VERGE: Formal Refinement and Guidance Engine for Verifiable LLM Reasoning

Vikash Singh[1]* Darion Cassel[2], Nathaniel Weir[2], Nick Feng[2], Sam Bayless[2]

[1]Case Western Reserve University [2]Amazon Web Services

Abstract

Despite the syntactic fluency of Large Language Models (LLMs), ensuring their logical correctness in high-stakes domains remains a fundamental challenge. We present a neurosymbolic framework that combines LLMs with SMT solvers to produce verification-guided answers through iterative refinement. Our approach decomposes LLM outputs into atomic claims, autoformalizes them into first-order logic, and verifies their logical consistency using automated theorem proving. We introduce three key innovations: (1) multi-sample consensus via formal semantic equivalence checking to ensure logic-level alignment between candidates, eliminating the syntactic bias of surface-form metrics, (2) semantic routing that directs different claim types to appropriate verification strategies: symbolic solvers for logical claims and LLM ensembles for commonsense reasoning, and (3) precise logical error localization via Minimal Correction Subsets (MCS), which pinpoint the exact subset of claims to revise, transforming binary failure signals into actionable feedback. Our framework classifies claims by their logical status and aggregates multiple verification signals into a unified score with variance-based penalty. The system iteratively refines answers using structured feedback until acceptance criteria are met or convergence is achieved. This hybrid approach delivers formal guarantees where possible and consensus verification elsewhere, advancing trustworthy AI. With the GPT-OSS-120B model, VERGE demonstrates an average performance uplift of 18.7% at convergence across a set of reasoning benchmarks compared to single-pass approaches.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across diverse reasoning tasks, from mathematical problem-solving (Lewkowycz et al., 2022; Hendrycks et al., 2021) to code generation (Chen et al., 2021; Austin et al., 2021) and logical inference (Tafjord et al., 2021). Despite these advances, ensuring the correctness of LLM-generated answers remains a critical barrier in high-stakes domains such as legal policy compliances, healthcare and finance etc. While recent models achieve impressive benchmark performance (OpenAI, 2025; Anthropic, 2025; Meta AI, 2025), they rely on statistical likelihood maximization (Ouyang et al., 2022) rather than logical deduction. Consequently, they operate with-

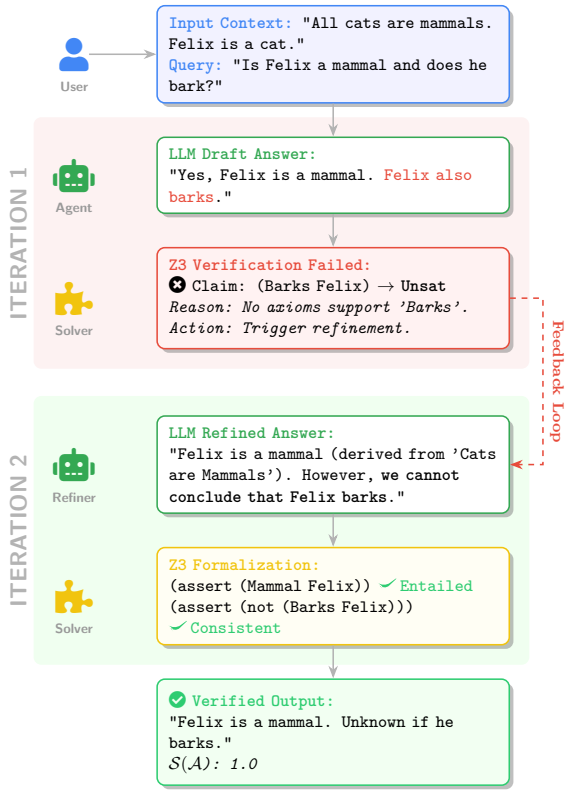

Figure 1: VERGE correcting LLM Hallucinations via Formal Verification. The solver detects an unsupported claim and guides the LLM to a consistent answer through MCS-based feedback.

*Work done during internship at Amazon Web Services

out mechanisms for provable correctness, making them prone to hallucinations and internal contradictions (Weng et al., 2023).

Current verification strategies such as self-consistency (Wang et al., 2022), process supervision (Lightman et al., 2023), and self-refinement (Madaan et al., 2023; Shinn et al., 2023) provide *heuristic* rather than *formal* guarantees. Even multi-agent debate frameworks (Du et al., 2023; Liang et al., 2023) merely achieve consensus, which does not imply correctness. To achieve verifiable reasoning, neuro-symbolic methods (Singh et al., 2026; Ganguly et al., 2025, 2024; Feng et al., 2025; Pan et al., 2023; Callewaert et al., 2025; Olausson et al., 2023; Garcez et al., 2019; Mao et al., 2019) and semantic parsing (Wang et al., 2026; McGinness and Baumgartner, 2024; Zettlemoyer and Collins, 2005; Dong and Lapata, 2016) have attempted to bridge natural language with formal logic. However, these approaches face a fundamental *semantic gap*: natural language is inherently ambiguous, and rigid formalization often fails on open-domain claims (Church, 1936; Turing, 1936).

We present **VERGE**, a framework that mitigates this gap by combining LLMs with Satisfiability Modulo Theories (SMT) solvers (Barrett et al., 2009; De Moura and Bjørner, 2008) to produce verification-guided answers with formal guarantees for logical/mathematical claims through iterative refinement, as illustrated in Figure 1. Unlike standard feedback loops, VERGE leverages the SMT solver's ability to extract *unsatisfied assertions* (Zhang and Malik, 2003; Nadel et al., 2014). This allows us to compute **Minimal Correction Subsets (MCS)** (Belov et al., 2012; Marques-Silva et al., 2013), identifying a minimal set of modifications to the atomic claims sufficient to restore consistency and move from ***probabilistic*** self-correction to ***provable self-consistency correction***.

The Expressivity Trade-off and Pragmatic Verification. A key insight of our work is that enforcing formal verification on all claims is fundamentally misaligned with the ambiguity of natural language. Rather than attempting to bridge this gap universally, a theoretically intractable goal, we adopt a **pragmatic stance**: Apply formal verification where the semantic gap is narrow (mathematical/logical claims), and fall back to consensus-based verification where it is wide (commonsense/-vague claims). VERGE introduces a verification cascade with semantic routing to implement this strategy. This hybrid approach provides formal guarantees for a verifiable subset of claims while maintaining the system's ability to handle broad, real-world reasoning tasks.

Our work introduces:

1. **High-Fidelity Consensus via SMT:** We bridge the semantic gap by enforcing semantic equivalence ($\phi_a \iff \phi_b$) among candidate formalizations. Unlike syntactic metrics (e.g., BLEU, Jaccard) which fail on variable renaming or structural permutation, we utilize the solver to prove that different candidate formulas yield identical truth tables, ensuring robust consensus.
2. **Actionable Feedback via MCS:** We adapt greedy MCS computation (Marques-Silva et al., 2013; Morgado et al., 2013; Bacchus and Katsirelos, 2015) to provide polynomial-time, specific feedback (e.g., "claim C_2 does not hold") rather than generic error signals.
3. **Flexible Neuro-symbolic Integration:** A semantic routing framework that balances the precision of SMT solvers with the flexibility of LLMs, avoiding the pitfalls of forcing undecidable language into decidable theories.

2 Related Work

Probabilistic vs. Formal Verification. Standard LLM reasoning strategies rely on *probabilistic* confidence. Methods like self-consistency (Wang et al., 2022) and process supervision (Ganguly et al., 2026; Chen et al., 2025; Lightman et al., 2023; Uesato et al., 2022; Yang et al., 2026) aggregate samples or train verifiers on human labels, but cannot guarantee logical soundness. Self-refinement approaches (Madaan et al., 2023; Shinn et al., 2023; Welleck et al., 2022) use the model to critique itself, often failing due to the faithfulness gap where reasoning does not match output (Lyu et al., 2023; Huang et al., 2024). Multi-agent debate (Du et al., 2023; Liang et al., 2023) achieves consensus, not truth, recent work shows self-correction can even degrade performance (Huang et al., 2024; Kamoi et al., 2024). In contrast, VERGE uses SMT solvers (Barrett et al., 2009; De Moura and Bjørner, 2008) to provide *mathematically proven* feedback. Unlike tool-augmented LLMs (Schick et al., 2023; Gou et al., 2024) that use tools for execution (e.g., calculators), we use tools for *consistency checking*, computing MCS. (Zhang and Malik, 2003; Nadel et al., 2014) to identify exactly which premises contradict the generated answer.

Neuro-Symbolic Integration and the Semantic Gap. Traditional semantic parsing (Zafar et al., 2026; Dong and Lapata, 2016; Berant et al., 2013; Zettlemoyer and Collins, 2005) maps language to executable logical forms but requires expensive supervision. Recent work extends this to theorem proving (Polu and Sutskever, 2020; Polu et al., 2022; Jiang et al., 2022; Azerbayev et al., 2023) and augmenting LLMs with symbolic solvers (Pan et al., 2023; Olausson et al., 2023; Callewaert et al., 2025), but these require fully formalizable domains. Prior neuro-symbolic integration (Mao et al., 2019; Garcez et al., 2019; Kautz, 2022) and grammar-based approaches (Ganguly et al., 2024) struggle with the semantic gap (Church, 1936; Turing, 1936) the mismatch between ambiguous natural language and rigid formal systems. VERGE targets *open-domain* natural language where full formalization is often impossible. We introduce **semantic routing**, rather than forcing vague or commonsense claims into rigid first-order logic, we route them to a consensus-based soft verifier. This treats the semantic gap as an inherent property of language requiring hybrid verification.

Automated Reasoning for Repair. Our feedback mechanism adapts MCS computation (Marques-Silva et al., 2013; Liffiton and Malik, 2008) from constraint programming to NLP. MCS identifies the minimal set of constraints to delete to restore satisfiability. Recent work applies MCS to constraint relaxation (Bacchus and Katsirelos, 2015) and automated debugging (Morgado et al., 2013). We innovate by translating MCS output into natural language feedback, guiding the LLM to *rewrite* specific atomic claims. This prioritizes interpretability and convergence speed ($O(m \times \text{SAT})$ greedy approximation) over theoretical optimality. To our knowledge, VERGE is the first to apply MCS-based feedback to guide iterative refinement in LLM reasoning, converting abstract unsat cores into actionable guidance (see Appendix B).

3 Methodology

Problem Formulation. Given a context $\mathcal{C} = \{p_1, \ldots, p_m\}$ of premise statements and a query q, we aim to produce a verified answer $\mathcal{A}^*$ composed of atomic claims $\{c_1, \ldots, c_n\}$ with maximal verification coverage. Here, $\mathcal{A}^*$ is a refined version of the candidate answer $\mathcal{A}$ obtained through verification. We formalize this as maximizing the verification score $\mathcal{S}(\mathcal{A}) \in [0, 1]$ subject to two constraints for each claim c_i: (1) *Consistency*: $\text{SAT}(\varphi_{\mathcal{C}} \wedge \varphi_{c_i}) = \text{true}$, and (2) *Entailment*: $\text{SAT}(\varphi_{\mathcal{C}} \wedge \neg\varphi_{c_i}) = \text{false}$, where φ denotes the logical formalization function that maps natural language statements to SMT constraints. The consistency constraint verifies that each claim is compatible with the context, while the entailment constraint ensures that each claim is a logical consequence of the context. These constraints provide formal equivalence guarantees where logic permits, while falling back to semantic consistency (soft verification, see §3.4) otherwise. **Pipeline.** The pipeline (Fig. 2) executes iteratively: (1) entity extraction, (2) generation, (3) decomposition, (4) formalization & verification, and (5) refinement.

3.1 Entity Extraction and Generation

We first extract entities $\mathcal{E} = \text{Extract}(\mathcal{C}, q)$ (e.g., 'Felix', 'Monday', 'Process A') to serve as typed constants in SMT. At iteration t, we generate answer $\mathcal{A}^{(t)}$ via a language model M:

$$\mathcal{A}^{(t)} = M(\mathcal{C}, q, \mathcal{A}^{(t-1)}, \mathcal{F}^{(t-1)}) \quad (1)$$

where $\mathcal{F}^{(t-1)}$ is structured feedback (see §3.5). The initial iteration ($t = 1$) utilizes zero-shot prompting (e.g., $A^0 = M(\mathcal{C}, q, \emptyset, \emptyset)$).

3.2 Claim Decomposition and Classification

We decompose $\mathcal{A}$ into atomic claims $\{c_1, \ldots, c_n\}$. To ensure our system is honest about what it can and cannot formally prove, we classify each claim c_i into a semantic type $\tau_i \in \mathcal{T}$ via M:

$$\tau_i = \arg \max_{\tau \in \{\tau_{\text{M}}, \tau_{\text{L}}, \tau_{\text{T}}, \tau_{\text{P}}, \tau_{\text{C}}, \tau_{\text{V}}\}} P_M(\tau \mid c_i) \quad (2)$$

where types correspond to *Mathematical, Logical, Temporal, Probabilistic, Commonsense, and Vague* claims. These categories align with standard distinctions in semantic parsing to differentiate verifiable facts from subjective or probabilistic statements. This classification is crucial for minimizing "false formalization", the error of forcing ambiguous natural language into rigid logic. Vague claims are identified by the model as containing subjective predicates (e.g., "likely", "possibly") that lack binary truth values, preventing brittle SMT assertions.

3.3 SMT Formalization with Consensus

For claims classified as logical or mathematical, we target the **QF_UF** (Quantifier-Free Uninterpreted Functions) and **QF_LIA** (Quantifier-Free

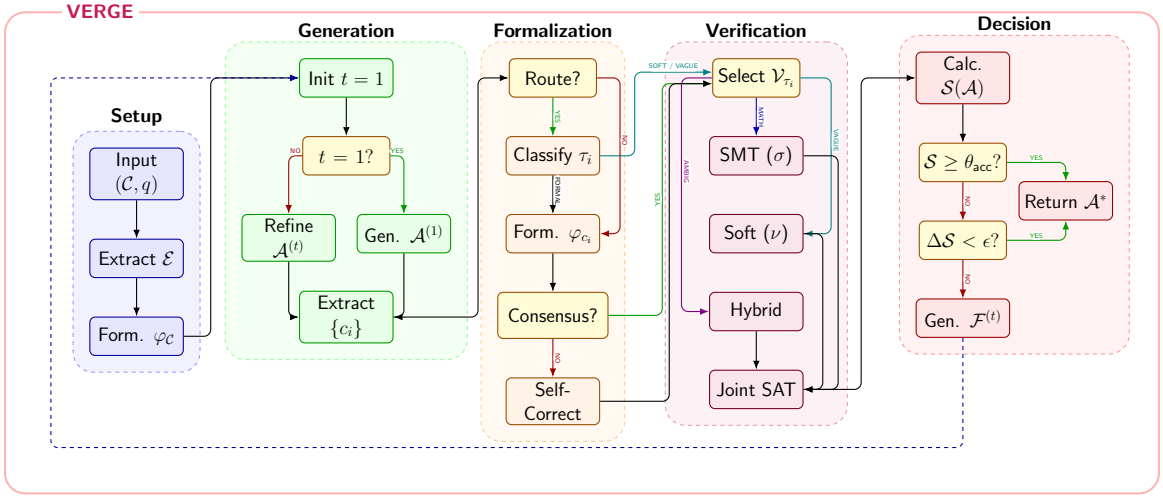

Figure 2: Overview of **VERGE**: The pipeline is structured into five distinct stages: Setup prepares the context $\mathcal{C}$ and entities $\mathcal{E}$; Generation produces and refines answers $\mathcal{A}^{(t)}$ iteratively; Formalization classifies claim types τ_i and generates SMT formulas φ_{c_i}; Verification routes claims to SMT, Soft, or Hybrid verifiers based on semantic type; and Decision computes the aggregate score $\mathcal{S}(\mathcal{A})$ to either accept the answer $\mathcal{A}^*$ or generate feedback $\mathcal{F}^{(t)}$ for the next iteration.

Linear Integer Arithmetic) fragments within the SMT-LIB2 standard. Given the context $\mathcal{C}$ and query q, we first extract entities $\mathcal{E}$ as a set of typed constants. For example, from the statement "All humans above age 18 are adults", we extract entities age of type $\mathbb{Z}$ and $adult$ of type AGE_GROUP, which are declared as uninterpreted constants of the corresponding sorts. We then use generate candidate formulas $\phi = M(c_i, \mathcal{E})$ for each claim c_i over the vocabulary in $\mathcal{E}$. To mitigate the stochastic nature of autoformalization, we generate $K = 3$ candidate formulas $\{\phi_1, \ldots, \phi_K\}$ and check for consensus. Instead of relying on brittle syntactic overlap, we compute consensus based on semantic equivalence (see Appendix C.2 for the formal definition). Two formulas ϕ_a and ϕ_b are deemed equivalent if and only if their bidirectional entailment is valid, which we verify by querying the SMT solver:

$$\text{Equiv}(\phi_a, \phi_b) \iff \text{UNSAT}\left(\Sigma_{\mathcal{E}} \wedge \neg(\phi_a \leftrightarrow \phi_b)\right) \tag{3}$$

where $\Sigma_{\mathcal{E}}$ represents the declarations of constants and uninterpreted functions derived from the entity extraction phase. This procedure constructs a semantic equivalence graph where edges represent proven logical identity.

A formalization is accepted only if a **majority consensus** is reached (i.e., the size of the largest semantic equivalence clique is $\geq \lceil K/2 \rceil$). Additionally, we employ **Round-Trip Translation** (SMT $\rightarrow$ Natural Language) as a semantic sanity check to ensure alignment with the source text. If consensus fails or the confidence (derived from clique size and round-trip similarity) is low, we trigger a self-correction step $\varphi^{\text{new}} = M(\varphi, \ldots)$, constrained such that $\varphi^{\text{new}} \models \varphi$. This ensures the system strictly strengthens (and never weakens) the constraints during refinement. The strengthening constraint is verified by checking the unsatisfiability of $\varphi^{\text{new}} \wedge \neg\varphi$. To ensure decidability, we perform this check over the finite domain of entities $\mathcal{E}$ extracted in the setup phase, effectively reducing the check to propositional logic or quantifier-free first-order logic(QF-FOL).

3.4 Verification Cascade

We employ a hybrid strategy that routes claims to the most rigorous verifier available for their type, prioritizing formal guarantees where applicable.

Semantic Routing (Flexibility Mechanism). We define a routing function $\mathcal{V}_{\tau_i}$:

$$\mathcal{V}_{\tau_i} = \begin{cases} \textit{SMT-Verify} & \text{if } \tau_i \in \{\tau_{\text{M}}, \tau_{\text{L}}, \tau_{\text{T}}\} \\ \textit{Soft-Verify} & \text{if } \tau_i \in \{\tau_{\text{C}}, \tau_{\text{V}}, \tau_{\text{P}}\} \\ \textit{Hybrid} & \text{else (or SMT error)} \end{cases} \tag{4}$$

While SMT provides provable correctness, it cannot natively handle vague predicates (e.g., 'is simi-

lar to') without excessive and fragile axioms. By routing these to Soft-Verify, VERGE preserves logical rigor where possible while preventing the system from falsely treating probabilistic or ambiguous reasoning as logical certainty.

SMT-Based Verification. For logic-amenable claims, we check satisfiability using the Z3 solver. We assign statuses based on rigorous logical tests:

- **Contradictory ($\sigma_{\mathbf{C}}$):** SAT($\varphi_{\mathcal{C}} \wedge \varphi_{c_i}$) = False. The claim violates the context.
- **Entailed ($\sigma_{\mathbf{E}}$):** SAT($\varphi_{\mathcal{C}} \wedge \neg\varphi_{c_i}$) = False. The claim is proven true (proof by contradiction).
- **Possible ($\sigma_{\mathbf{P}}$):** Consistent (SAT($\varphi_{\mathcal{C}} \wedge \varphi_{c_i}$) = True) but not entailed. The context allows the claim but does not force it.
- **Unknown ($\sigma_{\mathbf{U}}$):** Solver timeout or execution error.

Minimal Correction Sets (MCS). For contradictions (σ_{C}), we compute the MCS to generate precise feedback. Let φ_{c_i} be a set of clauses, the subset $S \subset of \varphi_{c_i}$ is a minimal correction set (MCS) if

- SAT($\varphi_{\mathcal{C}} \wedge \varphi_{c_i} \setminus S$) = true
- $\forall S' \subset S$, SAT($\varphi_{\mathcal{C}} \wedge \varphi_{c_i} \setminus S'$) = false

Intuitively, MCS is small subset of clauses to remove to restore satisfibility. This provides actionable guidance to address contradictions (e.g., "remove constraint X") rather than generic error messages. See Appendix B for details about MCS computation.

Soft Verification. For claims unsuitable for SMT, we use an ensemble of LLM judges. We compute a confidence weighted majority vote (using self-reported confidence of the judge) by defining verdicts $v \in$ {Supported, Plausible, Unsupported, Uncertain}.

Constraint: To penalize the lack of formal guarantees, soft-verified claims are capped at a lower maximum score contribution than SMT-verified claims (see §3.5).

Hybrid Verification. The Hybrid strategy acts as a robustness fallback. Claims routed to SMT that fail due to non-logical errors (e.g., syntax errors, undeclared variables, or timeouts) are automatically re-routed to Soft Verification. This prevents the pipeline from stalling on "Unknown" (σ_U) statuses due to correctable formalization issues, allowing the system to degrade gracefully from formal proof to probabilistic consensus.

3.5 Score Aggregation

We compute an aggregated verification score for the entire answer by combining verification results from both soft and hard verification across all atomic claims. Let $\mathcal{A}$ be an answer to query q given context $\mathcal{C}$. Suppose $\mathcal{A}$ is decomposed into atomic claims $c_1, \ldots, c_n$ and verified against q and $\mathcal{C}$ to produce verification results $\sigma_1, \ldots, \sigma_n$, respectively. The verification score $S(\sigma_i)$ for each result is defined as:

$$S(\sigma_i) = \begin{cases} 1.0 & \text{if } \textbf{Entailed}(\sigma_i) \\ 0.9 & \text{if } \textbf{Supported}(\sigma_i) \\ 0.7 & \text{if } \textbf{Possible}(\sigma_i) \\ 0.0 & \text{if } \textbf{Contradictory}(\sigma_i) \text{ or else} \end{cases} \tag{5}$$

These weights reflect verification rigor: formally entailed claims receive the maximum score (1.0), while soft-verified supported claims receive 0.9, ensuring the system favors provable logic over semantic consistency. The final aggregated score $\mathcal{S}(\mathcal{A})$ integrates a variance-based penalty (Eq. 6) to discourage "gaming" the system, where a model might generate claims that are individually confident but mutually contradictory under joint verification:

$$\mathcal{S}(\mathcal{A}) = \bar{S} \cdot \max\left(0.5, 1.0 - \frac{\sigma_S}{\bar{S} + 0.01}\right) \tag{6}$$

where $\bar{S}$ is the mean of the verification scores $\{S(\sigma_1), \ldots, S(\sigma_n)\}$ and σ_S is their standard deviation.

Iterative Refinement. At each iteration t, we generate feedback $\mathcal{F}^{(t)}$ containing: (1) **Unsat Cores & MCS** for contradictions, pinpointing exact logical conflicts; (2) **Joint Conflicts** for mutually incompatible claims; and (3) **Formalization Alerts** for low-confidence mappings. The process repeats until $\mathcal{S}(\mathcal{A}) \geq 0.75$ and `JointSAT=True` (where `JointSAT` is the boolean result of the joint satisfiability check), or until convergence ($\Delta\mathcal{S} < 0.01$). **For Joint Consistency**, soft-verified claims are treated as atomic boolean variables (b_i) within the SMT solver. To capture interactions between soft and hard claims, the context formalization $\varphi_{\mathcal{C}}$ includes **bridging axioms** generated by the LLM (e.g., assertions linking vague predicates like "small" to numerical bounds). This allows the solver to detect if a mathematically proven claim (τ_M) contradicts a commonsense claim (τ_C) (e.g., Mathematical Claim "$X > 10$" vs Commonsense

Claim "X is a small single-digit number" which implies $X < 10$), ensuring holistic consistency even without full formalization of the commonsense component. If claims are not jointly consistent, we compute the MCS over the claims as refinement feedback, signaling a minimal patch to restore joint consistency. Algorithm 1 (Appendix) details the complete refinement procedure.

4 Results

Benchmarking Neuro-Symbolic Reasoning. Table 1 presents a systematic evaluation of VERGE against state-of-the-art inference-time compute (prompting) strategies. To ensure comparable computational budgets, we configure Self-Consistency (SC) with $k = 3$ samples and Self-Refinement (SR) with $n = 3$ iterations with self-critique. Results for established neuro-symbolic baselines (Proof of Thought, LINC, Logic-LM) were computed using their officially released codebase.

Consistent with these parameters, VERGE operates with a maximum budget of $T_{\max} = 3$ iterations. This setting balances accuracy with computational cost, whereas baselines like CoT represent single-pass performance (for convergence analysis up to $T_{\max} = 10$, see Figure 3). Consequently, the improvements shown represent the specific value of *iterative verification-guided refinement*. To distinguish these gains from those achieved by iteration alone, we refer to our ablation study (Table 3). By comparing the full system against the "w/o MCS" and "w/o Routing" variants (both of which also operate iteratively), we isolate the substantial performance contributions of VERGE's verification and feedback mechanisms.

General Performance Trends and Robustness. As evidenced in Table 1, VERGE consistently outperforms standard prompting and existing neuro-symbolic baselines on 5 out of the 6 evaluated benchmarks. The method demonstrates robust scaling across model sizes, notably on the Humanities Last Exam (HLE), where it improves GPT-OSS-120B performance from 14.2% (CoT) to 30.5%. This contrasts with traditional neuro-symbolic baselines (DSB, LogicLM), which suffer from a "translation bottleneck", where invalid SMT specifications cause the solver to fail silently or reject valid reasoning. VERGE overcomes this via its Verification Cascade, which utilizes Minimal Correction Sets (MCS) to isolate specific formalization errors. By iteratively refining the context based on solver feedback (Unsat Cores) rather than discarding the entire proof, VERGE successfully salvages logical entailments that probabilistic baselines miss (see Appendix F).

A notable outlier is the ProofWriter dataset, where Proof of Thought (PoT) retains dominance (98.4% vs. 89.9%). This performance gap highlights a fundamental methodological distinction between monolithic execution and modular verification. PoT approaches reasoning as program synthesis, converting the full context into a single executable artifact to derive the answer in one pass. This is ideal for the rigid, deductive structure of synthetic datasets like ProofWriter. In contrast, VERGE treats reasoning as a **semantic entailment** task, employing a routing mechanism to decompose and verifying individual atomic claims. On purely synthetic tasks, this general-purpose machinery specifically the overhead of claim decomposition and semantic routing, introduces unnecessary complexity compared to PoT's direct solver execution. However, it is precisely this modular flexibility that allows VERGE to generalize to semantically complex domains like Law (AR-LSAT), where a monolithic translation to executable logic often fails due to linguistic ambiguity.

4.1 Compute Parity and Search Baselines

Table 2 in the appendix reports the per-problem cost on GPT-OSS-120B.VERGE at T=1 uses fewer tokens than SR yet beats it on every benchmark, isolating verification from refinement. SC at k=10 matches VERGE's T=3 token budget but trails by 46.6 pts on FOLIO, 1.9 on AR-LSAT, 15.2 on HLE. ToT-BFS (b=5, k=3, T=2) on GPT-OSS-20B reaches 64.7 (FOLIO) and 55.9 (AR-LSAT) versus VERGE's 88.5 and 90.1, at 4× the tokens and 2× the calls; search and verification are composable, not competitive.

4.2 Ablation study: The Role of MCS, Routing, and Feedback systems

Impact of Architectural Components. Table 3 isolates the contributions of VERGE's key components using GPT-OSS-120B. The full pipeline consistently outperforms all ablated variants, confirming that both Minimal Correction Sets (MCS) and Semantic Routing are integral to maximizing VERGE's performance. We observe that MCS is particularly critical for strict constraint satisfaction tasks. On AR-LSAT, removing MCS causes

		Prompting			Neuro-Symbolic Baselines				VERGE (Ours)		
Dataset	**Model**	**CoT**	**SC**	**SR**	**DSB**	**LogicLM**	**LINC**	**PoT**	**w/o MCS**	**w/o Rt**	**Full**
FOLIO	GPT-20B	40.4	52.2	34.0	53.7	29.9	18.6	48.8	73.9	86.7	**89.2**$_{\pm 1.1}$
	GPT-120B	32.0	35.5	42.9	14.0	27.5	47.5	54.2	80.7	81.6	**84.7**$_{\pm 0.9}$
	Sonnet-3.7	70.4	72.4	62.1	44.5	71.6	65.2	58.1	86.7	83.0	**87.9**$_{\pm 0.7}$
ProofWriter	GPT-20B	74.6	71.4	56.8	38.8	35.8	42.8	**94.0**	82.7	85.8	85.2$_{\pm 1.3}$
	GPT-120B	52.4	65.6	67.2	31.4	32.0	76.4	**98.4**	88.7	84.6	89.9$_{\pm 0.8}$
	Sonnet-3.7	71.6	86.6	74.0	42.8	64.7	71.1	**98.2**	90.2	88.0	93.0$_{\pm 0.5}$
ZebraLogic	GPT-20B	67.6	72.2	46.2	44.8	-	-	-	80.9	83.6	**87.3**$_{\pm 1.0}$
	GPT-120B	84.0	77.8	72.2	28.2	-	-	-	88.9	**90.9**	**91.0**$_{\pm 0.6}$
	Sonnet-3.7	52.0	51.0	58.8	48.0	-	-	-	58.0	62.8	**64.8**$_{\pm 1.4}$
AR-LSAT	GPT-20B	81.7	89.1	63.9	59.6	21.7	-	-	82.6	86.8	**89.5**$_{\pm 0.8}$
	GPT-120B	87.8	88.7	84.8	32.2	19.9	-	-	83.0	87.4	**91.7**$_{\pm 0.5}$
	Sonnet-3.7	61.7	58.7	73.5	67.8	31.2	-	-	88.2	87.7	**88.6**$_{\pm 0.9}$
BBEH	GPT-20B	34.4	38.4	28.0	10.0	-	-	-	42.2	43.7	**49.9**$_{\pm 1.5}$
	GPT-120B	38.4	41.8	37.4	20.2	-	-	-	54.1	50.2	**58.9**$_{\pm 1.2}$
	Sonnet-3.7	33.0	29.4	37.8	24.0	-	-	-	40.4	43.5	**45.9**$_{\pm 1.4}$
HLE	GPT-20B	9.6	15.0	8.6	1.4	-	-	-	12.2	13.7	**19.9**$_{\pm 1.1}$
	GPT-120B	14.2	14.0	12.8	6.4	-	-	-	21.0	15.2	**30.5**$_{\pm 1.6}$
	Sonnet-3.7	5.8	5.0	6.8	0.6	-	-	-	16.7	14.7	**17.2**$_{\pm 0.9}$

Table 1: **Comprehensive Performance Analysis.** Results are averaged over 5 independent runs, with error bars (subscript) indicating standard deviation. We compare VERGE against standard prompting (CoT, SC, SR) and neuro-symbolic baselines (DSB, LogicLM, LINC, PoT) across three different backbone models: GPT-OSS-20B, GPT-OSS-120B, and Claude 3.7 Sonnet. VERGE consistently outperforms baselines across diverse domains and model sizes, except on ProofWriter where the specialized PoT method remains dominant. " – " indicates that the baseline does not support the dataset.

Method	LLM calls	Solver calls	Tokens (I+O)	Wall (s)
CoT (1-pass)	1	0	2.1K	3.2
SC (k=3)	3	0	6.3K	8.7
SC (k=10)	10	0	21.0K	28.3
SR (n=3)	6	0	11.4K	15.3
PoT	2	1	4.8K	5.4
LINC	3	1	5.1K	6.1
ToT-BFS	40	0	63.0K	71.2
VERGE (T=1)	∼8	∼5	8.4K	11.8
VERGE (T=3)	∼17	∼15	18.2K	32.6

Table 2: Per-problem cost on GPT-OSS-120B, averaged across six benchmarks. Solver calls are CPU-only (<200ms each). VERGE at T=1 is cheaper than SR yet outperforms it; SC at k=10 matches VERGE's token budget but not its accuracy.

a sharp drop from 91.7% to 83.0%. This indicates that while the solver can detect contradictions, the model struggles to resolve complex scheduling conflicts without the precise, minimal deletion guidance provided by the MCS. Conversely, Semantic Routing proves advantageous for open-ended and commonsense reasoning. Removing the routing mechanism, thereby forcing all claims into formal logic, substantially impacts HLE and BBEH, with HLE scores halving from 30.5% to 15.2%. This lends support to the "Formalization Barrier" hypothesis (discussed in §4.4): attempting to formalize vague or fuzzy predicates leads to brittle systems that reject valid reasoning (e.g., a solver might reject a valid commonsense claim due to a missing axiom). Routing allows VERGE to fallback to soft verification when necessary, a benefit most pronounced in domains with high ambiguity like HLE.

Feedback Granularity. The right side of Table 3 demonstrates the value of high-resolution feedback. There is a clear hierarchy of performance correlated with feedback specificity. Unsat-Core Only feedback, which identifies conflicting constraints but does not prescribe a fix, lags behind the full model by 10.9% on average. Minimal Solver Feedback (MSF) performs worst, with a 15.3% average drop. This confirms that simply telling an LLM "you are wrong (UNSAT)" (binary feedback) is insufficient for complex reasoning; the model requires

	Architectural Components				Feedback Granularity	
Dataset	**Full**	**w/o MCS**	**w/o Routing**	**Soft-Only**	**Unsat-Core Only**	**MSF**
FOLIO	**84.7**	80.7	81.6	80.8	79.3	76.5
ProofWriter	**89.9**	88.7	84.6	75.8	82.4	80.1
ZebraLogic	**91.0**	88.9	90.9	70.2	85.3	72.5
AR-LSAT	**91.7**	83.0	87.4	84.6	89.4	89.2
BBEH	**58.9**	54.1	50.2	41.7	42.7	40.1
HLE	**30.5**	21.0	15.2	13.2	23.2	19.8
Avg. Drop	–	-6.8%	-8.3%	-22.8%	-10.9%	-15.3%

Table 3: **Ablation Study on Component Contributions (GPT-OSS-120B).** We isolate the impact of Minimal Correction Subsets (MCS), Semantic Routing (Rt), and the SMT Solver itself. *Full* represents the complete VERGE pipeline. *w/o Routing* forces SMT verification for all claims. *Soft-Only* removes the SMT solver entirely, relying solely on LLM consensus. *Unsat-Core-Only* and *MSF* vary the granularity of the feedback.

the actionable, structural guidance that VERGE provides.

To address the converse hypothesis whether formal verification is necessary at all we evaluated a **Soft-Only** variant where all claims are routed to the LLM consensus mechanism, bypassing the SMT solver entirely. As shown in Table 3, this results in the most significant performance degradation across the board (average drop: -22.8%). The impact is most severe on strict constraint satisfaction tasks like ZebraLogic (91.0% $\rightarrow$ 70.2%) and ProofWriter (89.9% $\rightarrow$ 75.8%), confirming that while soft verification is useful for ambiguity, it lacks the precision required to resolve complex logical dependencies. This effectively validates the necessity of the hybrid approach: VERGE requires SMT for precision and Semantic Routing for flexibility.

Semantic Router Reliability. To validate our routing mechanism, we evaluated the classifier on a stress-test dataset (also see Appendix C.3) of $N = 54$ claims, including adversarial edge cases such as idioms (e.g., "gave 110%") and vague quantifiers. The router achieved an overall **accuracy of 94%**, with strong discrimination between categories (SMT: F1=0.93; Soft: F1=0.95). Crucially, the error patterns reveal a *safe failure mode*: only 1 out of 32 soft claims was misrouted to SMT; a recoverable error that triggers autoformalization fallback. The high recall for Soft claims (0.97) and precision for SMT claims (0.95) confirm the system effectively shields the solver from ambiguity while preserving logical rigor.

Scoring sensitivity. The weights (1.0,0.9,0.7,0.0) encode an ordinal hierarchy, not a tuned operating point. Varying the Supported weight w_S on GPT-OSS-120B (Table 4) yields worst-case -1.3% and mean $|\Delta|<0.6\%$; collapsing w_S=1.0 slightly hurts, confirming the discount appropriately favors entailment over consensus.

4.3 Efficiency and Convergence

Fig. 3 (in Appendix) reveals a striking divergence in refinement dynamics. VERGE achieves monotonic improvement across all six datasets (Kendall's $\tau = 1.0$, $p < 0.001$), with average convergence at iteration 6.2 ($\sigma = 1.3$), beyond T=10 (Table 13) , the gains in performance are negligible ($< 0.3\%$ across all benchmarks), validating our convergence criterion ($\Delta S < \epsilon = 0.01$). In contrast, probabilistic self-refinement exhibits what we term the **correlation cliff** phenomenon: performance systematically degrades in 85.2% of trials ($\chi^2 = 26.7$, $p < 0.001$), characterized by a strong negative correlation between iteration and accuracy ($\tau = -0.84$ average, $p < 0.001$). This could be due to self-refinement (without formal verification) introducing hallucinations where the model "corrects" valid reasoning into invalid states. This convergence analysis provides the strongest statistical evidence for VERGE's value: while final accuracy gains are modest (average +9.3 points), the observed consistent monotonic improvement has high practical value in production systems where reliability matters more than peak performance.

4.4 Model Scalability and The Formalization Barrier

We observe that effectiveness is contingent on the model's ability to produce syntactically valid SMT code. We term this threshold the Formalization Bar-

rier. Our tested model under 20B parameters (GPT-OSS-20B) struggles, achieving only $\sim$30% syntax validity. In this regime, the solver acts merely as a spell-checker. However, models in the 120B+ and frontier regime (GPT-OSS-120B, Sonnet) cross the barrier (>90% validity). Here, solver feedback shifts from generic error messages to semantic logical contradictions, enabling VERGE's MCS mechanism to perform actual reasoning repairs. This suggests that neuro-symbolic verification is a capability that emerges with scale, and VERGE is uniquely positioned to leverage the next generation of highly capable reasoning models.

5 Conclusions

We present VERGE, a neuro-symbolic framework combining LLMs with SMT solvers for verified reasoning via iterative refinement. Our approach introduces three key innovations: (1) high-fidelity formalization via multi-sample consensus, (2) semantic routing that balances symbolic solvers with soft verification, and (3) actionable feedback via Minimal Correction Subsets (MCS) for precise error localization. Evaluation shows VERGE excels in formal reasoning and achieves convergence across all datasets, contrasting with the degradation often observed in probabilistic self-refinement. Our analysis identifies a "Formalization Barrier" at the 70B+ parameter scale where semantic verification becomes viable. By bridging neural generation with symbolic reasoning, VERGE provides a practical step toward trustworthy AI with provable correctness where logic permits.

6 Limitations

Computational Overhead. VERGE incurs significantly higher latency than single-pass generation. Each iteration requires claim decomposition, multiple formalization attempts ($K = 3$), consensus computation, SMT solver calls, and feedback generation. For problems with $n > 20$ atomic claims, the pipeline requires 15-30 seconds per iteration compared to > 2 seconds for standard Chain-of-Thought prompting. While our greedy MCS approximation reduces complexity from exponential $O(2^n)$ to linear $O(n \times \text{SAT})$, the multiplicative overhead remains substantial. This latency-accuracy trade-off limits deployment in interactive applications requiring sub-second response times (e.g., conversational AI, real-time decision support).

Formalization Barrier and Access Inequality. Our analysis reveals models under 20B parameters achieve only $\sim$30% formalization validity, restricting the solver to syntax checking rather than semantic verification. This creates a capability threshold where only organizations with access to frontier models ($\geq$70B parameters) can leverage VERGE's full potential. Additionally, restricting to decidable logics (QF-UF, LIA) sacrifices expressiveness claims requiring universal quantification, non-linear arithmetic, or recursive definitions cannot be formally verified and must fall back to soft verification. This undermines the framework's promise of provability for complex mathematical or algorithmic reasoning.

7 Ethical Considerations

Logical Correctness is not Ethical Correctness. VERGE verifies internal consistency and logical entailment, not moral soundness or factual truth. The system could formally prove harmful reasoning—discriminatory policies that satisfy legal constraints, exploit chains in cybersecurity, or conspiracy theories with internally consistent logic but false premises. SMT solvers are fundamentally value-neutral tools. Deploying VERGE in high-stakes domains (legal, medical, military decision-making) requires additional ethical oversight layers: premise provenance tracking, factuality verification orthogonal to logic, and human expert review for consequential decisions.

Risk of Overconfidence and Misplaced Trust. Labeling outputs as "verified" may induce false confidence in users unfamiliar with the distinction between formal and soft verification. Our scoring system assigns high scores (0.9) to soft-verified commonsense claims that lack mathematical guarantees. More critically, if autoformalization misrepresents a claim's semantics producing syntactically valid but semantically incorrect SMT code the solver verifies the wrong statement, creating "verified hallucinations." Users may over-rely on verification badges without understanding rigor gradations. Clear interface design distinguishing "formally proven" (σ_E) from "consensus-supported" (ν_S) claims is essential but insufficient if users lack technical literacy.

References

Anthropic. 2025. Introducing Claude Opus 4.5. Accessed: 2025-12-19.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. 2023. ProofNet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*.

Fahiem Bacchus and George Katsirelos. 2015. Finding a collection of MUSes incrementally. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pages 35–44. Springer.

Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. 2009. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185, pages 825–885. IOS Press.

Anton Belov, Mikoláš Janota, Inês Lynce, and Joao Marques-Silva. 2012. On computing minimal correction subsets. In *International Joint Conference on Artificial Intelligence*, pages 615–622.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Weicong Chen, Vikash Singh, Zahra Rahmani, Debargha Ganguly, Mohsen Hariri, and Vipin Chaudhary. 2025. K4: Online log anomaly detection via unsupervised typicality learning. In *2025 IEEE 32nd International Conference on High Performance Computing, Data, and Analytics (HiPC)*, pages 96–107.

Alonzo Church. 1936. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58(2):345–363.

Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 33–43.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. 2025. Vericot: Neuro-symbolic chain-of-thought validation via logical consistency checks. *arXiv preprint arXiv:2511.04662*.

Debargha Ganguly, Srinivasan Iyengar, Vipin Chaudhary, and Shivkumar Kalyanaraman. 2024. PROOF OF THOUGHT : Neurosymbolic program synthesis allows robust and interpretable reasoning. In *The First Workshop on System-2 Reasoning at Scale, NeurIPS'24*.

Debargha Ganguly, Sreehari Sankar, Biyao Zhang, Vikash Singh, Kanan Gupta, Harshini Kavuru, Alan Luo, Weicong Chen, Warren Morningstar, Raghu Machiraju, and Vipin Chaudhary. 2026. Trust the typical. *Preprint*, arXiv:2602.04581.

Debargha Ganguly, Vikash Singh, Sreehari Sankar, Biyao Zhang, Xuecen Zhang, Srinivasan Iyengar, Xiaotian Han, Amit Sharma, Shivkumar Kalyanaraman, and Vipin Chaudhary. 2025. Grammars of formal uncertainty: When to trust LLMs in automated reasoning tasks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Artur d'Avila Garcez, Marco Gori, Luis C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. 2019. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–632.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2024. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations (ICLR)*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, and 1 others. 2024. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *Advances in Neural Information Processing Systems*, 34:6901–6914.

Jie Huang, Xinyun Gu, Lisha Shen, Weijia Shi, Qizhe Yuan, Jiawei Wang, Xianjun Zhao, Keze Zhou, Linjun Zhang, Jianlong Yu, and 1 others. 2024. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*.

Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothee Lacroix, Yuhuai Wu, and Guillaume Lample. 2022. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*.

Ryo Kamoi, Olga Golovneva, Esin Durmus, Asli Celikyilmaz, and Yejin Cao. 2024. Can LLMs critique and correct their own outputs? *arXiv preprint arXiv:2406.01297*.

Henry Kautz. 2022. The third AI summer: AAAI Robert S. Engelmore memorial lecture. *AI Magazine*, 43(1):105–125.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Yuanzhu Peter Chen, and 1 others. 2025. Big-bench extra hard. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26473–26501.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, and 1 others. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 3843–3857.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu, and Shuming Shi. 2023. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*.

Mark H Liffiton and Ammar Malik. 2008. Algorithms for computing minimal unsatisfiable subsets of constraints. *Journal of Automated Reasoning*, 40(1):1–33.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. 2025. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *arXiv preprint arXiv:2301.13379*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *International Conference on Learning Representations*.

Joao Marques-Silva, Federico Heras, Mikoláš Janota, Alessandro Previti, and Anton Belov. 2013. On computing minimal correction subsets. In *International Joint Conference on Artificial Intelligence*, pages 615–622.

Lachlan McGinness and Peter Baumgartner. 2024. Automated theorem provers help improve large language model reasoning. *arXiv preprint arXiv:2408.03492*.

Meta AI. 2025. The llama 4 herd: The beginning of a new era of natively multimodal intelligence. Technical report, Meta. Technical Report.

Antonio Morgado, Federico Heras, Mark Liffiton, Jordi Planes, and Joao Marques-Silva. 2013. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534.

Alexander Nadel, Vadim Ryvchin, and Ofer Strichman. 2014. Efficient MUS extraction with resolution. In *Formal Methods in Computer-Aided Design*, pages 197–200. IEEE.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. 2023. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5153–5176.

OpenAI. 2025. Introducing OpenAI o3 and o4-mini. Accessed: 2025-12-19.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *Preprint*, arXiv:2305.12295.

Long Phan, Alice Gatti, Ziwen Han, Nathaniel Li, Josephina Hu, Hugh Zhang, Chen Bo Calvin Zhang,

Mohamed Shaaban, John Ling, Sean Shi, and 1 others. 2025. Humanity's last exam. *arXiv preprint arXiv:2501.14249*.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. 2022. Formal mathematics statement curriculum learning. In *International Conference on Learning Representations*.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36.

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

Vikash Singh, Debargha Ganguly, Haotian Yu, Chengwei Zhou, Prerna Singh, Brandon Lee, Vipin Chaudhary, and Gourav Datta. 2026. Toward guarantees for clinical reasoning in vision language models via formal verification. *Preprint*, arXiv:2602.24111.

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

Alan M. Turing. 1936. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Nengbo Wang, Tuo Liang, Vikash Singh, Chaoda Song, Van Yang, Yu Yin, Jing Ma, Jagdip Singh, and Vipin Chaudhary. 2026. Hugrag: Hierarchical causal knowledge graph design for rag. *Preprint*, arXiv:2602.05143.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. In *International Conference on Learning Representations*.

w_S	FOLIO	PW	ZL	AR-LSAT	BBEH	HLE
0.70	83.9	89.2	90.6	90.8	57.4	29.1
0.80	84.3	89.5	90.8	91.2	58.1	29.7
0.90	**84.7**	**89.9**	**91.0**	**91.7**	**58.9**	**30.5**
0.95	84.4	89.7	90.9	91.4	58.5	30.2
1.00	84.0	89.3	90.5	91.0	57.8	29.4

Table 4: Sensitivity to w_S (others fixed) on GPT-OSS-120B.

Jie Weng, Oumaima Kitouni, Fanghua Shi, Neel Gupta, Preetum Nakkiran, Boaz Barak, Sham Chaudhuri, and Shunyu Yao. 2023. Can large language models self-verify? *arXiv preprint arXiv:2310.11638*.

Wang Yang, Debargha Ganguly, Xinpeng Li, Chaoda Song, Shouren Wang, Vikash Singh, Vipin Chaudhary, and Xiaotian Han. 2026. Mid-think: Training-free intermediate-budget reasoning via token-level triggers. *Preprint*, arXiv:2601.07036.

Osama Zafar, Mina Namazi, Yuqiao Xu, Youngjin Yoo, and Erman Ayday. 2026. A user-centric, privacy-preserving, and verifiable ecosystem for personal data management and utilization. In *Computer Security – ESORICS 2025*, pages 395–414, Cham. Springer Nature Switzerland.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Conference on Uncertainty in Artificial Intelligence*, pages 658–666.

Lintao Zhang and Sharad Malik. 2003. Extracting small unsatisfiable cores from unsatisfiable boolean formula. In *International Conference on Theory and Applications of Satisfiability Testing*, volume 2003.

Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. 2021. Ar-lsat: Investigating analytical reasoning of text. *arXiv preprint arXiv:2104.06598*.

A VERGE: Algorithm / Pseudo code

B Rationale for Greedy MCS Computation

To provide actionable feedback when a generated answer is contradictory, VERGE must identify a Minimal Correction Subset (MCS): a minimal subset of atomic claims that, if removed, restores consistency. Formally, finding an MCS is equivalent to finding the complement of a Maximal Satisfiable Subset (MSS).

While exact algorithms ensure the theoretically smallest removal set, they require exploring the combinatorial search space of all subsets, leading to exponential time complexity ($O(2^n)$). For an

Algorithm 1 Iterative Verification and Refinement (VERGE)

Initialize $\mathcal{A}^{(0)} \leftarrow$ null, $\mathcal{S}^{(0)} \leftarrow 0$
Extract entities: $\mathcal{E} \leftarrow \text{EntityExtraction}(\mathcal{C}, q)$
Formalize context: $\varphi_{\mathcal{C}} \leftarrow \text{AutoFormalize}(\mathcal{C}, \mathcal{E})$
if $\neg\text{SAT}(\varphi_{\mathcal{C}})$ **then** $\varphi_{\mathcal{C}} \leftarrow \text{RefineContext}(\varphi_{\mathcal{C}})$ {Handle Translation Bottleneck}
best_answer $\leftarrow$ null, best_score $\leftarrow 0$
for $t = 1$ to $T_{\max}$ **do**
  Generate answer: $\mathcal{A}^{(t)} \leftarrow \text{LLM}(\mathcal{C}, q, \mathcal{F}^{(t-1)})$
  Extract claims: $\{c_i^{(t)}\} \leftarrow \text{Decompose}(\mathcal{A}^{(t)})$
  Classify claims: $\tau_i \leftarrow \text{SemanticRouter}(c_i^{(t)})$
  if $\text{IsFormal}(\tau_i)$ **then** $\varphi_{c_i}, \alpha_i \leftarrow \text{FormalizeConsensus}(c_i^{(t)})$
  else $\varphi_{c_i} \leftarrow \text{BooleanAbstract}(c_i^{(t)}), \alpha_i \leftarrow \text{SoftConf}(c_i^{(t)})$ {Boolean var for Soft claims}
  Verify individual claims: $s_i \leftarrow \text{RouteAndVerify}(c_i^{(t)}, \tau_i, \varphi_{c_i})$
  Identify valid subset: $V = \{c_i \mid s_i \text{ is not Contradictory}\}$
  Verify joint consistency: JointSAT, Core $\leftarrow \text{Solver}(\varphi_{\mathcal{C}} \wedge \bigwedge_{c_k \in V} \varphi_{c_k})$
  Compute score: $\mathcal{S}^{(t)} \leftarrow \text{AggScore}(\{s_i\}, \text{JointSAT}, \{\alpha_i\})$
  if $\mathcal{S}^{(t)} >$ best_score **then**
    Update best score: best_score $\leftarrow \mathcal{S}^{(t)}$, best_answer $\leftarrow \mathcal{A}^{(t)}$
  end if
  if $\mathcal{S}^{(t)} \geq \tau_{\text{acc}}$ **and** JointSAT **then**
    return $\mathcal{A}^{(t)}$
  end if
  if Convergence detected ($\Delta\mathcal{S} < \epsilon$) **then**
    return best_answer
  end if
  {**Feedback Generation**}
  $\mathcal{F}_{\text{indiv}} \leftarrow \text{GetIndividualErrors}(\{c_i\} \setminus V)$
  if $\neg$JointSAT **then**
    {Sorted Greedy MCS on Valid Claims}
    Sort V by confidence α **descending**
    MSS $\leftarrow \emptyset$, MCS $\leftarrow \emptyset$
    for $c_k \in V$ **do**
      if $\text{Solver}(\varphi_{\mathcal{C}} \wedge \text{MSS} \wedge \varphi_{c_k})$ is SAT **then**
        MSS $\leftarrow$ MSS $\cup \{c_k\}$
      else
        MCS $\leftarrow$ MCS $\cup \{c_k\}$ {Conflict found}
      end if
    end for
    $\mathcal{F}_{\text{joint}} \leftarrow \text{FormatFeedback}(\text{MCS}, \text{Core})$
  else
    $\mathcal{F}_{\text{joint}} \leftarrow \text{IdentifyWeakClaims}(V)$
  end if
  $\mathcal{F}^{(t)} \leftarrow \mathcal{F}_{\text{indiv}} \cup \mathcal{F}_{\text{joint}}$
end for
return best_answer

LLM inference pipeline where latency is critical, this approach is computationally intractable.

The Greedy Approximation. To address this, we employ a linear-scan approximation (c.f. Algorithm 1; (Marques-Silva et al., 2013)). The algorithm iterates through atomic claims $c_1, \ldots, c_n$ sequentially. For each claim, it checks if adding it to the current set of consistent claims preserves satisfiability (SAT). If $\text{SAT}(\text{Context} \wedge \text{Current\_Set} \wedge c_i)$ holds, c_i is kept; otherwise, it is marked for removal.

This reduces the complexity from exponential to linear ($O(n \times \text{SAT})$). As shown in Table 5, the computational disparity becomes prohibitive even for moderate claim counts ($n = 20$).

The trade-off is that the greedy approach is order-dependent and may not find the global maximum subset (e.g., it might recommend deleting 3 claims when deleting 2 would suffice). However, in the context of Iterative Refinement, **actionability prioritizes optimality**. Getting a "valid enough" correction signal in seconds is practically superior to waiting hours for a mathematically perfect one. The greedy MCS provides a specific, consistent sub-context that guides the LLM effectively, even if it is sub-optimal.

Stability & Order Dependence: Greedy MCS is order-dependent and to maximize the retention of

high-quality reasoning, we sort claims by their individual verification confidence (descending) before the greedy pass. This prioritizes keeping 'Entailed' or High-Confidence Soft claims and suggests modifying the weakest links first. In preliminary tests, this sorting reduced feedback variance significantly compared to random ordering.

C Semantic Claim Type Definitions

To ensure the **Semantic Router** directs claims to the appropriate verification strategy (as described in Section 2.2 and Eq. 2), we formally define the six semantic categories (τ) used by VERGE in Table (6).

The routing decision is binary based on these types: *Hard Verification* (SMT) is applied to deterministically provable claims (τ_M, τ_L, τ_T), while *Soft Verification* (Consensus) is applied to claims requiring world knowledge or subjective interpretation (τ_C, τ_V, τ_P).

C.1 Technical Implementation Details

To ensure reproducibility and formal rigor, we define the specific mechanisms used for claim decomposition, autoformalization, and consensus scoring.

C.1.1 Atomic Decomposition Strategy

We define an *Atomic Claim* c_i as the minimal semantic unit that carries a truth value independent of other claims. Given a generated answer $\mathcal{A}$, we employ a zero-shot decomposition function $f_{decomp} : (\mathcal{C}, \mathcal{A}) \to \{c_1, \ldots, c_n\}$. To prevent context loss, we enforce that every c_i must be *self-contained* (i.e., resolving pronouns like "he" to specific entities such as "Felix"). This is implemented via a structure-enforcing prompt that outputs a JSON list of claims, preventing the hallucination of non-existent dependencies.

C.1.2 Formalization into SMT-LIB2

For claims classified as τ_{Logic} or τ_{Math}, we target the **QF_UF** (Quantifier-Free Uninterpreted Functions) and **LIA** (Linear Integer Arithmetic) logics within the SMT-LIB2 standard. The translation process $\mathcal{T} : c_i \to \varphi_i$ operates under strict syntactic constraints:

1. **Type Declaration:** All entities extracted in the Setup phase are declared as uninterpreted constants of a generic sort `Object` or specific sorts (e.g., `Person`, `Number`) where applicable.

2. **Predicate Mapping:** Relations are mapped to boolean functions. For example, "*Felix eats food*" maps to `(assert (Eats Felix Food))`.

3. **Quantifier Handling:** While SMT solvers support quantifiers ($\forall, \exists$), they often lead to undecidability. Where possible, we instantiate universals over the finite set of extracted entities $\mathcal{E}$ to maintain decidability.

C.2 Formal Definition of Semantic Equivalence

In Section 3.3, we introduce **Semantic Equivalence Checking** to compute consensus among candidate formalizations. Here, we precisely define the logical framework and the equivalence condition.

C.2.1 Logical Framework

VERGE operates within the framework of **Many-Sorted First-Order Logic** (specifically, the SMT-LIB2 standard). However, to ensure decidability and efficiency, we restrict the formalization to specific fragments:

- **QF_UF** (Quantifier-Free Uninterpreted Functions): Used for abstract relationships and categorical claims.

- **QF_LIA** (Quantifier-Free Linear Integer Arithmetic): Used for numerical constraints and temporal sequencing.

- **Finite Domain Quantification**: Where universal quantifiers ($\forall$) are unavoidable in natural language, we instantiate them over the finite set of extracted entities $\mathcal{E}$ (see §3.1), effectively reducing them to conjunctions in Propositional Logic.

C.2.2 Equivalence Definition

Let $\mathcal{C}$ be the problem context and $\mathcal{E}$ be the set of extracted entities. We define a **Signature** $\Sigma_{\mathcal{E}} = (\mathcal{S}, \mathcal{F}, \mathcal{P})$ consisting of:

- $\mathcal{S}$: A set of sorts (types) derived from the context (e.g., `Student`, `Day`).

- $\mathcal{F}$: A set of function symbols (e.g., `age`: `Person` $\to$ `Int`).

- $\mathcal{P}$: A set of predicate symbols (e.g., `gives_report`: `Student` $\times$ `Day` $\to$ `Bool`).

Feature	MCS	Greedy MCS (Ours)
Optimality	Guaranteed Minimal	Approximation
Complexity	$O(2^n \times \text{SAT})$	$O(n \times \text{SAT})$
Space	Exponential	Linear
Estimated SAT Calls required:		
$n = 10$ atoms	1,024	**10**
$n = 20$ atoms	1,048,576	**20**
$n = 30$ atoms	$\sim$1 Billion	**30**

Table 5: Computational comparison between exact and greedy MCS strategies. For real-time LLM reasoning tasks (where n often exceeds 20), the exact approach is infeasible, whereas the greedy approach scales linearly.

Two candidate formulas ϕ_a and ϕ_b generated by the LLM are defined as **Semantically Equivalent** modulo $\Sigma_{\mathcal{E}}$ if and only if they share the same truth value in every possible interpretation $\mathcal{I}$ consistent with the signature:

$$\phi_a \equiv_\Sigma \phi_b \iff \forall \mathcal{I} \models \Sigma_{\mathcal{E}}, \quad [\![\phi_a]\!]^{\mathcal{I}} = [\![\phi_b]\!]^{\mathcal{I}} \tag{7}$$

C.2.3 Verification Implementation

In practice, we verify this condition using the SMT solver (Z3) by checking the unsatisfiability of the negated biconditional. We construct a query Q:

$$Q = \text{Declare}(\Sigma_{\mathcal{E}}) \wedge \neg(\phi_a \leftrightarrow \phi_b) \tag{8}$$

If SOLVE(Q) returns `UNSAT`, it implies there is no model where ϕ_a and ϕ_b differ; thus, they are logically equivalent.

This approach is robust to:

1. **Syntactic Permutation:** $(A \wedge B) \equiv (B \wedge A)$.
2. **Variable Renaming:** $\forall x.P(x) \equiv \forall y.P(y)$ (handled via canonicalization or finite instantiation).
3. **Tautological Variance:** $(P \rightarrow Q) \equiv (\neg P \vee Q)$.

Unlike string matching or embedding similarity, this provides a mathematically rigorous guarantee that the consensus candidates represent the exact same logical constraint.

C.3 Semantic Router Stress-Test Dataset

To rigorously evaluate the Semantic Router (Section 4.2), we constructed a diverse evaluation set of $N = 54$ atomic claims designed to probe the decision boundary between formalizable logic and natural language ambiguity. The dataset was composed of three distinct subsets:

1. **Logic & Math (Standard):** 22 claims sampled from FOLIO and AR-LSAT containing explicit logical operators, arithmetic constraints, and temporal sequences (e.g., "The meeting is at 2 PM," "x is greater than 70").
2. **Commonsense & Vague (Standard):** 20 claims involving subjective predicates, probability, or world knowledge not strictly definable in SMT (e.g., "It is likely to rain," "The painting is beautiful").
3. **Adversarial Edge Cases:** 12 manually crafted claims designed to trick keyword-based classifiers. These include:
   - **Numeric Idioms:** Phrases containing numbers that are not mathematical (e.g., "He gave 110% effort," "She was on cloud nine").
   - **Logical Homonyms:** Words like "follows" or "implies" used rhetorically rather than deductively (e.g., "It follows that he was angry").
   - **Perturbed Contexts:** Claims derived from logic puzzles where constraints were inverted to test stability (e.g., swapping "banned" for "permitted" to verify routing consistency remains robust under contradiction).

Ground truth labels (SMT-Amenable vs. Soft-Verification) were manually assigned by two authors with high inter-annotator agreement ($\kappa > 0.9$). Table 7 provides examples of these challenging edge cases.

D Adversarial Robustness and Context Faithfulness

A known failure mode of reasoning models is the "Faithfulness Gap," where the model ignores provided context in favor of its parametric mem-

Type	Definition	Example	Router
Group A: SMT-Amenable Claims (Hard Verification)			
τ_M	**Mathematical**: Claims involving arithmetic operations, algebraic constraints, numerical comparisons, or unit conversions.	*"x is a prime number greater than 5 and less than 20."*	SMT
τ_L	**Logical**: Claims involving formal entailment, set theory inclusion, boolean logic, or syllogistic structure.	*"All entities in set A must also belong to set B."*	SMT
τ_T	**Temporal**: Claims involving linear sequencing, specific timestamps, duration, or precedence constraints.	*"The event occurred 3 days after the signing."*	SMT
Group B: LLM-Consensus Claims (Soft Verification)			
τ_C	**Commonsense**: Claims relying on general world knowledge, causality, or physical properties not strictly definable by axioms.	*"Glass typically shatters when dropped on concrete."*	Soft
τ_V	**Vague**: Claims involving subjective predicates, qualitative descriptors, or attributes lacking a boolean truth value.	*"The painting is considered beautiful by most critics."*	Soft
τ_P	**Probabilistic**: Claims explicitly stating uncertainty, likelihood, or future predictions without deterministic data.	*"It is likely to rain tomorrow given the clouds."*	Soft

Table 6: Definitions of Semantic Claim Types (τ) used in the Routing Module. Claims in Group A are autoformalized into SMT-LIB2 logic; Claims in Group B are verified via multi-model consensus.

Category	Example Claim	Correct Route
Numeric Idiom	"The team gave **110%** effort."	Soft
Numeric (Real)	"The score was **110** points."	SMT
Logical Homonym	"It **follows** that he felt sad."	Soft
Logical (Real)	"It **follows** that $A \subset B$."	SMT
Vague Quantifier	"Many people attended."	Soft
Exact Quantifier	"More than 50 people attended."	SMT

Table 7: Examples from the Semantic Router Stress-Test Dataset, highlighting adversarial pairs used to test the router's discrimination capabilities.

ory (e.g., refusing to accept a counterfactual premise like "Cats are not mammals"). To evaluate VERGE's robustness to such logical perturbations, we conducted an adversarial probe using paired samples where the logical constraints were inverted or perturbed (e.g., changing "banned" to "permitted", or swapping temporal order).

As shown in Table 8, VERGE demonstrated **100% robustness** across the tested categories. Notably, the system maintained high verification scores (avg. 0.91) on the adversarial samples. This indicates that VERGE successfully overcame the "prior bias" of the LLM; rather than hallucinating the standard answer or failing verification, the MCS-guided feedback loop forced the model to align its generated answer with the *perturbed* context. This confirms that our formal verification constraint effectively enforces context-faithfulness over parametric memory.

E Reproducibility & Implementation Details

To ensure the replicability of VERGE, we provide detailed specifications of our prompt engineering, hyperparameters, and computational infrastructure.

E.1 Hyperparameters and Configuration

We utilize all models in Table 1 (via AWS Bedrock) as the backbone LLM for all generation and refinement steps due to its strong reasoning capabilities.

Perturbation Type	Original	Perturbed	Status	Adaptation
Logic Inversion	✓	✓	**Safe**	Correct Flip
Numeric Threshold	✓	✓	**Safe**	Correct Flip
Sequence Swap	✓	✓	**Safe**	Correct Flip
Mutually Exclusive	✓	✓	**Safe**	Correct Flip

Table 8: Adversarial Robustness results. "Safe" indicates the system either rejected the invalid premise or, in these cases, successfully adapted its reasoning to the counterfactual context (faithfulness), achieving high verification scores (> 0.9) on the perturbed inputs.

We use **Z3 (version 4.12.2)** as the SMT solver. Table 9 details the specific configuration used across all experiments.

Parameter	Value
LLM Generation	
Temperature (T_{gen})	1.0
Top-P	0.99
Max Output Tokens	10,000
Thinking Budget	4,000
VERGE *Pipeline*	
Max Iterations (T_{max})	3
Consensus Samples (K)	3
Consensus Threshold	0.70
Soft-Verify Judges (N)	5
Acceptance Score (τ_{acc})	0.75
Convergence ϵ	0.01
Judge Model	Sonnet-3.7

Table 9: Hyperparameters for the VERGE pipeline. These values were held constant across all datasets (FOLIO, ZebraLogic, etc.) to demonstrate robustness.

E.2 Prompt Templates

We employ a modular prompting strategy. The system prompts for the key components of the pipeline are provided below.

E.2.1 Claim Decomposition & Classification

This prompt breaks the raw generation into atomic units and routes them to the appropriate verifier.

System Prompt: Decomposition

```
You are an expert logician. Analyze the provided Answer and decompose it into atomic, verifiable claims. For each claim, assign a Semantic Type based on the following definitions:
  • MATHEMATICAL: Involves arithmetic, algebra, or numerical constraints.
  • LOGICAL: Involves first-order logic, set theory, or strict deduction.
  • COMMONSENSE: Involves general world knowledge or vague predicates.
Output format: JSON list of objects {"text": string,"type": string}.
```

E.2.2 Autoformalization (Natural Language →SMT-LIB2)

This prompt translates text into code. Note the instruction `(set-logic ALL)`, which allows the solver to handle mixed integer arithmetic and uninterpreted functions dynamically.

System Prompt: Formalization

```
Translate the following natural language context and claims into SMT-LIB2 format for the Z3 solver.
Constraints:
  1. Declare all sorts (e.g., Person, Object) explicitly.
  2. Use (set-logic ALL) to support mixed arithmetic and logic.
  3. Do not include (check-sat) commands; simply assert the facts.
  4. If a statement is ambiguous, use uninterpreted functions.
Input Context: [CONTEXT]
Input Claim: [CLAIM]
Output: <smt> ... code ... </smt>
```

E.2.3 Feedback Injection (Refinement Step)

When the SMT solver returns `UNSAT`, we calculate the Minimal Correction Set (MCS) and feed it back to the model using this template.

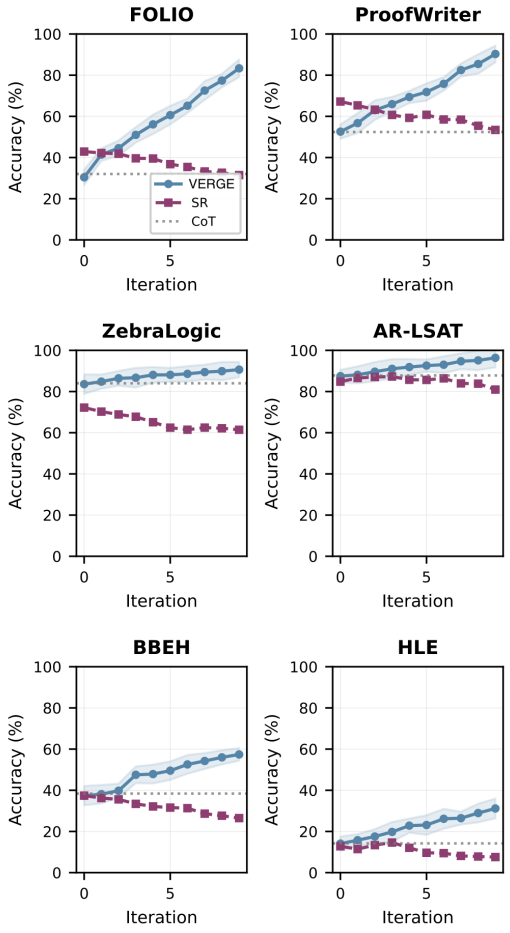

Figure 3: **The Correlation Cliff in Iterative Refinement.** Accuracy progression over 10 iterations (GPT-OSS-120B). VERGE exhibits perfect monotonic improvement (Kendall's $\tau = 1.0$, $p < 0.001$ across all datasets). Probabilistic Self-Refinement shows systematic degradation ($\tau = -0.84$ average, $p < 0.001$), stagnating below the CoT baseline in 85% of trials ($\chi^2 = 26.7$, $p < 0.001$). Shaded regions show 95% confidence bands.

System Prompt: Refinement

```
Your previous answer contained logical
contradictions verified by a formal solver.
Please refine your reasoning. Verification Re-
port:
  • Status: UNSATISFIABLE
  • Conflict Logic: [Z3_UNSAT_CORE]
  • Actionable Feedback: The set of claims
    {C1, C3} cannot be true simultaneously.
    Specifically, [MCS_DESCRIPTION].
Instruction: Rewrite your answer to resolve
this contradiction. Do not weaken the
premises; correct your derivation.
```

E.3 Consensus and Scoring Metrics

Formalization Consensus. To ensure the SMT code faithfully represents the natural language, we generate $K = 3$ candidate formalizations for every claim. We compute the **Semantic Equivalence** between candidates by querying the solver for the unsatisfiability of their negation (see Eq. 3). We construct an equivalence graph where edges represent proven logical identity. If a clique of size $\geq \lceil K/2 \rceil$ (majority consensus) is found, the representative formalization is accepted. Otherwise, the claim is flagged as "Ambiguous" and routed to Soft Verification.

Variance-Based Scoring. The final confidence score $\mathcal{S}(\mathcal{A})$ is penalized if high-confidence claims contradict each other. We define the score as:

$$\mathcal{S}(\mathcal{A}) = \mu_{claims} \cdot \max\left(0.5, 1.0 - \frac{\sigma_{claims}}{\mu_{claims} + \epsilon}\right) \tag{9}$$

where μ_{claims} is the weighted average verification status (Entailed=1.0, Soft-Pass=0.9, Possible=0.7, Fail=0.0) and σ_{claims} is the standard deviation. This penalizes answers that contain a mix of "True" and "False" claims, encouraging internal consistency.

Benchmark	Claims/Ans.	P	R
FOLIO	3.2	0.96	0.94
PW	4.1	0.97	0.95
ZL	5.8	0.93	0.91
AR-LSAT	6.4	0.94	0.92
BBEH	4.7	0.92	0.90
HLE	5.3	0.91	0.89

Table 10: Decomposition quality (80 audited trajectories).

E.4 Pipeline Reliability and Failure Modes

We audit the three LLM stages that could silently corrupt verification: decomposition, routing, and formalization.

Decomposition. On 80 audited trajectories (Table 10), errors are dominated by omissions, not fabrications: omitted claims that matter resurface as joint inconsistencies and are repaired by MCS feedback.

Formalization funnel. Table 11 traces hard-routed claims. Consensus is the central fidelity gate: a misformalization must independently appear in $\geq 2/3$ samples to survive. Round-trip alignment is a second gate; invalid SMT triggers soft fallback rather than silent failure.

Stage	20B	120B
Routed to hard (SMT)	68.1%	72.3%
Syntactically valid	30.2%	92.3%
Consensus ($\geq 2/3$)	41.2%	87.1%
Round-trip aligned	79.4%	94.6%
Verified by solver	12.1%	83.8%
Soft fallback	55.9%	16.2%
Solver timeout/error	4.6%	3.4%

Table 11: Formalization funnel by scale; VERGE degrades gracefully below the formalization barrier.

Benchmark	Hard	Soft	Hybrid
FOLIO	82.4	12.1	5.5
PW	91.3	4.2	4.5
ZL	88.7	6.8	4.5
AR-LSAT	76.3	16.2	7.5
BBEH	54.1	35.8	10.1
HLE	41.2	47.3	11.5

Table 12: Routing % on GPT-OSS-120B; Hybrid = hard-routed claims that fell back to soft.

Routing distribution. On GPT-OSS-120B (Table 12), logic-heavy benchmarks are >76% SMT-verified, leaving little exposure to consensus-on-falsehood; HLE/BBEH are soft-dominated by necessity, and VERGE honestly reflects this rather than overclaiming.

Error taxonomy. Three modes account for nearly all formalization failures, each caught by an existing safeguard: *(i)* predicate mismatches (Z3 parse error, soft fallback); *(ii)* quantifier scope (consensus disagreement); *(iii)* missing axioms (spurious UNSAT triggers context refinement, §3.3). The Soft-Only ablation (-22.8% avg) confirms these safeguards are load-bearing.

Entity extraction. Accuracy is $93-99\%$ across benchmarks, errors concentrated in type misassignment. A gold-entity study on AR-LSAT lifts consensus by 2.1% and accuracy by 0.7%, indicating robustness to extraction noise.

Shared hallucinations. Soft verification can fail under correlated judge errors. VERGE mitigates via the variance penalty (Eq. 6), a 0.9 cap on soft claim contributions (entailment cannot be overridden by consensus), and the joint check (soft claims contradicting hard ones are flagged). None of these eliminates shared hallucinations; we treat this as fundamental to any consensus-based verifier.

Bridges and cross-model consensus. Bridging axioms are LLM-generated and validated only conservatively (§3.5); principled validation via compositional translation remains open. Multi-sample consensus uses one formalizer; cross-model consensus is supported but not yet evaluated. Bridging axioms enter only the joint check, never individual verification. Adding constraints can destroy but never create satisfiability, so a faulty bridge triggers conservative over-refinement but cannot cause acceptance of an incorrect answer. Empirically, joint SAT disagrees with individual verification on 8.3% of GPT-OSS-120B problems; 72% are genuine conflicts and 28% bridge artifacts, with net accuracy impact below 0.5%.

Benchmark	Acc@T=10	Acc@T=15	Acc@T=20	Δ(T=10→20)
FOLIO	86.1	86.3	86.3	+0.2
ZebraLogic	92.3	92.4	92.5	+0.2
AR-LSAT	92.1	92.2	92.2	+0.1
HLE	31.8	32.0	32.1	+0.3

Table 13: **Extended iterations** $T > 10$**.** Beyond T=10 are negligible ($< 0.3\%$ across all benchmarks), validating the convergence criterion ($\Delta S < \epsilon = 0.01$).

E.5 Compute Infrastructure

All experiments were conducted on a standard workstation (64GB RAM, 16-core CPU). The Z3 solver component handles both verification (avg. $<$ 200ms) and equivalence checking (capped at 2.0s timeout per pair). The LLM inference was performed using the AWS Bedrock API.

E.6 Benchmark Descriptions

We used test split of each dataset mentioned below:

FOLIO (Han et al., 2024): First-order logic reasoning requiring entailment verification. Metric: Accuracy on predicting "Entailed/Contradictory/Unknown."

ProofWriter (Tafjord et al., 2021): Synthetic deductive reasoning. Metric: Proof step accuracy.

ZebraLogic (Lin et al., 2025): Constraint satisfaction puzzles (Einstein's riddle variants). Metric: Exact answer match.

AR-LSAT (Zhong et al., 2021): Analytical reasoning from LSAT exams. Metric: Multiple-choice accuracy.

BBEH (Kazemi et al., 2025): Big-Bench Extra Hard reasoning subset. Metric: Task-specific accuracy (varies by sub-task).

HLE (Phan et al., 2025): Humanity's Last Exam—diverse reasoning including ethics, physics,

literature (1,200 questions). Metric: Multiple-choice accuracy.

All reported numbers are **task accuracy** against gold labels, not internal verification scores. $S(\mathcal{A})$ is used only to decide whether to accept or refine during iteration.

F Detailed Verification Trajectories

Case Study 1: AR-LSAT (Temporal Scheduling)

Problem Context & Question

Context: Of eight students—George, Helen, Irving, Kyle, Lenore, Nina, Olivia, and Robert—in a seminar, exactly six will give individual oral reports during three consecutive days—Monday, Tuesday, and Wednesday. Exactly two reports will be given each day—one in the morning and one in the afternoon—according to the following conditions:

- Tuesday is the only day on which George can give a report.
- **Neither Olivia nor Robert can give an afternoon report.**
- If Nina gives a report, then on the next day Helen and Irving must both give reports...

Question: If Kyle and Lenore do not give reports, then the morning reports on Monday, Tuesday, and Wednesday, respectively, could be given by:
Options: A) Helen, George, Nina B) Irving, Robert, Helen C) Nina, Helen, Olivia D) Olivia, Robert, Irving E) Robert, George, Helen

Iteration 1 (Failed)

Generated Answer

Answer: E) Robert, George, and Helen

Extracted Claims & Verification Status

Idx	Claim Text	Type	Status	Score	Conf
0	The selected answer is Option E	LOGICAL	CONTRADICTORY	0.00	0.95
1	Robert gives the Monday Morning report	TEMPORAL	POSSIBLE	0.70	0.98
2	George gives the Tuesday Morning report	TEMPORAL	POSSIBLE	0.70	0.95
3	Helen gives the Wednesday Morning report	TEMPORAL	POSSIBLE	0.70	0.92
4	Olivia gives an afternoon report (Implicit)	LOGICAL	CONTRADICTORY	0.00	1.00

Detailed Failure Analysis

Claim 0 Failure: Constraint Violation

Issue: Option E assigns slots (Robert, George, Helen) to mornings. Since Kyle and Lenore are out, the remaining slots (Afternoons) must be filled by Nina, Irving, Olivia. This forces **Olivia** into an afternoon slot.

```
; Context Constraint: Neither Olivia nor Robert can give afternoon reports
(assert (not (exists ((d Day)) (gives_report Olivia d Afternoon))))

; Claim 0 implies:
(assert (gives_report Robert Monday Morning))
(assert (gives_report George Tuesday Morning))
(assert (gives_report Helen Wednesday Morning))
; This leaves Olivia with only Afternoon slots available -> CONTRADICTION
```

Z3 Result: `unsat`
Unsat Core: `['context_axiom_olivia', 'claim_0 part 0']`
Minimal Correction Subset (MCS): Remove `(gives_report Robert Monday Morning)` OR Remove `Option E`.

Metrics (Iteration 1)

Overall Score: 0.350 **Joint Cons:** 0.0% **Entailment:** 0.42 **Verified:** False

⇓ **VERGE Feedback: "Constraint Violation: Olivia forced into Afternoon. MCS: Change Option E or Robert's slot."** ⇓

Iteration 2 (Success)

Refined Answer

Answer: D) Olivia, Robert, and Irving
Reasoning: The previous attempt failed because it forced Olivia into an afternoon slot. To satisfy the condition "Neither Olivia nor Robert can give an afternoon report," both must be assigned to morning slots. Option D places Olivia (Mon AM), Robert (Tue AM), and Irving (Wed AM). This leaves valid afternoon slots for the others.

Extracted Claims & Verification Status

Idx	Claim Text	Type	Status	Score	Conf
0	The selected answer is Option D	LOGICAL	ENTAILED	1.00	0.95
1	Olivia gives the Monday Morning report	TEMPORAL	POSSIBLE	0.95	0.99
2	Robert gives the Tuesday Morning report	TEMPORAL	POSSIBLE	0.95	0.99
3	Irving gives the Wednesday Morning report	TEMPORAL	POSSIBLE	0.95	0.99
4	Constraint on Olivia/Robert is satisfied	LOGICAL	ENTAILED	1.00	1.00

Verification Success

Joint Consistency Verified

```
(assert (gives_report Olivia Monday Morning))
(assert (gives_report Robert Tuesday Morning))
(assert (gives_report Irving Wednesday Morning))
```

Z3 Result: `sat`.
Entailment Check: The conjunction of these assignments is consistent with all global constraints.

Final Metrics

Overall Score: 0.965 **Joint Cons:** 100% **Entailment:** 0.98 **Verified:** TRUE

Case Study 2: FOLIO (First-Order Logic)

Problem Context & Question

Context:

1. All guests at the party drink wine or beer. ($\forall x. Guest(x) => Wine(x) \vee Beer(x)$)
2. If a guest drinks wine, they eat cheese. ($\forall x. Guest(x) \wedge Wine(x) => Cheese(x)$)
3. **No one who eats cheese likes fish.** ($\forall x. Cheese(x) => \neg LikesFish(x)$)
4. Alan is a guest. ($Guest(Alan)$)
5. **Alan likes fish.** ($LikesFish(Alan)$)

Question: Does Alan drink beer?

Iteration 1 (Hallucination)

Generated Answer

No, Alan drinks wine. Since Alan is a guest, he drinks wine or beer. Guests usually drink wine with meals, and there is no rule against it.

Detailed Failure Analysis

Claim 0: "Alan drinks wine" is Contradictory

SMT Formalization:

```
(assert (Drinks Alan Wine))
```

Constraint Propagation: 1. `(Drinks Alan Wine)` => `(Eats Alan Cheese)` [Context Axiom 2] 2. `(Eats Alan Cheese)` => `(Not (Likes Alan Fish))` [Context Axiom 3] 3. Context Axiom 5 asserts: `(Likes Alan Fish)` 4. **Contradiction:** `(Likes Alan Fish)` AND `(Not (Likes Alan Fish))`
Z3 Result: `unsat`.
Unsat Core: `['Drinks_Alan_Wine', 'Axiom_2', 'Axiom_3', 'Axiom_5']`

Metrics (Iteration 1)

Score: 0.000 **Joint Cons:** 0.0% **Verified:** False

⇓ **VERGE Feedback: "Claim 'Alan drinks wine' causes contradiction with 'Alan likes fish'.**

MCS: Delete claim." ⇓

Iteration 2 (Success)

Refined Answer

Yes, Alan drinks beer. *Reasoning:* We know Alan is a guest, so he drinks Wine or Beer. If he drank Wine, he would eat Cheese. If he ate Cheese, he would not like Fish. However, we know Alan DOES like Fish. Therefore, he cannot drink Wine. Since he must drink Wine or Beer, and Wine is impossible, he must drink Beer.

Extracted Claims & Verification Status

Idx	Claim Text	Type	Status	Score	Conf
0	Alan drinks beer	LOGICAL	ENTAILED	1.00	0.98
1	Alan does not drink wine	LOGICAL	ENTAILED	1.00	0.98

Verification Success

Entailment Proof

```
; To prove "Alan drinks Beer" is entailed, we assert its negation and check for UNSAT
(push)
(assert (not (Drinks Alan Beer)))
(check-sat) ; Returns UNSAT -> Proves Entailment
(pop)
```

Final Metrics

Score: 1.000 **Joint Cons:** 100% **Entailment:** 1.00 **Verified:** TRUE

Case Study 3: ZebraLogic (Spatial Relations)

Problem Context & Question

Context: Three houses (Red, Blue, Green) are in a row, numbered 1 (left) to 3 (right).
- The Blue house is immediately to the right of the Red house.
- The Green house is somewhere to the left of the Blue house.

Question: What is the order of the houses from left to right?

Iteration 1 (Inconsistent)

Generated Answer

The order is Red, Blue, Green. *Reasoning:* Red is at 1. Blue is right of Red, so Blue is 2. Green is left of Blue, so Green is 3... wait.

Extracted Claims & Verification Status

Idx	Claim Text	Type	Status	Score	Conf
0	Red is at position 1	LOGICAL	POSSIBLE	0.70	0.90
1	Blue is at position 2	LOGICAL	POSSIBLE	0.70	0.90
2	Green is at position 3	LOGICAL	CONTRADICTORY	0.00	0.90

Detailed Failure Analysis

Joint Inconsistency

Issue: Claim 2 (Green=3) contradicts Claim 1 (Blue=2) given the constraint "Green is left of Blue".

```
(declare-fun pos (House) Int)
(assert (= (pos Blue) (+ (pos Red) 1)))  ; Blue right of Red
(assert (< (pos Green) (pos Blue)))      ; Green left of Blue

; Claims:
(assert (= (pos Blue) 2))
(assert (= (pos Green) 3))

; Check: 3 < 2 ? -> FALSE.
```

Z3 Result: `unsat`.
Unsat Core: `['Green_Left_Blue_Constraint', 'Claim_1', 'Claim_2']`

Metrics (Iteration 1)

Score: 0.466 **Joint Cons:** 0.0% **Verified:** False

⇓ VERGE Feedback: "Constraint violation: Green must be < Blue. Current: 3 < 2 (False)." ⇓

Iteration 2 (Success)

Refined Answer

The order is Green, Red, Blue. *Reasoning:* 1. Blue is right of Red -> $B = R + 1$. 2. Green is left of Blue -> $G < B$. Possible values: If R=1, B=2. Then G < 2 implies G=1. But houses are distinct, so G cannot be 1. Therefore, Red cannot be 1. Try R=2, B=3. Then G < 3 implies G=1 or G=2. Since R=2, G must be 1. Solution: Green(1), Red(2), Blue(3).

Extracted Claims & Verification Status

Idx	Claim Text	Type	Status	Score	Conf
0	Green is at position 1	LOGICAL	ENTAILED	1.00	0.95
1	Red is at position 2	LOGICAL	ENTAILED	1.00	0.95
2	Blue is at position 3	LOGICAL	ENTAILED	1.00	0.95

Verification Success

Entailment Verified

```
(assert (= (pos Green) 1))
(assert (= (pos Red) 2))
(assert (= (pos Blue) 3))
```

Z3 Result: `sat`. The solution uniquely satisfies all spatial constraints.

Final Metrics

Score: 1.000 **Joint Cons:** 100% **Entailment:** 1.00 **Verified:** TRUE