ERROR LOCALIZATION AND STEERING OF LANGUAGE MODEL QUANTITATIVE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

A single reasoning error in a large language model (LLM) can propagate through subsequent steps and lead to consequential failures in quantitative tasks. Existing correction methods rely largely on textual feedback, while formal verification can provide precise diagnostics without directly translating them into effective model interventions. We introduce our approach, a neuro-symbolic framework that turns formal diagnosis into targeted inference-time correction. A typed factor-graph intermediate representation grounds variables, relations, and dependencies in the source text, enabling a Satisfiability Modulo Theories (SMT) solver to localize the first conflicting reasoning step and identify the constraints responsible for the error. Our approach then converts this diagnosis into steering signals applied through the model's Key-Value (KV) cache, preserving the valid reasoning prefix while regenerating only the erroneous step with the same frozen model. Across four LLM backbones and four quantitative reasoning datasets, our approach improves both first-error localization and correction accuracy over three state-of-the-art baselines. Crucially, it provides effective guidance even for smaller models that struggle to act on textual critiques, showing that verifier-grounded steering can enhance corrections where language-only feedback falls short.

1 INTRODUCTION

Large language models (LLMs) are increasingly used to solve quantitative problems in safety-critical applications (Goodell et al., 2025), where errors can lead to catastrophic outcomes. However, LLMs have been observed to make logical and computational errors in quantitative reasoning (Ye et al., 2023). These errors can propagate across sequential steps and lead to erroneous consequential decisions (Zhou et al., 2024; Goodell et al., 2025) (Figure 1(a)).

To address this issue, researchers have proposed methods to detect reasoning errors (Zhou et al., 2024; Tyen et al., 2024) and intervene to correct subsequent reasoning (Madaan et al., 2023; Gou et al., 2024). Existing correction approaches provide textual critics from external LLMs (Madaan et al., 2023; Shinn et al., 2023), including error information and high-level refinement. However, these approaches rely on the reasoning ability of the external LLMs. Furthermore, the granularity of feedback is often insufficient to guide LLMs to converge on correct steps (Singh et al., 2026). Neuro-symbolic approaches leverage formal verification tools and represent the intermediate steps, including symbolic reasoning (Wooding et al., 2026), chain-of-thought entailment (An et al., 2026; Feng et al., 2026), and minimal correction subsets (Singh et al., 2026). These approaches advance from high-level critics to detailed formal diagnostics. However, the rich information exposed by formal verification is not fully translated into actionable guidance for the agent, leaving the resulting intervention too coarse-grained to directly correct the erroneous reasoning step. Corresponding challenges are illustrated in Figure 1(b).

Textual critics are not enough to guide the agent to converge to correct subsequent steps. Our experimental results in Table 2 show that vague descriptions may not be well comprehended in complex tasks or by smaller models. A body of research offers approaches to steer model output, including modifications to activations (Rimsky et al., 2024), decoding-time intervention (Liu et al., 2021) and the Key-Value (KV) Cache (Dathathri et al., 2020). Connecting formal diagnostics to inference-time steering requires an intermediate representation that exposes conflicts in reasoning and maps the implicated claims and constraints to their corresponding source text.

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.

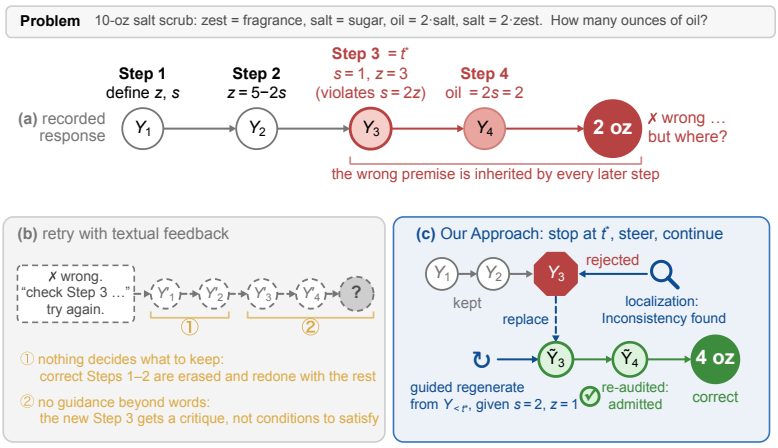


Figure 1: Real error made by Qwen2.5-7B-Instruct on GSM8K test case 000815 in abstracted form. (a) The error in Step 3 propagates to the final answer. (b) Existing approaches face two challenges: deciding which steps to keep and guiding regeneration beyond textual feedback. (c) Our approach preserves the correct steps, derives verified correction conditions, and uses them to guide regeneration of the erroneous step, producing a correct continuation in this example.

In this paper, we propose a neuro-symbolic framework built on a typed factor-graph intermediate representation. The representation records variable identities, mathematical relations, dependencies and source locations, while state transitions track admitted reasoning. our approach locates the first-error step containing conflicts to prevent snowball phenomena (Zhang et al., 2023) in which subsequent steps are poisoned by errors. For an identified conflict, our approach converts the formal diagnosis into targeted steering signals for regeneration. Rather than restarting the reasoning process or relying on a textual critic, it preserves the valid prefix and guides the same frozen model to revise the conflicting step under the diagnosed constraints (Figure 1(c)). This bridges the gap between formal error diagnosis and inference-time correction. We make the following contributions:

- We introduce a typed factor-graph intermediate representation that grounds variables and mathematical relations in their source text, providing a structured and traceable representation of evolving reasoning.
- The proposed source-grounded error localization method identifies the first reasoning step that contradicts the premises, while preserving the valid prefix and exposing the claims and constraints responsible for the conflict.
- We bridge formal diagnosis with inference-time steering, going beyond textual critique by using localized contradictions to provide verified correction conditions to guide how the erroneous step is revised.
- We comprehensively evaluate our approach against 3 state-of-the-art baselines over 16 scenarios (4 LLM backbones; 4 datasets). Our approach shows superior performance in locating the first error and steering LLMs to correct answers

2 Problem Formulation and Preliminaries

2.1 Problem formulation

A language model generates a stepwise solution to a quantitative reasoning problem. We aim to identify the first invalid step, if any, and steer the solution from that point while preserving the preceding valid steps. The model parameters remain fixed during steering.

Let $p = (q_p, a_p)$ denote a problem, where q_p is the natural-language statement and a_p specifies the target quantity and requested answer form. For stepwise error localization, we partition the model's original response as $Y = (Y_1, \ldots, Y_T)$, with Y_t denoting a reasoning step. Its prefix before step t is $Y_{<t} = (Y_1, \ldots, Y_{t-1})$.

Generated steps may contain unjustified assertions or definitions. We assess their validity under the intended mathematical interpretation of p.

Definition 1 (Step validity)**.** Given a valid prefix $Y_{<t}$, the step Y_t is *valid relative to* $(p, Y_{<t})$, written $V_p(Y_t \mid Y_{<t}) = 1$, if its definitions are admissible and every assertion follows from q_p, $Y_{<t}$ and those definitions. Otherwise $V_p(Y_t \mid Y_{<t}) = 0$.

A definition is admissible if it introduces fresh quantities without restricting previously established quantities, precedes its uses, and satisfies its domain obligations. Thus definitions are considered in dependency order. Prefix validity is inductive: $Y_{1:t}$ is valid when $Y_{<t}$ is valid and $V_p(Y_t \mid Y_{<t}) = 1$. The first invalid step is

$$t^\star = \min \{t \in \{1, \ldots, T\} : Y_{<t} \text{ is valid and } V_p(Y_t \mid Y_{<t}) = 0\}.$$

If an invalid step exists, we steer the response from $t^\star$.

Definition 2 (Steering)**.** Given $t^\star$, *steering* regenerates the response from step $t^\star$ with the same fixed model, using the valid prefix and localized error information.

The repaired response $\widetilde{Y} = (\widetilde{Y}_1, \ldots, \widetilde{Y}_{\widetilde{T}})$ has $\widetilde{T} \geq t^\star$ steps and retains the original prefix: $\widetilde{Y}_{<t^\star} = Y_{<t^\star}$. Steering succeeds when every step of $\widetilde{Y}$ is valid and $\widetilde{Y}$ satisfies the answer requirement a_p.

Problem 1. Given p, a fixed language model and its response Y, solve two tasks: (1) detect whether Y contains an invalid reasoning step and, if so, return its earliest index $t^\star$; and (2) steer the response from the valid prefix $Y_{<t^\star}$ to produce a valid continuation that answers p.

2.2 Preliminaries

Factor graphs. A factor graph represents a set of variables X and a set of mathematical relations Φ as two kinds of nodes. Each relation $\phi_j \in \Phi$ is a constraint on the variables it involves; edges connect those variables to its node. In a typed factor graph, each variable $x_i \in X$ has a *sort* specifying its mathematical domain.

Satisfiability and entailment. Let B be a conjunction of premise constraints and A the assertion to check. A *model* of B assigns each variable a value of its sort and satisfies every constraint. We write $\mathrm{SAT}(B)$ when a model exists and $\mathrm{UNSAT}(B)$ otherwise. Here $\wedge$ means conjunction and $\neg$ means negation. The entailment $B \models A$ means that every model of B satisfies A, equivalently $\mathrm{UNSAT}(B \wedge \neg A)$. A model of $B \wedge \neg A$ is a *countermodel*. When entailment fails, A contradicts B if $\mathrm{UNSAT}(B \wedge A)$; otherwise it is compatible but unsupported.

Minimal conflicts. Let $\mathcal{C}$ be a finite set of labelled constraints, with distinct labels for repeated formulas. For $U \subseteq \mathcal{C}$, $\bigwedge U$ conjoins its constraints; the empty conjunction is true. Write $\mathrm{SAT}(U)$ for $\mathrm{SAT}(\bigwedge U)$, and similarly for UNSAT. An *unsatisfiable core* is a subset U with $\mathrm{UNSAT}(U)$. It is a *minimal unsatisfiable subset* (MUS) if $\mathrm{SAT}(U \setminus \{\varrho\})$ for every $\varrho \in U$: deleting any one constraint makes the remaining constraints jointly satisfiable. Thus a MUS is minimal under set inclusion; it need not have the smallest number of constraints among all unsatisfiable subsets (Liffiton & Malik, 2013; Liffiton et al., 2016).

3 Error localization

To locate the first invalid step, we propose an error localization framework that represents reasoning as typed factor graphs and checks successive state updates. It retains the valid prefix and error evidence for steering.

Sub-problem (First-error localization). Given the problem p and response Y, determine whether Y contains an invalid reasoning step and, if so, return its earliest index $t^\star$ under Definition 1.

3.1 Intermediate representation of reasoning

Free-form reasoning can leave quantity meanings, relations and step dependencies implicit, so we need an explicit intermediate representation. Extraction translates the problem and the written

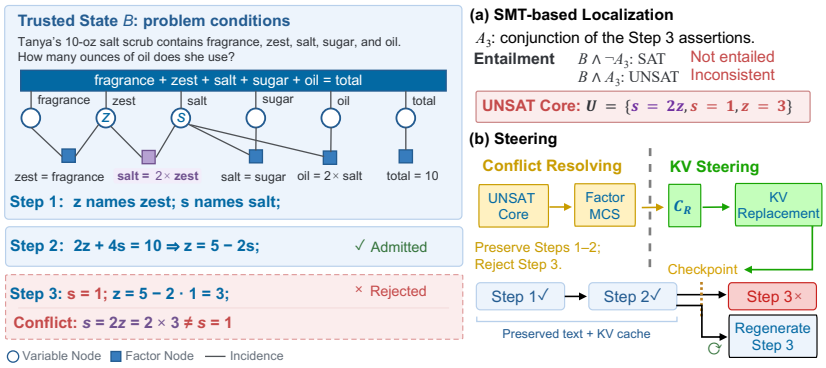


Figure 2: Typed factor graphs connect reasoning steps to the problem constraints. (a) Solver checks reject Step 3 and return an unsatisfiable core: $s = 1$ and $z = 3$ violate $s = 2z$. (b) Conflict resolution finds a factor-level minimal correction set and derives correction conditions C_R. Steering regenerates Step 3 from its checkpoint while preserving the text and key-value cache of Steps 1–2.

reasoning into mathematical relations that a solver can check. Stepwise translation with links to the source text also appears in VeriCoT (Feng et al., 2026). We organize these relations as typed factor graphs so that a failed claim can be traced to its quantities and location in the response. Appendix A.1 gives the representation interface.

Typed factor graphs. In Figure 2(a), the factors $s = 2z$ and $z = 5 - 2s$ (squares) share the salt s and zest z nodes (circles), following the factor-graph convention (Kschischang et al., 2001). Each *factor occurrence* is one recorded instance of a relation with its text location; repeated relations retain separate occurrences.

Let ψ_0 denote the problem expression and ψ_t the expression of step Y_t. Its logical content is $[\![\psi_t]\!] = \Delta_t \wedge A_t$, where Δ_t conjoins definitions and A_t conjoins assertions. A step may contain several factors, but admission applies to the whole step. Appendix A specifies the typed product, quantity identifiers, definition dependencies and source records.

We assume that the extraction is conducted faithfully. The faithful extraction includes quantity meanings and dependencies, and satisfies the full faithful-extraction contract in Appendix A.1.

Assumption 1 (Faithful extraction)**.** We assume that the extraction succeeds and preserves the mathematical meaning of the problem and steps

From steps to a state machine. Each state is a snapshot of the accumulated typed factor graph. The symbolic state machine is

$$\mathcal{M}_p = (\mathscr{S}_p, \Sigma_p, \delta_p, s_0).$$

Here $s_0 = \psi_0$ is the initial state, Σ_p contains fully specified symbolic updates, δ_p is their partial transition function, and $\mathscr{S}_p$ contains the reachable typed expressions. The symbol ι denotes insertion: $\iota[\psi_t]$ proposes $\delta_p(s_j, \iota[\psi_t]) = s_j \otimes \psi_t$ after j admitted operations. This product adds the step's factors while sharing established quantity nodes. Appendix A specifies structural compatibility and operation semantics.

Structural reachability does not establish validity. The admission gate $G_\kappa(s_j, \iota[\psi_t])$ takes the committed state and proposed operation and returns *admit*, *reject* or *unresolved*. The fixed configuration κ specifies the theory, solver and definition checker shared across steps. Error localization commits $s_{j+1} = s_j \otimes \psi_t$ only on admission and otherwise leaves s_j unchanged. Thus s_j contains the problem and admitted relations, and $j = t - 1$ before textual step t on an uninterrupted replay with one operation per step.

3.2 Error localization in the represented reasoning

Error localization checks whether each step's assertions follow from the trusted state. It advances the state only when the whole step is admitted and stops at the first rejected or unresolved step.

Error localization first establishes that $[\![\psi_0]\!]$ is satisfiable. Before step t, let $B = [\![s_j]\!]$ be the theory of the problem and admitted prefix. Certify the definitions Δ_t in dependency order under Assumption 2. A conservative extension lets every model of the preceding theory extend to satisfy the new definition without changing established quantities. Successful certification gives the temporary theory $B_t^+ = B \wedge \Delta_t$ for checking A_t; no part of the step is committed yet.

Assumption 2 (Sound solver and definition checks)**.** Every solver answer used is correct in the declared theory. Every admitted definition is fresh, well formed and a conservative extension with satisfied domain obligations. Uncertified definitions and undecided entailment queries remain unresolved.

The SMT solver Z3 (De Moura & Bjørner, 2008) tests $B_t^+ \models A_t$ as $\mathrm{UNSAT}(B_t^+ \wedge \neg A_t)$. An unsatisfiable answer admits the whole step. A satisfiable answer returns a countermodel and rejects it. The further query $B_t^+ \wedge A_t$ distinguishes an *inconsistent* assertion (unsatisfiable) from a *compatible but unsupported* assertion (satisfiable). If this query is undecided, rejection still holds but its classification remains unresolved. Failed extraction, incompatible products, uncertified definitions and undecided entailment checks provide no mathematical rejection. Appendix A.3 gives the complete gate.

Let $\hat{t}$ be the first step not admitted. Error localization freezes s_j and returns the stopped step with available source locations, a countermodel for rejection and an additional unsatisfiable core for a certified inconsistency; unresolved checks carry diagnostics.

Error localization in the motivating example. In Figure 2(a), Step 2 asserts $z = 5 - 2s$, which the trusted state entails, so Error localization admits it. Step 3 asserts $s = 1$ and $z = 3$. These contradict $s = 2z$, although the arithmetic $5 - 2 \cdot 1 = 3$ is correct. The entailment query returns a countermodel and the consistency query is unsatisfiable. Error localization rejects the whole step and retains the preceding state; the factor occurrences locate the conflicting assignments.

Proposition 1 (First-error localization)**.** Suppose the initial state is satisfiable and Assumptions 1 and 2 hold. If a first invalid step $t^\star$ exists, all proposed products through that step are structurally compatible, all encountered definitions are well formed and certified, and all required queries are decided, then Error localization stops exactly at that step: $\hat{t} = t^\star$.

The proof in Appendix D establishes validity and satisfiability of each admitted prefix before showing exact localization. An unresolved check can stop localization earlier. The implemented definition checker is weaker than Assumption 2 for some partial functions (Appendix A.4).

4 STEERING

When Error localization detects an inconsistency in a reasoning step, it identifies the factors involved. Steering uses these factors to guide the model in generating a replacement for the rejected step while preserving the valid prefix.

Sub-problem (Conflict-guided steering). Given a valid prefix and a localized conflict in the next step, perform steering as defined in Definition 2.

The repair targets the rejected step $Y_{\hat{t}}$ and leaves the admitted prefix $\Pi = Y_{<\hat{t}}$ unchanged. The frozen state s_j from Section 3 contains the problem and the relations admitted by j operations, represented as a typed factor graph. Its logical content is $B = [\![s_j]\!]$. Let ζ denote the generation checkpoint preceding $Y_{\hat{t}}$, storing the tokens and KV cache tied to Π.

Steering attempts a repair when Error localization finds $B_{\hat{t}}^+ \wedge A_{\hat{t}}$ unsatisfiable. If the step's assertions are consistent with $B_{\hat{t}}^+$ but are not entailed by it, the loop stops without repair. The repair proceeds in two stages. Conflict resolution (Section 4.1) determines which factors to revise and derives certified conditions for correcting the step. KV-based execution (Section 4.2) regenerates the rejected step from ζ. Error localization checks the regenerated step, which is committed only if admitted.

4.1 CONFLICT RESOLUTION

An inconsistent step can contain conflicting claims alongside correct calculations. Let $t = \hat{t}$ be the stopping index returned by Error localization. Earlier steps have already passed the admission checks

and remain fixed in $B = [\![s_j]\!]$. Steering reduces the unsatisfiable core returned by Error localization to a MUS U to explain the conflict, then searches for factors to revise within the rejected step Y_t.

Keeping B fixed, we search over the labelled formula factors $\mathcal{C}$ of the rejected step ψ_t. A minimal correction set (MCS) $R \subseteq \mathcal{C}$ is an inclusion-minimal set whose removal restores consistency. The retained factors $L = \mathcal{C} \setminus R$ satisfy $\mathrm{SAT}(B \wedge L)$ and $\mathrm{UNSAT}(B \wedge L \wedge \varrho), \forall \varrho \in R$. In these formulas, we use L to denote the conjunction of the retained logical formulas.

We propose entailment-guided search (EGS), which combines MCS enumeration with validity checks on the retained factors. EGS uses MARCO (Liffiton & Malik, 2013; Liffiton et al., 2016) to enumerate alternative deletion sets, reusing previous checking results to avoid redundant search. Restoring consistency means that the retained factors L can hold together with B, but does not establish that the retained claims follow from B and the definitions retained in the current step. Removing a factor may also remove a definition needed by another retained claim. We therefore check that every definition used by a retained claim is still available in B or among the retained factors. We then recheck the retained factors under the admission criteria of Section 3. A candidate R is eligible for repair only if all retained factors pass these checks.

An admissible candidate R identifies the factors to revise in the rejected step. Appendix B.3 specifies the search procedure. For each factor in R, we use solver checks to identify constraints in B and retained factors in L that jointly conflict with it. We collect these checked constraints into C_R to guide the model when revising the step. The factors in R are excluded from C_R. If a formula selected for revision contains a quantity x and the solver confirms that B uniquely determines its value as v, we include $x = v$ in C_R. In Figure 2, the model uses $s = 2$ from C_R to compute $2s = 4$ ounces of oil, and Error localization admits the regenerated step. Appendix B.5 documents the run. Appendix B.5 describes how these values are obtained and checked.

4.2 KV-BASED EXECUTION AND RE-CHECKING

Let $\mathcal{R}_t$ denote the set of repair records constructed by conflict resolution. Given a supplied record $r \in \mathcal{R}_t$, we restore checkpoint ζ at the beginning of the rejected step. This discards the rejected step's tokens and KV cache entries while preserving those of the admitted prefix.

We encode C_R as tokens and pass them through the model, appending their keys and values to the retained KV cache. The model then uses the updated cache to generate a replacement for the rejected step. Error localization converts the replacement into a symbolic representation and checks it against the unchanged state s_j. We accept the replacement and update the state only if it passes the admission checks and is consistent with B. If the replacement fails a check or a check cannot be completed, we leave s_j unchanged and restore ζ. We then retry using the same repair record r and checkpoint, up to the retry budget. After accepting a replacement, we rebuild the next checkpoint from the accepted reasoning text without the tokens for C_R. Appendix C states the conditions under which the earlier KV entries remain unchanged. Appendix B.4 details the generation and checking procedure.

Execution in the motivating example. For Step 3, conflict resolution constructs C_R, including the verified values $s = 2$ and $z = 1$. We restore checkpoint ζ at the beginning of the rejected step and feed C_R to the model as conditioning tokens. With these conditions in its context, the model uses $s = 2$ to compute the required oil quantity as $2s = 4$ ounces. Error localization checks and admits the regenerated step. Appendix B.5 provides details of the recorded run.

Algorithm 1 combines Error localization, conflict resolution, and KV-based execution. ENUMERATEREPAIRS performs the conflict resolution procedure in Section 4.1, including MUS extraction. It takes the problem expression ψ_0, the represented steps $(\psi_i)_{i=1}^{t}$, and the frozen state s_j, and returns repair records $\mathcal{R}_t$ with a status σ indicating whether enumeration is complete. Each record $r \in \mathcal{R}_t$ contains an MCS R that passes the retained-factor validity checks, the source locations of the factors in R, and the corresponding conditions C_R.

Theorem 1 (Validity under steering)**.** *Under Assumptions 1 and 2, an initially satisfiable problem state, and correct source and checkpoint bindings, every prefix committed by Algorithm 1 is valid, including after a regenerated step is admitted.*

This guarantee concerns committed reasoning; it does not ensure that generation finds an admissible replacement or completes an answer within the retry budget. Appendix D proves the theorem and the

Algorithm 1 Our approach: error localization and conflict-guided regeneration

Require: Problem p, fixed model M_θ, step limit $T_{\max}$, retry budget N
Initialize $j = 0$, trusted state s_j, admitted prefix Π, and checkpoint ζ
while Π has no terminal answer and $|\Pi| < T_{\max}$ **do**
    Set $t = |\Pi| + 1$; generate a tentative step y from ζ
    Error localization extracts ψ_t and checks the step against s_j
    if y is inconsistent **then**
        Freeze s_j; $(\mathcal{R}_t, \sigma) \leftarrow \textsc{EnumerateRepairs}(\psi_0, (\psi_i)_{i=1}^{t}, s_j)$
        output $(\mathcal{R}_t, \sigma)$; **resume only on input** $r \in \mathcal{R}_t$
        Try at most N regenerations from ζ using C_R from r; re-check each attempt
        Stop unless one is admitted and consistent with $[\![s_j]\!]$; let y be that replacement
    end if
    Stop unless y is admitted
    Commit y to obtain s_{j+1} and extend Π; increment j
    Advance ζ; after a repair, rebuild it from the accepted text without the tokens for C_R
end while
return Π with status complete or truncated

repair-certificate properties. For frozen B and finite $\mathcal{C}$, completed MARCO enumeration under sound, decided oracle queries and resolved survivor checks identifies exactly the MCS deletions that pass the specified checker (Appendix D.1). Returning a record for every such deletion additionally requires successful support-certificate and source-binding construction. This does not enumerate arbitrary replacement expressions; an empty $\mathcal{R}_t$ yields no executable candidate.

5 Evaluation

We evaluate our approach through two research questions: **RQ1:** How accurately does our approach identify the earliest invalid step in a reasoning trace? **RQ2:** How effectively does our approach steer generation from the valid prefix to recover a correct final answer?

Experimental setup. We evaluate our approach on 209 incorrect solutions generated by Qwen2.5-7B-Instruct across GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2020), and GSM-Symbolic (Mirzadeh et al., 2025). We compare against Reflexion (Shinn et al., 2023), SatLM (Ye et al., 2023), and Logic-LM (Pan et al., 2023), covering textual feedback and symbolic reasoning approaches. For RQ1, we evaluate first-error localization accuracy on shared erroneous traces to assess how reliably our approach identifies where reasoning first becomes invalid. For RQ2, we evaluate final-answer recovery accuracy on the same problem subsets using Qwen2.5-7B, Llama3.1-8B, GPT-OSS-20B, and Qwen3.8-27B to assess the effectiveness of our approach steering across backbones. Together, these evaluations examine our approach's ability to locate reasoning errors and guide subsequent generation toward correct answers. We report per-dataset accuracies and their macro-average. The macro-average is the sum of the GSM8K, SVAMP, ASDiv, and GSM-Symbolic accuracies divided by four, giving each dataset equal weight regardless of its sample size. We compute this average from unrounded accuracies and round the result to two decimal places. Dataset descriptions, baseline adaptations, and evaluation protocols are provided in Appendix E.

RQ1 Result: First-error localization. Table 1 shows that our approach achieves the highest localization accuracy on all four datasets, with a macro-average accuracy of 94.55%, compared with 79.47% for the strongest baseline configuration, Reflexion with Qwen3.8-27B. our approach exceeds the best baseline on GSM8K by 21.84%, on SVAMP by 3.57%, on ASDiv by 4.76%, and on GSM-Symbolic by 16.44%. Relative to the mean accuracy of the three baseline methods, our approach improves by 57.85% on GSM8K, 55.36% on SVAMP, 54.37% on ASDiv, and 49.09% on GSM-Symbolic. These results demonstrate the effectiveness of our approach in identifying the earliest invalid reasoning step across diverse mathematical word problems. Compared with Reflexion, which relies more heavily on the model's reasoning ability to critique a solution, our approach checks each step explicitly against the problem constraints and the admitted prefix, providing a precise criterion for identifying unsupported claims. For SatLM, even a correctly parsed formulation does not by itself

Table 1: First-error localization accuracy (%) on incorrect solutions from Qwen2.5-7B-Instruct. Header parentheses give the number of evaluated solutions; accuracy is the percentage of these solutions whose predicted first-error step exactly matches the reference. Macro avg. is the unweighted mean of the four dataset accuracies; bold marks the best score per column.

Approach	Reasoning	GSM8K (87)	SVAMP (28)	ASDiv (21)	GSM-Sym (73)	Macro avg.
Ours	Deterministic	**95.40%**	**96.43%**	**90.48%**	**95.89%**	**94.55%**
Reflexion	Qwen2.5-7B	19.54%	42.86%	28.57%	30.14%	30.28%
	Llama3.1-8B	28.74%	50.00%	38.10%	35.62%	38.11%
	GPT-OSS-20B	65.52%	71.43%	66.67%	79.45%	70.77%
	Qwen3.8-27B	73.56%	92.86%	85.71%	65.75%	79.47%
SatLM	Deterministic	51.72%	42.86%	38.10%	76.71%	52.35%
Logic-LM	Qwen2.5-7B	10.34%	7.14%	9.52%	16.44%	10.86%
	Llama3.1-8B	20.69%	25.00%	19.05%	8.22%	18.24%
	GPT-OSS-20B	12.64%	10.71%	19.05%	5.48%	11.97%
	Qwen3.8-27B	12.64%	21.43%	14.29%	13.70%	15.51%

identify where the original reasoning first goes wrong. Our approach models individual mathematical relations within each step and preserves their source locations, connecting solver evidence directly to the reasoning trace. These differences help explain its stronger first-error localization performance.

RQ2 Result: Answer recovery through steering. Table 2 shows that our approach achieves the highest macro-average answer recovery accuracy for every backbone: 73.81% with Qwen2.5-7B, 81.34% with Llama3.1-8B, 85.88% with GPT-OSS-20B, and 96.11% with Qwen3.8-27B. These results demonstrate the effectiveness of our approach steering across the evaluated backbones and mathematical word-problem datasets. Factor-graph modeling provides transition-level feedback for effective recovery with smaller backbones. With Qwen2.5-7B and Llama3.1-8B, our approach exceeds the strongest baseline for each backbone by 45.45% and 36.82% in macro-average recovery accuracy, respectively, and outperforms every baseline on all four datasets. Together with the 94.55% first-error localization accuracy in Table 1, these results suggest that **accurate diagnosis and transition-level feedback help smaller backbones recover from reasoning errors**. Our approach represents reasoning steps as updates to a typed factor graph, making their mathematical relations explicit and traceable to the source text so that invalid transitions can be precisely localized. Conflict resolution then derives certified correction conditions for the rejected transition, providing feedback for its refinement while preserving the valid prefix. By supplying both the error location and the mathematical conditions required for correction, our approach reduces the need for the generating model to independently diagnose the failure and determine how to correct it. By separating error localization and diagnosis from generation, our approach gives the model a precise direction for correcting the rejected transition, supporting effective recovery with smaller backbones.

6 Related Work

Correcting an LLM's reasoning trace requires both identifying where the reasoning fails and providing reliable guidance for its revision. Prompt-based methods such as Self-Refine (Madaan et al., 2023) and Reflexion (Shinn et al., 2023) use textual feedback to refine responses, but interpreting feedback and producing a successful revision still depend on the model's reasoning capabilities. Tree of Thoughts (Yao et al., 2023) and process supervision (Lightman et al., 2024) evaluate reasoning paths or intermediate steps, supporting selection and backtracking without scores themselves specifying which relations to revise. Logic-LM (Pan et al., 2023) and SatLM (Ye et al., 2023) delegate inference over formalized problems to solvers, with Logic-LM also using solver errors to refine formalization. Applying these results to trace repair additionally requires correspondence with the original textual claims. Neurosymbolic methods make diagnosis more explicit: ARc (An et al., 2026) checks statements against formal policies, VeriCoT (Feng et al., 2026) checks steps against accumulated premises, and VERGE (Singh et al., 2026) uses atomic claims and correction subsets to guide revision. However, incorrect claims can propagate into later steps. The representations should also ensure the

Table 2: Answer recovery accuracy (%). All methods and backbones use the same problems initially answered incorrectly by Qwen2.5-7B-Instruct. Header parentheses give the number of evaluated problems; accuracy is the percentage of these problems with a correct final answer after recovery, regardless of error-detection or localization success. Macro avg. is the unweighted mean of the four dataset accuracies; bold marks the best score per backbone and column, including ties.

Backbone	Approach	GSM8K (87)	SVAMP (28)	ASDiv (21)	GSM-Sym (73)	Macro avg.
Qwen2.5-7B	**Ours**	**75.86%**	**82.14%**	**61.90%**	**75.34%**	**73.81%**
	Reflexion	9.20%	14.29%	33.33%	38.36%	23.79%
	SATLM-SOL	13.79%	32.14%	42.86%	24.66%	28.36%
	Logic-LM	5.75%	28.57%	19.05%	50.68%	26.01%
Llama3.1-8B	**Ours**	**78.16%**	**78.57%**	**80.95%**	**87.67%**	**81.34%**
	Reflexion	19.54%	39.29%	42.86%	49.32%	37.75%
	SATLM-SOL	34.48%	50.00%	33.33%	60.27%	44.52%
	Logic-LM	19.54%	21.43%	23.81%	36.99%	25.44%
GPT-OSS-20B	**Ours**	**79.31%**	**92.86%**	**80.95%**	90.41%	**85.88%**
	Reflexion	71.26%	89.29%	66.67%	84.93%	78.04%
	SATLM-SOL	75.86%	82.14%	**80.95%**	**94.52%**	83.37%
	Logic-LM	48.28%	78.57%	52.38%	58.90%	59.53%
Qwen3.8-27B	**Ours**	**91.95%**	**100.00%**	**95.24%**	**97.26%**	**96.11%**
	Reflexion	86.21%	89.29%	90.48%	95.89%	90.46%
	SATLM-SOL	77.01%	89.29%	80.95%	95.89%	85.79%
	Logic-LM	70.11%	85.71%	61.90%	78.08%	73.95%

validation of the accumulative correctness. Our approach connects these decisions through a typed factor-graph IR preserving quantity identities, dependencies, and source correspondence. For certified conflicts, it identifies which claims to remove and derives verified conditions from valid prefix to guide regeneration from the corresponding checkpoint. Appendix G provides detailed comparisons.

7 Conclusion

We show that formal verification becomes substantially more useful for correcting LLM reasoning when its diagnostics can directly drive inference-time intervention. Our key insight is that this connection requires an intermediate representation that exposes reasoning conflicts and grounds them in the source text. Based on this insight, our approach uses a typed factor-graph representation to enable SMT-based localization of the first conflicting reasoning step and translates the resulting diagnosis into Key-Value (KV) cache steering without discarding the valid reasoning prefix. Across 16 scenarios of LLM backbones and quantitative reasoning datasets, our approach consistently improves both error localization and answer recovery, with particularly strong benefits for smaller models that struggle to follow textual critiques. These results show that formal verification can not only detect reasoning failures but also steer models with an intermediate representation that bridges formal diagnosis and inference-time intervention.

Limitations. The current formulation focuses on quantitative reasoning that can be expressed as typed constraints and verified with SMT. Evaluation relies on reviewed representations and selected incorrect solutions, while extraction faithfulness and fully automatic edit selection remain unmeasured. Validity remains conditional on the stated extraction, verification, and binding assumptions, and local repair does not guarantee recovery of the correct answer.

Statement of AI Use

Our evaluation dataset consists of LLM-generated artifacts, including reasoning traces and symbolic representations, derived from problems in existing open-source datasets. AI tools assisted with manuscript writing and language polishing, organization, figure design, and implementation checks. The authors remain responsible for the final manuscript and its evidence.

REFERENCES

Chenyang An, Sam Bayless, Stefano Buliani, Darion Cassel, Byron Cook, Duncan Clough, Rémi Delmas, Nafi Diallo, Ferhat Erata, Nick Feng, et al. A neurosymbolic approach to natural language formalization and verification. In *International Conference on Computer Aided Verification*, pp. 601–617. Springer, 2026.

Clark Barrett and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of model checking*, pp. 305–343. Springer, 2018.

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.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

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

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 *International Conference on Learning Representations*, volume 2026, pp. 133170–133208, 2026.

Alex J Goodell, Simon N Chu, Dara Rouholiman, and Larry F Chu. Large language model agents can use tools to perform clinical calculations. *NPJ digital medicine*, 8(1):163, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Nan Duan, Weizhu Chen, et al. Critic: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, volume 2024, pp. 57734–57811, 2024.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22895–22907. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/kambhampati24a.html.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

Mark H Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple muses quickly. In *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 160–175. Springer, 2013.

Mark H Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible mus enumeration. *Constraints*, 21(2):223–250, 2016.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6691–6706, 2021.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human language technologies*, pp. 4288–4299, 2021.

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. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 94743–94765, 2025.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

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*, pp. 3806–3824, 2023.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Vikash Singh, Darion Cassel, Nathaniel Weir, Nick Feng, and Sam Bayless. Verge: Formal refinement and guidance engine for verifiable llm reasoning. *arXiv preprint arXiv:2601.20055*, 2026.

Gladys Tyen, Hassan Mansoor, Victor Cărbune, Yuanzhu Peter Chen, and Tony Mak. Llms cannot find reasoning errors, but can correct them given the error location. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13894–13908, 2024.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models-a critical investigation. *Advances in neural information processing systems*, 36:75993–76005, 2023.

Ben Wooding, Kiersten Brennan, Anne M Tumlin, Hongchao Zhang, and Taylor T Johnson. A self-correcting neuro-symbolic ai reasoning framework. In *International Symposium on AI Verification*, pp. 191–211. Springer, 2026.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. Satlm: Satisfiability-aided language models using declarative prompting. *Advances in Neural Information Processing Systems*, 36:45548–45580, 2023.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A Smith. How language model hallucinations can snowball. *arXiv preprint arXiv:2305.13534*, 2023.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Processbench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1009–1024, 2025.

Jin Zhou, Charles Staats, Wenda Li, Christian Szegedy, Kilian Weinberger, and Yuhuai Wu. Don't trust: Verify–grounding llm quantitative reasoning with autoformalization. In *International Conference on Learning Representations*, volume 2024, pp. 2544–2563, 2024.

A Representation and Error Localization Details

Typed variables and occurrences. Let $\mathscr{T}$ be a finite sort set, with nonempty domain D_σ for each sort σ. A variable x_i^σ has identifier $i \in \mathscr{I}$ and sort σ; a finite variable set contains at most one variable with a given identifier. A valuation is $\nu \in \prod_{x_i^\sigma \in X} D_\sigma$. A local-factor occurrence φ_α has a unique identifier, ordered input and output scopes, a local function f_α, and immutable source records $\mathrm{src}(\varphi_\alpha) \subseteq \mathscr{R}$. For nonempty output scope, f_α maps the product of input domains to the product of output domains. An occurrence without output variables is relational, with codomain $\{0, 1\}$.

Product and satisfaction. The expression $e = \bigotimes_{\alpha \in \mathscr{A}} \varphi_\alpha$ retains distinct factor occurrences. Its partial product with e' is defined when shared variable identifiers have equal sorts and factor identifiers are disjoint. The product identifies shared variables and preserves all factors and source records. Structural equality preserves identifiers, scopes, functions and source records up to reindexing. Write $X(e)$ and $\gamma(e)$ for the variable and factor sets. A valuation satisfies a functional occurrence when the local function's output equals the valuation of its output variables, and satisfies a relational occurrence when its value is true. Then $\nu \models e$ means that every occurrence is satisfied; $[\![e]\!]$ denotes the corresponding conjunction. The factor graph follows directly from the scopes. Its incidence alone does not establish directed dependence.

Objective transitions and admission. Let $\mathscr{E}$ be the set of finite well-typed factorized expressions, $\mathscr{O}$ a set of operation symbols, and $\Theta_{p,\mu}$ the parameter set of $\mu \in \mathscr{O}$. The operation alphabet is $\Sigma_p = \bigsqcup_{\mu \in \mathscr{O}} \Theta_{p,\mu}$. A ground operation $\mu[\vartheta]$, with $\vartheta \in \Theta_{p,\mu}$, denotes a partial action $\xi_p(\mu[\vartheta]) : \mathscr{E} \rightharpoonup \mathscr{E}$. Starting at $s_0 = \psi_0$, let $\mathscr{S}_p$ contain the expressions reachable by defined actions, with $\delta_p(s, \mu[\vartheta]) = \xi_p(\mu[\vartheta])(s)$. Insertion uses $\delta_p(s, \iota[\psi]) = s \otimes \psi$. This is an objective transition, without an accepting set. The separate admission gate $G_\kappa(s, \mu[\vartheta])$ maps a state and proposed operation to admit, reject or unresolved under the fixed checker configuration κ. Error localization commits the transition only on admission. Consequently, structural representability does not imply textual validity under V_p.

Fresh definitions and solver assumptions. Ordinary entailment on established variables does not admit arbitrary fresh outputs. A well-formed definition must extend the existing interpretation conservatively: each valuation satisfying the prior state must admit an extension satisfying the definition, without adding a restriction on the old variables. A typed total function whose inputs belong to the prior state and whose outputs are fresh provides such an extension. The checker handles definitions separately from claims about established quantities. Source alignment, interpretation fidelity and the chosen theory remain assumptions of the encoded obligation. A decided solver answer concerns that obligation. Parsing failures, typing failures and undecided queries remain unresolved.

A.1 Extraction and source records

The representation interface takes the problem q_p, the admitted prefix $Y_{<t}$ with its established quantity bindings, and the next step Y_t with its supplied text boundaries. It records typed quantity nodes,

relation occurrences, definition or assertion roles, and source locations. A quantity reference binds to an established identifier or introduces a fresh quantity. A definition relates fresh outputs to its inputs; assertions are claims checked against the preceding theory and certified definitions.

The step expression is

$$\psi_t = \bigotimes_{\alpha \in \mathscr{A}_t} \varphi_{t,\alpha}, \qquad [\![\psi_t]\!] \equiv \Psi_t,$$

where $\mathscr{A}_t$ is a finite set of fresh occurrence identifiers and Ψ_t is the conjunction of the step's represented relations. Each occurrence $\varphi_{t,\alpha}$ records its relation, typed variables, role, definition dependencies and source records $\mathrm{src}(\varphi_{t,\alpha})$. Source spans identify the text expressing the relation; one occurrence may have several such spans. Repeated relations remain distinct occurrences. The typed product retains these records while its logical content conjoins the relations. Missing quantity or source bindings leave the step unresolved. Appendix A gives the occurrence and product semantics.

Full faithful-extraction contract. Assumption 1 requires extraction to succeed and preserve the problem conditions and every encountered step's meaning in the declared theory, including required background constraints, quantity identities and source coordinates. It distinguishes fresh definitions from assertions, preserves their scopes and dependencies, and orders definitions before their uses. Thus satisfying $[\![\psi_0]\!]$ and $[\![\psi_t]\!]$ agrees with satisfying the represented problem and step, respectively. This contract concerns the meaning of the written response. A typed, executable expression does not by itself certify that meaning or the validity of its assertions.

A.2 Generation and step segmentation

The decoder interface writes the fixed model as M_θ, where θ denotes its parameters. The input context contains the problem statement and the prescribed instructions. The configuration η specifies the decoding rule, sampling settings, generation limit and stopping conditions. Before step t, z_t denotes the model-side generation state for that context and the generated prefix, including any cached state used to continue generation.

For each response, the data interface supplies an ordered inventory of steps with their text and source spans. These boundaries remain fixed during error localization, so $t^\star$ refers to the earliest invalid supplied step. A boundary need not isolate a single assertion. Source spans connect factor occurrences to that inventory; token alignment separately determines the rollback point. Neither segmentation nor source alignment establishes faithful extraction.

The saved online rollout proposes boundaries at paragraph breaks, explicit step labels or an end-of-sequence marker. It buffers text until the accumulated chunk passes delimiter-balance and substantive-content checks, retaining an unfinished suffix for further generation. These are structural checks, not certificates of a complete mathematical argument. The recorded step inventory and this online boundary policy are interfaces to error localization; we do not claim a general method for finding semantic step boundaries.

A.3 Admission, indexing and first stopping step

Insertion proposes $\delta_p(s_j, \iota[\psi_t]) = s_j \otimes \psi_t$ under the product semantics in Appendix A. Shared quantity identifiers must have equal sorts, and new factor-occurrence identifiers must be disjoint from those already present. The admission gate $G_\kappa(s_j, \iota[\psi_t])$ returns admit, reject or unresolved for the state and proposed operation under the fixed checker configuration κ. Error localization commits s_{j+1} only on admission. The index j counts admitted operations; it equals $t-1$ only on an uninterrupted replay with one operation per response step. The symbolic state s_j differs from the model-side generation state z_t.

Establish satisfiability of $B = [\![s_j]\!]$ before checking a step. Split $[\![\psi_t]\!] = \Delta_t \wedge A_t$ into definitions and assertions. Write $\Delta_t = d_1 \wedge \cdots \wedge d_m$ in dependency order and let $B^{(0)} = B$. For each d_i, require freshness, well-formedness and domain validity, and certify that every valuation satisfying $B^{(i-1)}$ extends to one satisfying $B^{(i)} = B^{(i-1)} \wedge d_i$. These certificates imply that $B_t^+ = B^{(m)}$ is satisfiable. They are temporary until the assertions pass; rejection or an unresolved check leaves s_j unchanged. A total typed function on established inputs with fresh outputs supplies a sufficient certificate. Failure to obtain such a certificate need not mean that no conservative extension exists.

The assertion gate uses the following outcomes:

$$\begin{array}{lll} B_t^+ \wedge \neg A_t \text{ is UNSAT} & \Rightarrow & \text{admit,} \\ B_t^+ \wedge \neg A_t \text{ is SAT}, \quad B_t^+ \wedge A_t \text{ is UNSAT} & \Rightarrow & \text{reject: inconsistent,} \\ B_t^+ \wedge \neg A_t \text{ is SAT}, \quad B_t^+ \wedge A_t \text{ is SAT} & \Rightarrow & \text{reject: unsupported.} \end{array}$$

A satisfiable entailment query supplies a countermodel satisfying the trusted state and current definitions while violating an assertion. If its subsequent consistency query is undecided, the rejection remains justified but is not classified. Under Assumptions 1 and 2, a decided consistency query classifies the step as inconsistent exactly when the problem, admitted prefix and full current step have no joint model. An unsatisfiable core from this query explains the conflict; a core from the negated-assertion query instead explains entailment. Neither solver core is necessarily minimal. Error localization returns the inconsistency core; Steering performs its minimization and MUS certification as specified in Section 4.

Parsing failures, missing source bindings, malformed products, failed definition certification and undecided entailment queries leave error localization unresolved. The implemented checker can label some definition failures as refusals; that operational label alone is not a countermodel or a proof of semantic invalidity. The main-text guarantees apply to the stated semantic gate, subject to the implementation gap in Appendix A.4.

First stopping step. The first nonadmitted step $\hat{t}$ freezes the checkpoint; no later step is admitted until it is resolved. Proposition 1 identifies $\hat{t}$ with the semantic first error $t^\star$ only under faithful extraction, sound solver decisions, structurally compatible products, certified definitions and decided queries through that step. An earlier unresolved check can stop the procedure before $t^\star$. The proof and its scope are given in Appendix D.

Recorded example and displayed steps. The problem relations in Figure 2(a), $2z + 4s = 10$ and $s = 2z$, determine $z = 1$ and $s = 2$. Step 3's assignments $s = 1$ and $z = 3$ are therefore inconsistent with the trusted state, while the separate arithmetic identity $5 - 2 \cdot 1 = 3$ is true. Admission still applies to the whole supplied step. The figure retains the quantity bindings of Figure 1 and condenses the saved localization run's four admitted extracted steps and rejected fifth step into three displayed steps.

A.4 Definition-check implementation

The implemented validator checks freshness, known inputs, acyclic dependencies and encoder acceptability. For variable division, the encoder adds a nonzero-denominator guard without generally proving that the old state entails it. Thus a definition can receive an admission label without a conservative-extension certificate: $y := 1/x$ excludes old models with $x = 0$. The validity guarantees require these domain obligations to be discharged, or definitions to be restricted to functions total over the retained state.

B EGS and Regeneration Details

Relative conflicts and correction sets. Using the notation of Section 4, all subset checks keep B fixed and range over $\mathcal{C}$. A MUS relative to B is a subset $U \subseteq \mathcal{C}$ such that $B \wedge \bigwedge U$ is unsatisfiable and removing any member makes it satisfiable. A maximal satisfiable subset (MSS) is a subset $L \subseteq \mathcal{C}$ that is satisfiable with B and has no satisfiable proper superset within $\mathcal{C}$. The MCSs R defined in Section 4 are exactly the complements of these MSSs: $R = \mathcal{C} \setminus L$. Minimality and maximality refer to set inclusion, not cardinality (Liffiton & Malik, 2013; Liffiton et al., 2016).

These relative checks keep all constraints in B fixed. By contrast, the absolute support certificates constructed below explicitly include the background factors needed to explain a conflict.

B.1 Localization and repair handoff

Algorithm 1 checks each tentative step with the gate in Appendix A.3. The ordered expressions of committed steps accompany s_j and are updated at each commit. Extraction failure, unsupported assertions and unresolved checks stop the loop without changing the admitted state.

At a certified inconsistency, s_j, Π and ζ remain frozen while EnumerateRepairs$(\psi_0, (\psi_i)_{i=1}^t, s_j)$ constructs $(\mathcal{R}_t, \sigma)$. The inconsistency core, including its background factors, is reduced to an absolute MUS U, certified by $\mathrm{UNSAT}(\bigwedge U)$ and $\mathrm{SAT}(\bigwedge(U \setminus \{\varrho\}))$ for every $\varrho \in U$. Unresolved certification yields no executable record and an incomplete status. The MUS explains the conflict but does not restrict the enumeration universe.

The loop pauses until a record $r \in \mathcal{R}_t$ is supplied; it does not rank or select records. Execution checks the record's source and frozen-state bindings, then uses Algorithm 2 to regenerate the rejected step. An admitted replacement returns control to Algorithm 1.

B.2 Entailment filtering and search scope

Fix the satisfiable committed theory B and the finite set $\mathcal{C}$ of labelled formula factors in the rejected step. The problem and all admitted steps remain fixed in B. MARCO enumerates MCSs $R \subseteq \mathcal{C}$ relative to B. For each candidate, let $L = \mathcal{C} \setminus R$.

Let Δ_L and A_L conjoin the retained definitions and assertions, respectively. Every definition used by a retained claim must remain available in B or in L. The checker validates retained definitions in dependency and source order as fresh conservative extensions with satisfied domain obligations and unchanged established quantity bindings. It then requires

$$B \wedge \Delta_L \models A_L.$$

Write $E_B(L)$ for successful completion of these checks. A candidate R is admissible only when $E_B(L)$ holds. Missing dependencies, failed certification and undecided queries do not establish admissibility.

A repair record additionally requires checked support certificates and valid source bindings. Thus $\mathcal{R}_t$ contains the successfully constructed records for admissible MCSs. The completion status σ distinguishes exhaustive enumeration from a search stopped before completion. An initial MUS supplies an explanation but does not restrict the search universe to that MUS.

B.3 Factor refinement and support certificates

The search universe $\mathcal{C}$ contains all labelled formula factors of the rejected expression ψ_t. Formula decomposition must preserve its semantics; in particular, a disjunction is not split into jointly asserted factors. Declarations remain in the typed signature. Repeated formulas retain distinct occurrence labels.

Let $\mathcal{B}$ be the labelled factors of the problem and admitted prefix, so that $\bigwedge \mathcal{B} = B$. These factors remain fixed throughout repair search. For an admissible MCS $R \subseteq \mathcal{C}$, let $L = \mathcal{C} \setminus R$. The survivor checks are those in Appendix B.2. The following construction derives support certificates for the factors in R.

Single-factor add-back certificate. For each deleted factor $\varrho \in R$, require

$$\mathrm{SAT}\Big(B \wedge \bigwedge L\Big), \qquad \mathrm{UNSAT}\Big(B \wedge \bigwedge L \wedge \varrho\Big).$$

Fixed-order deletion removes dispensable factors from the explanation query over $\mathcal{B} \cup L \cup \{\varrho\}$, never removing ϱ. The resulting certificate has the form

$$U_\varrho = C_\varrho \cup J_\varrho \cup \{\varrho\}, \qquad C_\varrho \subseteq \mathcal{B}, \qquad J_\varrho \subseteq L.$$

Its conjunction is unsatisfiable, and deleting any member makes it satisfiable. Thus U_ϱ is inclusion-minimal. The support C_ϱ comes from the retained theory; J_ϱ records surviving factors of the target step that the explanation still needs. In general, $\mathrm{UNSAT}(\bigwedge C_\varrho \wedge \varrho)$ does not follow. Removing J_ϱ from the explanation requires a separate successful query. The procedure never removes problem constraints from the actual verification theory; explanation minimization only selects what to show the model.

Correction conditions. For the support certificates stored in a repair record, collect

$$C_R = \bigcup_{\varrho \in R} (C_\varrho \cup J_\varrho) \ \cup \ \mathcal{V}_R,$$

where $\mathcal{V}_R$ contains any certified value equalities included under Appendix B.5. The factors in R are excluded. The record preserves source identities, quantity bindings, definition dependencies and the certificates associated with these conditions. The notation C_R refers to the conditions of the supplied record; different support certificates for the same deletion set may yield different conditions.

A condition drawn from L may depend on retained definitions and need not follow from B alone. The logical conditions and their rendered conditioning text are distinct: rendering must preserve the meaning and bindings of every displayed condition. The generated step is re-extracted and checked against B; C_R is not added to the trusted state as a substitute for that check.

The recorded renderer includes retained assertions with their entailment status under B and validated retained definitions. It supplies sort, domain and alias bindings but does not render every retained definition equation; the conditioning text is therefore not a self-contained proof for fresh-symbol assertions. Re-extraction and re-checking must recover the full definition scope. For an intrinsic conflict $U_\varrho = \{\varrho\}$, no external support exists: the renderer records this and may request bounded recomputation, but supplies no invented equation.

B.4 Condition rendering, cached regeneration and revalidation

The repair record binds s_j, the MCS R, its source locations and C_R. Factor spans identify the rejected step and validate the record; execution rolls back to the beginning of that whole step. Stale checking results, unresolved source mappings or mismatched checkpoints block execution.

The renderer encodes C_R with quantity sorts, domains and aliases; optional source quotations are marked as provenance. Rejected factors, arbitrary solver assignments, reference answers and future reasoning are excluded. Numerical equalities require the checks in Appendix B.5. An origin map separates these auxiliary conditioning tokens from visible reasoning.

Every attempt restores a clone of ζ bound to the same prefix, positions and model, forwards C_R through ordinary cached decoding, and generates a complete replacement $\tilde{y}$. Its extraction $\tilde{\psi}$ must pass the admission gate and $\mathrm{SAT}(B \wedge [\![\tilde{\psi}]\!])$. This additional SAT check is redundant under the semantic assumptions and does not replace definition certification. Invalid boundaries, failed extraction or undecided checks prevent commitment. Checkpoint mutation, cache-provenance mismatch or a failed required replay comparison blocks generation.

Support entries remain with their descendants during the attempt. Retries discard both by restoring ζ. After acceptance, the controller commits $\tilde{y}$ and $\tilde{\psi}$, updates the ordered committed record, and rebuilds the cache from the problem context and accepted visible text, excluding support and discarded look-ahead text. Neither an admissible record nor a finite retry budget guarantees success.

B.5 Certified values from the pre-error state

Certified values may supplement C_R for already-bound quantities mentioned by factors in R. The configured answer symbol is excluded; this identifier-based rule does not establish exclusion of every semantic alias. The theory B is the unchanged committed state before the rejected step.

First establish $\mathrm{SAT}(B)$ and obtain a candidate value v for x from a model of B. The value must be representable as a closed, well-typed exact term with model-independent denotation. Then query $B \wedge x \neq v$. Only an unsatisfiable result certifies $B \models x = v$ and licenses including that equality in $\mathcal{V}_R$ and hence C_R. A satisfiable result shows that the proposed value is not uniquely determined; an undecided or failed query supplies no equality. A factor in R supplies candidate quantity identities only. Its equation, the reference answer and the future suffix never enter B.

In the saved salt-scrub run, the established relations imply $s = 2$ and $z = 1$. The model then generates $2s = 4$ ounces of oil; the oil answer is not supplied as a hint. The record uses whole-step rollback on one problem, one seed and only the EGS condition, with no rerun of a matched control. Its extraction procedure also differs from the earlier unsuccessful attempt. It therefore demonstrates this execution path without estimating accuracy or attributing the outcome solely to the numerical hints.

Algorithm 2 Regeneration and re-checking of a supplied repair candidate

Require: Frozen s_j, Π, ζ; admissible $r \in \mathcal{R}_t$ with matching bindings; retry budget N
Ensure: Admitted replacement and updated state, or unchanged state with diagnostics

$(v, d) \leftarrow$ (not checked, no attempt)
for $n = 1, \ldots, N$ **do**
  Restore ζ and forward conditioning tokens for C_R from r
  Generate a complete visible replacement $\tilde{y}$
  $(\tilde{\psi}, d) \leftarrow \textsc{Extract}(\tilde{y})$
  if extraction fails **then**
    $v \leftarrow$ unresolved; **continue**
  end if
  $(v, d) \leftarrow \textsc{Check}(s_j, \tilde{\psi})$
  if $v =$ admit **then**
    $(v_{\mathrm{sat}}, d) \leftarrow \textsc{CheckSat}([\![s_j]\!] \wedge [\![\tilde{\psi}]\!])$
    if $v_{\mathrm{sat}} =$ sat **then**
      Commit $\tilde{\psi}, \tilde{y}$; update the ordered record, s_{j+1} and Π; increment j
      Rebuild ζ from the problem context and accepted visible text
      return (admitted, (s_j, Π, ζ))
    end if
  end if
end for
Restore ζ; **return** (budget exhausted, (s_j, Π, ζ, v, d))

C Cache Conditions

The cache argument assumes fixed parameters and positional rules, deterministic causal inference, and complete retention of earlier key/value entries and cache metadata bound to the same model, prefix and positions. Copying and cropping must preserve every layer's state. Sliding-window, quantized or recurrent caches require separate justification. Under exact arithmetic, forwarding new conditioning or reasoning tokens preserves the earlier prefix entries. Appendix D.2 proves this property and exact cached recomputation; neither implies bitwise equality across floating-point execution schedules.

During a repair attempt, support entries remain with all descendants that attended to them. A retry restores the pre-support checkpoint and discards both. After admission, the controller rebuilds the cache from accepted visible text, as specified in Appendix B.4.

Numerical checks. A historical GPU/BF16 comparison failed five replay-versus-prefill checks at the replacement position, measuring logits, hidden outputs and new KV entries. Later small-model controls preserved prefix projections and retained entries in twenty configurations but did not reproduce that failure. Its cause remains unresolved; these checks do not establish retained-prefix corruption.

D Proofs

Notation and initialization. Write $B_0 = [\![\psi_0]\!]$ for the problem theory. Let $\Pi_j = (Z_1, \ldots, Z_j)$ denote the textual prefix after j commits, and let $B_j = [\![s_j]\!]$ be its committed theory. The Z_i are the actually committed steps, which can include replacements. Let $\mathcal{L}_j$ be their accumulated typed quantity signature, including the problem signature $\mathcal{L}_0$. All valuations below use the fixed sort domains and interpretation of the declared theory. If ν is a valuation on a larger signature, $\nu|_{\mathcal{L}}$ denotes its restriction to $\mathcal{L}$. The initial state $s_0 = \psi_0$ contains the problem and no committed step, so Π_0 is empty. Algorithm 2 resumes from the admitted prefix and matching state produced by Algorithm 1; those admissions justify its initialization. An arbitrary satisfiable state or a prefix merely labelled admitted does not supply this initialization.

Semantic equivalence. For a fixed background theory, interpret each query's B and A over a common typed signature. The standard semantic equivalences are (Barrett & Tinelli, 2018, Section 1.1)

$$\begin{aligned} B \models A \quad &\iff \quad \mathrm{UNSAT}(B \wedge \neg A), \\ B \not\models A \quad &\iff \quad \mathrm{SAT}(B \wedge \neg A). \end{aligned}$$

Model extension. Let B be a theory on signature $\mathcal{L}$, and let Δ contain finitely many definitions in dependency order, possibly none. Under Assumption 2, each certificate extends every preceding model without changing established quantities. Successive application therefore extends every model of B to $B \wedge \Delta$ while preserving $\mathcal{L}$. Assertion-only fresh quantities can be assigned arbitrary values in their fixed nonempty sorts; these unconstrained declarations preserve $B \wedge \Delta$. If $B \wedge \Delta \models A$ on the full signature, each resulting extension also satisfies A. Thus every old model extends to $B \wedge \Delta \wedge A$, and satisfiability is preserved.

Admission invariant. Under Assumptions 1 and 2 and a satisfiable B_0, every committed prefix satisfies all of the following:

1. Π_j is valid under Definition 1.
2. B_j is exactly the problem theory conjoined with the faithfully represented committed steps, with unchanged established quantity bindings.
3. B_j is satisfiable, and every model $\nu_0 \models B_0$ has an extension $\nu_j \models B_j$ such that $\nu_j|_{\mathcal{L}_0} = \nu_0$.

Proof. Induct on commits. At $j = 0$, the empty prefix is valid, $B_j = B_0$ is satisfiable, and the identity extension establishes the invariant.

For an admitted step Z with content $\Delta \wedge A$, the typed product is defined. Its dependency-ordered certificates ensure fresh, well-formed conservative definitions with satisfied domain obligations. Full faithful extraction preserves scopes, bindings and source definition/use order; it cannot repair an invalid earlier use by reordering. Thus the definitions are admissible. With $B^+ = B_j \wedge \Delta$, the sound unsatisfiable answer for $B^+ \wedge \neg A$ gives $B^+ \models A$ by the semantic equivalence. Faithfulness and prefix validity give $V_p(Z \mid \Pi_j) = 1$. Whole-step insertion yields exactly $B_{j+1} = B_j \wedge \Delta \wedge A$; rejected or temporary content is not committed. Each original model extends to B_j by induction and then to B_{j+1} by the model-extension property above, preserving all established quantities. Since B_0 has a model, B_{j+1} is satisfiable. All three items follow. □

Proof of Proposition 1. Fix Y and $t^\star$ under the proposition's hypotheses. Induct on $t < t^\star$: initially the prefix is empty. If $Y_{<t}$ has been admitted, the invariant gives its valid, satisfiable state. Structural compatibility and certified definitions let Y_t reach the assertion gate. Since Y_t is valid, faithful extraction gives $B_t^+ \models A_t$. The semantic equivalence makes $B_t^+ \wedge \neg A_t$ unsatisfiable. The query is decided and the solver is sound, so the whole step is admitted and preserves the invariant. Thus every $t < t^\star$ is admitted, giving $\hat{t} \geq t^\star$.

At $t^\star$, the product is again compatible and its certified definitions are admissible under the faithful-extraction and definition-check assumptions. The prefix is valid, so the step's invalidity must lie in an assertion: $B_{t^\star}^+ \not\models A_{t^\star}$. By the semantic equivalence, $B_{t^\star}^+ \wedge \neg A_{t^\star}$ is satisfiable. Its decided, sound query rejects the step, giving $\hat{t} \leq t^\star$. Therefore $\hat{t} = t^\star$. □

Countermodels and consistency classification. After certified definitions and a valid committed prefix, a countermodel $w \models B^+ \wedge \neg A$ satisfies the problem, prefix and definitions but falsifies at least one current assertion. Faithful extraction gives $V_p(Z \mid \Pi_j) = 0$. This need not be a joint contradiction. A correct satisfiable answer for $B^+ \wedge A$ means compatible but unsupported; a correct unsatisfiable answer means the problem, prefix and full step have no joint model. An undecided consistency query leaves classification unresolved without invalidating w. Failed definition certification remains unresolved and supplies neither the extension premise nor such a countermodel.

Scope of exact localization. The proof uses $j = t - 1$ only on this uninterrupted replay with one admitted operation for each earlier supplied step. Regeneration and failed attempts do not share that indexing rule. If the response has no invalid step and the same structural, certification and decision

conditions hold throughout, the induction admits every step; no first nonadmitted index then exists unless a separate end-of-response sentinel is defined. Certified definitions at $t^\star$ are essential: the proposition covers a first invalid assertion after admissible definitions. It does not assert completeness for invalid definitions, uncertified valid definitions or earlier unresolved queries.

Proof of Theorem 1. Use the initialization and invariant above, and induct over commit events rather than generation attempts. Between commits, the algorithm can generate a tentative step, check it, prepare a handoff, enumerate records, pause, or retry a replacement. None of these operations commits new textual or symbolic content. An unsuccessful attempt retains s_j and Π_j and restores the bound checkpoint. Correct source and checkpoint bindings ensure that the preserved text is the same prefix whose theory is B_j.

For a directly admitted step, the only commit appends the checked text and its extracted expression; the admission invariant applies. For a regenerated step, the algorithm checks the newly generated text against the same frozen state s_j, and commits only if that step is admitted and the required consistency check succeeds. The admission proof is independent of how the proposed text was generated, so it applies to this replacement as well. In particular, the support conditions and rejected factors are not added as trusted premises in place of re-checking the visible replacement.

Therefore each commit extends a valid prefix by a valid step. A pause, rejection, unresolved outcome or exhausted budget cannot change this invariant. Every committed prefix is valid, including after an admitted regenerated step, as claimed. No termination or successful generation conclusion is used in this induction. □

Satisfiability and terminal assertions. The invariant proves more than satisfiability: for every finite committed state, each original problem model has an extension to that state. Conversely, any model of B_j restricts to a model of B_0, because B_0 remains a conjunct. Thus the restrictions to $\mathcal{L}_0$ of the models of B_j are exactly the models of B_0.

Suppose an admitted terminal step asserts $a = v$, where a belongs to $\mathcal{L}_0$ and v is a numeral of the appropriate sort. Let B_k be the committed theory including that step. Fix any $\nu_0 \models B_0$. By the invariant it extends to some $\nu_k \models B_k$. The terminal assertion is a conjunct of B_k, so $\nu_k(a) = v$. Restriction preserves a and the interpretation of the numeral, hence $\nu_0(a) = v$. Since ν_0 was arbitrary, $B_0 \models a = v$; faithful extraction transfers this entailment to the original problem. More generally the same argument applies to any entailed terminal formula using only the original signature.

This conclusion requires the original-signature condition. A valid fresh definition can determine a newly introduced quantity without establishing that it is the problem's requested answer. A terminal marker alone is also insufficient: the asserted quantity must match the requested target, and the answer form in a_p requires its separate check. The theorem guarantees valid committed reasoning and does not guarantee completion of those obligations.

D.1 Repair certificates and finite enumeration

Enumeration and admissibility. Fix satisfiable B, the finite labelled factor set $\mathcal{C}$ of the rejected step, and the typed signature and declarations. Apply MARCO with the relative oracle $X \mapsto \mathrm{SAT}(B \wedge \bigwedge X)$. Each satisfiable seed grows to an MSS and blocks its subsets; each unsatisfiable seed shrinks to a MUS and blocks its supersets. With correct, decided queries, growth, shrinking and blocking, exhausting the map returns all relative MCSs as MSS complements (Liffiton et al., 2016, Sec. 5.1, Algorithm 1, Theorems 2–4 and Lemma 3). Early stopping or unresolved queries do not establish completeness.

Fix the survivor checker and quantity bindings. Applying its resolved predicate $E_B(\mathcal{C} \setminus R)$ to every raw MCS retains exactly the passing deletions. A sound checker need not accept every semantically valid survivor set. A record for every passing deletion additionally requires successful support-certificate and source-binding construction; enumerating deletions does not enumerate every certificate encoding. This family excludes nonminimal deletions and replacement expressions, so an empty family does not establish that no correct replacement exists.

For survivors $L = \mathcal{C} \setminus R$, certify Δ_L under the model-extension property above in dependency and source order, with domain obligations discharged and established quantity bindings preserved.

Evaluate $B \wedge \Delta_L \models A_L$ on the full retained signature. Every old model then extends to satisfy all survivors, so the retained theory is satisfiable. The same property covers no definitions and unconstrained fresh declarations over nonempty sorts; joint satisfiability alone does not discharge the extension obligation.

Absolute support certificates. Let $\mathcal{B}$ be the finite labelled factors of B, disjoint from $\mathcal{C}$. For a deleted ϱ, the add-back test certifies that $\mathcal{B} \cup L$ is satisfiable and $\mathcal{B} \cup L \cup \{\varrho\}$ is unsatisfiable. Complete a fixed-order deletion pass over every factor except ϱ, deleting exactly when the remaining conjunction is certified unsatisfiable. With correct, decided tests, the result $U_\varrho = C_\varrho \cup J_\varrho \cup \{\varrho\}$ remains unsatisfiable, where $C_\varrho \subseteq \mathcal{B}$ and $J_\varrho \subseteq L$. Each other retained factor had a satisfiable removal test; its final removal leaves a subset of that tested conjunction and remains satisfiable. Removing ϱ leaves a subset of the satisfiable base $\mathcal{B} \cup L$. Thus U_ϱ is an absolute MUS.

These queries include every premise, with no hidden hard background; relative MUS queries instead keep B fixed. Undecided removal tests do not certify MUS minimality. Explanation minimization does not change B, and omitting J_ϱ requires a separate successful check.

Certified values. For satisfiable pre-error B, an already-bound quantity x, and a closed, well-typed exact term v with model-independent denotation, $\mathrm{UNSAT}(B \wedge x \neq v)$ gives $B \models x = v$ by the semantic equivalence above. A single model only proposes v; an undecided query supplies no hint. Rendering must preserve the certified value and quantity binding.

D.2 Causal prefix preservation and exact recomputation

Decoder contract. Consider a fixed deterministic causal decoder in exact arithmetic. Let $u = (u_1, \ldots, u_n)$ be its complete token stream, including context, auxiliary conditioning and visible reasoning, with fixed positions p_i. At each layer, a position's state depends only on its preceding-layer state and the permitted keys and values at positions no later than itself. Normalization, positional transforms, attention, residual and feed-forward operations are fixed. The cache contains all earlier layers' reusable keys, values and required metadata. The contract excludes unrecorded recurrent state, changed positions and earlier outputs that depend on the eventual sequence length.

Prefix preservation. Two streams with identical tokens and positions before index ℓ have identical hidden states and cached keys and values at every position $i < \ell$.

Proof. At layer zero, the token and position inputs agree. Assume the earlier positions' states agree on entry to a layer. Their projections and positional transforms then agree. Causal attention at $i < \ell$ reads only positions $j \leq i$, whose keys and values also agree. Attention and the deterministic residual and feed-forward updates therefore produce equal output states. Induction over layers proves the claim. □

Exact cached continuation. Take a complete cache for $u_{<\ell}$ bound to the same model, positions and layer conventions. Forwarding a new token $\tilde{u}_\ell$ produces the same hidden states, new KV entries and next-token logits as a fresh computation on $(u_1, \ldots, u_{\ell-1}, \tilde{u}_\ell)$, in exact arithmetic.

Proof. Prefix preservation identifies the retained cache with the earlier entries in the fresh computation. Both executions receive the same new token and position. If their current states agree on entry to a layer, the current projections agree, and attention reads identical current and retained keys and values. The layer outputs and new cache entries agree. Induction through all layers and the output projection gives equal logits. Repeating this argument proves equality for any fixed continuation token sequence, including a support block followed by visible reasoning. Equal generated continuations additionally require identical deterministic token selection and tie-breaking; independent random sampling is not covered. Floating-point execution schedules need not be bitwise equivalent. □

Support lifetime and checkpoint restoration. Support tokens preserve the earlier prefix, but their cached descendants may depend on them. Deleting only support rows therefore does not reconstruct a cache for the visible text. Restoring the immutable pre-support checkpoint discards the support and all descendants together. After acceptance, a fresh computation on the visible text constructs the next

checkpoint without support. This changes the context for later decoding while retaining the accepted text, which was generated under the support conditions.

E Evaluation Setup

E.1 Datasets

We evaluate MOS on four mathematical word-problem datasets: GSM8K (Cobbe et al., 2021), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2020), and GSM-Symbolic (Mirzadeh et al., 2025). GSM8K contains 7,473 training and 1,319 test problems involving multistep arithmetic; SVAMP comprises 1,000 problems emphasizing variations in wording and quantity relationships; ASDiv contains 2,305 problems with diverse linguistic patterns and problem types; and the standard GSM-Symbolic benchmark contains 5,000 problems generated from 100 symbolic templates.

We use Qwen2.5-7B-Instruct to generate step-by-step solutions and evaluate MOS on subsets of problems that the model answers incorrectly: 87 from GSM8K, 28 from SVAMP, 21 from ASDiv, and 73 from GSM-Symbolic, totaling 209 cases. On these cases, we measure how often MOS correctly locates the first error in the original solution and how often steering recovers the correct final answer.

E.2 Baseline adaptations

We use mathematical-reasoning adaptations of Reflexion (Shinn et al., 2023), SatLM (Ye et al., 2023), and Logic-LM (Pan et al., 2023). Reflexion uses verbal feedback to diagnose or revise an incorrect solution. SatLM formalizes mathematical relations and uses solver results to identify conflicts or obtain an answer. Logic-LM translates the problem into a symbolic formulation and refines it using tool feedback. For localization, the symbolic baselines map their diagnostics to trace steps. For answer recovery, SatLM returns the solver-derived answer; Logic-LM solves a formulation generated from the problem; and Reflexion revises the original failed response.

E.3 Experimental protocols and metrics

RQ1: First-error localization. For error localization, all methods evaluate the same problems and original Qwen2.5-7B-Instruct solutions with fixed step boundaries, allowing localization performance to be compared on shared reasoning traces. MOS and SatLM use symbolic representations extracted by GPT-5.6 [1]. The extraction prompt instructs the model to faithfully represent the original reasoning traces, including any errors they contain. Localization accuracy requires exact agreement with the reference first-error index; an earlier or later prediction, or no prediction, counts as incorrect.

RQ2: Answer recovery through steering. We compare MOS with the baselines on the same subsets of incorrectly answered problems using four backbones: Qwen2.5-7B, Llama3.1-8B, GPT-OSS-20B, and Qwen3.8-27B. MOS and SatLM use the GPT-5.6-extracted symbolic representations described above. For steering, MOS guides generation from the retained prefix preceding the first invalid step. For each dataset, answer recovery accuracy is the proportion of initially incorrect cases for which the final generated answer matches the reference answer, irrespective of the validity of the intervening reasoning. The reported totals are unweighted means of the four dataset accuracies.

Backbone identity and evaluation scope. The reported `Qwen3.8` is a deployment label whose configuration declares `qwen3_5`. Recovery results combine saved closed-loop and one-shot runs and score final answers regardless of gate acceptance; they do not estimate uniform end-to-end success of Algorithm 1. The evaluated representations include reviewed oracle-assisted inputs, and input conditions differ across methods, as discussed in Section 7.

[1] The model is accessed through `https://developers.openai.com/api/docs/models/gpt-5.6-sol`

Table 3: Effect of MOS feedback on single-pass answer recovery with Qwen3.8-27B. Both conditions resume from the same pre-error prefix. Scores are percentages.

Condition	GSM8K (87)	SVAMP (28)	Total (115)
With MOS feedback	**93.10**	**100.00**	**94.78**
Without MOS feedback	80.46	92.86	83.48

Table 4: Solver-answer accuracy (%) on frozen MOS formulations from 28 SVAMP cases per backbone. Scores evaluate the solved target value, rather than the model's generated final answer. All three solvers receive the same mathematical constraints.

Backbone	Z3	cvc5	dReal ($\delta = 10^{-6}$)
Qwen2.5-7B	92.86	92.86	92.86
Llama3.1-8B	92.86	92.86	92.86
GPT-OSS-20B	100.00	100.00	100.00
Qwen3.8-27B	96.43	96.43	96.43

F Ablation Study

Effect of fine-grained feedback. MOS represents mathematical expressions as typed factor graphs, connecting individual mathematical relations to the quantities and reasoning steps they describe. This representation enables feedback grounded in localized conflicts and the established conditions that a correction must satisfy. We investigate whether this fine-grained feedback helps steer the model toward a correct answer when generation resumes from the pre-error prefix.

We compare continuation with and without MOS feedback using Qwen3.8-27B on 87 GSM8K and 28 SVAMP cases. Both conditions share the same problem, retained prefix, steering point, and paired random seed. The feedback condition receives MOS correction guidance, where the ablated condition continues without this guidance. Each condition generates one completion at temperature 0.2 and top-p 0.95, with a maximum of 4,096 new tokens and no subsequent checking or iterative repair. We measure answer recovery by final numerical answer accuracy.

Table 3 shows that MOS feedback improves answer recovery on both datasets, increasing accuracy from 83.48% to 94.78%, a gain of 11.30 percentage points. Because both conditions already resume from the same pre-error prefix, the improvement demonstrates the additional value of supplying correction guidance at that point. The results support the role of MOS feedback in translating symbolic diagnosis into actionable guidance: the established mathematical relations provide concrete conditions for the continuation, helping the model recover a correct answer while preserving the preceding reasoning.

Effect of solver choice. We examine whether the solution of MOS symbolic formulations depends on the choice of solver backend. We compare Z3, cvc5, and dReal on 28 SVAMP cases for each of the four backbones, yielding 112 frozen formulations. Each formulation combines the problem constraints with reasoning constraints retained from steering or extracted from the saved continuation. We hold these formulations fixed across solvers and apply equivalent syntax normalization, so the comparison isolates backend differences at the symbolic solving stage.

We measure solver-answer accuracy by exact numerical agreement between the solved target value and the reference answer, retaining all 28 cases per backbone in the denominator. Each query has a 10-second time limit. For dReal, we use $\delta = 10^{-6}$ and score the midpoint of the returned target interval. Its δ-satisfiability verdict has a different guarantee from the exact satisfiability verdicts of Z3 and cvc5.

Table 4 shows identical accuracy across the three solver backends for every backbone. All three recover the correct target value for the same 107 formulations and identify the remaining five as unsatisfiable, with no timeouts or unknown results. Z3 and cvc5 additionally establish target uniqueness for all 107 solved formulations; dReal returns singleton target intervals on these inputs. This agreement supports the portability of MOS's symbolic formulations across the tested solvers.

G Extended Related Work

Prompt-based self-correction. Self-Refine and Reflexion use natural-language feedback to condition subsequent attempts, drawing on self-critique or task feedback (Madaan et al., 2023; Shinn et al., 2023). Such feedback can guide revision but does not itself certify whether a claim follows from the problem; intrinsic self-correction can also fail without external feedback (Huang et al., 2024). Our approach grounds correction in symbolic checks against the problem and admitted reasoning, returning source-linked countermodels or conflict evidence for rejected claims.

Search and process supervision. Learned outcome and process verifiers score candidate solutions or intermediate steps (Cobbe et al., 2021; Lightman et al., 2024), while Tree of Thoughts uses model evaluations to guide search and backtracking (Yao et al., 2023). These scores guide candidate selection without specifying certified replacement conditions. ProcessBench evaluates first-error localization (Zheng et al., 2025), and supplying error locations can improve backtracking-based repair (Tyen et al., 2024).

Tool- and solver-guided reasoning. CRITIC uses external tools to inform revision (Gou et al., 2024), and LLM-Modulo frameworks couple candidate generation with external verification and feedback (Valmeekam et al., 2023; Kambhampati et al., 2024). Logic-LM and SatLM translate problems into symbolic formulations for solver-based answer derivation; Logic-LM also refines formulations using solver errors (Pan et al., 2023; Ye et al., 2023). A solver-derived answer does not itself establish the validity of the intervening textual steps, which our approach checks sequentially against the admitted state.

Neurosymbolic diagnosis. ARc checks natural-language question–answer pairs against an SMT policy model and provides relevant policy rules and countermodels (An et al., 2026). VeriCoT checks each reasoning step by entailment against accumulated premises (Feng et al., 2026). It distinguishes contradictions, ungrounded steps and translation failures, and attempts premise generation before declaring a step ungrounded. Its published algorithm continues after errors; in the ungrounded branch, it adds the proposed premise and current formula to the state used by later checks. Our approach stops at the first unadmitted step and retains the preceding state. Its handoff links the diagnostic to source occurrences, certified conditions and a generation checkpoint. Section 5 compares with Reflexion, SatLM and Logic-LM under the stated input conditions; ARc, VeriCoT and VERGE are not evaluated.

Conflict-guided repair. VERGE guides answer revision with one greedily computed minimal correction subset (Singh et al., 2026). Such a subset identifies claims whose removal restores consistency, but consistency alone does not establish that the surviving claims follow from the trusted state. For a certified inconsistency, our approach fixes the admitted prefix, enumerates minimal correction sets over the rejected step using MARCO (Liffiton & Malik, 2013; Liffiton et al., 2016), and validates surviving claims by entailment. The admission guarantees require faithful extraction and sound verification (Sections 3–4).

Decoding and activation steering. Decoding methods control generation through token probabilities or search constraints: DExperts combines expert and anti-expert distributions, while NeuroLogic enforces lexical constraints (Liu et al., 2021; Lu et al., 2021). Activation steering instead modifies internal representations, as in contrastive activation addition (Rimsky et al., 2024). These interventions control output properties without themselves certifying the mathematical validity of a reasoning step. Our approach provides that check through its symbolic checks and conditions ordinary decoding on certified repair information, retaining the admitted prefix and its KV cache.

Training-based steering. RLHF and direct preference optimization shape model behavior by updating parameters from preference supervision (Ouyang et al., 2022; Rafailov et al., 2023). Our approach addresses per-instance reasoning failures at inference time with a fixed model, making its verification and repair loop complementary to preference training.