Beyond Direct Gains: Matched Controls for Evaluating Concept Scaffolds

Jiangshan He [1] **Song Dai** [1] **Xiaolong Qiao** [1] **Jungang Li** [1] **Yibo Yan** [1] **Xuming Hu** [1]

Abstract

Reasoning scaffolds such as concepts and hints are often evaluated by asking whether a helpful scaffold improves accuracy. This direct comparison is under-identified: a gain may come from the scaffold's semantic content, from the presence of a scaffold-shaped prompt block, or from sensitivity to plausible but inapplicable auxiliary information. We propose a matched-control protocol for concept scaffolds that compares no-scaffold, format-only, helpful, misleading, and wrong-fact conditions. We evaluate it on 2,000 undergraduate mathematics problems with three randomized versions per problem, which lets us distinguish average instance-level gains from stable gains across variants of the same reasoning objective. In this nine-setting evaluation, helpful concepts improve over format-only controls for most settings under stable accuracy, but the average gain is modest ($G_{\text{helpful}} = +1.22$ points). We interpret this as a diagnostic helpful-over-format contrast rather than a pure estimate of semantic effect.

1. Introduction

Reasoning scaffolds such as rationales, plans, generated knowledge, concepts, and hints are often used to guide language-model reasoning (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022; 2023; Zhou et al., 2022; Yao et al., 2023; 2022; Liu et al., 2022). In mathematical reasoning, a concept scaffold may name a theorem, formula, definition, or method that appears relevant to a problem. If accuracy improves after adding such a scaffold, it is natural to ask whether the model benefited from the concept itself.

The direct gain from a helpful scaffold is not enough to answer this question. The same improvement may reflect the content of the concept, the format of an additional "Concepts" block, or a broader response change caused by plausible auxiliary information. Prior work on in-context learning, prompt sensitivity, misleading information, and rationale faithfulness gives related reasons to be cautious about attributing behavior to the stated semantic content alone (Min et al., 2022; Turpin et al., 2023; Lanham et al., 2023; Bhuiya et al., 2024; Chen et al., 2025).

We frame scaffold evaluation as two diagnostic questions. First, does a helpful concept still help after controlling for scaffold format? Second, does the effect persist across randomized versions of the same base problem? To answer them, we evaluate no-scaffold, format-only, helpful, misleading, and wrong-fact conditions on a 2,000-base-problem mathematics testbed with three randomized versions per problem. Our results show a consistent but modest helpful-over-format gain in stable accuracy, suggesting that helpful concepts carry signal beyond format alone, while direct helpful-vs.-no comparisons overstate what the evidence identifies.

Our contributions are:

- We recast concept-scaffold evaluation as an identification problem, where direct helpful-vs.-no gains conflate semantic content, prompt format, and sensitivity to auxiliary context.
- We propose a matched-control protocol that defines diagnostic gains against format-only, helpful, misleading, and wrong-fact concept conditions.
- We evaluate the protocol with both per-instance and stable accuracy over randomized problem variants, finding consistent but modest helpful-over-format gains and negative misleading-control effects in a concept-only UGMathBench setup across nine instruction-compatible model settings.

2. Matched-Control Protocol

Figure 1 illustrates the protocol. For each base problem q with gold answer y, we evaluate five matched input conditions across randomized problem versions. N gives only the problem. F preserves the scaffold block but replaces each concept with `[N/A]`. D gives the problem-level helpful concept. M gives mathematically valid but inapplicable

[1]The Hong Kong University of Science and Technology (Guangzhou). Correspondence to: Xuming Hu <xuminghu@hkust-gz.edu.cn>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

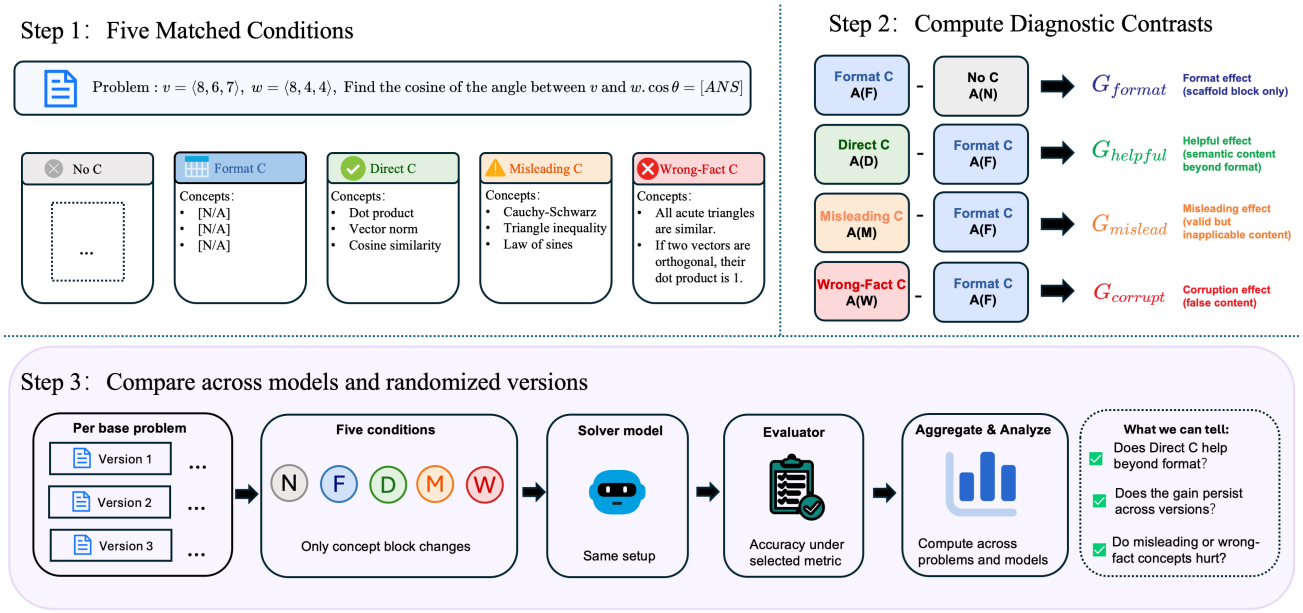

Figure 1. Matched-control protocol: for each base problem, we construct five concept conditions, compute diagnostic contrasts against no-concept or format-only controls, and aggregate across randomized versions and models.

concepts sampled from other problems. W gives subject-level false mathematical facts in the same scaffold format. These conditions are matched by problem and differ only in the concept block.

Design Desiderata. We design the protocol around three requirements that are important for controlled evaluation in mathematical reasoning. First, the control must preserve scaffold structure; otherwise a helpful-vs.-no comparison cannot distinguish semantic content from prompt-format effects. Second, the negative control must be mathematically plausible; otherwise a model can ignore it simply because it is obviously nonsensical. Third, the metric should separate average instance-level improvement from stable improvement over randomized versions of the same reasoning objective. The five conditions and the two metrics below instantiate these requirements.

For any chosen accuracy metric A, we compute four diagnostic deltas:

$$G_{\text{format}} = A(F) - A(N), \tag{1}$$
$$G_{\text{helpful}} = A(D) - A(F), \tag{2}$$
$$G_{\text{mislead}} = A(M) - A(F), \tag{3}$$
$$G_{\text{corrupt}} = A(W) - A(F). \tag{4}$$

The key diagnostic contrast is G_{helpful}, not $A(D) - A(N)$: a helpful scaffold may outperform no scaffold simply because the scaffold block changes format or response behavior. This contrast is more informative than the direct no-scaffold comparison, but it should not be read as a pure estimate of semantic effect. Similarly, G_{mislead} is diagnostic rather than a pure trigger estimate, because inapplicable content can also interfere with reasoning.

3. A Worked Control Example

Table 1 shows one complete control set. The example is deliberately simple enough for a reader to inspect, but it illustrates why the five conditions are not interchangeable. Direct C gives the definition and test procedure for primality; Misleading C gives true mathematical facts that preserve the look of a concept scaffold but are not useful for selecting small prime numbers; Wrong-Fact C keeps the same format while making the auxiliary content explicitly false. This example is also useful for interpreting the aggregate metrics: F asks whether a scaffold block changes behavior by itself, $D - F$ asks whether problem-relevant content adds value, and $M - F$ asks whether the model can ignore plausible but inapplicable mathematical content.

4. Experimental Setup

Data and Metrics. We instantiate the protocol on UG-MathBench using concept-only CHAMP-style pseudo-annotations (Xu et al., 2025; Mao et al., 2024): 2,000 base problems, three randomized versions per problem, and 6,000 solver prompts per condition. Concepts are generated from version 1 and reused for versions 1–3. Concept conditions,

Setting	Example concept block for a prime-number selection problem	Diagnostic role
No C	No concept block.	Baseline without scaffold content or scaffold format.
Format C	**Concepts:** 1. `[N/A]` 2. `[N/A]`	Controls scaffold format and concept count.
Direct C	**Concepts:** 1. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. 2. To determine if a number is prime, check divisibility by all integers up to the square root of the number.	Tests helpful semantic content without revealing the selected options.
Misleading C	**Concepts:** 1. The equation $a^n + b^n = c^n$ has no non-zero integer solutions for a, b, and c when $n > 2$. 2. For any integers a, b, and m, $(a * b) \bmod m = ((a \bmod m) * (b \bmod m)) \bmod m$.	Tests true but inapplicable mathematical content.
Wrong-Fact C	**Concepts:** 1. All prime numbers are even. 2. If a number is divisible by 3, then it must be divisible by 9.	Tests explicitly false auxiliary content.

Table 1. Worked example of the five matched concept conditions. The underlying problem asks which of several small integers are prime.

solver prompts, teacher prompts, and an example are in Section A; dataset construction, filtering, replacement rules, and the expert audit are in Section B. We report two metrics following UGMathBench's multi-version evaluation design: per-instance accuracy I corresponds to average accuracy over versioned instances, and stable accuracy S counts a base problem as correct only when all three randomized versions are answered correctly. S is primary because the paper asks whether a scaffold helps the underlying reasoning objective, not merely one sampled surface version. All G values in Table 3 are percentage-point differences in S.

Evaluation Questions. The experiment is organized around three questions. **Q1**: Does Direct C still help after controlling for scaffold format? This is measured by G_{helpful}. **Q2**: Do plausible but inapplicable concepts behave differently from format-only placeholders? This is measured by G_{mislead}. **Q3**: Are conclusions stable under both per-instance accuracy I and stable accuracy S? This checks whether scaffold gains reflect isolated version-level wins or more robust gains over the same base problem.

Uncertainty. We estimate uncertainty with paired bootstrap resampling over the 2,000 base problems. Each of 10,000 replicates samples base problems with replacement, preserves all three versions of each sampled problem, and recomputes all condition scores and matched gains on the same sampled problems. Section C gives the full setup.

Models and Evaluation. We evaluate nine instruction-compatible settings from Qwen2.5, Qwen2.5-Math, DeepSeek-R1 distilled models, Qwen3, Gemma 3, Phi-4-mini, and Llama 3.2; model reports are cited where available (Team, 2024; Yang et al., 2024; Guo et al., 2025; Yang et al., 2025; Team et al., 2025; Abouelenin et al., 2025). Qwen3 models are evaluated with thinking disabled, so they are compared in the same instruction-following regime as the other models rather than as long-reasoning systems. All runs use raw responses, deterministic decoding, and the same rule-based answer-equivalence evaluator. Most models use maximum generation length 8192; Qwen2.5-Math-7B-Instruct follows a 4096-token setting used for the Qwen2.5-Math family in our scripts. Section E lists exact model sources, serving versions, and decoding settings; Section D reports auxiliary diagnostic runs.

5. Pseudo-Concept Construction

Because UGMathBench does not provide CHAMP-style concept labels, we construct concept-only pseudo-annotations before solver evaluation. We start from 5,061 base problems, remove items whose answer type or option format is incompatible across randomized versions, and sample 2,000 base problems with subject-stratified proportional sampling using seed 42. Each sampled base problem contributes three solver versions, so every prompt condition has 6,000 rows. This sampling strategy keeps the evaluation large enough to estimate small matched gains while retaining the multi-version structure needed for stable accuracy.

For each selected base problem, Qwen2.5-72B-Instruct generates 1–3 reusable mathematical concepts from version 1, the problem metadata, and the reference answer. The reference answer is used only to validate the annotation: the teacher must solve the problem and mark the annotation invalid if the derived answer does not match the reference. Only cleaned concept text is copied into solver prompts; solution steps, reasoning objectives, self-check fields, and reference answers are discarded. This separation matters because the solver receives a concept scaffold, not a worked solution.

We apply automatic filtering before constructing any prompt set. Concepts are removed if they are empty, duplicate, contain the reference answer or option-answer patterns, collapse to a subject/topic/subtopic label, contain problem-specific phrases such as "this problem" or "the answer", or become vague advice such as "use algebra". Failed items are retried and then replaced from a reserve pool; the final set contains 2,000 valid base problems and 3,786 cleaned concepts. Misleading C is then sampled from the cleaned concept pool

Construction check	Value
Eligible base problems after compatibility filtering	4,977
Final sampled base problems	2,000
Solver rows per condition	6,000
Cleaned concepts	3,786
Average concepts per scaffolded prompt	1.893
Initially selected problems replaced	370
Direct C relevance audit	100/100
Direct C cross-version audit	98/100
Misleading C validity/inapplicability audit	91/100
Wrong-Fact C falsity audit	100/100

Table 2. Summary of pseudo-concept construction and expert audit. The audit samples 100 base problems and checks the concept blocks used to form Direct C, Misleading C, and Wrong-Fact C.

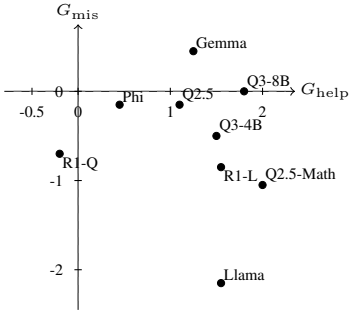

Figure 2. Per-model scaffold profiles under stable accuracy S; the lower-right quadrant indicates helpful-over-format gains with misleading-control harm.

while excluding the same base problem, exact direct-concept reuse, duplicates, and high-overlap candidates. Wrong-Fact C uses subject-level false facts with the same concept count as Direct C.

The audit in Table 2 is not meant to certify every annotation, but it targets the main failure modes reviewers should care about. Direct C passed relevance checks in all 100 audited examples, and Wrong-Fact C was false in all audited examples. The weaker point is Misleading C: 9/100 audited examples failed because a sampled concept was still applicable or was not valid under the needed conditions. We therefore interpret Misleading C as a strong diagnostic control rather than a perfect intervention.

6. Results

Overview. We report the results in the same order as the evaluation questions. Table 3 is the primary table and uses stable accuracy S. Table 4 checks whether the aggregate conclusions also hold under per-instance accuracy I. Figure 2 summarizes model-level scaffold profiles by placing each model according to helpful-over-format gain and misleading-control gain.

Q1: Direct Gains Mostly Survive the Format Control, but Remain Small. Table 3 shows raw condition scores and matched gains under stable accuracy. The average direct helpful-vs.-no comparison is +1.25 points ($S_D = 35.49$ vs. $S_N = 34.24$), but the controls decompose it into an almost-zero format component ($G_{\text{format}} = +0.03$, 95% CI [-0.34, 0.41]) and a helpful-over-format component ($G_{\text{helpful}} = +1.22$). Table 4 shows the same sign pattern under both I and S: format effects are near zero, helpful-over-format effects are positive, and misleading concepts reduce performance.

Q2: Misleading Controls Are an Informative Stress Test. G_{mislead} is negative for 7/9 models under S, with an average of -0.57 points and a 95% CI of [-0.99, -0.14]. This makes Misleading C informative because it preserves a plausible mathematical scaffold while removing intended applicability to the current problem. Figure 2 highlights the model-level profile: Gemma-3-4B is the main exception with a positive misleading effect (+0.45), showing that the misleading-control effect is contingent rather than universal. This is why a positive G_{helpful} alone should not be read as selective semantic use.

Q3: Stable Accuracy Separates Average and Robust Wins. The randomized variants do more than increase sample size: they let us ask whether a scaffold helps isolated surface versions or the base reasoning objective. If a condition improved I but not S, the gain would be weaker evidence for stable scaffold utility. Here Direct C improves over Format C under both metrics, so the helpful effect is not only a version-level artifact; however, the modest magnitude shows that concept utility should not be overstated.

Wrong-Fact Controls Are Secondary. Wrong-Fact C is a robustness check rather than the main evidence: its average G_{corrupt} is -0.17 points and its CI includes zero. Model-factor observations are also exploratory. The R1-distilled settings do not share a single scaffold profile, math specialization is associated with a different profile in our table, and Qwen3-8B has higher raw stable accuracy than Qwen3-4B while both have modest scaffold gains. We therefore treat scaffold utility as a behavior profile, not as a direct function of raw accuracy or size.

7. Exploratory Model-Factor Analysis

The model comparison lets us ask what kinds of factors might be associated with different scaffold profiles. We emphasize that these are not causal claims: the models differ in architecture, data, post-training, tokenizer, serving stack, and release recipe. Still, the matched-control view is useful because it separates raw accuracy from scaffold selectivity.

Model	Stable Acc. S					Diagnostic gains			
	N	F	D	M	W	G_{fmt}	G_{help}	G_{mis}	G_{cor}
Llama-3.2-3B	15.85	17.00	18.55	14.85	14.90	+1.15	+1.55	-2.15	-2.10
Gemma-3-4B	19.75	21.40	22.65	21.85	21.25	+1.65	+1.25	+0.45	-0.15
Phi-4-mini	37.95	38.30	38.75	38.15	38.30	+0.35	+0.45	-0.15	+0.00
Qwen2.5-7B	42.00	41.50	42.60	41.35	41.45	-0.50	+1.10	-0.15	-0.05
Qwen2.5-Math-7B-Inst.	40.75	39.60	41.60	38.55	39.55	-1.15	+2.00	-1.05	-0.05
Qwen3-4B	48.00	48.15	49.65	47.65	48.35	+0.15	+1.50	-0.50	+0.20
Qwen3-8B	53.35	52.65	54.45	52.65	53.20	-0.70	+1.80	+0.00	+0.55
R1-Qwen	25.60	25.30	25.10	24.60	25.35	-0.30	-0.20	-0.70	+0.05
R1-Llama	24.90	24.55	26.10	23.70	24.60	-0.35	+1.55	-0.85	+0.05
Average	34.24	34.27	35.49	33.71	34.11	+0.03	**+1.22**	**-0.57**	-0.17

Table 3. Main UGMathBench results using stable accuracy S (%). Left: raw scores for N, F, D, M, W; right: diagnostic gains. Colors encode signed gain magnitude; boldface marks the main aggregate contrasts.

Gain	I	95% CI	S	95% CI
G_{fmt}	+0.05	[-0.21, 0.32]	+0.03	[-0.34, 0.41]
G_{help}	**+1.22**	**[0.85, 1.60]**	**+1.22**	**[0.74, 1.72]**
G_{mis}	**-0.33**	**[-0.62, -0.04]**	**-0.57**	**[-0.99, -0.14]**
G_{cor}	-0.15	[-0.43, 0.13]	-0.17	[-0.57, 0.24]

Table 4. Model-average gains under I and S with paired bootstrap CIs over base problems. Colors and boldface highlight signed magnitude and main contrasts.

Raw Accuracy Is Not Scaffold Selectivity. The Qwen3 settings have the strongest raw stable accuracy in our table, with Qwen3-8B reaching $S_N = 53.35$ and Qwen3-4B reaching $S_N = 48.00$. Their scaffold gains, however, remain modest: G_{helpful} is +1.80 and +1.50, respectively, and G_{mislead} is 0.00 and -0.50. Conversely, Llama-3.2-3B has much lower raw stable accuracy ($S_N = 15.85$) but a clear selective profile: $G_{\text{helpful}} = +1.55$ and $G_{\text{mislead}} = -2.15$. This contrast suggests that a stronger solver is not necessarily a more scaffold-selective solver.

Math Specialization May Be Associated with Content Sensitivity. Qwen2.5-Math-7B-Instruct has the largest helpful-over-format gain in stable accuracy (+2.00), but it also has a negative misleading-control effect (-1.05). This pattern is consistent with the possibility that math-specialized post-training is associated with stronger responsiveness to mathematical auxiliary content in both directions: relevant concepts can help, while plausible but inapplicable concepts can interfere. The point is not that math specialization is harmful, but that specialization does not remove the need for content controls.

R1-Distilled Settings Do Not Share One Scaffold Profile. The two DeepSeek-R1 distilled settings behave differently. R1-Qwen has $G_{\text{helpful}} = -0.20$, while R1-Llama has $G_{\text{helpful}} = +1.55$ and $G_{\text{mislead}} = -0.85$. Since both share the same high-level distillation source but differ in base model family and release recipe, the results argue against treating "R1-distilled" as a homogeneous reasoning category for scaffold evaluation. Matched controls instead expose a per-model behavioral profile.

Exceptions Are Informative. Gemma-3-4B is the main exception to the misleading-control pattern: $G_{\text{mislead}} = +0.45$ under stable accuracy. This does not invalidate the protocol; it is exactly the kind of case the protocol is designed to reveal. A direct helpful-vs.-no comparison would show that Direct C improves over No C, but the matched profile shows that plausible inapplicable concepts can also improve this model, making semantic-use interpretations less secure.

8. Discussion

What the Controls Identify. The main lesson is not that concept scaffolds produce large gains, but that direct helpful-vs.-no comparisons are under-identified. In our results, the helpful-over-format gain is positive but modest, while misleading controls often reduce stable accuracy. This pattern supports reporting matched contrasts rather than treating Direct C gains as direct evidence of semantic use.

What the Testbed Does Not Identify. The pseudo-concepts are controlled scaffold inputs, not human explanations. Filtering and expert audit reduce obvious leakage, but they do not prove that every Direct C is equally helpful or every Misleading C is perfectly inapplicable. We therefore treat model-family observations as exploratory behavior profiles rather than causal claims about training recipes.

9. Related Work

Our setting is related to scaffolded reasoning methods that elicit or provide intermediate structure, including chain-of-thought prompting, self-consistency, plan-based prompting, tree-structured reasoning, ReAct-style prompting, self-refinement, and generated knowledge (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022; 2023; Zhou et al., 2022; Yao et al., 2023; 2022; Madaan et al., 2023; Shinn et al., 2023; Liu et al., 2022). We do not introduce a new

scaffold generator. Instead, we study how to interpret a gain after a scaffold has been supplied.

Our work also follows diagnostic evaluation traditions that test model behavior through controlled input changes rather than aggregate accuracy alone. Behavioral testing and contrast-set evaluation show how carefully designed perturbations can expose failures hidden by standard test scores (Ribeiro et al., 2020; Gardner et al., 2020). Work on in-context learning artifacts and prompt-format sensitivity further shows that model behavior can change for reasons unrelated to the semantic content intended by the experimenter (Min et al., 2022; Sclar et al., 2024). This motivates treating scaffold content itself as an experimental factor.

Unlike scaffolded-vs.-unscaffolded comparisons, our protocol reports matched contrasts against format-only and inapplicable-content controls. This connects to work on misleading information and rationale faithfulness, which cautions against directly attributing model behavior to stated rationales or auxiliary context (Turpin et al., 2023; Lanham et al., 2023; Bhuiya et al., 2024; Chen et al., 2025). CHAMP provides human concept and hint annotations for competition mathematics and includes misleading concepts (Mao et al., 2024). Our testbed is not a full CHAMP reproduction: it is concept-only and uses teacher-generated pseudo-concepts, but it allows a larger matched-control evaluation with randomized versions.

10. Conclusion

Helpful scaffold gains are not self-explanatory. In our concept-only UGMathBench setup, helpful concepts improve over format-only controls for most instruction-compatible settings under stable accuracy, but the average gain is modest and the same sign pattern appears under per-instance accuracy. Format-only controls are near zero, misleading controls usually reduce accuracy, and wrong-fact controls are mixed, motivating matched controls and stability metrics for scaffold evaluation.

11. Limitations

The protocol is diagnostic rather than causal. $D - F$ is more controlled than a direct helpful-vs.-no comparison, but it still differs in length, mathematical wording, topical cues, and plausibility, so it is not a pure estimate of semantic effect. Misleading C is mathematically valid but inapplicable, so its effect can combine response-mode changes and content interference; the manual audit also shows that this control is imperfect. Wrong-Fact C is not a pure adversarial condition, because models may ignore obviously false statements or treat the scaffold block as a general reasoning cue. UGMathBench concepts are teacher-generated pseudo-annotations rather than human labels: filtering removes explicit answer leakage, but cannot rule out all solution-path leakage. All conditions use the same rule-based evaluator, but evaluator errors are not separately audited in this version. Bootstrap intervals quantify uncertainty over sampled base problems, but they do not propagate annotation-audit uncertainty, such as the 9/100 Misleading-C audit failures. Thinking-enabled Qwen3 behavior is outside the present comparison because the main experiment standardizes instruction-compatible non-thinking decoding under a fixed generation budget; evaluating long-reasoning settings would require a separate budget and answer-extraction protocol. The Qwen2.5-Math-7B-Instruct token-budget exception may also affect its absolute scores and should be considered when interpreting the math-specialist profile. Finally, model-factor interpretations are based on reported model families and observed scaffold profiles; controlled ablations over pretraining and post-training recipes would be required for causal claims.

Impact Statement

This paper presents work whose goal is to improve diagnostic evaluation of machine-learning systems. The main societal implication is methodological: scaffolded evaluations should be less likely to over-attribute model gains to semantic reasoning when prompt-format or misleading-content controls have not been tested.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. This research work was supported by the HPC AI Intelligent Computing Platform of The Hong Kong University of Science and Technology (Guangzhou).

References

Abouelenin, A., Ashfaq, A., Atkinson, A., Awadalla, H., Bach, N., Bao, J., Benhaim, A., Cai, M., Chaudhary, V., Chen, C., et al. Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras. *arXiv preprint arXiv:2503.01743*, 2025.

Bhuiya, N., Schlegel, V., and Winkler, S. Seemingly plausible distractors in multi-hop reasoning: Are large language models attentive readers? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2514–2528, 2024.

Chen, G., Wang, P., Zhang, T., and Chen, F. Exploring the hidden reasoning process of large language models by misleading them. *arXiv preprint arXiv:2503.16401*, 2025.

Gardner, M., Artzi, Y., Basmov, V., Berant, J., Bogin, B., Chen, S., Dasigi, P., Dua, D., Elazar, Y., Gottumukkala,

A., et al. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323, 2020.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., Hubinger, E., Kernion, J., et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Liu, J., Liu, A., Lu, X., Welleck, S., West, P., Le Bras, R., Choi, Y., and Hajishirzi, H. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3154–3169, 2022.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Mao, Y., Kim, Y., and Zhou, Y. Champ: A competition-level dataset for fine-grained analyses of llms' mathematical reasoning capabilities. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13256–13274, 2024.

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 11048–11064, 2022.

Ribeiro, M. T., Wu, T., Guestrin, C., and Singh, S. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4902–4912, 2020.

Sclar, M., Choi, Y., Tsvetkov, Y., and Suhr, A. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *International Conference on Learning Representations*, volume 2024, pp. 25055–25083, 2024.

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., yeong Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S.,

Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Team, Q. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Turpin, M., Michael, J., Perez, E., and Bowman, S. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36: 74952–74965, 2023.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R. K.-W., and Lim, E.-P. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 2609–2634, 2023.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Xu, X., Zhang, J., Chen, T., Chao, Z., Hu, J., and Yang, C. Ugmathbench: A diverse and dynamic benchmark for undergraduate-level mathematical reasoning with large language models. *arXiv preprint arXiv:2501.13766*, 2025.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q., et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A. Prompt Conditions and Templates

The five concept conditions are defined in the main text and illustrated in Table 1. Below we provide the prompt templates used to instantiate them.

A.1. Prompt Templates

Solver Prompt. All solver prompts contain the UGMathBench problem, answer-type instruction, and final-answer format. The scaffold block is omitted for No C and inserted before the final-answer instruction for the other conditions:

> **Problem header.** The following is an undergraduate-level mathematical problem in {subject}. Complete all placeholders `[ANS]`. The answer type is {answer_type_description}.
> **Problem.** {problem}
> **Optional scaffold block.** You may find the following information useful. Concepts: {concept list}.
> **Final-answer instruction.** Use LaTeX for mathematical symbols and end with "The final answer is \boxed{ANSWER}".

In Format C, each concept is replaced by `[N/A]`; in Direct C, concepts are teacher-generated for the current base problem; in Misleading C, concepts are valid but inapplicable; in Wrong-Fact C, concepts are subject-level false facts.

Teacher Prompt. The teacher model sees only version 1 of each sampled base problem. The input fields are the problem id, subject, topic, subtopic, keywords, level, the V1 problem text, the V1 reference answer, answer type, and options when available. The reference answer is used only to validate the teacher's annotation: the teacher must solve the problem, check that its derived answer matches the reference answer, and return strict JSON. The required JSON fields are `annotation_status`, `reasoning_objective`, `solution_steps`, `final_answer_check`, and `concepts`. Each concept has a short name, a text field, boolean self-checks for generality and topic-label status, and linked step indices. The teacher is not used for solver evaluation.

What Reaches the Solver. Only the cleaned `concepts[*].text` fields are copied into Direct C prompts. The solution steps, derived answer, reference answer, reasoning objective, self-check fields, and step indices are discarded before solver prompt construction. Thus Direct C is a concept-only scaffold rather than a worked solution, hint, or answer-bearing annotation.

A.2. Teacher Annotation Prompt

The teacher prompt below is the template used to construct pseudo-concepts. We show the essential prompt text; braces indicate fields filled from each UGMathBench item.

System Prompt.

> You are a mathematical annotation expert. Your task is to create CHAMP-style concept annotations for undergraduate mathematics problems.
>
> Definitions: A Concept is reusable mathematical knowledge, such as a theorem, formula, identity, definition, standard method, or general mathematical principle. A Concept must be general and reusable. It must not be a topic label, keyword, or problem-specific instruction.
>
> Rules: First solve the problem step by step. The derived final answer must match the provided reference answer. Generate 1 to 3 Concepts. Each Concept must be useful for solving the problem, but must not reveal the final answer. Do not copy the subject, topic, subtopic, or keyword names as Concepts. Do not output vague concepts such as "solve step by step", "use algebra", or "think carefully". Do not include the final answer, option letter, or any answer-specific number in the Concept unless it is part of a general theorem. If the problem cannot be solved or the derived answer does not match the reference answer, mark the annotation as invalid. Return strictly valid JSON only.

User Prompt.

> Create CHAMP-style concept annotations for the following UGMathBench problem.
>
> Problem metadata: id={id}; subject={subject}; topic={topic}; subtopic={subtopic}; keywords={keywords}; level={level}.
>
> Problem version 1: {problem_v1}
>
> Reference answer for version 1: {answer_v1}
>
> Answer type: {answer_type_v1}
>
> Options, if any: {options_v1}
>
> Output strictly valid JSON with fields: `annotation_status`, `invalid_reason`, `reasoning_objective`, `solution_steps`, `final_answer_check`, and `concepts`. Each concept must contain `id`, `name`, `text`, `category`, `is_general_reusable`, `is_topic_label_only`, `contains_answer`, and `step_indices`.

B. Pseudo-Concept Construction and Quality Control

The original UGMathBench pool contains 5,061 base problems and three randomized versions per base problem. We first remove base problems whose answer type or option format is incompatible across V1–V3, leaving 4,977 eligible base problems. We then sample 2,000 base problems with subject-stratified proportional sampling using seed 42, and keep the remaining 2,977 eligible base problems as a reserve pool. Each selected base problem is expanded to V1–V3 only at solver time, yielding 6,000 prompts per condition.

Annotation Generation. For each selected base problem, Qwen2.5-72B-Instruct is queried through an OpenAI-compatible API with temperature 0.0 and top-p 1.0. The teacher is asked to output 1–3 concepts that are reusable mathematical knowledge, such as a theorem, formula, identity, definition, standard method, or general principle. The prompt explicitly forbids topic labels, vague advice such as "solve step by step", problem-specific instructions, final answers, option letters, and answer-specific numbers. If the teacher's derived answer does not match the reference answer, the annotation is marked invalid.

Validation and Replacement. Every teacher output is parsed as JSON and validated before prompt construction. We require `annotation_status=valid`, nonempty solution steps, 1–3 concepts, and `final_answer_check.matched=true`; we also run a local answer-equivalence check between the teacher's derived answer and the V1 reference answer. Invalid annotations are retried up to two times. If they still fail, the base problem is replaced from the reserve pool under the same annotation procedure. In the final construction, 370 initially selected problems are replaced, and the final set contains 2,000 valid base problems.

Leakage Filtering. Concepts are filtered before any solver prompt is built. We discard a concept if it is empty, duplicates another concept for the same problem, contains the reference answer string, contains an option-answer pattern, is only a subject/topic/subtopic/keyword label, is marked as topic-label-only, contains problem-specific phrases such as "this problem", "we need to find", or "the answer", or is vague advice such as "use algebra" or "think step by step". If all concepts for a base problem are removed, the annotation is rejected and the item is regenerated or replaced. This filtering is applied before Direct C, Misleading C, and the global concept pool are constructed. These checks cannot prove that every remaining concept is equally helpful or equally non-revealing, but they prevent direct answer copying and worked-solution leakage.

Cleaning and Prompt Construction. The final cleaned annotation set contains 3,786 concepts, with 1–3 concepts per base problem. Concepts longer than 40 words are shortened using the concept name, first sentence, or truncation to the length threshold, while retaining the original text for traceability. The cleaned set has no rejected annotations, no concepts over the length threshold, and no detected answer-instruction omissions in the generated prompts. Format C uses the same number of `[N/A]` placeholders as the cleaned direct-concept count. Misleading C samples from the cleaned concept pool, excluding the same base problem, exact direct-concept reuse, duplicate selections, and high-overlap candidates using token Jaccard similarity. Wrong-Fact C uses subject-level false facts with the same concept count. The final prompt sets contain 6,000 rows each for `no_c`, `format_c`, `direct_c`, `misleading_c`, and `wrong_fact_c`; the average number of concepts in scaffolded settings is 1.893.

B.1. Expert Manual Annotation Audit

We conducted an expert manual audit of 100 sampled base problems from the constructed UGMathBench prompt set. The auditor has graduate-level mathematical training and inspected the problem, reference answer, Direct C, Misleading C, Wrong-Fact C, and the three randomized versions for each sampled base problem. The audit checks whether Direct C is relevant, whether Direct C remains applicable across V1–V3, whether Misleading C is mathematically valid but inapplicable, and whether Wrong-Fact C is false. Table 5 summarizes the results.

Audit criterion	Pass / Total	Pass rate
Direct C is relevant	100 / 100	100%
Direct C applies to V1–V3	98 / 100	98%
Misleading C is valid and inapplicable	91 / 100	91%
Wrong-Fact C is false	100 / 100	100%

Table 5. Expert manual audit of 100 sampled UGMathBench pseudo-concept annotations.

The audit supports the Direct C and Wrong-Fact C construction, and it also clarifies the main limitation of Misleading C. The 9 Misleading C failures were not answer leaks; they came from two cases: a sampled concept was mathematically valid but still applicable to the target problem, or the concept was not valid under the conditions needed for the target problem. The 2 Direct C cross-version failures occurred when V2/V3 changed a subpart enough that a V1-derived concept no longer covered all variants.

C. Bootstrap Confidence Intervals

The main paper reports matched-gain intervals for both I and S. Table 6 gives the model-average stable-accuracy condition scores and paired bootstrap intervals.

Setup. The bootstrap is run on the nine completed main-model settings and the five prompt settings N, F, D, M, W. The common evaluation set contains 2,000 base problems for every model and setting. The sampling unit is `base_id`: each replicate samples 2,000 base problems with replacement, keeps the V1/V2/V3 rows together for each sampled base problem, and recomputes all condition scores before computing matched gains. We use 10,000 bootstrap replicates with random seed 42 and report percentile intervals using the 2.5 and 97.5 percentiles. The implementation computes I, S, and coverage accuracy; the main text reports

S as the primary metric and I as a robustness check.

Quantity	Estimate	95% CI
S_N	34.24	[32.71, 35.74]
S_F	34.27	[32.73, 35.79]
S_D	35.49	[33.95, 37.02]
S_M	33.71	[32.19, 35.20]
S_W	34.11	[32.59, 35.61]
G_{format}	+0.03	[-0.34, 0.41]
G_{helpful}	+1.22	[0.74, 1.72]
G_{mislead}	-0.57	[-0.99, -0.14]
G_{corrupt}	-0.17	[-0.57, 0.24]

Table 6. Model-average stable accuracy and matched gains with paired bootstrap 95% CIs over 2,000 base problems. Values are percentage points.

D. Auxiliary Results

Table 7 reports per-instance diagnostic gains over all 6,000 versioned prompts. The signs and relative patterns are close to the stable-accuracy table in the main text.

D.1. CHAMP Diagnostic Reference

We also ran the same control idea on CHAMP, which provides human concept and hint annotations (Mao et al., 2024). Because CHAMP has only 270 examples and the resulting deltas are noisy, we do not use CHAMP as a main-result table. The diagnostic pattern is nevertheless useful: it shows that scaffold effects can vary sharply with hints and model family.

D.2. Base-Model Prompt Compatibility

Qwen2.5-Math-7B base is useful for understanding the origin of DeepSeek-R1-Distill-Qwen-7B, but it is not directly comparable under the shared instruction-style scaffold prompt. In the UGMathBench run, the base model frequently copies answer placeholders rather than following the intended answer instruction. We therefore do not interpret its aggregate scores as scaffold utility. A fair before/after distillation comparison would require a separate completion-style prompt used consistently for both Qwen2.5-Math-7B base and DeepSeek-R1-Distill-Qwen-7B.

E. Model Sources and Implementation Details

Table 8 lists the model identifiers used in the main experiments and pseudo-concept construction. For checkpoints without a technical report, such as Llama-3.2-3B-Instruct, we report the Hugging Face source for reproducibility but do not use the model card as a scholarly citation. Qwen3 models are decoded with thinking disabled.

Model	G_{fmt}	G_{help}	G_{mis}	G_{cor}
Llama-3.2-3B	+0.65	+2.37	-1.70	-1.72
Gemma-3-4B	+1.90	+1.45	+1.27	+0.05
Phi-4-mini	+0.52	+0.42	-0.23	-0.42
Qwen2.5-7B	-0.07	+0.92	-0.75	-0.55
Qwen2.5-Math	-1.43	+2.38	-0.67	+0.58
Qwen3-4B	-0.38	+1.57	+0.15	+0.37
Qwen3-8B	-0.68	+0.70	-0.05	+0.12
R1-Qwen	+0.02	-0.12	-0.33	+0.05
R1-Llama	-0.03	+1.33	-0.67	+0.18
Average	+0.05	+1.22	-0.33	-0.15

Table 7. Auxiliary per-instance diagnostic gains I (%).

Name / role	Hugging Face checkpoint
Qwen2.5-7B	`Qwen/Qwen2.5-7B-Instruct`
Qwen2.5-Math	`Qwen/Qwen2.5-Math-7B-Instruct`
R1-Qwen	`deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`
R1-Llama	`deepseek-ai/DeepSeek-R1-Distill-Llama-8B`
Qwen3-4B	`Qwen/Qwen3-4B`
Qwen3-8B	`Qwen/Qwen3-8B`
Gemma-3-4B	`google/gemma-3-4b-it`
Phi-4-mini	`microsoft/Phi-4-mini-instruct`
Llama-3.2-3B	`meta-llama/Llama-3.2-3B-Instruct`
Annotation teacher	`Qwen/Qwen2.5-72B-Instruct`
Base diagnostic	`Qwen/Qwen2.5-Math-7B`

Table 8. Hugging Face checkpoints for the main settings and annotation teacher. Model reports are cited in the main setup where available.

E.1. Implementation Details

Solver inference uses Transformers v4.53.2 and vLLM v0.9.2. We use greedy decoding with temperature 0.0 and top-p 1.0. Unless otherwise noted, solver runs use `max_tokens=8192` and `max_model_len=8192`. The only main-table exception is Qwen2.5-Math-7B-Instruct, which uses `max_tokens=4096` and `max_model_len=4096` because this run follows the Qwen2.5-Math family serving profile used in our scripts. We report this exception explicitly rather than mixing it silently with the 8192-token configuration.

Evaluator Details. We evaluate all conditions with the same UGMathBench-compatible rule-based answer-equivalence evaluator. The evaluator is configured with strict final-answer extraction: it prioritizes explicit or boxed final answers and does not fall back to the last formula or number when no final answer is detected. It parses the last boxed answer when present, splits multi-answer predictions by top-level commas, and dispatches equivalence checks according to the UGMathBench answer type, including numeric, symbolic expression, equation, interval, ordered or unordered list, true/false, open-ended string, and single- or multiple-choice formats. Each judged prediction records the extracted answer and judge status.