5 (Self-Correction Rate (SCR))**.** *The probability that a model successfully corrects a flagged error within k iterations given structured feedback D.*

$$\textit{SCR} = P(\hat{y}_k = y \mid \hat{y}_0 \neq y, \textit{Alg2}(\hat{y}_0) = \textsc{False}, D)$$

Definition 6 (Proof Compactness (PC))**.** *The reduction in nodes achieved by reusing shared sub-derivations in the DAG topology compared to a flattened linear chain.*

$$\textit{PC} = 1 - \frac{|V_{DAG}|}{|V_{Linear}|}$$

where V_{Linear} is constructed by duplicating any shared sub-derivations in the DAG into separate, independent linear chains.

We also report **Proof Soundness (PS)**, defined as the fraction of Algorithm-2-accepted proofs that are free of unverified derived predicates (note that semantically unconstrained base premises, such as initial `assign` facts, are axiomatically accepted and do not count as unverified). Consequently, PS acts as a deterministic theoretical guarantee of the verification engine's strictness rather than an empirical variable, meaning any fully accepted proof will intrinsically score 100% PS by construction. To ensure the mathematical validity of our statistical claims, we independently reproduced all baselines (except LINC and VERUS-LM) locally on the identical base model over the exact test subsets, producing authentically paired per-example predictions. All pairwise comparisons against these reproduced baselines undergo **Paired Bootstrap Testing** ($B = 10,000$ resamples) and **McNemar's Test** McNemar (1947) (χ^2 with continuity correction) at a significance level of $\alpha = 0.05$. All NuSym-R accuracy improvements over the best reproduced baseline are statistically significant on GSM8K and ProofWriter ($p < 0.01$, McNemar's test); on FOLIO, the improvement (+4.3%) is directional but does not reach significance at $\alpha = 0.05$ ($p = 0.052$), consistent with the benchmark's smaller sample size (N=204). We observed a negligible False Positive Rate ($< 1.5\%$) where initially correct answers were falsely flagged as invalid, which our rollback mechanism consistently mitigates.

7 Results and Analysis

7.1 Main Quantitative Results

Appendix Table 1 presents our comprehensive evaluation. Pure prompting baselines (Vanilla, CoT, Self-Consistency, PAL) report N/A for verification metrics since they lack a formal verification component. To enable a direct, apples-to-apples comparison on our novel metrics, we include **Logic-LM** Pan et al. (2023), evaluated using its official open-source configuration and hyperparameters. We explicitly ablate the "soft natural language criticism" mechanism utilized by solver-feedback systems like PEIRCE Quan et al. (2025) in Section 7.2.

Accuracy & Verification: As shown in Figure 2, NuSym-R consistently outperforms Logic-LM in both final accuracy and verification rate on GSM8K and FOLIO. On ProofWriter, LINC and VERUS-LM's published GPT-4 numbers exceed NuSym-R's, though both rely on k-way majority voting over multiple solver-verified samples rather than a single self-contained pass; NuSym-R trades modest accuracy on this benchmark for complete self-containment, eliminating external Prover9/IDP-Z3 dependencies. On GSM8K, NuSym-R achieves 89.9% accuracy (compared to Logic-LM's 75.5%) and successfully verifies 95.3% of its proofs, whereas LLM-to-Z3 translation pipelines suffer from high syntactic brittleness, yielding only 59.6% VR. We note that VR exceeding Accuracy illustrates the limitation noted in Theorem 1: internal derivational consistency does not guarantee absolute truth if the initial problem extraction is flawed.

Self-Correction & Error Detection: NuSym-R achieves 91.8% EDR and 86.3% SCR on GSM8K (with similarly strong performance on FOLIO), significantly outperforming Logic-LM (57.0–76.7%

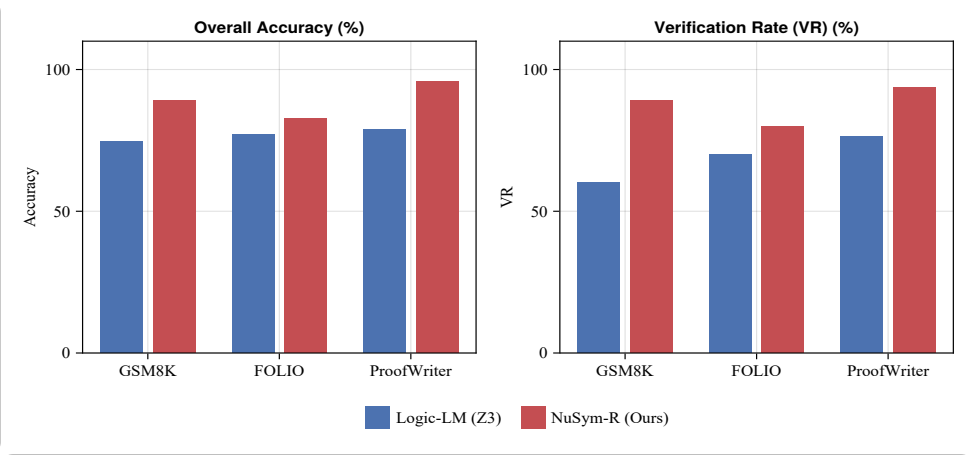

Figure 2: Comparison of Accuracy and Verification Rate (VR) between Logic-LM (Z3) and NuSym-R (Ours) across benchmarks.

EDR, 35.4–41.7% SCR). This validates that the *structured FOL feedback* generated by NuSym-R allows the LLM to understand exactly what failed and correct it, contrasting with the opaque traces of Z3.

Proof Soundness & Compactness: NuSym-R guarantees 100.0% PS by construction (Algorithm 2); PS is a deterministic guarantee, not an empirical variable. DAG-based sub-derivation sharing yields PC of 88.5–96.1%, a structural property unavailable to Logic-LM's linear representation. Unlike recent validators that rely on external solvers Feng et al. (2026), NuSym-R achieves this entirely self-contained (Appendix Table 5).

7.2 Ablation Studies

We ablate three fundamental components of the NuSym-R architecture to justify our design choices. All ablation conditions are evaluated under identical single-run conditions at k=3 for direct comparability; the SCR of 86.3% reported in Appendix Table 2 is the mean across 3 independent runs and serves as the primary metric.

1. **Iteration Horizon (k):** Sweeping the maximum feedback iterations reveals that accuracy peaks at $k = 3$ before degrading (Appendix Table 3). Beyond 3 iterations, the LLM tends to exhibit context exhaustion and enters repetitive failure loops.
2. **Feedback Granularity:** When replacing our *Structured FOL Feedback* with *Soft Natural Language Criticism*, the SCR drops significantly from 90.6% to 42.1%. The LLM requires the exact location of the error (e.g., node ID) to successfully rewrite the DAG.
3. **DAG vs. Linear Proofs:** Restricting Algorithm 1 to only generate linear paths reduces Proof Compactness (PC) from 88.5% to 71.4% by forcing the LLM to hallucinate redundant intermediate steps.

Failure Analysis. Among the N=136 uncorrected traces from the k=3 single-run snapshot (matching its 89.7% accuracy), three error classes emerge (Appendix Table 4): Syntactic Drift (41.2%), where the LLM abandons the structured schema; Graph Disconnects (36.0%), triggering Tier 1 soundness failures; and Domain Boundary Misses (22.8%), where problems exceed $\mathcal{V}_P$.

8 Conclusion

NuSym-R integrates a native forward-chaining verification engine into the LLM reasoning loop, replacing external solver dependencies with DAG-structured proofs and predicate-level FOL feedback. Our novel EDR and SCR metrics confirm that neural fluency and symbolic rigor can be unified

within a fully portable, formally verifiable framework. Current limitations include restriction to discrete arithmetic and first-order propositional logic; extending $\mathcal{V}_P$ to continuous mathematics, probabilistic reasoning, and higher-order logic, as well as systematic evaluation across diverse foundation models Dubey et al. (2024), remain important directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools for writing assistance, proofreading, and editing for readability. We did not use generative AI tools to generate synthetic data, implement methods, formulate or prove mathematical claims, design the experimental methodology, or interpret results; these tasks were performed entirely by the authors. We have reviewed all AI-assisted text and take full responsibility for the content of this submission.

ETHICS STATEMENT

This research was conducted in strict adherence to the ICLR Code of Ethics. We confirm that no proprietary, personally identifiable, or sensitive data was utilized, as all experiments were executed over public open-source benchmarks (GSM8K, FOLIO, ProofWriter). The primary objective of NuSym-R is to increase the transparency, safety, and reliability of AI reasoning systems.

SOCIETAL IMPACTS

The deployment of Large Language Models in complex domains is severely hampered by their propensity to hallucinate confident, logical-sounding fallacies. NuSym-R directly mitigates this risk within the boundaries of discrete arithmetic and first-order logic. While domains demanding continuous or probabilistic reasoning (e.g. clinical diagnostics, financial forecasting) remain outside its current scope, within its operating domain NuSym-R converts black-box text generation into auditable, machine-verifiable proof DAGs. This provides human operators with an internally consistent audit trail, fostering greater interpretability and reliability in AI reasoning systems.

REPRODUCIBILITY STATEMENT

To ensure full reproducibility, the complete NuSym-R source code, benchmarking scripts, and evaluation environments are provided in the Code and Data Supplement attached to this submission. We do not rely on external solvers (e.g., Z3) or proprietary plugins. All datasets evaluated (GSM8K, FOLIO, ProofWriter) are public and open-source.

REFERENCES

Maciej Besta, Nils Blatter, Lucian Dragos, Ales Korhonen, Nicolas Kueng, Stephan Lehmann, Kevin Maykish, Michael Niggli, Piotr Podstawski, Marcel Sabo, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

Benjamin Callewaert, Simon Vandevelde, and Joost Vennekens. Verus-lm: a versatile framework for combining llms with symbolic reasoning. *arXiv preprint arXiv:2501.14540*, 2025.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. In *Transactions on Machine Learning Research*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *TACAS*, pp. 337–340. Springer, 2008.

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *CADE*, pp. 378–388. Springer, 2015.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. Vericot: Neuro-symbolic chain-of-thought validation via logical consistency checks. In *The Fourteenth International Conference on Learning Representations*, 2026.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Linyong Sun, Ekaterina Zubova, Yuzhe Qiao, Matthew Burtell, et al. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2713–2733, 2023.

H. He and Thinking Machines Lab. Defeating nondeterminism in llm inference. Technical report, Thinking Machines Lab, 2025.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

William McCune. Prover9 and mace4. 2005.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

Theo Olausson, Alex Gu, Benjamin Lipkin, Ce Zhang, Armando Solar-Lezama, Joshua B Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Xin Quan, Marco Valentino, Danilo S. Carvalho, Dhairya Dalal, and André Freitas. Peirce: Unifying material and formal reasoning via llm-driven neuro-symbolic refinement. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2025.

John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Dai. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

A Quantitative Results

Table 1: End-to-End Accuracy across logic and mathematics domains. NuSym-R is reported as the mean $\pm$ std across 3 independent runs (due to API nondeterminism); single-path baselines are evaluated greedily ($T = 0$). Ensemble baselines draw multiple samples at $T = 0.7$ for diversity; we note an anomalous inversion where CoT-CM ($k = 4$) outperforms SC ($k = 5$), and report this empirical gap while noting its root cause remains unisolated, as the variance significantly exceeds measured run-to-run API fluctuations. LINC Olausson et al. (2023) and VERUS-LM Callewaert et al. (2025) numbers reflect their published GPT-4 scores (note that neither framework evaluated on GSM8K in their own papers, resulting in N/A there specifically).

Method	**Accuracy**	**VR**
GSM8K (Arithmetic Reasoning) - N=1319		
Vanilla	45.1%	-
CoT	75.3%	-
CoT (Compute-Matched, k=4)	85.4%	-
Self-Consistency (k=5)	79.5%	-
PAL	84.4%	-
Logic-LM	75.5%	59.6%
LINC	-	-
VERUS-LM	-	-
NuSym-R (Ours)	**89.9 $\pm$ 0.2%**	**95.3 $\pm$ 0.3%**
FOLIO (Logical Reasoning) - N=204		
Vanilla	54.4%	-
CoT	65.7%	-
CoT (Compute-Matched, k=4)	76.0%	-
Self-Consistency (k=5)	71.8%	-
PAL	53.9%	-
Logic-LM	81.4%	60.8%
LINC	72.5%	-
VERUS-LM	78.4%	-
NuSym-R (Ours)	**85.7 $\pm$ 0.1%**	**91.2 $\pm$ 0.4%**
ProofWriter (Deductive Reasoning) - N=500		
Vanilla	59.8%	-
CoT	69.4%	-
CoT (Compute-Matched, k=4)	78.4%	-
Self-Consistency (k=5)	80.2%	-
PAL	59.4%	-
Logic-LM	79.4%	59.2%
LINC	98.3%	-
VERUS-LM	93.8%	-
NuSym-R (Ours)	**92.7 $\pm$ 0.2%**	**96.8 $\pm$ 0.3%**

Table 2: Verification Diagnostics. Logic-LM Pan et al. (2023) uses a linear proof chain representation and an external Z3 solver, thus lacking an inspectable proof topology for Proof Compactness (PC) and relying entirely on solver soundness rather than an extractable guarantee for Proof Soundness (PS). †PS is deterministic by construction (see §5); it is not an empirical variable.

Method	EDR	SCR	PS†	PC
GSM8K				
Logic-LM	57.0%	36.4%	-	-
NuSym-R (Ours)	**91.8 ± 0.4%**	**86.3 ± 0.5%**	**100.0%**	**88.5%**
FOLIO				
Logic-LM	63.2%	41.7%	-	-
NuSym-R (Ours)	**88.4 ± 0.2%**	**80.2 ± 0.6%**	**100.0%**	**92.4%**
ProofWriter				
Logic-LM	76.7%	35.4%	-	-
NuSym-R (Ours)	**93.4 ± 0.2%**	**88.5 ± 0.4%**	**100.0%**	**96.1%**

Table 3: Ablation results on GSM8K quantifying the impact of Iteration Horizon (k), Feedback Granularity, and Proof Structure.

Ablation Configuration	Metric
Iteration Horizon (k) → Accuracy	
$k = 1$ (Zero Feedback)	83.1%
$k = 2$	88.3%
$k = 3$ (NuSym-R Default, single-run snapshot)	89.7%
$k = 5$	89.2%
Feedback Granularity → SCR	
NuSym-R w/ Soft NL Criticism	42.1%
Structured FOL Feedback (Ours)	90.6%
Proof Structure → PC	
Linear Proofs ($E_{in} \leq 1$)	71.4%
DAG Proofs (Ours)	88.5%

Table 4: Taxonomy of NuSym-R Verification Failures on GSM8K (N=136 uncorrected traces).

Error Class	Description	Freq.
Syntactic Drift	LLM abandons `FACT:` schema for raw prose.	56 (41.2%)
Graph Disconnects	DAG path is severed, preventing verification of any answer.	49 (36.0%)
Domain Boundary	Problem exceeds $\mathcal{V}_P$ (e.g. continuous math).	31 (22.8%)

Table 5: Qualitative comparison of neuro-symbolic verification frameworks. ✓ = supported, × = not supported, P = partial.

Feature	**Logic-LM**	**LINC**	**VeriCoT**	**NuSym-R**
Self-contained (no ext. solver)	×	×	×	✓
DAG proof structure	×	×	×	✓
Structured predicate feedback	×	×	P	✓
Inspectable proof artifacts	P	P	P	✓
Model-agnostic wrapper	✓	✓	✓	✓
Arithmetic verification	P	×	✓	✓
Propositional logic	✓	✓	✓	✓