AtomicMath: Skill-Aware Traceback Supervision for Mathematical Reasoning

Anonymous ACL submission

Abstract

Eliciting reliable reasoning in large language models (LLMs) remains challenging for complex problems. We present **AtomicMath**, a fine-tuning framework that enhances compositional mathematical reasoning via a skill-diagnostic supervision paradigm. Rather than merely imitating unstructured chains of thought, AtomicMath recasts process supervision as a diagnostic signal: each reasoning step is explicitly attributed to a reusable atomic skill. This structured traceback enables a closed-loop training strategy, where an adaptive curriculum dynamically identifies and remediates weak competencies. Evaluated on a novel 96K skill-aware dataset, AtomicMath consistently outperforms representative baselines. It improves final-answer accuracy, step-skill precision, and resilience on hard compositional subsets, providing an effective mechanism for structured and diagnosable mathematical reasoning.

1 Introduction

Eliciting reliable reasoning in large language models (LLMs) remains a central challenge (Huang and Chang, 2022). While chain-of-thought prompting improves performance (Wei et al., 2022; Wang et al., 2023; Zhou et al., 2023), models frequently default to superficial pattern matching rather than systematic compositional reasoning on complex tasks (Dziri et al., 2024; Hendrycks et al., 2021). This brittleness is exacerbated by standard supervised fine-tuning (SFT), which trains models to mimic unstructured text solutions without explicitly supervising the underlying skills (Nye et al., 2021), often concealing foundational weaknesses.

Recent process supervision methods evaluate step-level correctness (Cobbe et al., 2021; Lightman et al., 2023; Lin et al., 2025), yet they typically do not explicitly identify *which* specific skill is applied. This leaves no direct mechanism to isolate and remediate conceptual deficits during training. To address this limitation, we propose **AtomicMath** (Figure 1), which recasts process supervision as a **skill-diagnostic supervision** loop. By anchoring each intermediate step to an explicit `source_skill`, AtomicMath encourages models to represent reasoning as a structured composition of reusable competencies. This granular attribution enables an active, phased training loop (Bengio et al., 2009; Zhang et al., 2025): the model dynamically diagnoses weak skills on a remedial development set and injects targeted supplemental data to address these blind spots.

Our work makes three core contributions:

- **Skill-Diagnostic Process Supervision:** We shift the training paradigm from monolithic outcome imitation to a diagnostic loop. By linking every reasoning step to its supporting atomic skill, the training objective encourages structured, skill-grounded reasoning traces.

- **Adaptive Remedial Curriculum & Dataset:** We construct a novel 96K-instance skill-aware dataset with step-level JSON traceback annotations. To leverage this, we propose a closed-loop training strategy that uses periodic validation feedback to identify weak skill slices and inject targeted supplemental examples, promoting broader coverage across a 120-way taxonomy.

- **Empirical Effectiveness:** AtomicMath demonstrates strong performance gains over **representative process-supervision baselines**. On the MATH benchmark, it boosts LLaMA-3 (8B) to 37.52% and Qwen3.5 (9B) to 57.88%, showing particular resilience in long-tail skill slices and hard compositional subsets. All data and codebases will be publicly released.

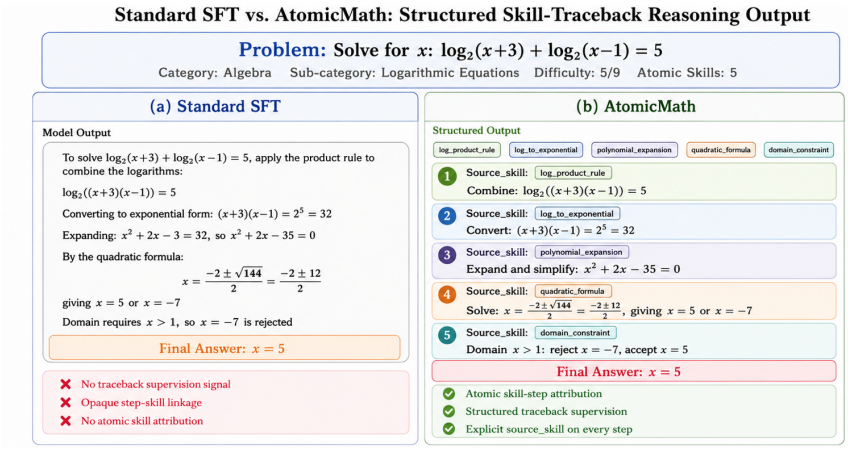


Figure 1: Illustrative comparison between standard SFT and **AtomicMath**. While SFT produces unstructured reasoning, AtomicMath organizes the solution as a structured traceback sequence with explicit `source_skill` attribution, turning process supervision into a diagnosable and remediable signal.

2 Related Work

2.1 Eliciting Internal Reasoning

Mathematical reasoning serves as a rigorous testbed for LLM capabilities. Despite advancements from prompting strategies like chain-of-thought and self-consistency (Wei et al., 2022; Wang et al., 2023; Zhou et al., 2023), models remain brittle on complex, multi-step problems (Cobbe et al., 2021; Hendrycks et al., 2021). While program-aided methods effectively offload rigorous calculations to external interpreters (Gao et al., 2023; Chen et al., 2022), enhancing the *internal* parametric reasoning of LLMs remains a fundamental goal. This necessitates stronger structural alignment during fine-tuning, moving beyond inference-time elicitation.

2.2 Compositionality and Skill Decomposition

Complex reasoning is inherently compositional (Lake and Baroni, 2018; Keysers et al., 2020), requiring the synthesis of multiple reusable sub-skills. However, standard supervised fine-tuning (SFT) typically trains models to imitate complete solutions as unstructured, free-form text sequences (Ouyang et al., 2022; Taori et al., 2023). This limits the model's ability to generalize compositionally to unseen distributions. We address this gap by explicitly formulating reasoning as a structured composition of reusable atomic skills.

2.3 Skill-Aware Process Supervision

Early analyses demonstrated the superiority of process-based feedback over outcome-based rewards for reasoning tasks (Uesato et al., 2022). Recent process supervision methods further improve reliability via step-level verification and rewards (Cobbe et al., 2021; Lightman et al., 2023; Wang et al., 2024; Lin et al., 2025). Yet, these approaches primarily judge step correctness without identifying the foundational skill invoked. AtomicMath extends process supervision by introducing skill-aware traceback annotations, explicitly supervising *which atomic skill* supports each step to provide structured signals for targeted remediation.

2.4 Adaptive Curriculum Learning

Curriculum learning typically presents examples progressing from easier to harder instances (Bengio et al., 2009), with recent adaptations dynamically aligning the training order with LLM competence (Zhang et al., 2025). AtomicMath tightly couples this concept with structured skill supervision. Rather than relying on a static easy-to-hard schedule, our framework actively monitors evaluation performance to dynamically identify and reinforce specific weak conceptual building blocks through an adaptive training loop.

3 Methodology

We present **AtomicMath**, a structured fine-tuning framework designed to elicit compositional mathematical reasoning. AtomicMath upgrades conventional process supervision into a **skill-diagnostic paradigm**—a closed-loop of attribution, diagnosis, and targeted remediation. Rather than merely imitating unstructured chains of thought, our approach disentangles supervision into skill-attributed traceback sequences and final-answer tokens. This struc-

tured alignment, governed by a difficulty-aware curriculum and an adaptive remedial loop, encourages the model to express *how* each step is derived through explicit conceptual anchoring (Figure 2).

3.1 Problem Formulation and Skill-Aware Representation

Given a complex query x, we fine-tune a model to generate a structured reasoning trace and a final answer. Each instance is defined as $\mathcal{D}_i = \{x_i, a_i, \mathcal{S}_i, \mathcal{T}_i, d_i, m_i\}$, where x_i is the problem, a_i the answer, $\mathcal{S}_i$ the requisite skills, $\mathcal{T}_i$ the traceback sequence, d_i the difficulty, and m_i metadata. Unlike SFT which optimizes unstructured solutions, AtomicMath employs a format designed to make the reasoning structure explicit. Each traceback step $t_{i,j} \in \mathcal{T}_i$ is represented as a tuple:

$$t_{i,j} = \{\texttt{step} : j,\ \texttt{source_skill} : s_{i,j},\ \texttt{content} : c_{i,j}\}$$

where $s_{i,j} \in \mathcal{S}_i$ anchors the j-th step to a specific skill from our 120-way taxonomy. This granular interface enables the training loop to analyze subskill-level failures rather than binary outcomes (Cobbe et al., 2021), providing a more fine-grained supervision signal than monolithic, free-form chains of thought (Wei et al., 2022; Nye et al., 2021). (See Table 7 in the Appendix for schema details).

3.2 Skill-Traceback Supervision as Regularization

We jointly optimize over traceback and final-answer tokens, excluding prompt tokens from the loss. We define masked token-level cross-entropy losses:

$$L_{\text{trace}} = \text{CE}(y^{\text{trace}}, \hat{y}^{\text{trace}}), \quad L_{\text{final}} = \text{CE}(y^{\text{final}}, \hat{y}^{\text{final}})$$

The total objective is $L_{\text{total}} = \lambda_{\text{trace}} L_{\text{trace}} + \lambda_{\text{final}} L_{\text{final}}$. Unlike generic process supervision (Lightman et al., 2023; Wang et al., 2024), this formulation embeds explicit skill attribution directly into the target. Crucially, we postulate that process supervision should function as a structural regularizer that guides the reasoning trajectory, rather than the dominant objective overshadowing outcome optimization. This insight informs our use of λ_{trace} to maintain a balance between logical rigor and final-answer accuracy.

3.3 Phased Curriculum and Adaptive Remediation

To address hard compositional subsets, we structure data into a phased curriculum (Bengio et al., 2009; Zhang et al., 2025). Problems are stratified into *basic*, *advanced*, and *hard* buckets. Training progresses through three stages: **Foundation** (basic competency), **Logic** (subskill composition), and **Challenge** (complex synthesis). We gradually shift the probability mass toward complex reasoning by adjusting difficulty-bucket priors.

To resolve conceptual blind spots in long-tail skill slices, we implement an adaptive remedial loop. Periodic evaluation on a strictly separated remedial development set ($\mathcal{D}_{remedial_dev}$) identifies categories and skills falling below a performance threshold. Targeted supplemental examples ($\mathcal{D}_{\text{sup}}$) are then dynamically injected into the active training batch. This transforms fine-tuning into a self-correcting diagnostic system where structured supervision enables fine-grained detection and immediate remediation (Algorithm 2). Note that $\mathcal{D}_{remedial_dev}$ is used exclusively for internal diagnostic routing and remains disjoint from the final test set.

Algorithm 1: Masked Traceback and Final-Answer Loss

Input: Problem x_i, traceback $\mathcal{T}_i$, answer a_i, tokenizer τ
Output: Total loss L_{total}
Build prompt p_i, traceback z_i^{trace}, and answer z_i^{final} strings
Tokenize: $u_i = \tau(p_i),\ v_i = \tau(z_i^{\text{trace}}),\ w_i = \tau(z_i^{\text{final}})$
Concatenate target: $y_i = [u_i,\ v_i,\ w_i,\ \texttt{eos}]$
Construct masks m_i^{trace} (ones on v_i) and m_i^{final} (ones on w_i)
Run model to obtain logits $\hat{y}_i$
$L_{\text{trace}} \leftarrow \text{CE}(\hat{y}_i, y_i; m_i^{\text{trace}})$
$L_{\text{final}} \leftarrow \text{CE}(\hat{y}_i, y_i; m_i^{\text{final}})$
return $\lambda_{trace} L_{trace} + \lambda_{final} L_{final}$

Algorithm 2: AtomicMath Adaptive Training

Input: Train set $\mathcal{D}_{train}$, supplemental pool $\mathcal{D}_{\text{sup}}$, remedial dev set $\mathcal{D}_{remedial_dev}$
Output: Trained model θ
Initialize weak sets $\mathcal{C}_{\text{weak}} \leftarrow \emptyset$, $\mathcal{S}_{\text{weak}} \leftarrow \emptyset$
foreach *phase* $p \in \{foundation, logic, challenge\}$ **do**
  for $e = 1$ **to** *epochs in* p **do**
    Sample active batch from $\mathcal{D}_{train}$ via phase ratios
    Inject targeted examples from $\mathcal{D}_{\text{sup}}$ for $\mathcal{C}_{\text{weak}}$ and $\mathcal{S}_{\text{weak}}$
    foreach *mini-batch* b **do**
      Compute L_{total} (Algorithm 1) and update θ
    Evaluate on $\mathcal{D}_{remedial_dev}$ to compute fine-grained accuracy
    Update $\mathcal{C}_{\text{weak}}$ and $\mathcal{S}_{\text{weak}}$ via threshold
    Adjust next-round ratios based on difficulty-bucket performance
return θ

4 Experiments

To empirically validate **AtomicMath**, we design our experiments to answer three core questions: (1) Does skill-aware traceback supervision effectively elicit compositional reasoning capabilities compared to standard fine-tuning paradigms? (2) Does the framework generalize across different foundational model architectures and standard public benchmarks? (3) Which components (traceback

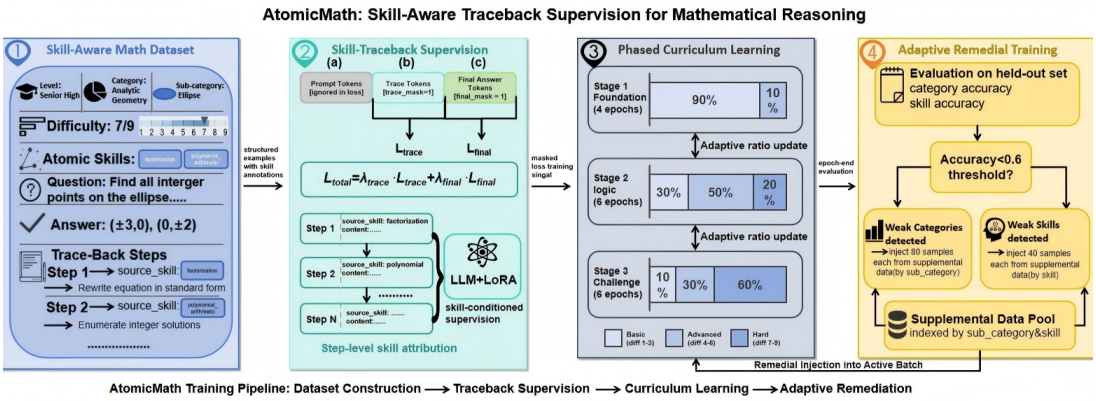


Figure 2: Overview of **AtomicMath**. Left: a skill-aware reasoning example where each step is attributed to a specific `source_skill`. Middle: supervision via L_{trace} (structural regularizer) and L_{final} (outcome objective). Right: the self-correcting loop where performance on $\mathcal{D}_{remedial_dev}$ triggers targeted remediation from $\mathcal{D}_{\text{sup}}$.

supervision, curriculum, remediation) are the primary drivers of these improvements?

4.1 Experimental Setup

We fine-tune all models with LoRA adapters (Hu et al., 2022) on 4-bit quantized backbones (Dettmers et al., 2024). Optimization settings, data splits, decoding protocols, and random seeds are fixed where applicable; parameter-matched and compute-matched controls are reported in Appendix F.

The data are divided into four disjoint splits: $\mathcal{D}_{train}$ for main training, $\mathcal{D}_{sup}$ for targeted remediation, $\mathcal{D}_{remedial_dev}$ for weak-skill diagnosis, and $\mathcal{D}_{test}$ for final evaluation, including hard compositional subsets. Training proceeds through a three-phase curriculum (*foundation, logic, challenge*) over difficulty-stratified examples, where traceback supervision serves as a structural regularizer rather than a standalone objective. Full hyperparameters and dataset statistics are given in Appendix Table 8.

4.2 Dataset Construction and Stratification

To operationalize our **skill-diagnostic supervision** paradigm, we developed a diagnostic-oriented data synthesis and curation pipeline. Because standard datasets lack the step-level `source_skill` attribution required for dynamic remediation, we constructed a novel 96K main training set and a 7.5K supplemental remedial pool, strictly constrained by a custom architectural prompt designed to facilitate compositional mathematical reasoning.

Data Synthesis and the Decoupling Principle. A core design principle is the strict decoupling of an educational stage (from Elementary to Graduate) from its **stage-conditioned cognitive load** (Levels 1–9), so that a low-load graduate problem and a low-load elementary problem may both fall into Levels 1–3 without implying the same absolute difficulty. Unlike unstructured solutions, our pipeline encourages teacher LLMs to output strictly formatted JSON objects where every solution is decomposed into discrete `trace_back_steps`, each anchored to our 120-way taxonomy. This helps ensure that even graduate-level theorems and elementary logic puzzles are uniformly structurally grounded, providing the necessary signals for process-level diagnosis and **structural regularization** during training.

Quality Assurance and Human Filtering. To reduce noise in the diagnostic annotations, we applied full-pass programmatic heuristic checks for JSON compliance, mathematical formatting, and schema consistency across all generated instances. This was followed by instance-level human-in-the-loop screening, where evaluators verified that reasoning traces were logically sound and skill attributions were accurate. Examples containing logical leaps or misaligned skills were discarded or manually rewritten to reduce the risk of learning heuristic shortcut patterns.

Curriculum Stratification (Difficulty Buckets). We map our 1-to-9 stage-conditioned cognitive load scale into three deterministic difficulty buckets to operationalize phased curriculum learning:

- **Basic (Levels 1–3)**: Stage-appropriate direct recall or single-formula application. These establish mastery in individual long-tail skill slices.

- **Advanced (Levels 4–6)**: Standard applications requiring the composition of 2–3 logical steps (e.g., function intersections).
- **Hard (Levels 7–9)**: Highly complex problems requiring deep synthesis or formal proofs (e.g., Sobolev Embedding Theorem). These form the hard compositional subsets where our structural advantage is most pronounced.

This objective stratification ensures that compositionality is introduced at a pedagogically sound pace, allowing process supervision to guide the reasoning trajectory. For a comprehensive breakdown of our 120-skill taxonomy, as well as exhaustive details regarding our human quality audit and anti-contamination de-duplication protocols, please refer to Appendix C.1.

4.3 Evaluation Methodology

To evaluate the effectiveness of our **skill-diagnostic paradigm** in eliciting compositional mathematical reasoning, we assess models across three dimensions (see Section D in the Appendix for formal mathematical definitions):

- **Outcome Metrics**: We compute standard final-answer accuracy (**Acc**$_{\text{final}}$) via exact-match extraction.
- **Process Metrics**: We report Per-skill Answer Accuracy (macro-average across specific atomic skills) to observe performance on long-tail skill slices, and Category accuracy to gauge stability across the taxonomy. To explicitly evaluate the model's diagnostic attribution capabilities, we introduce **Step-skill Precision**, measuring the precision of autonomously generated skill tags against the required taxonomy.
- **LLM-as-a-Judge**: Recognizing that exact-match metrics may fail to capture structural quality, we utilize an LLM-as-a-judge (Zheng et al., 2024). The judge evaluates (i) answer correctness, (ii) logical coherence, and (iii) skill alignment (detailed in Section D.1).

Inference Protocols and Autonomous Attribution. To ensure a rigorous and fair evaluation against representative process-supervision baselines, all main experiments consistently employ a **No-skill-prompt** setting. In this zero-shot configuration, the model is provided solely with the raw question x_i and remains blind to the ground-truth skill list $\mathcal{S}_i$. The model must autonomously attribute the `source_skill` for each step, allowing us to test whether process supervision induces skill-grounded reasoning traces without oracle skill prompts. We reserve the *Oracle-skill* setting for upper-bound ablation analysis.

All generator-only methods use greedy decoding (temperature = 0.0) to reduce sampling variance during evaluation. The verifier-reranking baseline follows its standard reranking protocol, generating $K = 16$ candidates and selecting the highest-scoring solution, as detailed in Appendix E. To validate our judge-based metrics, we report high human-judge consistency (Pearson $r = 0.88$) and implement a blind evaluation protocol (see Appendix C.3).

4.4 Baselines

We benchmark AtomicMath against both outcome-driven and process-driven fine-tuning paradigms. To rigorously isolate the structural effects of our framework, we compare against five baseline categories: (1) **Base LLM** (establishing the pre-tuned reasoning floor), (2) **Standard SFT** (Ouyang et al., 2022), (3) **Curriculum-only SFT** (ablating traceback and remediation), (4) **Structured CoT SFT** (Wei et al., 2022), and (5) **Process-Supervision Baselines**, which encompass Verifier-based reranking (Cobbe et al., 2021), Math-Shepherd-style step feedback (Wang et al., 2024), and STEP-KTO-style hybrid feedback (Lin et al., 2025). All trainable baselines share the same backbone, QLoRA setup, data splits, and answer parser. Process-supervision baselines use the same curated traces but without explicit `source_skill` labels; detailed implementations are provided in Appendix E.

4.5 Main Results and Analysis

Tables 1 and 2 present results on $\mathcal{D}_{test}$ under the zero-shot **No-skill-prompt** setting. AtomicMath achieves 24.11% answer accuracy, establishing a distinct structural advantage over both Standard SFT (19.55%) and representative process baselines (e.g., 17.47% for STEP-KTO). Critically, our *compute-matched* control groups suggest that these gains are not solely driven by extended training; even with equalized computational budgets, baseline performance remains stagnant, failing to capture the compositional logic elicited by our skill-diagnostic paradigm.

The structural superiority of AtomicMath is most

Table 1: Main results on the strictly held-out test set ($\mathcal{D}_{test}$). Higher is better. Methods are grouped into general baselines and **representative process-supervision baselines**. Compute-matched variants (marked with *) equalize the total training duration to that of AtomicMath.

Method	Ans. Acc.	Cat. Acc.	Per-skill Acc.	Basic	Adv.	Hard	Judge
General baselines							
Base LLM	13.21	12.35	6.44	27.59	10.71	1.72	0.466
Standard SFT	19.55	18.25	12.40	39.83	12.52	4.81	0.542
Standard SFT (Compute-matched)*	19.92	18.47	12.62	40.11	12.73	4.87	0.547
Structured CoT SFT	18.42	18.85	14.61	32.13	19.27	2.15	0.535
Structured CoT (Compute-matched)*	18.78	19.16	14.85	32.51	19.52	2.23	0.538
Curriculum-only SFT	20.66	21.42	13.82	31.55	22.48	4.67	0.548
Representative process baselines							
Verifier / Reranking	16.85	14.53	10.22	28.43	15.37	6.80	0.512
Math-Shepherd-style	17.15	18.20	11.48	30.61	18.83	2.47	0.495
STEP-KTO-style	17.47	18.92	12.88	29.54	17.55	3.11	0.508
Our method							
AtomicMath (Ours)	**24.11**	**24.71**	**20.89**	**43.10**	**25.33**	**7.51**	**0.638**

Table 2: Pairwise judge-based comparison against the base model (N=1000). Δ metrics represent the absolute improvement over the Base LLM.

Method	Δ Judge	Δ Acc.	Improved	Tied	Worse
Standard SFT vs. Base	+0.076	+6.34	485	320	195
AtomicMath vs. Base	**+0.172**	**+10.90**	**620**	**245**	**135**

evident in hard compositional subsets, where it maintains 7.51% accuracy compared to the 4.81% of standard SFT. Furthermore, our method excels in long-tail skill slices, nearly doubling the Per-skill Answer Accuracy of traditional paradigms (20.89% vs. $\sim$12.40%). AtomicMath further achieves a Step-skill Precision of **84.2%**. These results indicate that process supervision, when framed as a structural regularizer, can reduce heuristic shortcut patterns and improve the consistency of skill-grounded deductions.

Adaptive Remedial Dynamics. Beyond static metrics, our adaptive remedial loop actively patches conceptual deficits. We observe that for specific atomic skills where the model initially exhibits "blind spots," the targeted injection of supplemental data triggers a sharp recovery. This confirms the efficacy of the diagnostic-remedial cycle: structured attribution allows for fine-grained failure detection, which in turn facilitates precise knowledge reinforcement. **Detailed visualizations of these training dynamics are provided in Figure 4 of the Appendix.**

Contextualizing Performance. Given the extreme difficulty of tasks spanning elementary arithmetic to graduate-level theorems, where the Base LLM yields only 13.21%, AtomicMath's leap to 24.11% represents a substantial improvement in eliciting **compositional mathematical reasoning**. By outperforming representative baselines, these results provide evidence that explicit skill-diagnostic signals offer a more effective optimization path than monolithic solution imitation.

Stability and Knowledge Retention. Our remedial intervention avoids catastrophic forgetting of "non-weak" topics. As shown in Appendix C.2, accuracy on stable topics recovers rapidly after targeted injections. This resilience confirms that our curriculum serves as a global anchor, allowing remedial examples to act as corrective micro-adjustments without destabilizing the primary training objective.

4.6 Generalization Across Architectures and Domains

To evaluate architecture-agnostic and out-of-domain generalization, we apply AtomicMath to two distinct base models: LLaMA-3 (8B) (Dubey et al., 2024) and Qwen3.5 (9B) (Qwen Team, 2026). Crucially, models are fine-tuned consistently on our custom dataset and evaluated zero-shot on the public GSM8K and MATH benchmarks.

As shown in Table 3, AtomicMath consistently outperforms all baselines. Notably, on the rigorous MATH benchmark, it boosts LLaMA-3 from 22.36% to 37.52%, and Qwen3.5 from 36.42% to an impressive 57.88%. These results indicate that skill-aware traceback supervision can transfer to standard mathematical reasoning benchmarks.

Table 3: Performance evaluation on widely adopted public mathematical reasoning benchmarks across different base architectures. Accuracy (%) is reported.

	LLaMA-3 (8B)		Qwen3.5 (9B)	
Method	**GSM8K**	**MATH**	**GSM8K**	**MATH**
Base LLM	68.42	22.36	77.88	36.42
Standard SFT	76.53	28.48	85.16	44.51
Structured CoT SFT	78.16	30.65	86.63	47.15
Curriculum-only SFT	77.82	29.88	86.20	45.98
Math-Shepherd-style	79.20	32.15	88.95	50.44
AtomicMath (Ours)	**84.18**	**37.52**	**90.20**	**57.88**

Data De-duplication. To ensure these gains reflect genuine generalization rather than memorization, we implemented a strict de-duplication pipeline prior to training. We verified that no examples in our training or supplemental pools exhibit an exact string match or high 13-gram overlap with the GSM8K or MATH test sets.

4.7 Ablation Studies

We systematically deconstruct AtomicMath to isolate the drivers of its performance across our **skill-diagnostic paradigm** (Table 4).

Beyond Structure: The Power of Semantic Grounding. A pivotal question is whether AtomicMath's resilience on hard compositional subsets stems from JSON formatting or explicit skill grounding. Training with *JSON-CoT only* (no `source_skill`) drops accuracy to 18.65%. More tellingly, using *Shuffled source_skill tags* severely degrades performance (14.31%), confirming that the model benefits from the correct semantic mapping between deduction and competency, rather than mere structural mimicry.

Closing the Diagnostic Loop. Removing the remedial loop (*No injection*) collapses accuracy to 9.65%, demonstrating that static training undercovers long-tail skill slices. While *random remedial injection* and *static oversampling* (19.87%) improve results, they fail to match the targeted precision of the full diagnostic cycle. This supports the role of targeted remediation in improving coverage across the 120-skill taxonomy.

Process Supervision as Structural Regularization. Table 5 reveals a fundamental tension in process-supervised alignment. At the optimal $\lambda_{\text{trace}} = 0.65$, process supervision acts as a structural regularizer to guide the reasoning trajectory without overshadowing the outcome optimization. When λ_{trace} is excessively high (1.50), we observe *representation collapse*, where the model fixates on formatting and tag retrieval at the expense of computational accuracy. Conversely, too low a weight allows a reversion to **heuristic-driven guessing** typical of standard SFT. This confirms that skill-diagnostic signals should reinforce, rather than replace, the underlying problem-solving objective. Detailed remedial dynamics for specific subskills are visualized in Figure 4 of the Appendix.

4.8 Qualitative and Error Analysis

Fig. 3 illustrates how AtomicMath can shift model outputs from heuristic pattern-matching toward more skill-grounded deductions.

Mitigating Heuristics via Regularization. Standard SFT often relies on heuristic-driven guessing. In Case 1, SFT overlooks interval endpoints, whereas AtomicMath's λ_{trace} acts as a **structural regularizer**, encouraging the explicit invocation of the `closed_interval_method`. Similarly, Case 2 shows AtomicMath replacing flawed indexing with rigorous `geometric_sequence_definition` anchoring, ensuring the reasoning trajectory adheres to formal axioms.

Actionable Failure Diagnosis. AtomicMath transforms failures into actionable diagnostic signals (Case 3). While SFT's hallucinations are opaque, our **skill-diagnostic paradigm** isolates conceptual bottlenecks: the model succeeds at `conditional_probability` but fails at `combinatorial_counting`. This granularity allows our adaptive loop to target specific long-tail skill slices invisible in binary outcome labels.

Bottlenecks in Hard Subsets. Two primary challenges persist in hard compositional subsets. First, compounding errors: in long-horizon reasoning (>8 steps), early misattributions irreversibly corrupt downstream logic. Second, skill blending:

Table 4: Component ablations and structural control groups evaluated on $\mathcal{D}_{test}$. We decompose the framework into structural/semantic and training dynamic factors to isolate the drivers of performance gains in **compositional mathematical reasoning**.

Variant	Final-answer Acc.	Mean Per-skill Acc.	Hard Acc.	Avg. Judge Score
AtomicMath (full)	**24.11**	**20.89**	**7.51**	**0.638**
Structural & Semantic Controls				
w/o traceback loss ($\lambda_{\text{trace}} = 0$)	12.52	6.94	3.47	0.460
JSON-CoT format only (w/o `source_skill`)	18.65	14.54	2.47	0.532
Shuffled `source_skill` tags	14.31	8.13	1.82	0.450
Remediation & Curriculum Controls				
w/o adaptive remedial training (No injection)	9.65	5.21	3.45	0.467
w/ random remedial injection	18.42	13.23	4.14	0.524
w/ static difficulty oversampling	19.87	15.62	5.23	0.551
w/o phased curriculum (Static uniform)	11.04	8.24	1.72	0.465

Table 5: Sensitivity to the traceback-loss weight λ_{trace} with $\lambda_{\text{final}} = 1.0$. Process supervision acts as a **structural regularizer** at the optimal $\lambda_{\text{trace}} = 0.65$.

λ_{trace}	Fin-ans Acc.	Mean Skill Acc.	Hard Acc.
0.00	12.52	6.94	3.45
0.25	17.38	13.42	5.12
0.50	21.65	18.27	6.78
0.65	24.11	20.89	7.51
1.00	23.14	19.56	6.93
1.50	19.42	15.84	5.47

when problems demand tightly coupled, simultaneous skills (e.g., integration by parts with trig-identities), the model struggles to discretize them into sequential steps. This suggests that deeply intertwined operations still challenge strict step-level discretization.

4.9 Computational Efficiency and Trade-offs

As detailed in Appendix F (Table 11), we compare training overhead under a standardized 16-epoch schedule. AtomicMath requires more training time than standard SFT (41.12 vs. 22.76 hours) due to the expanded sequence lengths required by step-level traceback tokens and the dynamic injection of remedial data.

However, this temporal trade-off is empirically supported by the resulting performance leap. Crucially, AtomicMath introduces zero additional memory overhead. Sharing the exact same trainable parameter count (20.185M) and LoRA adapter size (80.8 MB) as standard SFT, the framework remains fully trainable on a single consumer-grade GPU.

5 Conclusion

We presented **AtomicMath**, a framework for compositional mathematical reasoning that upgrades process supervision into a skill-diagnostic paradigm. By anchoring reasoning steps to a formal taxonomy, process signals can serve as a structural regularizer that encourages skill-grounded reasoning trajectories and reduces heuristic shortcut patterns. Combined with an adaptive curriculum, AtomicMath effectively patches long-tail skill slices, outperforming representative baselines on hard compositional subsets. Ultimately, AtomicMath bridges the gap between structured supervision and robust generalization, providing a foundation for more diagnosable reasoning systems. Future works will explore cross-domain scalability and multimodal architectures.

Figure 3: SFT vs. AtomicMath. Structured attribution mitigates heuristics (Cases 1 & 2). Failures (Case 3) are localized to specific subskills, enabling targeted remediation.

Limitations

While **AtomicMath** significantly improves compositional reasoning, our framework has several limitations:

Domain Specificity. We evaluate our framework strictly on mathematical reasoning, an ideal testbed due to its objective metrics. Extending our rigid, static 120-skill taxonomy to more ambiguous, open-ended domains (e.g., common-sense reasoning or creative writing) remains an open challenge.

Data Synthesis Costs and Bias. Constructing our highly structured 96K dataset relies on proprietary LLMs (e.g., GPT-4). This introduces financial overhead for data generation and risks inheriting the latent reasoning ceilings and heuristic biases of the teacher models.

Computational Overhead. Generating explicit traceback tokens and dynamically injecting remedial data increases training duration by approximately 80% compared to standard SFT. While VRAM usage remains constant, this temporal overhead may slow down rapid hyperparameter iteration.

Lack of Inference-Time Rollback. While structured rationales improve training diagnostics, they bind the model to its logical trajectory during greedy decoding. If an early step is flawed, the model currently lacks an internal backtracking or search-based self-correction mechanism to recover from cascading errors.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

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

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

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

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, and et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Luyu Gao, Aman Maman, Shikhar Banerjee, and et al. 2023. Pal: Program-aided language models. In *International Conference on Machine Learning (ICML)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hessel Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinashov, Livio Stanton, and et al. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations (ICLR)*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International conference on machine learning*, pages 2873–2882.

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

Yen-Ting Lin, Di Jin, Tengyu Xu, Tianhao Wu, Sainbayar Sukhbaatar, Chen Zhu, Yun He, Yun-Nung Chen, Jason E. Weston, Yuandong Tian, Arash Rahnama, Sinong Wang, Hao Ma, and Han Fang. 2025. Stepkto: Optimizing mathematical reasoning through stepwise binary feedback. In *Proceedings of the 3rd Workshop on Mathematical Natural Language Processing*, pages 11–24.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

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

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents. https://qwen.ai/blog?id=qwen3.5.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. *GitHub repository*.

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

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9446–9460.

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

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

Enci Zhang, Xingang Yan, Wei Lin, Tianxiang Zhang, and Qianchun Lu. 2025. Learning like humans: Advancing LLM reasoning capabilities via adaptive difficulty curriculum learning and expert-guided self-reformulation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng Ying, Shiyang Hao, Sean Macavaney, and et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

Appendix

A Notation and Parameter Definitions

Table 6 summarizes the key mathematical symbols and parameters used throughout the AtomicMath methodology and algorithms.

B Skill-Aware Dataset Schema

Table 7 outlines the JSON schema. This format ensures every reasoning trace is explicitly grounded in atomic competencies, providing step-level traceback annotations and topic metadata for targeted remediation.

C Detailed Experimental Setup

All experiments were conducted on a single NVIDIA GeForce RTX 4090 (24GB VRAM) GPU Table 8 details the hyperparameters, LoRA configurations, and dataset statistics used to fine-tune and evaluate AtomicMath.

C.1 Detailed Dataset Taxonomy, Quality Audit, and De-duplication

Expanding on Section 4.2, Table 9 details the full taxonomy of our 96.6K training set. Spanning 24 consolidated categories and 120 atomic skill topics, this distribution—ranging from elementary arithmetic to graduate-level theorems—illustrates our design principle of decoupling educational stages from stage-conditioned cognitive load.

Taxonomy Derivation. The 120-skill taxonomy detailed in Table 9 was constructed following a strict education-driven principle: **(1) Syllabus Integration:** We aggregated standard mathematical syllabi (e.g., Common Core Standards) and core textbooks ranging from Elementary to Graduate levels. **(2) LLM Distillation:** Advanced LLMs were utilized to cluster and summarize the knowledge points from these syllabi across multiple iterations. **(3) Human Calibration:** Finally, domain experts reviewed and finalized the 120 mutually exclusive and collectively exhaustive (MECE) skill tags.

Table 6: Summary of notations and parameters used in the AtomicMath framework.

Symbol	Description
Data Representation	
$\mathcal{D}_i$	The i-th training example tuple.
x_i	The natural-language reasoning query / mathematical problem.
a_i	The target final answer for problem x_i.
$\mathcal{S}_i$	The set of required atomic skills to solve problem x_i.
$\mathcal{T}_i$	The structured traceback reasoning sequence for problem x_i.
d_i	The difficulty level of the problem.
m_i	Auxiliary metadata (e.g., educational level, category, subcategory).
$t_{i,j}$	The j-th reasoning step within the traceback sequence $\mathcal{T}_i$.
$s_{i,j}$	The specific atomic skill (where $s_{i,j} \in \mathcal{S}_i$) supporting step $t_{i,j}$.
$c_{i,j}$	The textual reasoning content of step $t_{i,j}$.
Supervision & Loss Formulation	
y_i	The full concatenated target sequence for the model.
y_i^{prompt}	The tokenized prompt segment (excluded from loss).
y_i^{trace}	The tokenized structured reasoning trace segment.
y_i^{final}	The tokenized final answer segment.
$\hat{y}_i$	The predicted logits output by the model.
L_{trace}	Masked cross-entropy loss computed over the traceback tokens.
L_{final}	Masked cross-entropy loss computed over the final-answer tokens.
L_{total}	The joint overall training objective.
λ_{trace}	Weighting hyperparameter for the traceback supervision loss.
λ_{final}	Weighting hyperparameter for the final-answer outcome loss.
τ	The tokenizer function used to process text into token sequences.
m_i^{trace}	Binary mask to isolate traceback tokens during loss computation.
m_i^{final}	Binary mask to isolate final-answer tokens during loss computation.
Training Loop & Remediation	
θ	The trainable parameters of the pre-trained language model.
$\mathcal{D}$	The primary training dataset.
$\mathcal{D}_{\text{sup}}$	The supplemental data pool used exclusively for targeted remediation.
$\mathcal{D}_{\text{val}}$	The held-out evaluation set for diagnosing skill-level weaknesses.
$\mathcal{P}$	The phased curriculum schedule (Foundation, Logic, Challenge).
$\mathcal{C}_{\text{weak}}$	The set of dynamically identified weak problem categories.
$\mathcal{S}_{\text{weak}}$	The set of dynamically identified weak atomic skills.

Data Audit Metrics. To improve the reliability of the training and evaluation signals, we conducted comprehensive automated screening and expert-level human auditing on the initial pool of 116,000 generated samples. The auditing was executed by two researchers with extensive backgrounds in computer science and mathematics:

- **Discard Rate:** From the initial 116,000 instances, 1,058 samples were outright discarded due to logical leaps, JSON format parsing failures, or mathematical inaccuracies.
- **Manual Rewriting:** For 324 samples that contained minor narrative flaws or imprecise `source_skill` attributions but possessed valid core logic, the researchers conducted manual corrections and rewrites.
- **Inter-Annotator Agreement:** In the initial cross-auditing phase, the two annotators achieved Cohen's $\kappa = 0.92$ on skill attribution and difficulty grading, ensuring a highly consistent annotation standard.

Anti-Contamination Protocol. To explicitly prevent data contamination—specifically to ensure the model had not seen variants of the GSM8K or MATH test sets during training—we implemented a robust three-tier de-duplication defense:

1. **N-gram Matching:** We executed standard 13-gram character-level overlap detection to filter

Table 7: Core fields in the AtomicMath training format.

Field	Type	Description
`id`	string	Unique example identifier.
`difficulty`	integer	Difficulty score on a 1–9 scale.
`school_level`	string	Educational stage, e.g., Elementary, Junior High, Senior High.
`category`	string	High-level topic family.
`sub_category`	string	Fine-grained problem type.
`atomic_skills`	list[string]	Set of reusable atomic skills required by the problem.
`question`	string	Natural-language mathematical problem.
`answer`	string	Final target answer.
`trace_back_steps`	list[object]	Step-level traceback annotations. Each step contains `step`, `source_skill`, and `content`.

Table 8: Comprehensive experimental setup. Qwen2.5-7B-Instruct serves as our primary backbone for all controlled experiments and ablations, while extended backbones are used for generalization testing.

Item	Value	Item	Value
Main backbone	Qwen2.5-7B-Instruct	Ext. backbones*	Qwen3.5-9B, LLaMA-3 (8B)
LoRA rank r	32	LoRA alpha	16
LoRA dropout	0.05	LoRA target modules	attention only (default)
Quantization	4-bit NF4	Batch size	1
Gradient accumulation	16	Learning rate	2×10^{-4}
Weight decay	0.01	Max sequence length	768
Max generation length	256	Epoch size	6000
Eval samples / phase	500	Phase epochs	$(4, 6, 6)$
Seed	42	Trace loss weight λ_{trace}	0.65
Final loss weight λ_{final}	1.0	Remedial threshold	0.60
Max weak skills	12	Category remedial count	80
Skill remedial count	40	Curriculum adapt rate α	0.35
Judge model	`gpt-4o-mini`		
Main training set size	96,668	Supplemental pool size	7,533
Remedial dev set ($\mathcal{D}_{remedial_dev}$)	5,000	Final test set ($\mathcal{D}_{test}$)	5,741
Consolidated categories	24	Number of Unique atomic skills	120

**Note.* The extended backbones (Qwen3.5-9B and LLaMA-3 8B) are utilized exclusively for cross-model out-of-domain generalization testing on public benchmarks (GSM8K and MATH) as reported in Section 4.6.

out instances with high exact-match similarity.

2. **Semantic De-duplication:** We utilized pre-trained embedding models to compute cosine similarity between samples. Pairs exhibiting a similarity score above 0.95 underwent manual inspection to prevent the inclusion of "template-swapped" questions.

3. **Paraphrase Detection:** We applied the MinHash algorithm to perform dimensionality reduction and fingerprinting on the logical structures, specifically designed to detect logically equivalent paraphrases of MATH benchmark problems.

C.2 Analysis of Remedial Training Dynamics

To investigate the impact of the adaptive remedial loop, we track the accuracy of three initially weak atomic skills (Figure 4). During the initial epochs (1–6), these skills exhibited stagnant growth (below 10% accuracy). At Epoch 6, the adaptive mechanism detected these bottlenecks and injected targeted supplemental data. This intervention triggered a sharp, sustained recovery, with accuracies rising to 22–28% by Epoch 16. These improvements confirm that our closed-loop system effectively diagnoses and remediates cognitive gaps that static training schedules miss.

C.3 Human-Judge Consistency and Evaluation Protocol

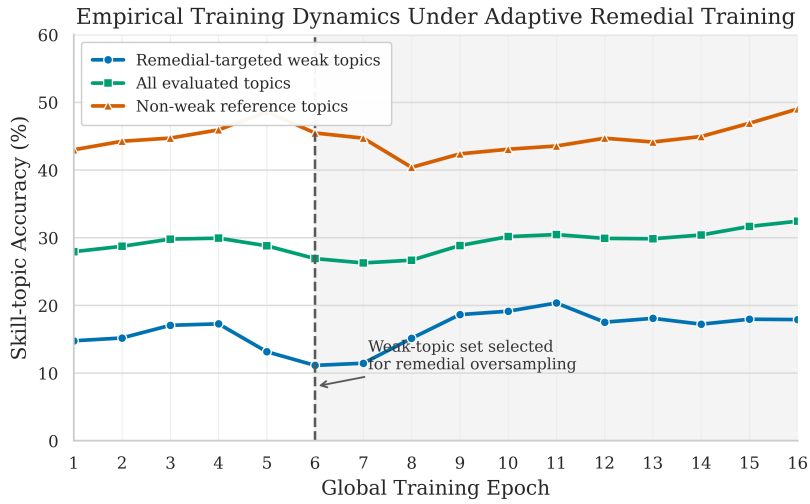
To validate the reliability of our LLM-as-a-judge, we conducted a blind human-check on a random sample of 1000 evaluation instances. The Pearson correlation between the judge's composite scores and the expert human ratings reached 0.88, demonstrating high alignment in identifying logical coherence and skill attribution accuracy.

Furthermore, for all pairwise judge-based comparisons reported in our main results, we implemented a strict blind evaluation protocol: the iden-

Table 9: Taxonomy of the AtomicMath training dataset. The table reports 24 consolidated synthesis categories and 120 unique atomic skill topics. Category labels are pedagogical groupings rather than mutually exclusive skill labels.

School Level	Category	#Topics	#Problems	Share	Sub-categories / Atomic Skill Topics
Elementary	Calculation	8	6,459	6.68%	Addition and Subtraction within 100; Decimal Arithmetic; Estimation and Approximation; Fraction Arithmetic (+, -, *, /); Fraction-Decimal Conversion; Mental Math Strategies (Rounding/Associative Law); Multi-digit Multiplication and Division; Order of Operations (PEMDAS)
Elementary	Geometry Basics	7	5,678	5.87%	Area of Parallelograms and Trapezoids; Area of Triangles; Circumference and Area of Circles; Geometry Transformations (Translation/Rotation/Symmetry); Perimeter and Area of Rectangles/Squares; Surface Area and Volume (Cube/Cuboid); Volume of Cylinders and Cones
Elementary	Number Theory Basics	7	5,655	5.85%	Divisibility Rules (2, 3, 5, 7, 11); Factors and Multiples; GCD and LCM; Odd and Even Numbers; Place Value System; Prime and Composite Numbers; Properties of Remainders
Elementary	Word Problems	12	6,491	6.71%	Boat and Current Problems; Chicken and Rabbit Problem; Estimation and Approximation; Mental Math Strategies (Rounding/Associative Law); Multi-digit Multiplication and Division; Order of Operations (PEMDAS); Profit, Discount, and Interest; Sum-Difference-Multiple Problems; Travel Problems (Meeting/Pursuit); Tree Planting Problem; Work/Engineering Problems; Working Backwards (Reverse Calculation)
Junior High	Algebra Basics	8	6,458	6.68%	Absolute Value; Algebraic Fractions; Exponent Rules; Factorization (Grouping/Cross Method); Multiplication Formulas (Square of Sum/Diff); Polynomial Arithmetic; Radical Expressions; Rational Numbers and Number Line
Junior High	Equations and Inequalities	6	4,841	5.01%	Linear Equations in One Variable; Linear Inequalities; Quadratic Equations (Factoring/Formula); Rational Equations; System of Linear Equations (Two Variables); Vieta's Formulas
Junior High	Functions	5	3,978	4.12%	Cartesian Coordinate System; Inverse Proportional Functions; Linear Functions (Slope/Intercept); Quadratic Functions (Vertex/Standard Form); Transformations of Quadratic Graphs
Junior High	Geometry & Proofs	6	4,812	4.98%	Circle Theorems (Chords/Tangents/Angles); Congruent Triangles (SSS, SAS, ASA, HL); Introduction to Trigonometry (SOH CAH TOA); Properties of Parallelograms/Rectangles/Rhombuses; Pythagorean Theorem; Similar Triangles
Senior High	Advanced Functions	4	3,211	3.32%	Exponential and Logarithmic Functions; Inverse Functions; Monotonicity and Parity; Zeroes of Functions (Bisection Method)
Senior High	Analytic Geometry	5	4,048	4.19%	Ellipse (Definition and Standard Equation); Equation of Lines and Circles; Hyperbola (Asymptotes); Line and Conic Section Intersections; Parabola (Directrix)
Senior High	Calculus Introduction	4	3,231	3.34%	Derivatives of Elementary Functions; Fundamental Theorem of Calculus; Geometric Meaning of Derivative; Maxima and Minima
Senior High	Probability and Combinatorics	5	4,039	4.18%	Binomial Theorem; Conditional Probability; Discrete Random Variables; Normal Distribution; Permutations and Combinations
Senior High	Sequences and Series	3	2,407	2.49%	Arithmetic Sequences; Geometric Sequences; Summation Techniques (Telescoping Series)
Senior High	Sets and Logic	3	2,399	2.48%	Necessary and Sufficient Conditions; Quantifiers (Universal/Existential); Set Operations (Union/Intersection)
Senior High	Trigonometry	4	3,217	3.33%	Law of Sines and Cosines; Trigonometric Graphs ($A\sin(\omega x+\phi)$); Trigonometric Identities (Sum/Difference/Double Angle); Unit Circle and Radians
Senior High	Vectors and Complex Numbers	2	1,614	1.67%	Complex Numbers (Modulus/Conjugate/Argand Plane); Dot Product and Angle Between Vectors
Undergraduate	Calculus & Analysis	7	5,635	5.83%	Epsilon-Delta Limit Definition; Fourier Series; L'Hôpital's Rule; Line and Surface Integrals (Green/Stokes); Multiple Integrals (Double/Triple); Partial Derivatives; Taylor and Maclaurin Series
Undergraduate	Differential Equations	3	2,405	2.49%	First Order Linear ODEs; Laplace Transforms; Second Order Homogeneous ODEs
Undergraduate	Linear Algebra	6	4,799	4.96%	Determinants and Rank; Diagonalization; Eigenvalues and Eigenvectors; Gram-Schmidt Process; Matrix Inversion and Transpose; Vector Spaces and Subspaces
Undergraduate	Probability & Statistics	4	3,203	3.31%	Central Limit Theorem; Hypothesis Testing (t-test/Chi-square); Maximum Likelihood Estimation (MLE); PDF and CDF
Graduate	Abstract Algebra	3	2,416	2.50%	Galois Theory; Group Theory (Sylow Theorems); Ring Theory and Ideals
Graduate	Partial Differential Equations	4	3,215	3.33%	Heat Equation; Laplace Equation; Sobolev Spaces; Weak Solutions
Graduate	Real & Functional Analysis	5	4,032	4.17%	Banach and Hilbert Spaces; Hahn-Banach Theorem; Lebesgue Measure and Integral; Lp Spaces; Spectral Theory
Graduate	Topology	3	2,425	2.51%	Compactness and Connectedness; Fundamental Group and Homotopy; Separation Axioms (Hausdorff)

Note. The table summarizes 96,668 training problems. Percentages are relative to the full training split. The 120 atomic skills are obtained after de-duplicating repeated sub-category entries across pedagogical groupings. Specifically, *Estimation and Approximation*, *Mental Math Strategies*, *Multi-digit Multiplication and Division*, and *Order of Operations* appear in both Elementary Calculation and Elementary Word Problems, since they function both as basic arithmetic skills and as supporting skills for word-problem solving.



Curves are computed from saved evaluation JSON files; values show a 3-epoch rolling mean. Weak topics are those below 25% at epoch 6 with at least 3 eval examples.

Figure 4: Remedial recovery dynamics for three initially weak atomic skills. The vertical dashed line at Epoch 6 indicates the point where the adaptive remedial mechanism detected low accuracy and injected targeted supplemental data into the training batch.

tities of the models were anonymized, and the order of responses (A/B) was randomly shuffled for each instance to eliminate position bias.

D Formal Evaluation Metrics

To rigorously quantify model performance across our multi-dimensional evaluation methodology, we formally define the metrics as follows. Let $\mathcal{D}_{\text{test}}$ represent the evaluation dataset containing N instances. Each instance consists of a problem x_i, the ground-truth final answer a_i, and the requisite atomic skills $\mathcal{S}_i$. Let $\hat{y}_i = \{\hat{\mathcal{T}}_i, \hat{a}_i\}$ denote the model's generated traceback reasoning steps and predicted final answer, respectively.

Outcome Metrics. The standard final-answer accuracy evaluates strict exact-match correctness:

$$\text{Acc}_{\text{final}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[\hat{a}_i \equiv a_i]$$

where $\mathbf{1}[\cdot]$ is the indicator function, returning 1 if the predicted answer exactly matches (or is mathematically equivalent to) the ground truth, and 0 otherwise.

Outcome-based Process Metrics. To gauge performance on complex tasks, we compute final-answer accuracies over specific conceptual slices. Let $\mathcal{D}_k \subseteq \mathcal{D}_{\text{test}}$ denote a subset of problems belonging to a specific category, atomic skill, or difficulty bucket k. The slice-specific accuracy is defined as:

$$\text{Acc}_k = \frac{1}{|\mathcal{D}_k|} \sum_{x_i \in \mathcal{D}_k} \mathbf{1}[\hat{a}_i \equiv a_i]$$

For **Category Accuracy** and **Per-skill Answer Accuracy** (previously colloquially referred to as "Skill Accuracy"), we report the macro-average of Acc_k computed across all distinct categories or required skills present in the evaluation set. This prevents highly frequent topics from dominating the aggregate metric.

Structural Process Metrics (Step-skill Precision). To explicitly evaluate the model's diagnostic attribution capabilities—i.e., whether the model actually predicts the correct mathematical concepts—we introduce **Step-skill Precision**. Let $\hat{\mathcal{S}}_i$ be the unique set of `source_skill` tags autonomously declared by the model across its generated traceback steps $\hat{\mathcal{T}}_i$. The step-skill precision measures the proportion of declared skills that belong to the problem's ground-truth requisite taxonomy:

$$P_{\text{skill}} = \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{\mathcal{S}}_i \cap \mathcal{S}_i|}{|\hat{\mathcal{S}}_i|}$$

This metric strictly isolates the model's structural understanding from its calculation capabilities, directly penalizing hallucinated skill tags that do not apply to the given problem.

D.1 Formalization of the LLM-as-a-Judge Scoring

To ensure reproducibility and mitigate subjective variance, our LLM evaluator J (implemented via `gpt-4o-mini` with API parameters `temperature=0.0` and `response_format={"type": "json_object"}`) operates under a strict deterministic setting, computing a composite score based on three formal dimensions. Let $\hat{\mathcal{T}}_i = (\hat{t}_{i,1}, \ldots, \hat{t}_{i,M})$ denote the generated traceback sequence of length M for problem x_i, where each step $\hat{t}_{i,j} = \langle \hat{s}_{i,j}, \hat{c}_{i,j} \rangle$ consists of a declared skill $\hat{s}_{i,j}$ and reasoning content $\hat{c}_{i,j}$.

Dimension 1: Answer Correctness (C_i). The judge computes semantic and mathematical equivalence between the predicted final answer $\hat{a}_i$ and the ground truth a_i:

$$C_i = \mathbf{1}[\hat{a}_i \equiv a_i] \in \{0, 1\}$$

where $\equiv$ denotes mathematical equivalence (handling format variations such as $1/2$ and 0.5).

Dimension 2: Logical Coherence (L_i). This dimension evaluates the causal validity of the reasoning trajectory. The judge assigns a step-level validity score $v_{i,j} \in [0, 1]$ representing the logical continuity from the preceding step $\hat{t}_{i,j-1}$ to $\hat{t}_{i,j}$. The trajectory-level coherence is defined as:

$$L_i = \frac{1}{M} \sum_{j=1}^{M} v_{i,j} \in [0, 1]$$

Any "hallucinated leaps" or omitted requisite calculations heavily penalize the local step score $v_{i,j}$.

Dimension 3: Skill Alignment (S_i). To explicitly evaluate our framework's structural constraint, the judge scores the alignment between the declared atomic skill and the actual mathematical operation performed. Let $u_{i,j} \in [0, 1]$ be the alignment score for step j:

$$S_i = \frac{1}{M} \sum_{j=1}^{M} u_{i,j} \in [0, 1]$$

A score of $u_{i,j} = 1$ requires that the execution in $\hat{c}_{i,j}$ perfectly instantiates the conceptual anchor defined by $\hat{s}_{i,j}$ without conflating distinct skills.

Composite Score with Hallucination Penalty. The final composite score Score_i is calculated as a weighted linear combination of the three dimensions. Following our strict evaluation protocol, the weights are explicitly set to $w_c = 0.5$, $w_l = 0.3$, and $w_s = 0.2$. Crucially, to penalize scenarios where the model encourages a correct final answer from a fundamentally flawed intermediate premise (i.e., heuristic guessing), we introduce a non-linear penalty term $\Omega(C_i, L_i)$:

$$\text{Score}_i = \underbrace{0.5C_i + 0.3L_i + 0.2S_i}_{\text{Weighted Sum}} - \underbrace{\lambda \cdot C_i \cdot (1 - L_i)^{\alpha}}_{\text{Penalty: } \Omega(C_i, L_i)}$$

The parameter λ controls the penalty magnitude, and $\alpha \geq 1$ controls the sensitivity. This formulation ensures that $C_i = 1$ combined with a low logic score L_i heavily degrades the overall objective, consistently rewarding only robust, transparent deductions.

Evaluator Prompt Template. To ensure full transparency, the exact system and user prompts provided to the LLM judge are detailed below. The model is constrained to output a structured JSON evaluation.

System Prompt:
You are an objective math evaluator.
Score the model output from 0.0 to 1.0 using:
1) answer correctness (0.5)
2) reasoning coherence (0.3)
3) skill-trace alignment to source_skill/atomic_skills (0.2)
Return JSON only:
`{"score": float, "reason": string}`.

User Prompt Template:
Question:
{question}

Atomic skills:
{atomic_skills}

Gold:
`{"trace_back_steps": [...], "answer": "..."}`

Prediction:
{prediction}

E Detailed Baseline Configurations

To evaluate AtomicMath under controlled conditions, we compare against outcome-driven, curriculum-based, structured-reasoning, and process-supervision baselines. Table 10 summarizes the implementation of each baseline. Unless otherwise specified, all trainable baselines use the same backbone, QLoRA configuration, optimizer, data splits, answer-extraction script, and

evaluation protocol as AtomicMath. To isolate the contribution of skill-diagnostic attribution, all non-AtomicMath baselines are trained without access to ground-truth `source_skill` labels.

Base LLM. The untuned instruction-following base model is evaluated directly without task-specific fine-tuning. This baseline establishes the native reasoning floor of the underlying model before mathematical alignment.

Standard SFT. Standard supervised fine-tuning is trained on the same main training split as AtomicMath but uses conventional unstructured solution targets followed by the final answer. This baseline isolates the effect of outcome-driven imitation learning without explicit traceback supervision, curriculum adaptation, or skill-level remediation (Ouyang et al., 2022).

Curriculum-only SFT. This baseline follows the same phased easy-to-hard curriculum schedule as AtomicMath, using the basic, advanced, and hard difficulty buckets defined in Section 4.2. However, it removes traceback labels, `source_skill` annotations, and adaptive remedial injection. It therefore tests whether curriculum pacing alone can explain the observed gains.

Structured CoT SFT. Structured CoT SFT trains the model to produce ordered step-by-step rationales before the final answer (Wei et al., 2022). The output format preserves the decomposition of a solution into intermediate reasoning steps, but all explicit atomic skill tags are removed. This baseline separates the benefit of generic step-wise reasoning from the benefit of skill-grounded traceback supervision.

Verifier / Reranking. Following verifier-based mathematical reasoning methods (Cobbe et al., 2021), we train a scalar verifier to score complete candidate solutions. Positive instances are constructed from curated gold solutions in the training split, while negative instances are obtained from model-generated or automatically perturbed solutions with incorrect final answers or inconsistent intermediate reasoning. During inference, the generator produces $K = 16$ candidate solutions for each problem, and the verifier selects the highest-scoring candidate. This baseline can use complete reasoning traces for solution-level scoring, but it does not receive `source_skill` annotations or skill-level diagnostic feedback.

Math-Shepherd-style Step Feedback. We implement a Math-Shepherd-style baseline by converting curated solutions into step-level correctness supervision (Wang et al., 2024). Validated gold steps are treated as positive steps, while negative steps are derived from model-generated or automatically perturbed traces that introduce algebraic, logical, or formatting errors. The model is trained to favor reasoning trajectories with locally correct intermediate steps. Unlike AtomicMath, this baseline supervises whether a step is valid but does not identify which atomic skill supports that step.

STEP-KTO-style Hybrid Feedback. We construct a STEP-KTO-style baseline using stepwise desirable and undesirable feedback signals (Lin et al., 2025). Desirable examples are derived from validated gold reasoning steps, while undesirable examples are obtained from generated or perturbed steps that lead to incorrect reasoning trajectories. The model is optimized with step-level binary feedback under the same parameter-efficient fine-tuning setup. This baseline tests whether generic step-level feedback is sufficient, in contrast to AtomicMath's explicit `source_skill`-grounded traceback supervision and adaptive remediation.

AtomicMath. AtomicMath is trained to generate a skill-attributed traceback sequence followed by the final answer. Each reasoning step is paired with a `source_skill` label from the 120-way taxonomy, and the model is optimized with the joint traceback and final-answer objective described in Section 3. In addition, AtomicMath uses the remedial development set only for diagnostic routing: weak categories and atomic skills are identified periodically, and targeted examples are injected from the supplemental pool during training.

Fairness Controls. All baselines are evaluated on the same held-out test set and use the same answer parser and judge protocol. For process-supervision baselines requiring negative or feedback-labeled steps, negative examples are constructed only from the training split and are never drawn from $D_{\text{remedial_dev}}$ or D_{test}. The remedial development set is used only by AtomicMath for internal diagnostic routing and is not used for final evaluation. Thus, the key distinction between AtomicMath and the process-supervision baselines is not access to additional test information, but the availability of explicit skill-diagnostic attribution and targeted remedial injection.

Method	Training Target	Extra Supervision	Inference	Access to `source_skill`
Base LLM	No task-specific fine-tuning	None	Greedy decoding	No
Standard SFT	Unstructured solution followed by final answer	Final-answer imitation	Greedy decoding	No
Curriculum-only SFT	Unstructured solution and final answer under the same phased curriculum	Difficulty-bucket curriculum	Greedy decoding	No
Structured CoT SFT	Ordered step-by-step rationale followed by final answer	Step-wise rationale format	Greedy decoding	No
Verifier / Reranking	Scalar correctness score over complete candidate solutions	Positive gold solutions and negative generated or perturbed solutions	Selects the highest-scoring solution among $K = 16$ candidates	No
Math-Shepherd-style	Step-level correctness supervision	Positive gold steps and negative generated or perturbed steps	Greedy decoding	No
STEP-KTO-style	Stepwise desirable/undesirable feedback optimization	Binary feedback over intermediate reasoning steps	Greedy decoding	No
AtomicMath	Skill-attributed traceback followed by final answer	Step-level `source_skill` attribution and adaptive remedial injection	Greedy decoding	Yes

Table 10: Implementation summary of AtomicMath and all baselines. Process-supervision baselines receive generic step-level signals but do not receive explicit skill-diagnostic `source_skill` supervision.

F Computational Efficiency Details

Table 11 provides a detailed comparison of the training overhead, parameter counts, and memory requirements across all evaluated fine-tuning methods. To ensure a fair and transparent comparison, we discuss our configuration choices below.

Parameter Fairness and Capacity. As detailed in Table 11, AtomicMath utilizes a LoRA rank of $r = 32$ (yielding 20.185M trainable parameters). We emphasize that this capacity is necessitated by the increased structural complexity of our target output, which requires the model to simultaneously manage JSON syntax, step-level reasoning, and a 120-way skill taxonomy. In contrast, outcome-driven process-supervision baselines (e.g., Math-Shepherd-style) converge optimally with $r = 16$, as they lack these structural constraints. Crucially, as shown by our *Standard SFT ($r = 32$)* baseline, doubling the parameter count of the standard SFT approach yields only marginal gains (+0.30%) and fails to close the significant performance gap with AtomicMath, confirming that our improvements are architectural rather than merely parametric.

Computation-Matched Comparisons. AtomicMath requires longer wall-clock training time (41.12 hours) primarily due to the increased sequence length of structured traceback tokens and the dynamic injection of remedial data. To isolate the effects of this increased compute budget, we introduced *Compute-matched* baselines for both Standard SFT and Structured CoT. In these settings, we extended the training duration of the baselines (up to 28–29 epochs) to match the total training hours of AtomicMath. Despite this exhaustive optimization, the performance of these compute-matched baselines remains substantially lower than AtomicMath. This result demonstrates that the "AtomicMath advantage" stems from the qualitative quality of skill-aware supervision and targeted remediation, rather than a quantitative increase in training tokens or time.

Memory and Deployment. Despite the structural complexity, AtomicMath introduces zero additional memory overhead compared to a standard LoRA SFT of the same rank. All experiments were conducted on a single consumer-grade GPU (e.g., NVIDIA RTX 4090), and the resulting adapters are lightweight ($\sim$80.8 MB), ensuring that the enhanced reasoning capabilities are accessible for real-world deployment without requiring high-performance computing clusters.

Table 11: Computational efficiency and parameter comparison. Training hours estimate active time (excluding evaluation). To accommodate the structural complexity of JSON-traceback sequences, our method and structured baselines use a LoRA rank of $r = 32$. Process-supervision baselines utilize $r = 16$ as they lack these structural constraints. We introduce parameter-matched ($r = 32$) and compute-matched baselines to ensure fair comparisons.

Method	**Train Hours**	**h/Epoch**	**Epochs**	**Final Acc. (%)**	**Judge**	**Trainable Params (M)**	**Adapter MB**
Standard SFT ($r = 16$)	22.76	1.42	16	19.55	0.542	10.093	40.4
Standard SFT ($r = 32$)*	23.45	1.46	16	19.63	0.545	20.185	80.8
Standard SFT (Compute-matched, $r = 32$)*	41.12	1.46	28	19.92	0.547	20.185	80.8
Structured CoT ($r = 32$)	34.60	2.16	16	18.42	0.535	20.185	80.8
Structured CoT (Compute-matched, $r = 32$)*	41.12	2.16	19	18.78	0.538	20.185	80.8
Curriculum SFT ($r = 32$)	32.08	2.01	16	20.66	0.548	20.185	80.8
Math-Shepherd-style ($r = 16$)	9.36	0.59	16	17.15	0.495	10.093	40.4
STEP-KTO-style ($r = 16$)	9.44	0.59	16	17.47	0.508	10.093	40.4
AtomicMath (Ours, $r = 32$)	**41.12**	**2.57**	**16**	**24.11**	**0.638**	**20.185**	**80.8**