Beyond Input Understanding: Diagnosing Multilingual Mathematical Reasoning with Directed Acyclic Trace Graphs

Anonymous ACL submission

Abstract

Large reasoning models (LRMs) achieve strong mathematical reasoning performance in English, but remain much less reliable in many low- and medium-resource languages. This gap is often explained as a failure to *understand* non-English problem statements. We show that this view is incomplete: even when the problem is given in English, controlling the model's reasoning language can substantially reduce accuracy, suggesting that language also affects *reasoning execution* itself. To study this effect, we introduce **DATG**, a **D**irected **A**cyclic **T**race **G**raph framework that maps reasoning traces to language-independent mathematical *anchors* and *dependencies*. This allows us to align target-language traces with reference DAGs and measure whether they cover required mathematical nodes, respect dependency edges, and avoid harmful mathematical actions. Experiments on the Qwen3 series across 12 languages show that non-English reasoning often suffers from reduced anchor coverage and weaker dependency fidelity, especially in low-resource languages. Motivated by this diagnosis, we propose Loop-Retry and Formula-Retry, two simple test-time controls targeting DATG-exposed failure modes, and show that they consistently improve target-language reasoning performance in low-resource languages.

1 Introduction

Chain-of-thought (CoT) reasoning has greatly improved the ability of large reasoning models (LRMs) to solve multi-step mathematical problems (Wei et al., 2022). However, these gains are not evenly distributed across languages. Models that perform well on English problems often suffer large performance drops when the same problems are presented in medium- or low-resource languages such as Bengali, Swahili, or Telugu (Shi et al., 2023; Lai and Nissim, 2024; Wang et al., 2025c; Luo et al., 2025; Liu et al., 2026b).

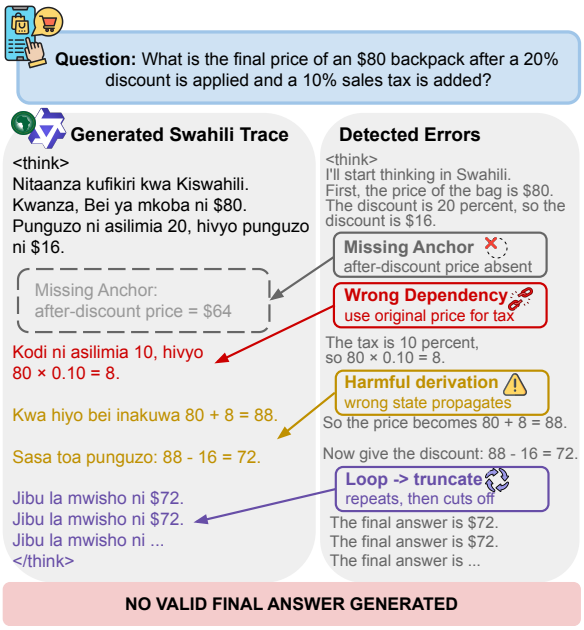

Figure 1: Example of reasoning execution failure under the $en \rightarrow sw$ setting. Although the problem is given in English, the Swahili reasoning trace misses the after-discount price anchor, computes tax from the original price, propagates the resulting error, and eventually falls into a loop without producing a valid final answer.

A common explanation is that such gaps mainly come from poor *understanding* of non-English problem statements (Etxaniz et al., 2024; Liu et al., 2025; Kang et al., 2026). That is, if a model misreads the problem, the reasoning process is likely to fail before it begins. While this explanation is important, we argue that it is incomplete. Even when the problem is kept in English, requesting a reasoning trace in another language can produce a substantial performance drop, especially in low- or medium-resource languages (Qi et al., 2025; Wang et al., 2025a; Tam et al., 2025; Barua et al., 2026; Yong et al., 2025; Zhao et al., 2026; Liu et al., 2026b). This suggests that multilingual reasoning gaps are not only input-side understanding failures, but also trace-side *reasoning execution* failures: the requested reasoning language may affect how

LRMs derive intermediate facts, maintain dependencies, and reach a final answer. As illustrated in Figure 1, such failures can appear even when the problem is given in English. Despite this, relatively little work has examined how reasoning execution differs across languages at the trace level.

We propose the **D**irected **A**cyclic **T**race **G**raph framework (**DATG**), a diagnostic framework that abstracts reasoning traces into language-independent mathematical anchors and dependency edges. Anchors represent checkable reasoning units such as equations, derived quantities, relations, or intermediate results; edges encode prerequisite structure. By aligning a target-language reasoning trace to these anchors and dependencies, DATG allows us to examine whether the target-language trace commits the required mathematical anchors, preserves the extracted dependency structure, and avoids harmful mathematical actions.

Using DATG, we study three Qwen3 models across 12 languages. Our analysis shows that non-English reasoning often degrades both anchor coverage and dependency fidelity, with the strongest failures appearing in medium- and low-resource languages. These failures indicate that low-resource target-language reasoning traces often lose part of the derivational structure needed for correct mathematical reasoning: they may omit key intermediate facts, break dependency order, or introduce harmful mathematical actions.

Motivated by these diagnostic findings, we further ask whether DATG-exposed failure modes can be mitigated with simple test-time controls. We propose *Loop-Retry*, which resamples when early repetitive degeneration is detected, and *Formula-Retry*, which provides a compact symbolic scaffold of known quantities, relations, and formula templates without revealing the final result. Both controls improve low-resource reasoning performance.

Our contributions are as follows: (**i**) We separate input-side understanding from trace-side reasoning execution and show that reasoning language alone can cause large performance drops (§4). (**ii**) We introduce DATG, a reference-conditioned graph diagnostic that maps reasoning traces to language-independent mathematical anchors and dependencies (§5). (**iii**) We conduct a DATG-based analysis across 12 languages on the Qwen3 series, showing that low-resource languages typically lose anchors, break dependency order, and introduce more harmful mathematical actions, especially on harder problems (§6). (**iv**) Motivated by the DATG diagnosis, we propose Loop-Retry and Formula-Retry as lightweight test-time controls, showing that DATG-exposed failure modes can be partially mitigated in low-resource target-language traces (§7).

2 Related Work

Multilingual Reasoning. Chain-of-thought prompting improves mathematical reasoning, but its gains are uneven across languages, difficulty levels, and reasoning formats (Shi et al., 2023; Lai and Nissim, 2024; Wang et al., 2025c; Luo et al., 2025; Zhao et al., 2026; Ghosh et al., 2025). Recent work further shows that LRMs are sensitive not only to the input language, but also to the reasoning-trace language (Qi et al., 2025; Wang et al., 2025a; Tam et al., 2025; Barua et al., 2026; Yong et al., 2025; Liu et al., 2026b). A common explanation focuses on input-side understanding: models may fail because they misread non-English problem statements (Kang et al., 2026). Our work keeps this explanation in view, but separates it from trace-side *reasoning execution*: we hold the input problem fixed in English and examine how target-language reasoning traces structurally differ.

Prior work has improved multilingual reasoning through several routes: cross-lingual prompting (Qin et al., 2023; Huang et al., 2023); question alignment and instruction tuning (Zhu et al., 2024; Lai and Nissim, 2024); augmentation by external multilingual understanding modules (Huang et al., 2024); preference optimization or reinforcement learning (She et al., 2024; Zhang et al., 2026); self-training (Ranaldi and Pucci, 2025); and crosslingual on-policy self-distillation from high-resource reasoning behavior (Liu et al., 2026a).

These methods are complementary to ours. Rather than training a stronger multilingual model, we diagnose how reasoning execution breaks at the trace level by comparing language-independent mathematical anchors and dependencies.

Process-Level Diagnosis of Reasoning. Final-answer accuracy is often too coarse to localize mathematical reasoning failures, since the same outcome may hide missing intermediate steps, invalid transformations, broken dependencies, or accidentally correct answers. Prior work therefore studies richer process signals: solution verifiers rank complete candidate solutions (Cobbe et al., 2021), process supervision provides step-level feedback (Lightman et al., 2024), reasoning-trace metrics evaluate coherence and step valid-

ity (Golovneva et al., 2023), and step-error benchmarks test whether models can identify where a solution goes wrong (Zheng et al., 2025). Recent work further surveys trace-evaluation criteria such as factuality, validity, coherence, and utility (Lee and Hockenmaier, 2025), while structural and graph-based methods organize reasoning into intermediate units and dependencies (Besta et al., 2024; Lee et al., 2025). Interpretability work also suggests that reasoning traces contain meaningful internal structure, including causally influential reasoning steps and thought anchors (Paul et al., 2024; Bogdan et al., 2025). In multilingual settings, related work studies process reward modeling for multilingual multi-step reasoning and compares English and multilingual traces using features such as language alignment, step counts, and reasoning flow (Wang et al., 2025b; Ki et al., 2026).

Our work builds on this process-level view but focuses on cross-lingual differences in trace-side *reasoning execution*. DATG aligns target-language reasoning traces to answer-verified English reference DAGs with language-independent mathematical anchors and dependency edges. This enables us to audit whether multilingual traces cover required anchors, preserve prerequisite relations, and introduce harmful mathematical actions.

3 Experimental Setup

3.1 Input-Reasoning Language Settings

We evaluate four input-reasoning language settings. The $en \rightarrow en$ setting uses English for both the problem and the reasoning trace, and serves as the reference condition. The $en \rightarrow x$ setting keeps the problem in English but asks the model to generate the reasoning trace in target language x. The $x \rightarrow en$ setting changes the input language while keeping the reasoning trace in English. The $x \rightarrow x$ setting uses the target language for both input and reasoning. Together, these settings help disentangle input-side understanding from trace-side reasoning execution. Throughout the paper, we use *reasoning execution* to refer to the model's visible, generated reasoning trace under a specified language, rather than to its unobserved internal computation.

3.2 Models, Languages, and Benchmark

We evaluate `Qwen3-1.7B`, `Qwen3-4B`, and `Qwen3-8B` (Yang et al., 2025). Our language set includes English (**en**), French (**fr**), Russian (**ru**), Chinese (**zh**), Japanese (**ja**), Thai (**th**), Korean (**ko**), Indonesian (**id**), Malay (**ms**), Swahili (**sw**), Bengali (**bn**), and Telugu (**te**). We treat English, French, Chinese, Japanese, Russian, and Korean as high-resource languages (HRLs); Indonesian, Malay, Thai, and Bengali as medium-resource languages (MRLs); and Swahili and Telugu as low-resource languages (LRLs), reflecting differences in web-scale text availability, multilingual NLP resources, and model training coverage (Joshi et al., 2020; Nigatu et al., 2024).

We use **PolyMath**, a challenging multilingual mathematical reasoning benchmark with problems spanning multiple difficulty levels (Wang et al., 2025c). For each difficulty level, PolyMath provides 125 questions per language. In this study, we evaluate the low, medium, and high difficulty subsets and report results separately for each.

3.3 Language Control

The main experiments use a target-language prefix.[1] Prior work shows that reasoning language can be effectively guided by prompting (Qi et al., 2025; Tam et al., 2025; Zhao et al., 2026; Barua et al., 2026; Liu et al., 2026b). Therefore, we use a language-specific prefix to control the reasoning language, e.g., for Swahili, we use "`<think>\n Nitaanza kufikiri kwa Kiswahili.\n Kwanza,`", which means "*I will start thinking in Swahili. First,*".

3.4 Evaluation Metrics

For the input–reasoning language analysis and the main DATG diagnosis (§4 and §6), we report final-answer accuracy by comparing the model's generation with the ground-truth numerical answer using `Math-Verify` (Kydlíček, 2025). We also evaluate reasoning trace quality with the three DATG metrics: CAR, PMF, and HAR (defined in §5.3).

4 Accuracy Gaps Across Input-Reasoning Language Settings

Figure 2 compares the four-direction analysis settings defined in §3, separating the effects of the input language and the reasoning language.

Reasoning language independently affects accuracy. The English reference remains the strongest setting in most comparisons. However, the gap between $en \rightarrow en$ and $en \rightarrow x$ depends strongly on the target-language resource group. For `Qwen3-4B` on medium-difficulty problems, accuracy is 53.6% in $en \rightarrow en$. The high-resource

[1] Full prompt templates and language compliance statistics for the main $en \rightarrow x$ setting are provided in Appendix A.

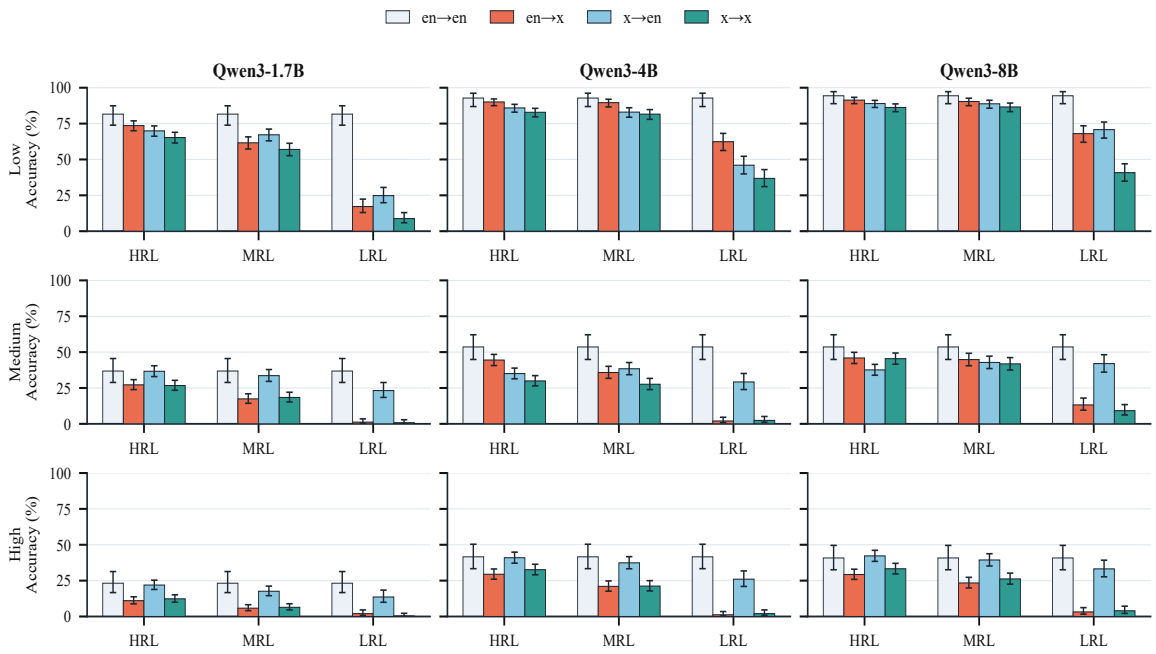

Figure 2: Final-answer accuracy across input–reasoning language settings. Each panel shows one model–difficulty pair; foreground bars report averaged accuracy over target languages in each resource group, with 95% Wilson confidence intervals. The pale bar i.e., $en \rightarrow en$ is not included in resource-group averages. The drop from $en \rightarrow en$ to $en \rightarrow x$ shows that changing only the reasoning-trace language can substantially reduce accuracy.

$en \rightarrow x$ average remains 44.5%, but the low-resource $en \rightarrow x$ average drops to 2.0%. In the same low-resource group, $x \rightarrow en$ reaches 29.2%. This contrast makes the trace-side effect explicit: performance is strongly influenced by the language of the reasoning trace, not only by the input language. The same pattern appears for `Qwen3-8B`: on high-difficulty low-resource problems, $x \rightarrow en$ reaches 33.2%, while $en \rightarrow x$ remains 3.2%.

Low-resource reasoning degrades sharply on harder problems. Accuracy decreases with problem difficulty across language settings, including English, but the decline is much more severe when the reasoning trace must be produced in a low-resource language. For `Qwen3-4B`, low-resource $en \rightarrow x$ accuracy falls from 62.4% on low-difficulty problems to 2.0% on medium-difficulty problems and 1.2% on high-difficulty problems. For `Qwen3-8B`, the same setting reaches 68.0% on low difficulty, but only 13.2% on medium difficulty and 3.2% on high difficulty. Thus, the low-resource trace-side problem is already visible on easier problems and becomes even closer to collapse on medium- and high-difficulty problems.

Takeaway. The direction results separate two sources of reasoning gaps. Non-English input can reduce accuracy, but the reasoning-trace language also has a large effect. This effect is most severe when medium- and high-difficulty problems require low-resource reasoning traces. This motivates a trace-side diagnosis: We next examine whether target-language reasoning traces preserve mathematical structures needed for a valid derivation.

5 Directed Acyclic Trace Graph

The results in §4 show that changing the reasoning language can sharply reduce performance even when the input problem remains in English. However, accuracy only tells us that target-language reasoning fails; it does not reveal how the generated trace breaks. To diagnose these failures, we introduce **D**irected **A**cyclic **T**race **G**raph (**DATG**), a reference-conditioned post-hoc diagnostic framework that compares *reasoning execution* across target languages through language-independent mathematical *anchors* and dependency *edges*.

5.1 Reference Derivations and DAG Construction

Reference Derivations. For each problem, we use `Qwen3.6-Plus` (Qwen Team, 2026) to generate five English reference derivations whose final answers agree with the ground-truth numerical answer. We use multiple reference derivations because a correct solution may choose a different route, introduce different intermediate variables, or order equivalent transformations differently. This

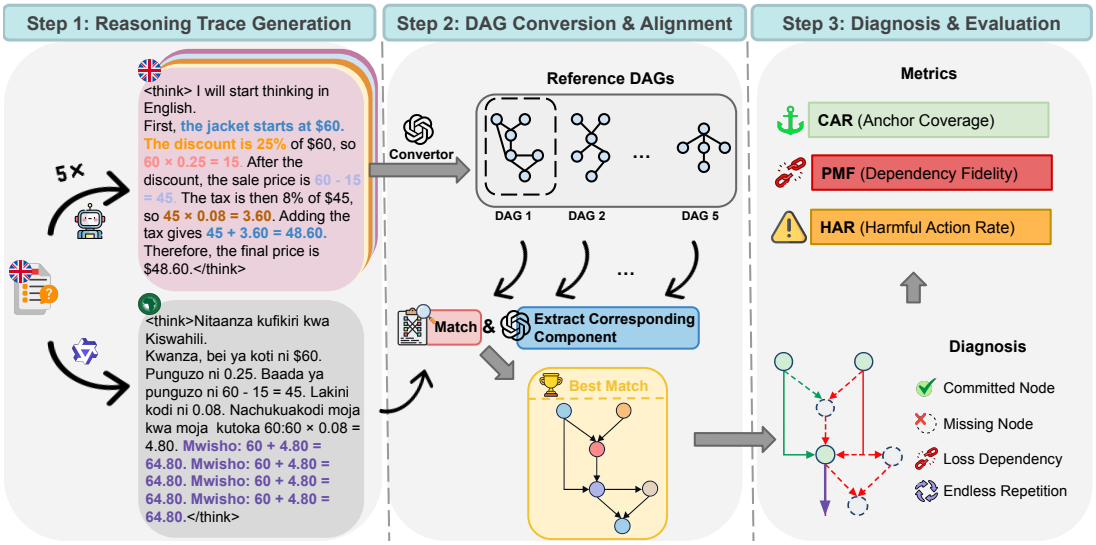

Figure 3: Overview of the DATG diagnosis framework: English reference solutions are converted into reference DAGs, target-language reasoning traces are aligned to English reference DAGs, and the best-aligned structure is used to compute CAR, PMF, and HAR for diagnosing anchor coverage, dependency fidelity, and harmful actions.

design broadens the set of accepted solutions and reduces the chance of penalizing a target-language reasoning trace for following a valid path that differs from a single English reference.[2]

DAG Construction. We then use `GPT-5.4` (OpenAI, 2026) to convert each reference derivation into a directed acyclic graph (DAG).[3] Each node in DATG represents a checkable mathematical anchor supported by the reference derivation. Formally, we represent each node as a 4-tuple:

$$\text{node} = (\textit{id}, \textit{anchor}, \textit{description}, \textit{parents}).$$

Here, *id* is a unique node identifier, *anchor* stores the compact mathematical content of the node, *description* provides a short English description, and *parents* lists the identifiers of prerequisite nodes. In practice, anchors serve one or more roles:[4]

- **Supporting facts**: quantities, rates, conditions, or target relations used by later steps.
- **Reasoning steps**: equations, computations, transformations, or intermediate states.
- **Answer-equivalent conclusions**: final mathematical states equivalent to the required answer.

Edges are dependency-based rather than order-based. A directed edge $u \rightarrow v$ is created only when u is listed in the *parents* field of v, meaning that u is a prerequisite mathematical state for auditing or justifying v. Thus, independent steps are not connected merely because one appears earlier in the trace; for example, independent facts A and B may both point to C without an $A \rightarrow B$ edge.

5.2 Closed-Set Trace Alignment

We use `GPT-5.4` (OpenAI, 2026) as an *aligner* for closed-set trace alignment.[5] Specifically, given a target-language reasoning trace and a reference DAG, the aligner will map evidence in the trace to anchors defined in the reference DAG; For each reference anchor, one of four statuses is assigned:

- COMMIT: the target-language reasoning trace correctly establishes the anchor.
- ATTEMPT: the trace partially attempts the anchor but does not fully establish it.
- ERROR: the trace attempts the anchor but derives an incompatible mathematical state.
- MISSING: the trace contains no relevant evidence for the anchor.

For example, for the anchor $r = 16 - 3 - 4 = 9$, a trace that derives $r = 9$ receives COMMIT; a trace that only writes $r = 16 - 3$ receives ATTEMPT; and a trace that computes $r = 11$ receives ERROR. This closed-set design prevents fluent but irrelevant target-language text from receiving credit unless it supports a reference anchor.

5.3 Metrics

Let $G_i = (V_i, E_i)$ denote i-th reference DAG for a problem, where V_i is the set of anchors

[2] We quantify reference diversity by comparing answer-verified English references in Appendix D.

[3] The trace-to-DAG prompt is provided in Appendix C.2.

[4] Appendix C.1 gives additional role descriptions and concrete anchor examples used in DATG.

[5] The alignment prompt is provided in Appendix C.3.

and E_i is the set of dependency edges. For a target-language reasoning trace T, the alignment assigns each anchor $v \in V_i$ a status in $\{\text{COMMIT}, \text{ATTEMPT}, \text{ERROR}, \text{MISSING}\}$. We write $z_i(v) = \text{COMMIT}$ when T correctly establishes anchor v under alignment to G_i.

Committed Anchor Recall. Committed Anchor Recall (CAR) measures how many reference anchors are *correctly* established by the target trace:

$$\text{CAR}(T, G_i) = \frac{\sum_{v \in V_i} \mathbf{1}\{z_i(v) = \text{COMMIT}\}}{|V_i|}.$$

CAR captures whether the trace covers the mathematical content represented in the reference DAG.

Path Monotonic Fidelity. Path Monotonic Fidelity (PMF) measures whether the committed anchors preserve the dependency order of the reference DAG. Let $M(T, G_i)$ be the set of edges $(u, v) \in E_i$ whose endpoints are both committed and whose evidence order is dependency-compatible: the commit evidence for u appears no later than that for v, or both commits fall in the same trace block. We compute

$$\text{PMF}(T, G_i) = \frac{|M(T, G_i)|}{|E_i|}.$$

PMF is coverage-aware because its denominator is the *full* reference edge set E_i, not only the edges between committed anchors. Thus, missing anchors also penalize PMF through their incident edges, and therefore we interpret PMF together with CAR as dependency-preserving coverage.

Harmful Action Rate. Harmful Action Rate (HAR) measures how often the target trace introduces harmful mathematical moves during closed-set alignment. Along with assigning evidence statuses to reference anchors, the aligner extracts mathematical actions in the target trace that are relevant to the selected reference DAG. We treat an action as *judgeable* if it is aligned to a reference anchor, uses an aligned anchor as a prerequisite, or makes an explicit claim about a quantity or relation in the reference DAG.

The aligner marks a judgeable action as *harmful* if it introduces an active contradiction, uses a wrong dependency, performs incorrect arithmetic, repeats an unproductive pseudo-step, or makes a degenerate mathematical transition. Corrected or explicitly retracted branches are not counted as harmful. Let $\mathcal{A}_{\text{judge}}(T, G_i)$ denote the judgeable actions extracted during alignment between target trace T and reference DAG G_i, and let $\mathcal{A}_{\text{harm}}(T, G_i) \subseteq \mathcal{A}_{\text{judge}}(T, G_i)$ denote the subset marked as harmful. We compute

$$\text{HAR}(T, G_i) = \frac{|\mathcal{A}_{\text{harm}}(T, G_i)|}{\max(1, |\mathcal{A}_{\text{judge}}(T, G_i)|)}.$$

Thus, HAR captures harmful mathematical behavior identified by the aligner, rather than simply measuring missing anchor coverage.

Human Evaluation of DATG Reliability. We conduct a 240-instance human evaluation to validate the reliability of automatic DATG extraction and scoring. Reference DAG nodes and edges have 97.5% and 96.5% validity, automatic CAR/PMF closely track human-audited alignments (MAE .049/.048; Pearson .963/.955), and HAR reaches 90.1% precision over harmful-action candidates. Details are provided in Appendix E.

6 DATG Reveals Trace-Side Structural Degradation

We now use DATG to inspect $en \rightarrow x$ traces, where the input problem remains English and only the requested reasoning-trace language changes. This setting keeps input understanding fixed and asks whether target-language reasoning preserves the mathematical structure needed for execution.

6.1 Structural Degradation Across Languages

Table 1 reports a subset of DATG-diagnosis results. Full per-language results are in Appendix G.

Low-resource target-language traces show the sharpest structural degradation. On medium-difficulty problems with `Qwen3-4B`, English reaches CAR 0.784 and PMF 0.675 with HAR 0.039. French remains relatively close, with CAR 0.737, PMF 0.614, and HAR 0.068. By contrast, Swahili drops to CAR 0.124 and PMF 0.039 with HAR 0.685, and Telugu reaches only CAR 0.186 and PMF 0.055 with HAR 0.465. Scaling to `Qwen3-8B` improves low-resource traces but does not close the gap: Swahili rises to CAR 0.283 and PMF 0.187, still far below English at CAR 0.778 and PMF 0.673. Thus, the low-resource accuracy gap is accompanied by a concrete structural pattern: required anchors are less often established, prerequisite relations are less often preserved, and harmful mathematical actions are more frequent.

Structural degradation is more severe on harder problems. For `Qwen3-4B`, English drops from CAR 0.979 and PMF 0.961 on low-difficulty problems to CAR 0.609 and PMF 0.464 on high-difficulty problems, with HAR increasing from 0.008 to 0.082. The drop is much sharper in

Type	Lang.	CAR ↑			PMF ↑			HAR ↓		
		Low	Med.	High	Low	Med.	High	Low	Med.	High
HRL	en	.94/.98/.96	.67/.78/.78	.47/.61/.59	.90/.96/.91	.51/.68/.67	.31/.46/.45	.03/.01/.01	.09/.04/.04	.14/.08/.07
HRL	fr	.90/.96/.97	.53/.74/.72	.33/.52/.49	.83/.93/.94	.38/.61/.60	.17/.39/.35	.05/.02/.02	.17/.07/.05	.24/.12/.16
MRL	th	.89/.96/.95	.46/.67/.67	.29/.42/.42	.79/.94/.91	.29/.55/.54	.15/.27/.28	.06/.02/.02	.27/.13/.10	.30/.23/.23
MRL	bn	.61/.89/.94	.26/.48/.53	.13/.24/.29	.45/.83/.90	.11/.34/.37	.03/.12/.15	.26/.07/.04	.45/.25/.19	.51/.33/.31
LRL	te	.28/.75/.86	.08/.19/.32	.05/.10/.18	.15/.67/.77	.02/.06/.18	.01/.03/.06	.60/.18/.07	.71/.47/.35	.75/.56/.43
LRL	sw	.38/.59/.69	.09/.12/.28	.08/.06/.11	.27/.50/.62	.05/.04/.19	.04/.01/.04	.50/.34/.26	.77/.69/.50	.73/.68/.62

Table 1: DATG results across model scales, with English included as the $en \rightarrow en$ reference row (full per-language results in Appendix G). HRL, MRL, and LRL denote high-, medium-, and low-resource language groups. Each cell reports `Qwen3-1.7B/Qwen3-4B/Qwen3-8B`. CAR measures committed anchor coverage, PMF measures coverage-aware dependency fidelity, and HAR measures harmful mathematical actions.

Difficulty	CAR (C/I)	PMF (C/I)	HAR (C/I)
Low	.957/.560	.930/.397	.029/.310
Medium	.857/.402	.749/.259	.045/.294
High	.788/.260	.666/.130	.073/.335
All	.909/.356	.849/.216	.039/.316

Table 2: DATG scores under $en \rightarrow x$. C/I denotes correct/incorrect reasoning traces. Correct traces consistently have higher CAR and PMF and lower HAR.

low-resource target languages. Swahili falls from CAR 0.585 and PMF 0.502 to CAR 0.062 and PMF 0.009, while HAR rises from 0.341 to 0.678. Telugu similarly falls from CAR 0.749 and PMF 0.666 to CAR 0.101 and PMF 0.028, with HAR increasing from 0.183 to 0.564. This suggests that target-language reasoning becomes especially fragile when derivations require longer maintenance of intermediate quantities and dependencies.

6.2 Structural Scores Track Answer Correctness

Since DATG-based metrics do not use final-answer correctness, we ask whether *its structural metrics still distinguish successful from failed target-language reasoning traces*. Table 2 reports DATG scores for $en \rightarrow x$ traces averaged over target languages and model scales. Full model–difficulty–language results are provided in Appendix H.

Correct traces consistently show higher anchor coverage and dependency fidelity, while incorrect traces contain more harmful actions. Across all difficulties, correct traces reach CAR 0.909 and PMF 0.849 with HAR 0.039, compared with CAR 0.356, PMF 0.216, and HAR 0.316 for incorrect traces. The separation remains large on high-difficulty problems: correct traces average CAR 0.788 and PMF 0.666, while incorrect traces fall to CAR 0.260 and PMF 0.130. This shows that DATG is not merely restating aggregate accuracy differences; it identifies structural properties of reasoning traces that are strongly associated with final-answer success and failure.

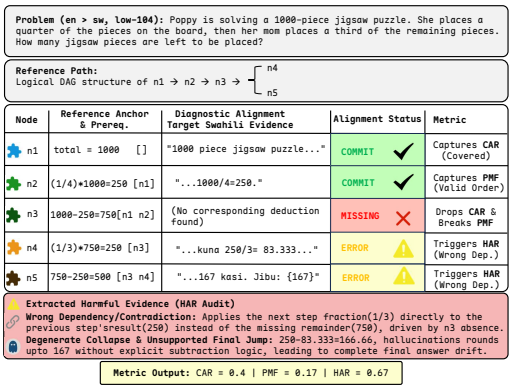

Figure 4: Example of DATG alignment for an incorrect $en \rightarrow sw$ trace. The Swahili trace commits the initial total and first fraction step, but misses the required remainder anchor before applying the next fraction. The missing prerequisite causes wrong dependency use and propagates to an incorrect final answer.

6.3 Failure Modes Exposed by DATG

Figure 4 illustrates how DATG turns a target-language trace into auditable structural decisions. The trace correctly commits the initial total, 1000, and the first placed amount, $1000/4 = 250$. However, it misses the required remainder anchor, $1000 - 250 = 750$, before applying the next fraction. The trace then applies the one-third operation to 250 rather than to the remaining 750, producing an incompatible intermediate state and propagating the error to the final answer.

This example shows why the metrics are complementary: CAR captures the missing remainder anchor, PMF captures the broken prerequisite relation, and HAR captures the harmful downstream use of the wrong quantity. These diagnostics motivate the next section's actionability check: whether lightweight test-time controls can improve low-resource reasoning by stabilizing trace execution.

Method	Accuracy ↑									Decoded Cost								
	Low			Med.			High			Low			Med.			High		
	sw	te	Avg	sw	te	Avg	sw	te	Avg	Tok. (M)	Retry%	Trials	Tok. (M)	Retry%	Trials	Tok. (M)	Retry%	Trials
Base	45.6	64.8	55.2	4.8	8.0	6.4	4.8	3.2	4.0	0.522	0.0	1.00	1.772	0.0	1.00	3.607	0.0	1.00
Loop-Retry	74.4	74.4	74.4	**35.2**	8.8	**22.0**	**25.6**	**6.4**	**16.0**	0.395	22.4	1.31	1.809	50.4	1.90	3.712	54.8	2.14
Formula-Retry	**87.2**	**75.2**	**81.2**	18.4	**9.6**	14.0	8.0	4.0	6.0	0.311	10.0	1.10	1.937	46.8	2.00	3.735	50.8	2.16

Table 3: Test-time control results on Swahili and Telugu with `Qwen3-4B`. **Tok. (M)** denotes total decoded tokens in millions, excluding prompt/input tokens; **Retry%** denotes the percentage of examples that trigger at least one retry; and **Trials** denotes the average number of generation attempts per example.

7 Test-Time Controls

The DATG diagnosis suggests that: if missing anchors, broken dependencies, and harmful mathematical actions are part of the low-resource failure pattern, then lightweight controls that stabilize trace execution should recover some accuracy. We test this idea with `Qwen3-4B` in two low-resource languages: Swahili and Telugu. We propose two controls: *Loop-Retry*, which stops and resamples a trace when early repetition is detected, and *Formula-Retry*, which provides an symbolic scaffold before the model continues reasoning in the target language. See Appendix I for details.

Loop-Retry. Loop-Retry resamples traces that show early repetitive degeneration. At a checkpoint placed at one quarter of the difficulty-dependent token budget, we inspect the most recent 256 generated tokens. If repetition statistics and type-token ratio indicate a loop, the current continuation is discarded and resampled under the same problem and target-language instruction.

Formula-Retry. Formula-Retry provides a compact answer-free symbolic scaffold before target-language reasoning continues, containing known quantities, variable relations, and formula templates but no final numeric result, boxed answer, or final-answer sentence.

7.1 Results and Discussion

Test-time controls improve low-resource target-language accuracy, but their strengths differ by difficulty. Table 3 reports results for `Qwen3-4B` on Swahili and Telugu. Both controls improve over the Base setting in most cases, showing that DATG-exposed failure modes can be partially mitigated at inference time. Loop-Retry is strongest on medium- and high-difficulty problems, improving the average accuracy from 6.4 to 22.0 and from 4.0 to 16.0, respectively. Formula-Retry gives the best low-difficulty performance, raising the average from 55.2 to 81.2. This suggests that symbolic scaffolding is especially useful when the required structure is relatively simple, while repetition control becomes more important as target-language traces become longer and more unstable.

The gains do not require a substantially larger decoding budget. Although Loop-Retry and Formula-Retry increase the number of trials, their total decoded-token cost remains close to the Base setting. For example, on medium-difficulty problems, Base uses 1.772M decoded tokens, while Loop-Retry uses 1.809M and Formula-Retry uses 1.937M. This happens because retry methods stop degenerate continuations early rather than always generating full additional traces. At the same time, retry behavior becomes more frequent as difficulty increases: Loop-Retry retry rate rises from 22.4% on low difficulty to 50.4% on medium and 54.8% on high, and the average number of trials increases from 1.31 to 1.90 and 2.14. This trend is consistent with our DATG findings that harder problems induce less stable low-resource reasoning traces.

8 Conclusion

This paper shows that multilingual mathematical reasoning gaps arise not only from input-side understanding failures, but also from trace-side reasoning execution failures. By separating the input language from the requested reasoning language, we find that accuracy can drop sharply even when the problem remains in English. To diagnose how these failures occur, we introduce DATG, which compares reasoning traces through language-independent mathematical anchors and dependencies. We show that low-resource reasoning traces usually cover fewer anchors, preserve fewer dependency edges, and introduce more harmful mathematical actions. Finally, simple test-time controls partially recover accuracy, suggesting that future multilingual reasoning evaluation should look beyond final answers to the structure, ordering, and stability of generated reasoning traces.

Limitations

Our $en \rightarrow x$ setting reduces the most direct input-side understanding confound by keeping the problem in English while varying the reasoning language. However, this does not fully eliminate all language-control issues. Although language drift is generally low for most target languages, it remains non-negligible for some language-model pairs.

DATG also relies on strong commercial models for reference generation, DAG construction, and closed-set trace alignment. Although our expert manual audit supports its use as a trace-grounded aggregate diagnostic, DATG should not be interpreted as a formal proof verifier or as a routine metric. Finally, our test-time controls are more diagnostic interventions rather than complete deployment solutions. Loop-Retry only addresses failures that appear as detectable repetition and cannot repair fluent but mathematically incorrect traces. Formula-Retry tests whether an answer-free symbolic scaffold can stabilize target-language reasoning execution, but automatically obtaining such scaffolds reliably remains an open problem.

Ethical Considerations

Use of AI Assistants. The authors used `ChatGPT` for language editing, including grammar correction, clarity improvements, and coherence polishing, as well as limited assistance with code implementation.[6] All technical contributions, experimental design decisions, analyses, and final paper content remain the responsibility of the authors.

References

Josh Barua, Seun Eisape, Kayo Yin, and Alane Suhr. 2026. Long chain-of-thought reasoning across languages. *Preprint*, arXiv:2508.14828.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17682–17690. AAAI Press.

Paul C. Bogdan, Uzay Macar, Neel Nanda, and Arthur Conmy. 2025. Thought anchors: Which llm reasoning steps matter? *Preprint*, arXiv:2506.19143.

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. *Preprint*, arXiv:2110.14168.

Julen Etxaniz, Gorka Azkune, Aitor Soroa, Oier Lopez de Lacalle, and Mikel Artetxe. 2024. Do multilingual language models think better in English? In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 550–564, Mexico City, Mexico. Association for Computational Linguistics.

Akash Ghosh, Debayan Datta, Sriparna Saha, and Chirag Agarwal. 2025. A survey of multilingual reasoning in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 8920–8936, Suzhou, China. Association for Computational Linguistics.

Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. ROSCOE: A suite of metrics for scoring step-by-step reasoning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Haoyang Huang, Tianyi Tang, Dongdong Zhang, Xin Zhao, Ting Song, Yan Xia, and Furu Wei. 2023. Not all languages are created equal in LLMs: Improving multilingual capability by cross-lingual-thought prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12365–12394, Singapore. Association for Computational Linguistics.

Zixian Huang, Wenhao Zhu, Gong Cheng, Lei Li, and Fei Yuan. 2024. Mindmerger: Efficiently boosting LLM reasoning in non-english languages. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online. Association for Computational Linguistics.

Deokhyung Kang, Seonjeong Hwang, Daehui Kim, Hyounghun Kim, and Gary Geunbae Lee. 2026. Why do multilingual reasoning gaps emerge in reasoning language models? *Preprint*, arXiv:2510.27269.

[6] `https://chatgpt.com/`

Dayeon Ki, Kevin Duh, and Marine Carpuat. 2026. What makes good multilingual reasoning? disentangling reasoning traces with measurable features. *Preprint*, arXiv:2604.04720.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Hynek Kydlíček. 2025. Math-Verify: Math verification library. Software library.

Huiyuan Lai and Malvina Nissim. 2024. mCoT: Multilingual instruction tuning for reasoning consistency in language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12012–12026, Bangkok, Thailand. Association for Computational Linguistics.

Jinu Lee and Julia Hockenmaier. 2025. Evaluating step-by-step reasoning traces: A survey. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1789–1814, Suzhou, China. Association for Computational Linguistics.

Jinu Lee, Sagnik Mukherjee, Dilek Hakkani-Tur, and Julia Hockenmaier. 2025. Reasoningflow: Semantic structure of complex reasoning traces. *Preprint*, arXiv:2506.02532.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Chaoqun Liu, Wenxuan Zhang, Yiran Zhao, Anh Tuan Luu, and Lidong Bing. 2025. Is translation all you need? a study on solving multilingual tasks with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9594–9614, Albuquerque, New Mexico. Association for Computational Linguistics.

Yihong Liu, Raoyuan Zhao, Michael A. Hedderich, and Hinrich Schütze. 2026a. Crosslingual on-policy self-distillation for multilingual reasoning. *Preprint*, arXiv:2605.09548.

Yihong Liu, Raoyuan Zhao, Hinrich Schütze, and Michael A. Hedderich. 2026b. Large reasoning models are (not yet) multilingual latent reasoners. *Preprint*, arXiv:2601.02996.

Wenyang Luo, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2025. MMATH: A multilingual benchmark for mathematical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11187–11202, Suzhou, China. Association for Computational Linguistics.

Hellina Hailu Nigatu, Atnafu Lambebo Tonja, Benjamin Rosman, Thamar Solorio, and Monojit Choudhury. 2024. The Zeno's paradox of 'low-resource' languages. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17753–17774, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI. 2026. Introducing GPT-5.4. OpenAI release note. `https://openai.com/index/introducing-gpt-5-4/`. Published March 5, 2026.

Debjit Paul, Robert West, Antoine Bosselut, and Boi Faltings. 2024. Making reasoning matter: Measuring and improving faithfulness of chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 15012–15032, Miami, Florida, USA. Association for Computational Linguistics.

Jirui Qi, Shan Chen, Zidi Xiong, Raquel Fernández, Danielle Bitterman, and Arianna Bisazza. 2025. When models reason in your language: Controlling thinking language comes at the cost of accuracy. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 20279–20296, Suzhou, China. Association for Computational Linguistics.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2695–2709, Singapore. Association for Computational Linguistics.

Qwen Team. 2026. Qwen3.6-Plus: Towards real world agents. Qwen Blog. `https://qwen.ai/blog?id=qwen3.6`. Published April 2, 2026.

Leonardo Ranaldi and Giulia Pucci. 2025. Multilingual reasoning via self-training. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11566–11582, Albuquerque, New Mexico. Association for Computational Linguistics.

Shuaijie She, Wei Zou, Shujian Huang, Wenhao Zhu, Xiang Liu, Xiang Geng, and Jiajun Chen. 2024. MAPO: Advancing multilingual reasoning through multilingual-alignment-as-preference optimization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10015–10027, Bangkok, Thailand. Association for Computational Linguistics.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das,

and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Zhi Rui Tam, Cheng-Kuang Wu, Yu Ying Chiu, Chieh-Yen Lin, Yun-Nung Chen, and Hung yi Lee. 2025. Language matters: How do multilingual input and reasoning paths affect large reasoning models? *Preprint*, arXiv:2505.17407.

Mingyang Wang, Lukas Lange, Heike Adel, Yunpu Ma, Jannik Strötgen, and Hinrich Schuetze. 2025a. Language mixing in reasoning language models: Patterns, impact, and internal causes. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 2637–2665, Suzhou, China. Association for Computational Linguistics.

Weixuan Wang, Minghao Wu, Barry Haddow, and Alexandra Birch. 2025b. Demystifying multilingual reasoning in process reward modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 9775–9788, Suzhou, China. Association for Computational Linguistics.

Yiming Wang, Pei Zhang, Jialong Tang, Haoran Wei, Baosong Yang, Rui Wang, Chenshu Sun, Feitong Sun, Jiran Zhang, Junxuan Wu, Qiqian Cang, Yichang Zhang, Fei Huang, Junyang Lin, Fei Huang, and Jingren Zhou. 2025c. Polymath: Evaluating mathematical reasoning in multilingual contexts. *Preprint*, arXiv:2504.18428.

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 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zheng-Xin Yong, M. Farid Adilazuarda, Jonibek Mansurov, Ruochen Zhang, Niklas Muennighoff, Carsten Eickhoff, Genta Indra Winata, Julia Kreutzer, Stephen H. Bach, and Alham Fikri Aji. 2025. Crosslingual reasoning through test-time scaling. *Preprint*, arXiv:2505.05408.

Xue Zhang, Yunlong Liang, Fandong Meng, Songming Zhang, Kaiyu Huang, Yufeng Chen, Jinan Xu, and Jie Zhou. 2026. Think natively: Unlocking multilingual reasoning with consistency-enhanced reinforcement learning. *Preprint*, arXiv:2510.07300.

Raoyuan Zhao, Yihong Liu, Hinrich Schuetze, and Michael A. Hedderich. 2026. A comprehensive evaluation of multilingual chain-of-thought reasoning: Performance, consistency, and faithfulness across languages. In *Findings of the Association for Computational Linguistics: EACL 2026*, pages 5223–5247, Rabat, Morocco. Association for Computational Linguistics.

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

Wenhao Zhu, Shujian Huang, Fei Yuan, Shuaijie She, Jiajun Chen, and Alexandra Birch. 2024. Question translation training for better multilingual reasoning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8411–8423, Bangkok, Thailand. Association for Computational Linguistics.

A Prompt Templates and Language Control

A.1 Direction Probing Prompts

We evaluate four input–reasoning language settings by independently varying the problem language and the requested reasoning-trace language. Here, *reasoning trace* refers to the model's generated CoT after the opening thinking tag. Each query consists of three parts: a language-specific system message, the problem statement as the user message, and a language-specific assistant-side direct-first prefix.

Part	Content
System	Language-specific solver instruction in the requested reasoning language.
User	Problem statement in English or the target language, depending on the direction.
Assistant	Direct-first prefix that starts the reasoning trace in the requested reasoning language.

Setting	Problem statement	Prompt language
$en \rightarrow en$	English	English
$en \rightarrow x$	English	Target language x
$x \rightarrow en$	Target language x	English
$x \rightarrow x$	Target language x	Target language x

In all four settings, the system message and assistant prefix use the requested reasoning-trace language, while the user message supplies the problem in the selected input language. This keeps input language and reasoning trace language separable without changing the rest of the query format.

A.2 Language-Specific Prompts

Figure 5 lists the exact multilingual system prompt templates and assistant-side direct-first prefixes used in the direction probes. The prompt tables are rendered as vector PDFs to avoid Unicode font issues under `pdflatex`.

A.3 Reasoning Language Compliance

We report reasoning language consistency using the PolyMath-compatible language consistency protocol. We evaluate only the generated CoT/reasoning trace under $en \rightarrow x$, not the final answer. For each trace, we remove mathematical spans from `think_trace`, including display and inline math delimited by `$$...$$`, `\(...\)`, `\[...\]`, and `$...$`. We then apply `langdetect.detect_langs`. A trace is counted as language-consistent when the detector returns a single language matching the requested reasoning language. Following PolyMath, Chinese is mapped to `zh-cn`. Because `langdetect` systematically maps Malay text to Indonesian, we prespecify a Malay–Indonesian alias: `ms` accepts {`ms`, `id`}, and `id` accepts {`id`, `ms`}. Language consistency is reported separately from mathematical accuracy and is not used to mark an answer incorrect. Each cell in Table 4 is computed from 125 samples.

B Decoding and Generation Details

B.1 Target Trace Generation and Decoding

For the tested-model traces used in direction probing and DATG diagnosis, we use a difficulty-specific maximum generation budget:

$$B_{\text{low}} = 4096,$$
$$B_{\text{med}} = 8192,$$
$$B_{\text{high}} = 16384.$$

Across languages, directions, and difficulty levels, reasoning continuations use the same decoding configuration: temperature 0.6, top-$p = 0.95$, and the corresponding maximum budget B. This keeps the direction comparisons focused on input and reasoning language rather than on decoding variation.

C DATG Construction and Alignment Prompts

C.1 Node and Edge Semantics

A DATG node is a trace-supported, language-agnostic mathematical state that can be audited against a reasoning trace. Nodes are not free-form sentences: their `anchor` fields use equations, inequalities, assignments, set relations, or controlled predicates. The `description` field is a short readability gloss; it is not used as a string-matching target or as an independent scoring field.

- **Given or fact relation**: an immutable quantity, condition, rate, or target relation from the problem or trace that is needed to audit later steps; e.g., $eggs_day = 16$.
- **Equation or setup relation**: an algebraic relation that defines how quantities are connected before or during computation; e.g., $remaining = eggs_day - eaten - muffins$.
- **Arithmetic transformation**: an explicit computation or simplification that changes one expression into an evaluated form; e.g., $16 - 3 - 4 = 9$.
- **Derived intermediate result**: a computed value or symbolic state used by downstream reasoning; e.g., $remaining = 9$.
- **Constraint, set, or predicate relation**: a checkable condition such as divisibility, membership, inequality, recurrence, or feasibility; e.g., $\mathrm{Div}(56) \cap \{d : d > 16\} = \{28, 56\}$.
- **Case, bound, or witness state**: a trace-stated case condition, extremal bound, candidate construction, or witness evaluation; e.g., $x_{76} - x_{16} \leq 41/800$.
- **Contradiction or feasibility conclusion**: a controlled mathematical conclusion that a branch is impossible or satisfies a stated condition; e.g., $bad_coloring(k) \rightarrow contradiction$.
- **Final answer-equivalent state**: the terminal mathematical state in the requested answer form; e.g., $m + n = 841$.

We do not create nodes for boxed-answer markers, output-format artifacts, standalone constants, or purely verbal restatements that do not define a checkable mathematical state. A DAG edge is defined by a node's `parents`: each parent is a prerequisite mathematical state for the child. These parent-defined edges are the dependency relations used by PMF.

C.2 DAG Construction Prompt

For each problem, the reference set contains five English reference derivations generated by `Qwen3.6-Plus`; each reference derivation has a final answer that agrees with the accepted answer under our answer-verification protocol. We then

Lang.	Solver system prompt template
en	Please always think in {language}. Finally, put your final answer in \boxed{}.
fr	Veuillez toujours réfléchir en {language}. Enfin, mettez votre réponse finale dans \boxed{}.
ru	Пожалуйста, всегда думайте на {language}. В конце поместите окончательный ответ в \boxed{}.
zh	请始终用{language}思考。最后，将您的最终答案放在\boxed{}中。
ja	必ず{language}で考えてください。最後に、最終的な答えを\boxed{}に入れてください。
th	โปรดคิดเป็น{language}เสมอ สุดท้ายใส่คำตอบสุดท้ายของคุณไว้ใน \boxed{}
sw	Tafadhali fikiri kwa {language} kila wakati. Mwishowe, weka jibu lako la mwisho katika \boxed{}.
bn	অনুগ্রহ করে সর্বদা {language}-এ চিন্তা করুন। শেষে, আপনার চূড়ান্ত উত্তরটি \boxed{} এর মধ্যে দিন।
te	దయచేసి ఎల్లప్పుడూ {language}లో ఆలోచించండి. చివరగా, మీ తుది జవాబును \boxed{} లో ఉంచండి.
ko	항상 {language}로 생각하세요. 마지막으로 최종 답을 \boxed{} 안에 넣으세요.
id	Harap selalu berpikir dalam {language}. Terakhir, letakkan jawaban akhir Anda dalam \boxed{}.
ms	Sila sentiasa berfikir dalam {language}. Akhir sekali, letakkan jawapan akhir anda dalam \boxed{}.

Lang.	Assistant-side direct-first prefix
en	<think> I will start thinking in English. First,
fr	<think> Je vais commencer à réfléchir en français. D'abord,
ru	<think> Я начну думать по-русски. Во-первых,
zh	<think> 我将开始用中文思考。 首先，
ja	<think> 日本語で思考を始めます。 まず，
th	<think> ฉันจะเริ่มคิดเป็นภาษาไทย ก่อนอื่น
sw	<think> Nitaanza kufikiri kwa Kiswahili. Kwanza,
bn	<think> আমি বাংলায় চিন্তা শুরু করব। প্রথমে,
te	<think> నేను తెలుగులో ఆలోచించడం ప్రారంభిస్తాను. ముందుగా,
ko	<think> 한국어로 생각을 시작하겠습니다. 먼저,
id	<think> Saya akan mulai berpikir dalam bahasa Indonesia. Pertama,
ms	<think> Saya akan mula berfikir dalam bahasa Melayu. Pertama,

Figure 5: Language-specific solver system prompts and assistant-side direct-first prefixes.

Pair	1.7B			4B			8B		
	Low	Med.	High	Low	Med.	High	Low	Med.	High
en-fr	98.4	93.6	95.2	99.2	100.0	99.2	89.6	75.2	79.2
en-ru	97.6	99.2	98.4	100.0	99.2	100.0	99.2	99.2	98.4
en-zh	84.8	84.0	92.0	85.6	84.0	95.2	85.6	82.4	96.8
en-ja	97.6	99.2	98.4	100.0	97.6	100.0	98.4	97.6	98.4
en-th	100.0	100.0	98.4	100.0	100.0	99.2	99.2	98.4	98.4
en-sw	93.6	94.4	93.6	96.0	96.8	96.8	96.8	84.8	92.8
en-bn	98.4	99.2	100.0	100.0	98.4	99.2	100.0	99.2	99.2
en-te	98.4	100.0	97.6	100.0	100.0	100.0	100.0	100.0	99.2
en-ko	92.0	85.6	86.4	96.8	84.0	91.2	93.6	81.6	84.0
en-id	98.4	98.4	99.2	91.2	93.6	92.8	95.2	78.4	72.8
en-ms	99.2	96.8	98.4	93.6	78.4	79.2	96.0	61.6	56.8

Table 4: Reasoning language compliance under $en \rightarrow x$. Values are percentages; each cell contains 125 traces.

use a fixed construction prompt to convert each reference derivation into a trace-grounded DAG of language-agnostic mathematical anchors.

DAG construction prompt

```
System message:
You are a logic-faithful mathematical trace extractor.
Your task is to convert a verified Chain-of-Thought (CoT)
into a Directed Acyclic Graph (DAG) of language-agnostic
mathematical reasoning anchors.

CORE PRINCIPLE (TRACE-GROUNDED, MINIMAL-SUFFICIENT
REASONING GRAPH):
The DAG should represent the core reasoning steps plus the
minimal set of trace-supported input facts needed to make
those steps auditable. Each node must be anchored to a
language-neutral mathematical logic state supported by the
trace. Prefer computation anchors such as equations,
arithmetic transforms, derived intermediate results, and
final numeric conclusions that already appear in the trace.
However, do not delete a trace-stated quantity, rate,
condition, or target relation if a later calculation depends
on it and the dependency would become unclear without that
fact node. Do not invent a new symbolic system, solve the
problem again, or replace the trace with a cleaner
derivation unless the trace itself uses it.

NODE SCHEMA:
- node_id: n1, n2, ...
- anchor: one compact symbolic or controlled mathematical
  anchor, not a natural-language sentence
- description: a very short English gloss for readability only
- parents: prerequisite node_ids

Anchor forms are descriptive and non-mutually-exclusive. An
anchor may be a given fact, equation, arithmetic
transformation, derived intermediate result, necessary
relation, or answer-equivalent conclusion. Do not output an
anchor_type field.
```

SYMBOLIC ANCHOR RULE:
The anchor field must be language-independent in serialization: use equations, inequalities, set relations, assignments, or controlled predicates. Use compact variable names such as eggs_day, remaining, price, S_+, D_i(k), or target. Put all English prose in description. Do not write natural English clauses in anchor such as "The question asks...", "The condition means...", "there are...", or "we get...".

Examples of acceptable anchors:
eggs_day = 16
remaining = eggs_day - eaten - muffins
16 - 3 - 4 = 9
Div(56) cap {d:d>16} = {28,56}
max(x_{76}-x_{16}) = 41/800
m+n = 841
bad_coloring(k) -> contradiction

Examples of unacceptable anchors:
Janet's ducks lay 16 eggs per day.
The divisors of 56 greater than 16 are 28 and 56.
The question asks how far he is from home.
The greatest value that x_76-x_16 can achieve is 41/800.

TRACE-FAITHFUL EXTREMAL CLAIMS:
For maximum, minimum, supremum, infimum, best/worst, or sharp-bound problems, the goal is faithful extraction rather than proof completion. Preserve the trace's explicit bound, construction, witness, candidate value, limiting argument, or trace-stated final conclusion as it appears. Do not invent missing inequalities, missing constructions, missing optimality arguments, or a cleaner proof. If the trace states an extremal conclusion without spelling out every proof component, keep it as a trace-stated or asserted conclusion rather than rewriting it as a fully proven theorem.

FINAL NODE RULE:
The final_node_id must point to the trace's terminal answer-equivalent mathematical state. Candidate values, witness evaluations, constructions, and one-sided bounds should normally be support nodes, not the final node, when the trace later states a final answer or asserted conclusion. For incomplete-but-correct traces, the terminal node may be an asserted conclusion or trace-stated answer, but it must correspond to the trace's final answer state rather than an earlier partial result.

FINAL RESPONSE USE POLICY:
You may consult the final response only as a boundary cue for the terminal answer-equivalent state. Do not use the final response to invent missing intermediate reasoning or to replace the trace's derivation. The DAG must still be extracted from the reasoning trace, with the final response used only to disambiguate the terminal answer when the reasoning trace is clipped or contains multiple local candidates.

MULTIPLE-PATH AND ENDPOINT-COUNT RULES:
If the trace contains two equivalent solution strategies, extract one settled minimal path and drop alternate checks that do not support the selected final node. If the trace first derives a core count or intermediate expression and later adds an endpoint, boundary case, or requested-form conversion, keep the core count as support and make the final node the later answer-equivalent conclusion.

FALSE STARTS AND ANSWER TARGETS:
Reasoning traces may contain false starts, abandoned branches, exploratory guesses, or statements later corrected by phrases such as "wait", "error", "actually", "this failed", "not optimal", or "final check". Do not use an abandoned intermediate conclusion as the final node. Follow the trace's settled conclusion near the end. If the trace converts an intermediate result into the requested answer form, such as m+n, a residue, a count, a maximum value, or a simplified expression, the final node must include that requested answer-equivalent value.

RULES:
1) Graph must be acyclic.
2) Keep formulas as close as possible to the trace wording and arithmetic.
3) Do not rename variables unless the trace itself defines those variables.
4) If the trace uses plain arithmetic such as 16 - 7 = 9, keep that plain arithmetic instead of inventing notation like E_total.
5) Prefer computation anchors over verbal restatements of givens, but keep key immutable facts when they are needed to audit a later calculation.
6) Keep a given/fact node when it is a necessary dependency for later calculations and no downstream equation fully subsumes the same information with explicit numbers, units, and relation.
7) If the trace contains a combined equation such as 2 + 1 = 3, prefer that combined equation over splitting it into separate result-only nodes.
8) Do not create nodes for boxed answers, final-answer markers, think tags, or other output-format artifacts.
9) Do not create standalone constant-only nodes unless they are unavoidable as trace-supported arithmetic anchors.
10) If a given/fact node must be kept, express it as a full relation from the trace rather than a bare literal.
11) If the trace includes both a setup form and a fully evaluated form for the same step, prefer the single more informative evaluated anchor unless the setup is needed as a separate dependency.
12) Preserve meaningful reasoning-path diversity across different traces, but within one DAG suppress purely notational, stylistic, formatting, verification, or alternate-solution branches.
13) Preserve the trace's logical granularity. Do not split or merge steps unless needed for DAG validity.
14) final_node_id must point to the final mathematical answer state or the last explicit answer-equivalent conclusion in the trace.
15) Follow the trace's final settled conclusion and ignore abandoned false-start conclusions.
16) For requested forms such as m+n, residues, counts, or simplified final expressions, make the final node contain that requested answer-equivalent form.
17) For extremal or sharp-bound conclusions, keep the trace's own proof status explicit in description, not as prose in anchor.
18) Do not hide uncertainty in the trace. If the trace contains a plausible but compressed conclusion, preserve it as a trace-stated conclusion; do not add a missing proof.
19) Output JSON only.

INTERNAL VALIDITY CHECK BEFORE OUTPUT:
- Emit nodes in dependency/topological order.
- Every retained node must support final_node_id directly or indirectly.
- final_node_id must be a terminal sink with no outgoing dependencies.
- If the trace evaluates the final expression, include the evaluated final value in the final anchor.
- Do this check silently; output only the final JSON object.

User message:

[Reasoning Trace to Parse]
<<<
{verified_reference_cot}
>>>

[Settled Trace Tail - final-node priority]
<<<TAIL
{reasoning_trace_tail}
TAIL>>>

[Final Response - answer boundary only]
<<<FINAL_RESPONSE
{final_response}
FINAL_RESPONSE>>>

[Task]
Extract one trace-faithful mathematical DAG and return exactly one JSON object:

Output strict JSON only:
```
{
  "final_node_id": "nK",
  "nodes": [
    {
      "node_id": "n1",
      "anchor": "symbolic_or_controlled_math_state",
      "description": "short English gloss",
      "parents": []
    }
  ]
```

```
}

Hard constraints:
1. JSON only.
2. Do NOT output question_id or sample_id.
3. parents must reference only earlier nodes.
4. DAG must be acyclic and nodes must be emitted in
   dependency/topological order.
5. The DAG should capture a minimal-sufficient reasoning
   graph.
6. Each node must correspond to a mathematical logic anchor
   explicitly supported by the trace.
7. anchor must be symbolic or a controlled mathematical
   predicate, not an English sentence.
8. English prose is allowed only in description.
9. Reuse original formulas, arithmetic, and explicit numeric
   conclusions whenever possible.
10. If the trace states a fact in prose, normalize it into a
    compact relation.
11. Avoid paraphrasing formulas into newly invented symbolic
    notation.
12. Prefer computation anchors over verbal given/fact
    restatements.
13. Keep a given/fact node when it is needed as an input
    dependency.
14. If a given/fact node is kept, write it as a symbolic
    relation.
15. Prefer combined equations over split result-only nodes.
16. Prefer the single more informative evaluated anchor when
    possible.
17. Keep description very short.
18. Do NOT create nodes for \boxed{...}, Final Answer, think
    tags, or formatting artifacts.
19. Do NOT create standalone constant-only nodes unless
    unavoidable.
20. Preserve meaningful path differences across traces.
21. Ignore abandoned false starts and corrected conclusions.
22. final_node_id must point to a terminal sink node.
23. Requested answer forms such as m+n, residue, count, or
    simplified expression must be final.
24. Every node must support the final answer-equivalent node
    directly or indirectly.
25. If the trace evaluates the final expression, the final
    anchor must contain the evaluated final value.
26. If you add an evaluated final-answer node, update
    final_node_id to that node.
27. For extremal traces, do not require proof completion.
28. If the trace only evaluates a candidate or one side of a
    bound, preserve that limitation.
29. Faithfulness to the trace is more important than elegance
    or proof completeness.
30. Do not output anchor_type.
31. Before output, silently verify parent-first order, no
    cycles, all retained nodes reach final_node_id, and
    final_node_id is a terminal sink.

Budget guidance:
- target_max_nodes: {target_max_nodes}
- target_max_desc_chars_per_node:
     {target_max_desc_chars_per_node}
- target_max_pre_nodes_per_node:
     {target_max_pre_nodes_per_node}
```

C.3 Closed-Set Alignment Prompt

For each target trace, the aligner maps trace evidence to a fixed reference DAG in a closed set. It can only assign evidence to existing nodes and cannot create new mathematical anchors.

Closed-set alignment prompt

```
System message:
You are a closed-set mathematical reasoning extractor for
three metrics only: CAR (Committed anchor recall), PMF (path
monotonic fidelity), and HAR (harmful action rate).

Your job is extraction only, not scoring.

Main principles:
- Work in a closed set for node alignment: only map to the
  provided DAG nodes.
- Judge mathematical equivalence, not prose similarity.
- Symbol names may differ. Align by mathematical role,
  object identity, dependencies, and value, not by exact
  variable names.
- For invariant scalar quantities such as lengths, counts,
  ratios, divisibility conditions, and final arithmetic,
  COMMIT evidence must establish the same value or an
  algebraically identical form.
- For coordinate or auxiliary representation nodes,
  different coordinate conventions are allowed when the trace
  is internally consistent.
- Do not align nearby but different objects, auxiliary
  quantities, or generic intermediate equations to a DAG node
  merely because they occur in the same solution region.
- Return only the earliest useful events for each node.
- A node becomes activated only at its earliest valid COMMIT.
- If an early occurrence is partial or wrong, mark it ATTEMPT
  or ERROR.
- Do not use a later polished recap, final-answer summary, or
  post-hoc restatement as COMMIT evidence if an earlier
  equivalent computation already establishes the node.
- If a downstream computation correctly uses an intermediate
  value required by a parent node, that same local evidence
  may serve as the parent's COMMIT when no earlier explicit
  statement is available.
- Do not compute totals, rates, correctness labels, or final
  metric values.
- Do not segment the trace. Do not invent positions.

Allowed node event statuses:
- COMMIT: correctly establishes the DAG node.
- ATTEMPT: partially expressed or intended but not sufficient.
- ERROR: mathematically wrong derivation of that DAG node.

Strict COMMIT rule:
- The evidence must establish the node's anchor, not merely a
  related step in the same solution.
- If the evidence uses a different symbol or representation,
  require enough local context to identify the same
  mathematical role or object.
- A bare mention of a number, variable, expression, or example
  substitution is not COMMIT.
- Verbal mathematical statements in any language can be COMMIT
  when the same quantity/state/update is locally checkable.
- If the trace reaches the right final answer through an
  alternative path not represented by the DAG, leave unmatched
  DAG nodes empty rather than forcing a match.

For node events, provide only:
- evidence: exact short quote copied from the trace, <= 80
  characters.
- evidence_span: exact continuous quote copied from the
  trace, <= 120 characters.

For HAR, keep only local harmful mathematical actions:
- contradictory_steps: harmful contradictions, wrong
  dependency use, or wrong DAG-node attempts that remain
  active in the final reasoning state.
- harmful_loop_steps: harmful repeated math acts that revisit
  an established state without adding justified state.
- degenerate_steps: one short repeated/garbled pseudo-math
  unit if the trace collapses.

Do not output off-path branches, harmless redundancy lists,
verification steps, or benign restatements.
Do not copy long repeated paragraphs.
Each HAR list may contain at most 2 items, and
degenerate_steps at most 1 item.
If an ERROR node event can change the active reasoning state
or final answer, also include its same short evidence in the
appropriate HAR list.
Do not include a wrong intermediate branch in HAR if the
trace explicitly marks it as a mistake and replaces it before
using it for the final answer.
If uncertain, leave the node/list empty. Do not guess.
Output strict JSON only.

User message:
Target trace language: {target_lang}

[Reference Mathematical DAG]
{dag_json}

[Target Reasoning Trace]
<<<TRACE
{raw_trace}
TRACE>>>
```

```
Task:
Extract only what is needed for CAR, PMF, and HAR.
For each DAG node, return only the earliest useful
mathematically supported occurrences in chronological order.
This is a closed-set alignment task: only align to the
provided node inventory. Do not invent nodes.
A node is activated only by its earliest valid COMMIT. Later
re-checks or restatements are irrelevant and must not be
returned. Use ATTEMPT for partial or incomplete derivations,
ERROR for mathematically wrong derivations, and COMMIT only
for a correct and sufficient establishment of the node.
Variable names may differ; align by mathematical role, object
identity, dependencies, and value, not exact names.
If the trace reaches the right final answer through an
alternative path not represented by this DAG, leave unmatched
DAG nodes empty rather than forcing a match.
For HAR, return only local harmful mathematical actions in
contradictory_steps, harmful_loop_steps, or degenerate_steps.
If uncertain, leave the node empty and do not guess.
Do not estimate total_math_steps, effective_math_acts, or any
score field. Extraction only.

Return strict JSON only:
{
  "audit_results": {
    "n1":[{"status":"COMMIT","evidence":"...",
           "evidence_span":"..."}],
    "n4":[{"status":"ATTEMPT","evidence":"...",
           "evidence_span":"..."},
          {"status":"COMMIT","evidence":"...",
           "evidence_span":"..."}],
    "n5":[]
  },
  "contradictory_steps": [
    {"evidence":"...","evidence_span":"...",
     "category":"contradiction"}
  ],
  "harmful_loop_steps": [
    {"evidence":"...","evidence_span":"...",
     "category":"loop"}
  ],
  "degenerate_steps": [
    {"evidence":"...","category":"collapse"}
  ]
}
```

An empty list for a node in `audit_results` represents a missing anchor; the deterministic scorer maps such nodes to MISSING. Thus Missing is a scoring status rather than an aligner-emitted label.

For each target trace, we compute CAR, PMF, and HAR against all five candidate reference DAGs. We select the most compatible alignment by maximizing CAR, then PMF, and finally minimizing HAR. The LLM aligner performs closed-set semantic alignment against the provided node inventory, using the `anchor` as the primary mathematical target and `description` only as a readability gloss; it is not asked to compute metric values or to string-match descriptions. The aligner returns status labels and `evidence_spans`. The scoring code then locates each span in the original trace to obtain `start_char`, `end_char`, `commit_progress`, and `block_id`. PMF is computed from the parent-defined DAG edges: for each parent-to-child edge, the edge is ordered only if both endpoint nodes are COMMIT and the parent commit progress is no later than the child commit progress, or both commits fall in the same trace block. The final PMF score is the number of ordered edges divided by the total number of DAG edges.

D Reference-Side Diversity

To quantify variation among the five answer-verified English references for the same problem, we perform reference-to-reference alignment. For each English reference trace, we align it against the other four English reference DAGs for the same problem, exclude its own DAG, and select the most compatible "best-other" alignment by maximizing CAR, then PMF, and finally minimizing HAR. This measures whether different English references provide alternative derivation paths while preserving compatible core mathematical structure.

Difficulty	Best-other CAR ↑	Best-other PMF ↑	Best-other HAR ↓
Low	1.000 ± 0.000	0.995 ± 0.035	0.006 ± 0.040
Medium	0.989 ± 0.035	0.926 ± 0.116	0.000 ± 0.000
High	0.949 ± 0.125	0.836 ± 0.197	0.002 ± 0.013
Overall	0.979 ± 0.078	0.919 ± 0.148	0.003 ± 0.024

Table 5: Reference-side diversity among answer-verified English reference derivations. Values are mean $\pm$ standard deviation over best-other reference-to-reference alignments.

E Human Evaluation of DATG

We use human evaluation as a diagnostic reliability check for DATG, not as a benchmark for formal proof verification. Two annotators independently audited reference DAG quality, closed-set trace alignment, and HAR candidates. The expanded audit set covers French, Swahili, Bengali, and Telugu, with 60 samples per language. For each language, we sample 20 instances from each difficulty level. These languages include both relatively stable multilingual settings and the lower-resource settings where the automatic DATG analysis shows the largest degradation.

Inter-annotator agreement is computed at the annotation layer rather than only for the final aggregate scores. We report MAE and correlation for CAR/PMF because these are continuous scores derived from committed-node alignments rather than categorical labels.

Table 6 provides direct support for DATG as an aggregate, trace-grounded diagnostic representation. The reference DAGs have consistently high node and edge validity across all four languages, including Bengali and Telugu, indicating that the extracted reference structures are mathematically well-formed enough for aggregate diagnostic use.

The CAR and PMF columns show that the automatic scores track human-audited alignments closely, with low MAE and strong Pearson/Spearman correlations in the aggregate. This is the key calibration result for our use case: DATG is not claimed to certify formal proof validity, but its closed-set anchor alignment is reliable enough to support comparisons of coverage and dependency order across languages, difficulties, and model scales. For HAR, the automatic extractor produces 162 harmful-action candidates, of which 146 are confirmed by human annotators, yielding a precision of 90.1%. This supports HAR as an aggregate signal for harmful mathematical actions.

DATG should therefore be read as reference-conditioned, trace-grounded diagnosis rather than formal proof verification. It does not define a formal proof language, proof obligations, or mechanically checkable inference rules. PMF checks whether committed anchors appear in an order compatible with a reference DAG; it does not prove that every child node is logically entailed by its parents. The reference DAGs are LLM-extracted from reference derivations, so they provide diagnostic structure, not sound proof certificates.

Lang.	n	DAG		CAR		PMF		HAR
		Node	Edge	MAE	r	MAE	r	
fr	60	98.7	96.1	.079	.886/.871	.118	.846/.857	65.4
sw	60	97.7	95.3	.034	.965/.935	.014	.985/.904	100.0
bn	60	96.8	98.1	.046	.967/.943	.043	.970/.966	85.0
te	60	96.8	96.4	.036	.969/.918	.015	.985/.954	97.7
All	240	97.5	96.5	.049	.963/.955	.048	.955/.954	90.1

Agreement	Node	Edge	Node-event	HAR
Pooled Cohen's κ	.753	.770	.938	.798

Table 6: Human evaluation of DATG. DAG validity is in percentages; CAR/PMF report MAE and Pearson/Spearman correlation against human-audited alignments. HAR precision is over 162 candidates, with all 240 double-annotated instances in the aggregate row.

F Full Direction-Probing Accuracy

Difficulty	Direction	en	fr	ru	zh	ja	ko	id	ms	th	bn	sw	te
1.7B													
low	en-en	81.60	–	–	–	–	–	–	–	–	–	–	–
low	en-x	–	73.60	75.20	77.60	67.20	74.40	71.20	58.40	72.00	44.80	27.20	7.20
low	x-en	–	72.80	78.40	71.20	63.20	64.00	73.60	75.20	64.80	55.20	8.80	40.80
low	x-x	–	63.20	68.80	74.40	59.20	60.80	67.20	61.60	64.80	34.40	4.00	13.60
medium	en-en	36.80	–	–	–	–	–	–	–	–	–	–	–
medium	en-x	–	31.20	32.00	36.80	18.40	17.60	24.80	16.80	22.40	5.60	1.60	0.80
medium	x-en	–	38.40	39.20	40.80	36.00	28.80	37.60	35.20	31.20	30.40	21.60	24.80
medium	x-x	–	24.00	32.80	36.80	25.60	14.40	25.60	26.40	14.40	7.20	1.60	0.00
high	en-en	23.20	–	–	–	–	–	–	–	–	–	–	–
high	en-x	–	11.20	8.80	19.20	10.40	5.60	9.60	5.60	8.00	0.00	3.20	0.80
high	x-en	–	17.60	20.00	25.60	23.20	23.20	16.80	20.00	16.00	17.60	12.00	15.20
high	x-x	–	13.60	13.60	20.00	6.40	8.00	8.80	10.40	4.80	1.60	0.80	0.00
4B													
low	en-en	92.80	–	–	–	–	–	–	–	–	–	–	–
low	en-x	–	88.80	89.60	91.20	91.20	89.60	92.00	92.00	90.40	84.00	52.80	72.00
low	x-en	–	87.20	91.20	85.60	78.40	87.20	84.80	87.20	80.80	79.20	24.00	68.00
low	x-x	–	83.20	88.80	84.80	76.80	80.80	86.40	85.60	78.40	76.00	24.00	49.60
medium	en-en	53.60	–	–	–	–	–	–	–	–	–	–	–
medium	en-x	–	43.20	51.20	49.60	41.60	36.80	47.20	44.80	35.20	16.00	0.00	4.00
medium	x-en	–	36.80	36.80	31.20	36.00	34.40	40.80	40.80	39.20	32.80	25.60	32.80
medium	x-x	–	32.80	32.00	24.80	28.80	31.20	33.60	31.20	29.60	16.00	2.40	2.40
high	en-en	41.60	–	–	–	–	–	–	–	–	–	–	–
high	en-x	–	31.20	32.80	36.00	23.20	24.00	30.40	26.40	16.80	10.40	0.80	1.60
high	x-en	–	39.20	40.00	43.20	39.20	43.20	38.40	41.60	33.60	36.00	20.80	31.20
high	x-x	–	36.00	37.60	42.40	24.80	22.40	25.60	29.60	19.20	10.40	0.80	3.20
8B													
low	en-en	94.40	–	–	–	–	–	–	–	–	–	–	–
low	en-x	–	92.00	94.40	92.00	91.20	87.20	90.40	92.00	92.00	87.20	59.20	76.80
low	x-en	–	88.80	92.00	88.00	86.40	89.60	92.00	89.60	89.60	84.00	64.80	76.80
low	x-x	–	86.40	92.00	88.80	80.00	84.00	88.00	88.00	89.60	80.80	23.20	58.40
medium	en-en	53.60	–	–	–	–	–	–	–	–	–	–	–
medium	en-x	–	50.40	49.60	54.40	39.20	36.00	48.00	48.00	48.80	34.40	16.00	10.40
medium	x-en	–	34.40	34.40	36.80	33.60	48.80	52.00	49.60	28.00	41.60	35.20	48.80
medium	x-x	–	47.20	45.60	48.00	46.40	40.00	50.40	47.20	43.20	26.40	5.60	12.80
high	en-en	40.80	–	–	–	–	–	–	–	–	–	–	–
high	en-x	–	29.60	36.80	39.20	22.40	18.40	28.00	27.20	25.60	12.80	2.40	4.00
high	x-en	–	39.20	43.20	41.60	45.60	41.60	40.00	39.20	43.20	35.20	32.00	34.40
high	x-x	–	33.60	37.60	39.20	32.00	24.00	30.40	35.20	27.20	12.00	4.00	4.00

Table 7: Full direction-probing accuracy by model, difficulty, direction, and language. Values are percentages; each non-empty cell is evaluated on 125 PolyMath problems.

G Full DATG Results

		CAR			PMF			HAR		
Pair	Difficulty	1.7B	4B	8B	1.7B	4B	8B	1.7B	4B	8B
en-en	High	.469	.609	.586	.309	.464	.448	.142	.082	.072
	Medium	.668	.784	.778	.514	.675	.673	.090	.039	.042
	Low	.942	.979	.960	.897	.961	.913	.026	.008	.007
en-fr	High	.334	.524	.487	.170	.388	.350	.236	.119	.158
	Medium	.529	.737	.724	.376	.614	.600	.174	.068	.053
	Low	.899	.956	.966	.829	.930	.940	.049	.016	.016
en-zh	High	.428	.592	.542	.267	.451	.402	.173	.085	.088
	Medium	.619	.796	.791	.481	.678	.671	.113	.033	.030
	Low	.930	.969	.976	.884	.936	.953	.043	.016	.008
en-ja	High	.310	.467	.400	.163	.319	.254	.305	.202	.180
	Medium	.505	.730	.597	.335	.605	.431	.232	.086	.131
	Low	.886	.977	.958	.804	.968	.928	.082	.013	.016
en-ru	High	.372	.545	.547	.217	.396	.400	.243	.115	.110
	Medium	.597	.776	.711	.448	.655	.567	.164	.063	.078
	Low	.896	.978	.977	.812	.959	.954	.059	.011	.011
en-ko	High	.268	.446	.338	.117	.309	.200	.315	.235	.200
	Medium	.494	.696	.570	.317	.553	.405	.229	.105	.122
	Low	.882	.974	.930	.789	.949	.872	.079	.015	.033
en-id	High	.329	.523	.471	.188	.389	.339	.268	.133	.165
	Medium	.528	.759	.693	.356	.643	.572	.179	.048	.117
	Low	.889	.966	.947	.815	.943	.902	.060	.015	.020
en-ms	High	.305	.501	.445	.164	.343	.296	.267	.148	.169
	Medium	.491	.730	.672	.316	.607	.543	.167	.059	.079
	Low	.888	.972	.944	.799	.947	.910	.061	.016	.032
en-th	High	.291	.424	.421	.145	.268	.284	.301	.227	.229
	Medium	.459	.670	.665	.294	.553	.540	.266	.125	.104
	Low	.890	.962	.948	.790	.944	.912	.056	.018	.019
en-bn	High	.128	.243	.293	.034	.123	.154	.506	.331	.308
	Medium	.257	.480	.526	.108	.339	.374	.451	.245	.190
	Low	.612	.892	.941	.448	.833	.904	.256	.072	.042
en-sw	High	.075	.062	.108	.037	.009	.039	.732	.678	.617
	Medium	.088	.124	.283	.045	.039	.187	.768	.685	.504
	Low	.375	.585	.693	.270	.502	.617	.500	.341	.259
en-te	High	.048	.101	.175	.005	.028	.059	.751	.564	.425
	Medium	.078	.186	.322	.015	.055	.180	.705	.465	.349
	Low	.281	.749	.855	.149	.666	.765	.601	.183	.074

Table 8: Full DATG results by language pair, difficulty, model scale, and metric. HAR uses the local harmful mathematical action definition from the main text.

H Correctness-Stratified DATG Results

		Qwen3-1.7B				Qwen3-4B				Qwen3-8B			
Pair	Diff.	Acc.	CAR	PMF	HAR	Acc.	CAR	PMF	HAR	Acc.	CAR	PMF	HAR
en-en	High	.232	.792/.371	.635/.211	.093/.157	.416	.857/.433	.732/.272	.038/.114	.408	.849/.404	.722/.260	.052/.086
	Med.	.368	.876/.547	.767/.367	.042/.118	.536	.912/.635	.831/.495	.034/.045	.536	.944/.586	.869/.446	.013/.075
	Low	.816	.981/.769	.966/.590	.004/.126	.928	.992/.802	.984/.676	.003/.072	.944	.980/.638	.945/.371	.005/.036
en-fr	High	.112	.644/.294	.450/.135	.226/.237	.312	.813/.392	.690/.251	.079/.138	.296	.870/.326	.771/.173	.051/.203
	Med.	.312	.748/.429	.607/.271	.060/.225	.432	.885/.625	.786/.483	.028/.098	.504	.895/.550	.792/.406	.015/.093
	Low	.736	.937/.794	.890/.660	.021/.127	.888	.979/.774	.966/.640	.005/.104	.920	.989/.699	.976/.531	.005/.143
en-zh	High	.192	.755/.350	.590/.191	.060/.200	.360	.853/.445	.741/.289	.024/.120	.392	.853/.341	.723/.195	.003/.143
	Med.	.368	.864/.477	.766/.314	.045/.152	.496	.921/.672	.823/.536	.010/.055	.544	.925/.631	.840/.469	.007/.058
	Low	.776	.962/.817	.937/.700	.020/.124	.912	.979/.861	.954/.745	.016/.018	.920	1.000/.705	.998/.428	.002/.084
en-ja	High	.104	.612/.274	.442/.130	.203/.317	.232	.856/.350	.724/.197	.058/.246	.224	.765/.295	.641/.142	.105/.202
	Med.	.184	.807/.437	.647/.265	.092/.264	.416	.911/.601	.801/.466	.045/.115	.392	.842/.440	.712/.250	.029/.197
	Low	.672	.957/.742	.917/.573	.034/.180	.912	.993/.812	.989/.745	.006/.084	.912	.988/.644	.974/.461	.006/.120
en-ru	High	.088	.651/.345	.489/.191	.082/.259	.328	.774/.434	.668/.264	.035/.153	.368	.857/.366	.722/.212	.031/.155
	Med.	.320	.814/.495	.697/.331	.071/.208	.512	.917/.628	.820/.482	.038/.089	.496	.891/.533	.785/.352	.011/.144
	Low	.752	.943/.754	.902/.539	.034/.137	.896	.993/.854	.981/.772	.003/.075	.944	.997/.645	.989/.357	.003/.148
en-ko	High	.056	.514/.253	.326/.104	.330/.314	.240	.804/.333	.730/.176	.102/.277	.184	.726/.251	.590/.112	.059/.232
	Med.	.176	.747/.440	.587/.259	.110/.254	.368	.845/.610	.718/.458	.044/.141	.360	.815/.432	.681/.250	.038/.169
	Low	.744	.945/.699	.900/.467	.041/.190	.896	.990/.831	.975/.723	.004/.109	.872	.984/.561	.961/.268	.000/.259
en-id	High	.096	.706/.289	.562/.148	.114/.284	.304	.853/.379	.769/.223	.055/.167	.280	.871/.315	.765/.173	.031/.218
	Med.	.248	.758/.452	.617/.270	.102/.205	.472	.916/.618	.824/.482	.025/.068	.480	.898/.503	.814/.349	.032/.195
	Low	.712	.962/.709	.937/.514	.020/.157	.920	.994/.638	.988/.423	.005/.128	.904	.980/.630	.961/.345	.000/.211
en-ms	High	.056	.758/.278	.652/.135	.119/.276	.264	.821/.386	.690/.218	.039/.187	.272	.796/.314	.636/.169	.031/.220
	Med.	.168	.694/.450	.556/.268	.114/.177	.448	.900/.592	.805/.446	.023/.088	.480	.895/.466	.813/.293	.006/.147
	Low	.584	.962/.783	.915/.637	.034/.099	.920	.990/.755	.980/.570	.009/.095	.920	.983/.495	.969/.232	.013/.250
en-th	High	.080	.720/.253	.571/.108	.173/.312	.168	.743/.360	.620/.197	.079/.257	.256	.786/.295	.679/.148	.082/.279
	Med.	.224	.749/.375	.612/.202	.108/.312	.352	.843/.576	.741/.450	.047/.168	.488	.883/.458	.781/.311	.011/.193
	Low	.720	.961/.707	.908/.487	.021/.146	.904	.979/.810	.965/.750	.015/.048	.920	.989/.472	.975/.185	.002/.223
en-bn	High	.000	–/.128	–/.034	–/.506	.104	.620/.199	.500/.079	.090/.359	.128	.783/.221	.672/.078	.090/.340
	Med.	.056	.556/.239	.339/.095	.315/.459	.160	.790/.421	.690/.272	.133/.266	.344	.840/.361	.739/.183	.035/.271
	Low	.448	.690/.549	.562/.356	.202/.300	.840	.948/.597	.909/.436	.039/.243	.872	.995/.571	.989/.330	.000/.328
en-sw	High	.032	.183/.071	.125/.034	.750/.732	.008	.000/.063	.000/.009	1.000/.675	.024	.639/.095	.611/.025	.000/.632
	Med.	.016	.000/.090	.000/.046	.750/.768	.000	–/.124	–/.039	–/.685	.160	.819/.181	.746/.080	.129/.575
	Low	.272	.650/.273	.530/.173	.258/.590	.528	.717/.438	.652/.333	.213/.485	.592	.954/.313	.933/.158	.061/.547
en-te	High	.008	.000/.049	.000/.005	.000/.757	.016	.312/.098	.250/.024	.583/.564	.040	.655/.155	.600/.036	.233/.433
	Med.	.008	.000/.079	.000/.015	1.000/.702	.040	.351/.179	.069/.054	.267/.474	.104	.798/.267	.709/.119	.106/.378
	Low	.072	.511/.263	.333/.135	.467/.611	.720	.806/.602	.732/.495	.143/.284	.768	.966/.490	.944/.174	.003/.308

Table 9: Correctness-stratified DATG results under $en \rightarrow x$ by language pair, difficulty, and model scale. For CAR, PMF, and HAR, each cell reports C/I for correct and incorrect traces; "–" indicates no correct trace in that cell. DATG scoring does not use final-answer correctness.

I Controlled-Execution Details

Cost accounting. Controlled-execution cost reports decoded output tokens only. For retry-based methods, decoded tokens include aborted continuations; for Formula-Retry, they also include scaffold generation. Prompt/input tokens are not included, so this is a decoded-token budget rather than total serving cost. In Table 3, token totals are aggregated over Swahili and Telugu within each difficulty level and reported in millions.

I.1 Loop-Retry Trigger and Example

Loop-Retry is an early-checkpoint repetition monitor, not a terminal failure detector. It does not use the gold answer, answer correctness, Math-Verify, DATG alignment, reference DAGs, language-compliance labels, selector labels, or boxed-answer absence as retry triggers.

Early checkpoint. For a difficulty-specific generation budget B, Loop-Retry inspects the reasoning trace after the first quarter of the budget:

$$C = B/4.$$

Thus $C = 1024, 2048, 4096$ tokens for low, medium, and high difficulty, respectively. In the reported high-difficulty run, $\texttt{check_tokens} = 4096$. We do not use a pre-check in the final setting ($\texttt{precheck_tokens} = 0$).

Surface repetition detector. At the checkpoint, the detector computes repetition statistics over the most recent 256 generated tokens. Let rep_n be the repeated n-gram rate:

$$\mathrm{rep}_n = \frac{\sum_g \max(0, \mathrm{count}(g) - 1)}{N_n},$$

where g ranges over token n-grams in the 256-token window and N_n is the number of observed n-grams. We use rep_{16}, rep_{32}, and the type-token ratio

$$\mathrm{ttr}_{256} = \frac{|\text{unique tokens}|}{256}.$$

We also detect suffix motifs: a repeated suffix unit of length 4–32 tokens repeated at least three consecutive times.

A partial trace is marked as a surface loop if

$$\mathrm{rep}_{16} \geq 0.65 \quad \text{or} \quad \mathrm{rep}_{32} \geq 0.30,$$

or a suffix motif is detected,

or if the softer joint condition holds:

$$\mathrm{rep}_{16} \geq 0.50 \quad \text{and} \quad \mathrm{ttr}_{256} \leq 0.16.$$

Mathematical progress guard. To avoid aborting legitimate long derivations, Loop-Retry also checks whether the recent tail introduces new mathematical information. We inspect the last 1600 characters and extract equations, formula-like expressions, variable assignments, and local number contexts. After normalization, if the tail contains a mathematical span not previously seen in the prefix, we set $\texttt{math_progress} = 1$. The retry trigger fires only when a surface loop is detected and $\texttt{math_progress} = 0$. A partial trace that already contains a boxed answer is not aborted. A partial trace shorter than $0.8C$ tokens is also not aborted, because it ended before the intended checkpoint.

Concrete Loop-Retry example. Loop-Retry keeps the original target-language prompt unchanged. If the first continuation enters a repetitive loop at the checkpoint, that continuation is discarded and the model resamples from the same prompt.

```
Problem:
Janet's ducks lay 16 eggs per day. She eats 3 for breakfast
and uses 4 for muffins. How many eggs are left after 7 days?

Target language: Swahili

Trial 1 continuation:
<think>
Kwanza tunahesabu mayai yanayobaki kila siku.
16 - 3 - 4 = 9.
Mayai yaliyobaki ni 9. Mayai yaliyobaki ni 9.
Mayai yaliyobaki ni 9...
[detector: repeated suffix, no new math progress]
=> abort and retry

Trial 2 continuation:
<think>
Kwanza tunahesabu mayai yanayobaki kila siku:
16 - 3 - 4 = 9.
Kwa siku 7:
9 \times 7 = 63.
</think>
\boxed{63}
```

I.2 Formula-Retry Scaffold Prompt and Example

Formula-Retry uses a two-phase protocol. Phase 1 elicits a compact English/symbolic pre-answer scaffold inside an execution-state block. Phase 2 continues the reasoning trace in the target language. The scaffold may contain known quantities, variables, relations, and formula skeletons, but it must not contain the final numeric result, a boxed answer, or a final-answer sentence. The scaffold is generated once and kept fixed across retries; if the continuation loops, only the target-language reasoning after `</execution_state>` is resampled.

Formula scaffold prompt

```
System message:
You are a mathematical reasoning assistant. Use a
two-phase reasoning protocol.
Phase 1 is a compact English/symbolic PRE-ANSWER
execution plan inside <execution_state>...</execution_state>.
It may use English words, formulas, variables, known
quantities, relations, and plan steps. The Plan section
should contain compact formula skeletons with unknown
results, e.g., "Calculate remaining eggs: 16 - 3 - 4 = ?".
```

```
You may include arithmetic operators, but all computed
intermediate and final numeric results must be masked as ?.
Do NOT compute the final numeric result in Phase 1. Do NOT
write a final answer, boxed answer, or any line named
Final/Answer. Do not use target-language prose in Phase 1.
Phase 2 is a step-by-step solution entirely in {language}.
Use the plan from Phase 1, perform the actual arithmetic in
{language}, and write both the reasoning trace and any
final-answer sentence in {language}. The only
English/symbolic part allowed in Phase 2 is the mathematical
expression inside \boxed{}. Do not write an English
explanation after </think>; after the reasoning trace,
output only \boxed{} or a short {language} final sentence
with \boxed{}. The execution state is a one-time planning
tool: never open or mention another execution-state block in
Phase 2.

User message:
{problem_statement}

Assistant prefill for Phase 1:
<think>
<execution_state>
Known:
-

After Phase 1 scaffold generation, the execution state is
closed and Phase 2 begins with the target-language starter:
</execution_state>
{target_language_starter}
```

Concrete Formula-Retry example. The scaffold contains only answer-free structure. In a retry, the same scaffold is reused and only the continuation after `</execution_state>` changes.

```
Problem:
Janet's ducks lay 16 eggs per day. She eats 3 for breakfast
and uses 4 for muffins. How many eggs are left after 7 days?

Phase 1 scaffold:
<think>
<execution_state>
Known:
- eggs_per_day = 16
- eaten_per_day = 3
- muffins_per_day = 4
- days = 7

Plan:
- remaining_per_day = 16 - 3 - 4 = ?
- total_remaining = remaining_per_day * 7 = ?
</execution_state>

Trial 1 after scaffold:
Kwanza tunatumia mpango uliopo.
16 - 3 - 4 = 9.
9 ni mayai yaliyobaki. 9 ni mayai yaliyobaki...
[detector: loop, no new math progress]
=> keep scaffold, retry only after </execution_state>

Trial 2 after same scaffold:
Kwanza tunahesabu mayai yanayobaki kila siku:
16 - 3 - 4 = 9.
Kisha kwa siku 7:
9 \times 7 = 63.
</think>
\boxed{63}
```

I.3 Shared Retry Policy and Decoding

Retry policy. Each retry method uses at most five trials. If the checkpoint trigger fires before the final trial, the current partial continuation is discarded and the model is resampled from the same prompt context. For Formula-Retry, the `<execution_state>` scaffold is generated once and kept fixed; only the reasoning continuation after `</execution_state>` is resampled. If the trigger still fires on the final trial, the final trial is accepted and marked as an unhealthy forced accept; no additional retry is performed.

All reasoning continuations use temperature 0.6, top-$p = 0.95$, and the maximum budget B. The first trial uses repetition penalty 1.08, and retry trials use repetition penalty 1.13. Formula-scaffold generation uses temperature 0.2, top-$p = 0.95$, repetition penalty 1.02, and a maximum of 4096 tokens, stopping at `</execution_state>`. We use vLLM with `max_num_seqs` $= 16$, `max_num_batched_tokens` $= 8192$, and GPU memory utilization 0.82. All controlled-execution experiments are run with three independent sampling seeds, 42, 215, 316.

Retry prompts. Base and Loop-Retry use the same target-language solver prompt and the same assistant-side direct-first starter; Loop-Retry changes only the test-time sampling procedure when the repetition trigger fires. Formula-Retry uses the two-phase scaffold prompt in Appendix I.2. Formula-Retry keeps the sanitized `<execution_state>` fixed across retries and resamples only the continuation after `</execution_state>`.

J Environment and Hyperparameters

All experiments are conducted on a server equipped with 8 NVIDIA RTX A6000 GPUs. Model inference is performed with `vLLM` (Kwon et al., 2023).