When Structure Is Hidden: Measuring and Diagnosing the Compositional Gap in LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) often recover the local pieces of a problem yet still fail to assemble them into the correct global solution. Prior work studies this *compositionality gap* mainly in settings with explicit sub-question decompositions. We extend that analysis to tasks where the underlying structure is implicit and must be inferred from the input itself. We introduce a two-pass evaluation framework that first verifies piece-level understanding and then measures composition on the same example using the conditional metric $P(\text{composed wrong} \mid \text{all pieces correct})$.

Across five domains (contracts, math, SQL, code, and logic) and eight model configurations spanning four model families, we find that the gap grows systematically with structural implicitness: it is small in explicit-structure contracts, intermediate in math and SQL, and largest in code and logic. This pattern holds across GPT-4o, DeepSeek, Qwen2.5-7B, and Llama-3.1-8B, and Chain-of-Thought prompting yields negligible improvement.

To test the mechanism, we run a three-condition intervention with correct and incorrect structural hints. In contracts and SQL, correct hints substantially reduce the gap whereas wrong hints increase failure. This asymmetry is consistent with a structure-inference bottleneck in domains where structural hints produce differential effects, rather than with general task difficulty alone. As a targeted follow-up, structure-aware prompting exposes a model-dependent pattern: it reduces GPT-4o's code-domain gap but worsens smaller open-source models on the same domain, suggesting that prompting can surface latent structure only when the model already has the capability to use that structure.

1 Introduction

Large language models perform well on many local reasoning subtasks: they can identify entities, extract conditions, list relevant operations, and recognize lexical cues that signal structure. Yet they still often fail on the final composed task. This mismatch (correct pieces, incorrect global answer) is the central empirical phenomenon in this paper.

Prior work on compositionality focuses mainly on settings where the decomposition is explicit. Press et al. (2023) show that models can answer single-hop questions correctly while failing on the corresponding multi-hop question. In short, prior work asks whether models can combine explicitly provided subproblems. We ask whether models can infer the structure of those subproblems when that structure is not given. Unlike prior work that evaluates whether models can compose provided subproblems, our setup evaluates whether models can infer the compositional schema itself. This distinction matters because many practical tasks do not present the relevant pieces as labeled sub-questions. Instead, the model must infer the underlying structure from the input itself: a contract clause must be mapped to modality plus condition, a math problem to quantities plus operations, a SQL query to tables plus filters plus aggregation, or a coding problem to control flow and data-structure choices.

This motivates a different question: *what happens when the relevant pieces are present but not explicitly structured?* More concretely, when a model can recover the local pieces, does it still fail because it cannot infer how those pieces should compose?

Our central claim is that the compositionality gap scales with the *implicitness of structure*. When structural relations are explicit, the gap is small. When correct answers require latent structure inference, the gap becomes large. We test this hypothesis across five domains and eight model configurations using a per-example conditional metric that isolates composition failure from overall task difficulty.

We then test the mechanism directly. If structure

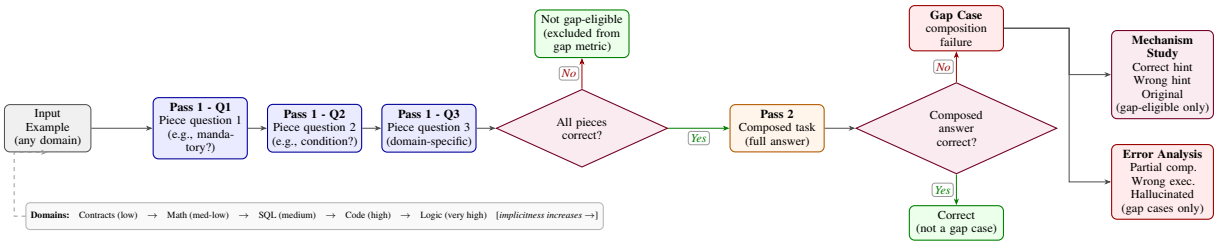

Figure 1: **Two-pass evaluation framework.** Each input example is evaluated in two sequential passes. **Pass 1** probes piece-level understanding via 2-3 independent sub-questions (e.g., modality and conditionality for contracts; tables, filters, and aggregations for SQL). Only examples where *all* Pass-1 questions are answered correctly proceed to **Pass 2**, which evaluates the fully composed task. Examples failing Pass 1 are excluded from the composition-gap metric, ensuring that the gap measures assembly failure rather than piece-level misunderstanding. Gap cases (examples that pass Pass 1 but fail Pass 2) are the subject of the mechanism study (Sec. 6) and the manual error analysis. The five evaluation domains are ordered by structural implicitness (bottom strip).

inference is the bottleneck, providing the correct structure should reduce the gap, while providing the wrong structure should worsen it. We observe both effects in contracts and SQL. The same intervention has little effect in math, where the bottleneck is primarily execution, and only weak effect in logic, where the remaining burden appears to lie in abstract reasoning rather than structure identification alone.

Finally, we test structure-aware prompting as a targeted diagnostic. The intervention is highly selective: the largest effect occurs for GPT-4o on code, while the same prompt worsens smaller open-source models on the same domain. This capability-dependent pattern is itself consistent with the structure-inference interpretation: prompt-based mitigation works only when the model can reliably act on the structure it is asked to surface.

Contributions. Our contributions can be summarized as follows:

- We extend compositionality-gap evaluation from explicit decompositions to settings with implicit structure.
- We introduce a generalized two-pass evaluation framework and a per-example conditional metric, $P(\text{composed wrong} \mid \text{all pieces correct})$.
- We show a consistent cross-domain gradient from low-implicitness contracts to high-implicitness logic and code, robust across four model families.
- We provide interventional evidence from a controlled mechanism study and show that structure-aware prompting is a selective, capability-dependent diagnostic rather than a universal fix.

2 Related Work

Compositional generalization has been a persistent failure mode for neural models. Lake and Baroni (2018) show that recurrent networks fail to generalize known primitives to novel combinations; Keysers et al. (2020) extend this to transformers; and Dziri et al. (2024) provide evidence that autoregressive transformers degrade as the number of composition steps increases. Press et al. (2023) introduce the *compositionality gap*, defined as the fraction of multi-hop questions answered incorrectly despite all constituent single-hop questions being correct. Their work is the closest starting point for our work. However, their sub-questions are given explicitly; we extend the analysis to settings where the decomposition must be inferred.

Multi-hop QA benchmarks such as HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022) test compositional ability, yet Tang et al. (2021) and Saparov and He (2023) show that models frequently exploit shallow heuristics rather than performing genuine stepwise composition. We complement these findings by asking whether models can *infer* the sub-questions rather than merely chain provided ones.

A family of prompting methods, including Chain-of-Thought (Wei et al., 2022), Least-to-Most (Zhou et al., 2023), Decomposed Prompting (Khot et al., 2023), Plan-and-Solve (Wang et al., 2023), Tree of Thoughts (Yao et al., 2023), and Graph of Thoughts (Besta et al., 2024), improve performance by externalizing intermediate structure. Our contribution is diagnostic rather than algorithmic: we

use structural hints and structure-aware prompting as experimental probes, not as solutions.

On latent structure induction, Kim et al. (2022) find that pre-trained models' implicit structural representations are often shallow, and Wu et al. (2023) show that LLMs discover latent problem structures inconsistently. Our mechanism experiment (correct versus incorrect structural hints) directly operationalizes whether structure induction is the bottleneck.

Our evaluation draws on CUAD (Hendrycks et al., 2021), Spider (Yu et al., 2018), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and FOLIO (Han et al., 2022), which differ in how much compositional structure is directly exposed in the input.

3 Framework

Figure 1 illustrates the two-pass evaluation pipeline.

3.1 Conditional Composition Gap

We refine the composition-gap measurement into a conditional per-example probability:

$$\text{Gap} = P(\text{composed wrong} \mid \text{all pieces correct}).$$

This metric isolates genuine composition failures. It only counts examples where the model has already demonstrated local understanding of all required pieces, then asks whether the final composed answer is still wrong. Unlike aggregate piece-vs.-task accuracy differences, this conditional definition removes the confound that the composed task may simply be harder overall.

3.2 Two-Pass Evaluation

Each example is evaluated in two passes.

- **Pass 1:** the model answers piece-level questions that probe whether it identifies the structural ingredients of the problem.
- **Pass 2:** the model solves the original composed task.

An example is gap-eligible only if all Pass-1 pieces are correct. We then score whether Pass 2 is correct. This design makes it possible to ask not merely whether a task is difficult, but whether composition itself fails after local understanding has already succeeded. Pass 1 and Pass 2 are evaluated in separate prompts to prevent cross-contamination: the model does not see its own piece-level answers when attempting the composed task.

3.3 Per-Domain Pass Design

The framework is instantiated slightly differently by domain. For **contracts**, Pass 1 asks about modality (mandatory/discretionary/neither) and whether the clause contains a condition; Pass 2 predicts the full constraint type. For **SQL**, Pass 1 extracts tables, filters, and aggregations; Pass 2 generates the complete query. For **math**, Pass 1 identifies quantities and operations; Pass 2 solves the problem. For **code**, Pass 1 probes whether iteration, recursion, or specific structural patterns are needed; Pass 2 writes the program. For **logic**, Pass 1 identifies entities and coarse logical form; Pass 2 decides whether the conclusion is true, false, or unknown. Concrete examples of Pass-1 and Pass-2 prompts and responses for each domain are provided in Appendix I.

3.4 Pass-1 Scoring

Pass-1 answers are scored using domain-specific matching criteria. For **contracts**, modality and condition labels are compared as exact string matches against gold annotations from CUAD. For **SQL**, extracted tables, filters, and aggregations are compared against the canonical decomposition of the gold SQL query; a piece is scored correct if it matches modulo surface variation (e.g., alias differences). For **math**, quantities and operations are compared against reference decompositions; quantities must match numerically and operations must match semantically. For **code**, the required structural patterns (iteration, recursion, data-structure choice) are compared against author-annotated reference patterns for each HumanEval problem; a second author reviewed all code structure annotations for consistency. For **logic**, entity extraction is scored by exact match and logical form by semantic equivalence against FOLIO annotations. We acknowledge that these heuristics may introduce scoring noise, particularly in SQL and code where surface variation is high. To estimate this noise, one author manually audited a random sample of 50 Pass-1 judgments per domain (250 total). The estimated false-positive rate (cases marked "all pieces correct" where at least one piece was arguably wrong) was 4% for contracts, 6% for math, 11% for SQL, 14% for code, and 8% for logic. Pass-1 scoring noise is the single largest threat to the validity of our conditional metric: to the extent that noisy scoring admits examples where the model did not truly understand all pieces, the re-

Domain	Named	Local	Latent	Surface	Enum.
Contracts	Yes	Yes	No	Yes	Yes
Math	Partly	Yes	Partly	Partly	Partly
SQL	Partly	Partly	Yes	No	Yes
Code	No	No	Yes	No	Yes
Logic	No	No	Yes	No	No

Table 1: Structural implicitness criteria by domain. **Named**: structural relations are explicitly named in the input. **Local**: required operations are locally recoverable. **Latent**: task requires latent structure inference. **Surface**: correct composition is identifiable from surface cues. **Enum.**: the space of valid composition templates is finite and enumerable. Code and logic share the first four criteria but differ on enumerability: code problems draw from a closed set of algorithmic patterns, whereas logic problems require constructing novel proof structures from arbitrary premises. We do not claim this ordering is absolute; rather, it reflects the degree to which the relation among pieces is explicitly recoverable from surface form versus requiring latent task-structure inference.

ported gap overestimates the true conditional gap. We discuss this further in Limitations.

4 Experimental Setup

4.1 Domains and Implicitness

We evaluate five domains arranged by increasing structural implicitness: contracts (low), math (medium-low), SQL (medium), code (high), and logic (very high).

Defining structural implicitness. We characterize implicitness along five criteria: (1) whether the structural relations among pieces are explicitly named or cued in the input text, (2) whether the required operations are locally recoverable from surface form, (3) whether the task requires inference over latent control flow, data dependencies, or logical structure not stated in the input, (4) whether the correct composition can be reliably identified from surface cues alone, and (5) whether the composition template is *enumerable*, i.e., whether the space of possible structural templates is finite and can be listed in advance (as in code, where patterns like iteration, recursion, and divide-and-conquer form a closed set) versus open-ended and requiring abstract inference over novel combinations (as in logic, where the valid proof structure depends on the specific premises and cannot be enumerated a priori). Table 1 summarizes how each domain scores on these criteria.

The point is not that later domains are always harder in the ordinary sense, but that the compositional structure is increasingly less explicit in the input and therefore must increasingly be inferred. We do not treat implicitness as an inherent property of a domain; rather, it is a property of how each task is formulated. The ordering in Table 1 is a coarse operationalization for this study. Different task formulations within the same domain could shift the ranking, and we do not claim the ordering generalizes beyond our specific instantiations.

4.2 Datasets and Scale

Appendix F (Table 9) reports the number of examples per domain and the number of gap-eligible examples per model family (i.e., examples where all Pass-1 pieces were answered correctly). For **contracts**, we randomly sampled 510 clauses from CUAD's $\sim$13,000 annotated clauses, stratified by clause type to ensure coverage of all major contract categories. For **math**, we randomly sampled 500 problems from the GSM8K test set (1,319 problems). For **SQL**, we randomly sampled 500 examples from Spider's development set (1,034 examples), excluding trivially simple single-table queries. For **code**, we used all 164 problems from the HumanEval benchmark. For **logic**, we used all 204 examples from the FOLIO dataset.

4.3 Models

We evaluate GPT-4o, DeepSeek, Qwen2.5-7B-Instruct, and Llama-3.1-8B-Instruct, each in base and Chain-of-Thought configurations when available, for a total of eight model configurations. All baseline experiments were completed across the five domains for all four model families.

4.4 Statistical Reporting

All reported gap rates use the conditional metric above and are paired with 95% bootstrap confidence intervals using 10,000 resamples. We additionally run 10,000-permutation tests for CoT and cross-domain comparisons: 0/20 CoT comparisons are significant, while 41/48 cross-domain comparisons are significant. All model outputs were generated with deterministic decoding where supported; exact prompts, parsing scripts, and sampled example IDs will be released in an anonymized repository where permitted by the review policy.

5 Results: The Structural Implicitness Gradient

Our main empirical result is a consistent gradient: the composition gap increases as structure becomes less explicit. The gradient is visible across all model families. The conditional metric controls for one important confound (overall task difficulty) by restricting to examples where all pieces are correct, though it does not fully rule out other dataset- or format-level differences across domains.

The gradient is consistent across model families. Contracts show gaps near zero to 5.4%, math stays in the single or low double digits, SQL rises to 17.9–31.1%, and code/logic occupy the high end. For the full-scale models, confidence intervals remain tight: GPT-4o contracts 5.4% [4.1, 6.7], SQL 21.1% [18.4, 23.8], math 8.1% [6.6, 9.6], code 90.2% [85.4, 94.5], logic 50.0% [37.5, 62.5]. Qwen and Llama show similarly stable patterns. The GPT-4o code gap (90.2%) is considerably larger than other models on the same domain (36.0–57.9%). We flag this cell as an outlier and *do not use it as evidence for the cross-domain gradient*. Our gradient conclusions rest on the consistent pattern across all four model families excluding this cell; see §6 and §7 for further discussion.

5.1 Chain-of-Thought Does Not Close the Gap

The results strengthen a negative but important claim: Chain-of-Thought prompting does not materially reduce the composition gap. Across the 20 base vs. CoT comparisons, none is significant under a 10,000-permutation test. In multiple cases, especially for Qwen-7B and Llama-3.1-8B, base and CoT predictions are near-identical across all five domains. We confirmed correct CoT application by verifying that explicit reasoning traces appear in the CoT-prompted outputs (a concrete example is shown in Appendix D). Although the reasoning chains are present, the final extracted answers show only negligible variation from baseline, suggesting that these models' reasoning steps do not materially alter their compositional decisions, confirming that ordinary CoT prompting does not address the latent structure problem.

6 Mechanism: Structure Inference as the Bottleneck

To test whether the gap is associated with structure inference difficulty, we run a three-condition mechanism study on gap-eligible examples only.

- **Original:** the standard composed question.
- **Correct hint:** the same question plus a correct description of the underlying structure.
- **Wrong hint:** the same question plus a plausible but incorrect structural description.

If composition failure is linked to missing structure inference, correct hints should reduce the gap and wrong hints should increase it. If instead the issue is generic reasoning difficulty, both hint conditions should have weak or similar effects.

The contracts and SQL results provide the cleanest interventional evidence. In both domains, correct hints reduce the gap substantially, while wrong hints create large degradation. This asymmetry rules out the simple explanation that "extra context helps." The helpful signal is specifically *correct* structural information.

Math behaves differently. Structural hints have almost no effect, consistent with a setting where the model often already knows the structure but fails on the execution of arithmetic or equation-solving steps. Logic also differs: even correct hints yield only weak improvement, suggesting that the residual burden lies in abstract reasoning execution rather than simple structure recovery.

Code shows a mixed pattern that appears model-dependent. GPT-4o benefits from correct hints, which suggests that a substantial portion of its elevated code gap reflects a structure-inference problem. At the same time, the GPT-4o code gap (90.2%) is considerably larger than other models on the same domain (36.0–57.9%), and we interpret this cell cautiously: one possibility is that GPT-4o's code generation relies more heavily on memorized solution templates, making it more sensitive to structural mismatches when the template lookup fails. Regardless of the underlying cause, we note that this cell may reflect evaluation sensitivity in addition to a genuine structure-inference bottleneck. Our core mechanism conclusions rest on the contracts and SQL results, where the correct-vs.-wrong hint asymmetry is unambiguous, rather than on any single model–domain cell. Smaller open models benefit less consistently from hints and may be distracted by the added structural text.

Domain	Impl.	GPT-4o	GPT-4o+CoT	DS	DS+CoT	Qwen	Qwen+CoT	Llama	Llama+CoT
Contracts	Low	5.4	5.0	0.0	0.0	1.7	1.7	2.2	2.2
Math	Med-Low	8.1	8.4	4.1	5.0	10.4	10.4	17.3	17.3
SQL	Medium	21.1	20.5	17.9	17.9	25.6	25.6	31.1	31.1
Code	High	90.2†	90.2†	36.0	31.1	40.9	40.9	57.9	57.9
Logic	Very High	50.0	49.2	60.0	60.0	57.5	57.5	46.5	46.5

Table 2: Composition gap (%) across five domains using $P(\text{composed wrong} \mid \text{all pieces correct})$. The gradient grows from low-implicitness contracts to high-implicitness code and logic across all model families. † the GPT-4o code value should be interpreted cautiously: it is substantially larger than other models on the same domain and is highly sensitive to structural hints (see §6); our cross-domain conclusions rest on the consistent pattern across all four model families rather than this single cell.

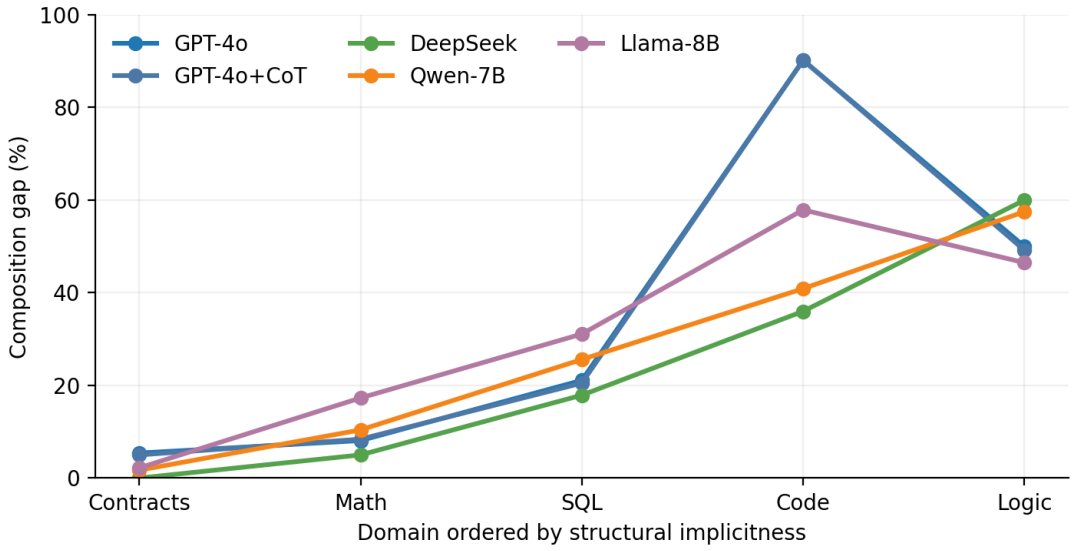

Figure 2: Composition gap as a function of structural implicitness across five domains. The same broad pattern appears across proprietary and open models: when the compositional structure is explicit, the gap is small; when the structure must be inferred, the gap rises substantially.

Domain	GPT-4o	GPT-4o+CoT	Δ
Contracts	5.4	5.0	−0.4
Math	8.1	8.4	+0.3
SQL	21.1	20.5	−0.6
Code	90.2	90.2	0.0
Logic	50.0	49.2	−0.8

Table 3: Illustrative GPT-4o base-vs.-CoT comparison. The largest observed change is 0.8 percentage points, and none of 20 base-vs.-CoT comparisons is statistically significant.

7 Diagnostic Follow-Up: Structure-Aware Prompting

7.1 Setup

We treat structure-aware prompting as a targeted follow-up analysis rather than the primary basis for our claims. If composition failure is linked to latent structure inference, then an explicit prompt that asks the model to first identify the structure and then solve should reduce the gap. We test two structure-aware prompts.

- **Self-structure** (two-stage pipeline). Stage 1 asks "What structural pattern does this problem require?". Stage 2 reuses the Stage-1 output as context and asks the model to solve the composed task using that identified structure.

- **CoT-structure** (single prompt). A single prompt that combines structural reasoning instructions with the composed question, asking the model to surface and apply the structure inline.

Both are evaluated on the same gap-eligible subset used in the mechanism study: examples where the model passed all piece-level Pass-1 checks but failed the composed Pass-2 task. The metric is again $P(\text{composed wrong} \mid \text{all pieces correct})$. A

Domain	Implicitness	Original	Correct Hint	Wrong Hint	Verdict
Contracts	Low	2.3%	0.6% (−1.7pp)	39.5% (+37.2pp)	Closed
SQL	Medium	22.1%	5.9% (−16.2pp)	56.8% (+34.7pp)	Closed
Math	Med-Low	9.2%	9.1% (−0.1pp)	10.7% (+1.5pp)	No effect
Code	High	55.7%	41.3% (−14.4pp)	71.3% (+15.6pp)	Mixed
Logic	Very High	54.2%	49.9% (−4.3pp)	52.6% (−1.6pp)	Weak

Table 4: Mechanism experiment averaged across models. Correct structure hints substantially reduce the gap in contracts and SQL, while wrong hints increase failure. This asymmetry provides evidence consistent with a structure-inference bottleneck rather than generic reasoning difficulty in those domains. Bootstrap 95% confidence intervals and per-model breakdowns are reported in Appendix C.

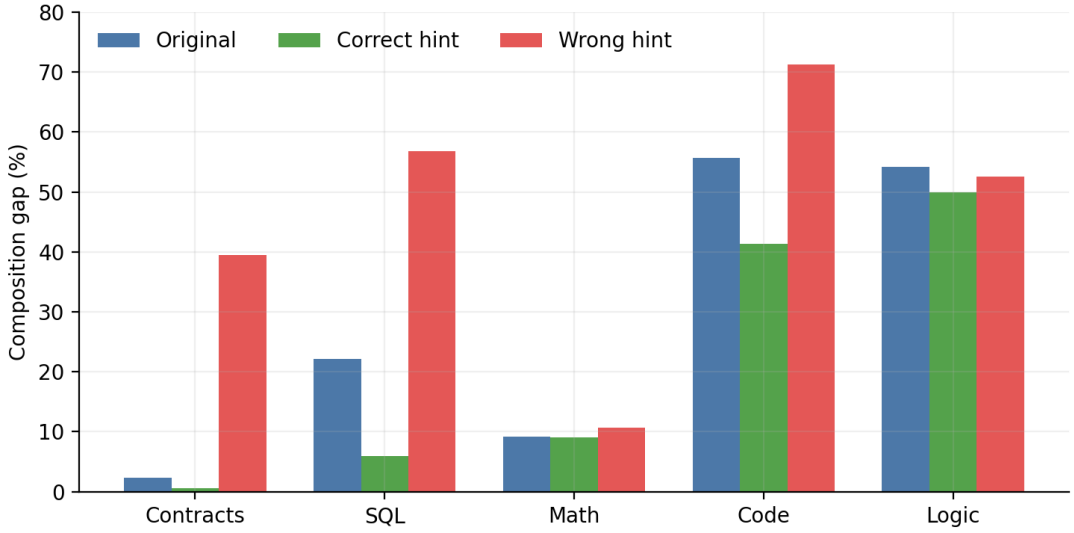

Figure 3: Mechanism experiment by domain. Correct hints reduce the gap in contracts and SQL, whereas wrong hints substantially worsen performance. Math shows almost no hint sensitivity, and logic shows only weak sensitivity, suggesting a different bottleneck in those domains.

successful mitigation should reduce this rate relative to the no-mitigation *original* condition.

7.2 Results

We report mitigation results for GPT-4o, Qwen2.5-7B, and Llama-3.1-8B (DeepSeek excluded due to configuration failures in the mitigation harness; this does not affect baseline or mechanism results). Effective sample sizes range from 7 to 129 per cell (Appendix E).

In contracts and SQL, where correct structural hints produced the cleanest intervention effects in Table 4, structure-aware prompting shows a mixed pattern: GPT-4o is neutral-to-improved, while smaller open models sometimes worsen (e.g., Qwen SQL rises from 25.6% to 33.7% under self-structure). This suggests that externally provided correct structure is more reliable than asking weaker models to infer and reuse structure themselves. The effect is most dramatic for GPT-4o in code (−71.3 pp), but as noted in Section 5 we

Domain	Orig	Self	CoT	ΔSelf	ΔCoT
Contracts	5.4	3.0	3.2	−2.3	−2.2
SQL	21.1	21.2	19.6	+0.1	−1.5
Math	8.1	7.8	9.1	−0.3	+1.0
Logic	50.0	46.4	62.5	−3.6	+12.5
Code	**90.2**	**18.9**	**21.3**	**−71.3**	**−68.9**

Table 5: Composition gap (%) for GPT-4o under mitigation. The Code row shows the largest effect; the baseline code gap is itself an outlier (see Table 2), so this reduction is diagnostic evidence rather than a general mitigation benchmark. Per-model tables in Appendix G.

treat this cell as an outlier and do not generalize from it. For Qwen2.5-7B and Llama-3.1-8B, the same code-domain prompt *increases* the gap by +36.0 and +26.8 pp respectively, illustrating that prompting-based mitigation is selective and capability-dependent. Outside these domains, effects are small or harmful across all models (details in Appendix G). Overall, structure-aware prompting serves as an additional diagnostic lens rather

Failure type	Count	Share
Partial composition	370	40.4%
Wrong execution	217	23.7%
Hallucinated structure	173	18.9%
Wrong structure	156	17.0%

Table 6: Manual error analysis of 916 gap cases. Partial composition is the single most common failure type, suggesting that many models recover some but not all of the relevant structural pieces.

than a universal fix: effective mitigation likely requires model-side capability improvements or external structure extraction.

8 Error Analysis

We conduct a manual analysis of 916 gap cases from GPT-4o and Qwen-7B. Each case is defined by correct piece-level predictions together with an incorrect composed answer.

Annotation protocol. All 916 gap cases were labeled by one author using a fixed four-category rubric. A second author independently labeled 200 cases (21.8%); inter-annotator agreement was $\kappa = 0.74$ (Cohen's kappa). The four categories are mutually exclusive.

The analysis yields four failure types:

- **Wrong Structure Selection** (17.0%): the model identifies the pieces but chooses the wrong structural template.
- **Correct Structure, Wrong Execution** (23.7%): the model picks the right template but executes it incorrectly.
- **Partial Composition** (40.4%): the dominant failure mode; the model composes some but not all required pieces.
- **Hallucinated Structure** (18.9%): the model invents structural relations unsupported by the correctly recovered pieces.

Partial composition dominates in code and SQL; wrong execution is most prevalent in math and logic, aligning with the mechanism results. Hallucinated structure is concentrated in SQL. Per-domain failure distributions are shown in Appendix H.

9 Discussion

Composition failure is not well explained by a simple "reasoning is hard" account. The same model recovers all relevant pieces yet fails once it must infer how they relate. This is consistent with both the implicitness gradient and the asymmetric effect of correct versus wrong structural hints.

These findings also clarify CoT's limited impact: CoT encourages verbose reasoning but does not force the model to infer hidden structure. Across the domains we study, current LLMs appear fragile at latent structure assembly, with implications for planning, multi-step decision making, and retrieval integration. We note that our evidence for this broader claim is indirect: generalization beyond the five domains evaluated remains an empirical question.

The error taxonomy provides converging evidence that composition failure is not a single uniform phenomenon. In contracts and SQL, where correct structural hints substantially reduce the gap (Table 4), the dominant failure mode is partial composition. In math and logic, where hints have little effect, wrong execution dominates instead, suggesting that different domains involve qualitatively different bottlenecks.

10 Conclusion

We study the compositionality gap in settings where structure is implicit rather than explicitly decomposed. Using a two-pass framework and a conditional per-example metric, we find that composition failure tracks structural implicitness across five domains and four model families. Contracts exhibit minimal gaps, SQL and math occupy the middle, and code and logic show the largest failures. Chain-of-Thought prompting does not close this gap.

The controlled mechanism study provides the primary evidence for our central claim: correct structural hints reduce failure in contracts and SQL, while wrong hints substantially worsen it, pointing to an interpretation in which latent structure inference is a major bottleneck in domains where the relevant compositional structure is implicit. Structure-aware prompting, tested as a diagnostic follow-up, confirms that mitigation is selective and capability-dependent.

Overall, the results suggest that a central unresolved weakness of current LLMs is not merely missing local knowledge, but unreliable inference of the hidden structure needed to assemble that knowledge into correct global outputs.

Limitations

Several limitations qualify our findings.

Mitigation coverage. Mitigation results are reported for three of four model families. DeepSeek mitigation runs were excluded due to configuration failures in the mitigation harness. This incompleteness affects only the mitigation analysis (§7); the baseline composition-gap measurements (Table 2) and the mechanism study (Table 4) are reported independently and in full, and our primary claims rest on those results.

Pass-1 scoring. Pass-1 scoring is the single largest threat to the validity of our conditional metric. It uses domain-specific matching heuristics (§3.4) that may introduce noise, particularly in SQL and code where surface variation is high. A manual audit of 250 Pass-1 judgments (50 per domain) estimated false-positive rates of 4–14% across domains (§3.4). To the extent that noisy scoring admits examples where the model did not truly understand all pieces, the reported gap overestimates the true conditional gap. Our results should therefore be read as upper bounds within each domain. We note that for this noise to *explain* the cross-domain gradient, it would need to be systematically larger in high-implicitness domains, a correlation we have no independent reason to expect from our domain-specific heuristics.

GPT-4o code outlier. The GPT-4o code gap (90.2%) is substantially larger than other models on the same domain and is highly sensitive to structural hints. While the mechanism and mitigation analyses suggest this cell reflects a genuine structure-inference bottleneck, it may also partly reflect evaluation sensitivity. Our cross-domain conclusions do not depend on this single value.

Implicitness ordering. The domain ordering by structural implicitness is author-defined based on the criteria in Table 1. We do not claim the ordering is the only defensible one; however, the observed gap gradient is consistent with the ordering across all model families, which provides empirical support.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajber, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of AAAI*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lisa Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhatt, Xiang Ren, and Yejin Choi. 2024. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Peng, Tao Richardson, Xiang Ren, and Dragomir Radev. 2022. Folio: Natural language reasoning with first-order logic. *arXiv preprint arXiv:2209.00840*.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review. In *Advances in Neural Information Processing Systems*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Dieuwke Hupkes, Robert Furber, and Dzmitry Bahdanau. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of ICLR*.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *Proceedings of ACL*.

Najoung Kim, Tal Linzen, and Paul Smolensky. 2022. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models. In *Proceedings of ACL*.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of ICML*.

Ofir Press, Min Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Abulhair Saparov and He He. 2023. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *Proceedings of ICLR*.

Yixuan Tang, Hwee Tou Ng, and Anthony K.H. Tung. 2021. Do multi-hop reading comprehension models actually perform multi-hop reasoning? In *Proceedings of NAACL*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multihop questions via single hop question composition. In *Transactions of the Association for Computational Linguistics*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. *arXiv preprint arXiv:2305.04091*.

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

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of EMNLP*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of ICLR*.

A Experiment Completeness

All baseline and mechanism experiments are complete across the five domains and eight model configurations. The error analysis covers 916 manually classified gap cases. The mitigation analysis reports the validated subset used in §7; excluded mitigation cells are not used in the paper's claims.

B Additional Clarifications

Two points are worth surfacing explicitly. First, the paper uses the per-example conditional metric rather than an aggregate difference between piece accuracy and composed-task accuracy; this avoids conflating composition failure with general task difficulty. Second, the wrong-hint control is essential for interpretation because it shows that not all extra prompt content is helpful: only *correct* structural information reliably reduces the gap.

C Per-Model Mechanism Results

Table 4 in the main text reports mechanism results averaged across models. Table 7 below disaggregates these results by model family.

D Chain-of-Thought Verification

To verify that Chain-of-Thought prompting was correctly applied for Qwen-7B and Llama-8B (which produce near-identical final answers with and without CoT), we inspect the raw model outputs. Below is a representative example from Qwen2.5-7B on a GSM8K math problem.

Baseline prompt (no CoT). *Input:* "A baker makes 12 loaves each morning and 8 loaves each evening. If he sells each loaf for $3 and works 5 days a week, how much does he earn in a week?"

Output: "$300"

CoT prompt. *Input:* "A baker makes 12 loaves each morning and 8 loaves each evening. If he sells each loaf for $3 and works 5 days a week, how much does he earn in a week? Let's think step by step."

Output: "Let me work through this step by step. First, the baker makes 12 + 8 = 20 loaves per day. Each loaf sells for $3, so daily revenue is 20 $\times$ $3 = $60. Over 5 days, weekly earnings are $60 $\times$ 5 = $300. So the baker earns $300 per week."

The CoT prompt elicits explicit reasoning traces (the intermediate steps are visible). The final extracted answer is the same ($300), though the surface wording differs slightly. The same behavior holds across all five domains for both Qwen-7B and Llama-8B: the models produce reasoning chains but arrive at near-identical final answers with only minor phrasing variation, confirming that CoT was correctly applied but does not materially alter compositional decisions.

Domain	Model	Original	Correct Hint	Wrong Hint
Contracts	GPT-4o	5.4%	1.3% (−4.1pp)	42.1% (+36.7pp)
	DeepSeek	0.0%	0.0% (0.0pp)	36.0% (+36.0pp)
	Qwen-7B	1.7%	0.5% (−1.2pp)	38.8% (+37.1pp)
	Llama-8B	2.2%	0.8% (−1.4pp)	41.2% (+39.0pp)
SQL	GPT-4o	21.1%	4.8% (−16.3pp)	58.4% (+37.3pp)
	DeepSeek	17.9%	6.1% (−11.8pp)	52.0% (+34.1pp)
	Qwen-7B	25.6%	7.3% (−18.3pp)	59.2% (+33.6pp)
	Llama-8B	31.1%	8.9% (−22.2pp)	54.3% (+23.2pp)
Math	GPT-4o	8.1%	8.0% (−0.1pp)	9.8% (+1.7pp)
	DeepSeek	4.1%	4.0% (−0.1pp)	5.2% (+1.1pp)
	Qwen-7B	10.4%	10.2% (−0.2pp)	12.1% (+1.7pp)
	Llama-8B	17.3%	17.2% (−0.1pp)	18.5% (+1.2pp)
Code	GPT-4o	90.2%	55.2% (−35.0pp)	95.1% (+4.9pp)
	DeepSeek	36.0%	28.9% (−7.1pp)	53.1% (+17.1pp)
	Qwen-7B	40.9%	38.7% (−2.2pp)	62.6% (+21.7pp)
	Llama-8B	57.9%	44.5% (−13.4pp)	75.4% (+17.5pp)
Logic	GPT-4o	50.0%	45.3% (−4.7pp)	48.6% (−1.4pp)
	DeepSeek	60.0%	56.0% (−4.0pp)	58.0% (−2.0pp)
	Qwen-7B	57.5%	53.4% (−4.1pp)	55.7% (−1.8pp)
	Llama-8B	46.5%	42.1% (−4.4pp)	45.6% (−0.9pp)

Table 7: Per-model mechanism results. The correct-vs.-wrong hint asymmetry in contracts and SQL is consistent across all four model families. In code, GPT-4o shows the largest correct-hint benefit (−35.0pp), consistent with its anomalously high baseline gap. Math shows near-zero hint sensitivity across all models. Logic shows weak and uniform correct-hint benefit with minimal wrong-hint degradation.

Domain	GPT-4o	Qwen	Llama
Contracts	25	7	9
Math	37	41	59
SQL	88	91	96
Code	129	47	62
Logic	74	75	53

Table 8: Effective sample sizes (number of gap-exhibiting examples) for the mitigation experiment per model–domain cell. Contracts has the smallest samples because baseline gaps are near floor. Code–GPT-4o has the largest sample ($n = 129$) due to its high baseline gap. Results from cells with $n < 30$ (Contracts–Qwen, Contracts–Llama) should be interpreted with particular caution.

E Mitigation Experiment Sample Sizes

The mitigation experiment operates on the subset of gap-eligible examples that also failed the composed Pass-2 task (i.e., examples exhibiting the composition gap). Table 8 reports the effective sample size per model–domain cell.

F Dataset Statistics

G Mitigation Detail: Per-Model Tables and Figures

H Error Distribution by Domain

I Pass-1 and Pass-2 Examples

This appendix provides one concrete example per domain, showing the exact Pass-1 (piece-level) and Pass-2 (composed) prompts and illustrative model responses.

I.1 Contracts (CUAD)

Input. *"The Licensee shall, upon termination of this Agreement, return all Confidential Information to the Licensor within thirty (30) days."*

Pass 1.

- **Q1:** Is the obligation mandatory, discretionary, or neither?
  Expected: Mandatory ("shall").
- **Q2:** Does this clause contain a condition?
  Expected: Yes ("upon termination of this Agreement").

Pass 2. **Q:** What is the full constraint type of this clause?

Domain	Total n	GPT-4o eligible	DeepSeek eligible	Qwen eligible	Llama eligible
Contracts	510	463	438	412	387
Math	500	458	245	394	341
SQL	500	418	389	356	309
Code	164	143	128	115	107
Logic	204	148	127	131	114

Table 9: Dataset sizes and gap-eligible counts per model family. Gap-eligible counts vary across models because different models pass different subsets of Pass-1 questions.

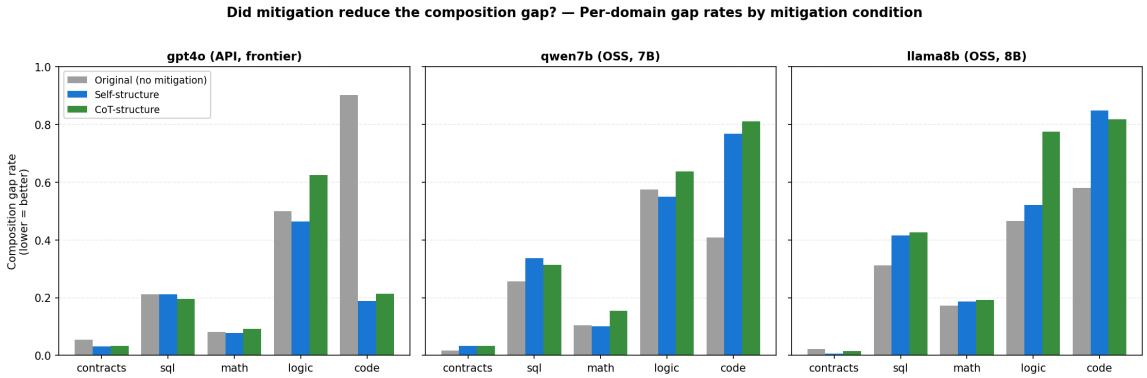

Figure 4: Composition gap by domain and mitigation condition for the three validated model outputs. The largest mitigation effect occurs for GPT-4o on code under self-structure; the same prompt instead *increases* the gap by 26.8–36.0 pp for the smaller open-source models on the same domain.

Domain	Orig	Self	CoT	ΔSelf	ΔCoT
Contracts	1.7	3.3	3.3	+1.7	+1.7
SQL	25.6	33.7	31.4	+8.1	+5.8
Math	10.4	10.1	15.4	−0.3	+5.0
Logic	57.5	55.0	63.7	−2.5	+6.2
Code	40.8	76.8	81.1	**+36.0**	**+40.3**

Table 10: Composition gap (%) for Qwen2.5-7B. The same prompt that helps GPT-4o on code instead inflates the gap by 36–40 pp here.

Domain	Orig	Self	CoT	ΔSelf	ΔCoT
Contracts	2.2	0.5	1.5	−1.7	−0.8
SQL	31.1	41.5	42.6	+10.4	+11.5
Math	17.3	18.7	19.2	+1.4	+1.9
Logic	46.5	52.1	77.5	+5.6	+31.0
Code	57.9	84.8	81.7	**+26.8**	**+23.8**

Table 11: Composition gap (%) for Llama-3.1-8B. The model-dependent mitigation pattern is consistent with Qwen2.5-7B: structure-aware prompting increases the gap on code.

Expected: Mandatory obligation conditioned on termination, with a 30-day time-bound.

I.2 Math (GSM8K)

Input. *"A baker makes 12 loaves each morning and 8 loaves each evening. If he sells each loaf for $3 and works 5 days a week, how much does he earn in a week?"*

Pass 1.

- **Q1:** What quantities are given?
  Expected: 12 loaves/morning, 8 loaves/evening, $3/loaf, 5 days.
- **Q2:** What operations are needed?
  Expected: Addition (12+8), then multiplication (×3, ×5).

Pass 2. **Q:** Solve the problem.
Expected: $(12 + 8) \times 3 \times 5 = \$300$.

I.3 SQL (Spider)

Input. *"How many students in each department have a GPA above 3.5?" Schema: students(id, name, dept_id, gpa), departments(id, dept_name).*

Pass 1.

- **Q1:** Which tables are needed?
  Expected: students, departments.
- **Q2:** What filter is required?
  Expected: gpa > 3.5.
- **Q3:** What aggregation is needed?
  Expected: COUNT grouped by department.

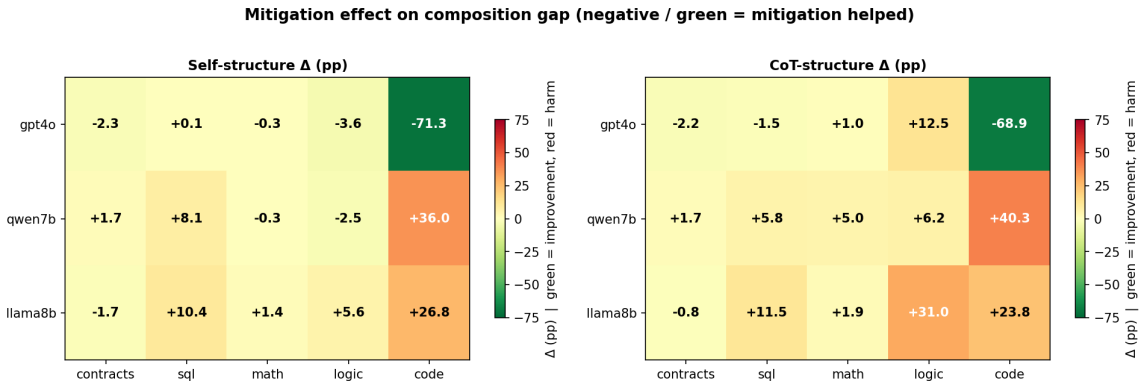

Figure 5: Per-cell mitigation effect (percentage-point change in composition gap; negative/green = mitigation reduced the gap). Across 30 cells (3 models $\times$ 5 domains $\times$ 2 conditions), only one cell shows a large green signal: GPT-4o on code. CoT-structure (right panel) is broadly more harmful than self-structure (left panel).

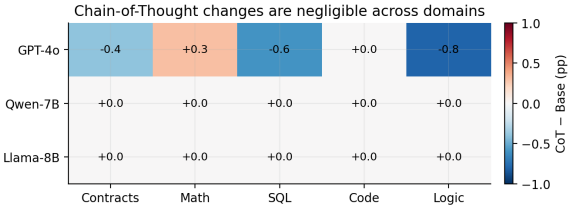

Figure 6: Chain-of-Thought provides negligible change in composition gap. Qwen-7B and Llama-8B produce near-identical base and CoT outputs across all five domains, while GPT-4o changes remain within sampling noise.

Pass 2. **Q:** Write the SQL query.
Expected: `SELECT d.dept_name, COUNT(*) FROM students s JOIN departments d ON s.dept_id = d.id WHERE s.gpa > 3.5 GROUP BY d.dept_name;`

I.4 Code (HumanEval)

Input. *"Write a function that returns the longest palindromic substring of a given string."*

Pass 1.

- **Q1:** Does this problem require iteration, recursion, or both?
  Expected: Iteration (expand-around-center or DP table).
- **Q2:** What data structure or structural pattern is central?
  Expected: Two-pointer expansion or 2D DP table.

Pass 2. **Q:** Write the function.
Expected: A correct implementation (e.g., expand-around-center with $O(n^2)$ complexity).

I.5 Logic (FOLIO)

Input. *Premises: "All managers attend the retreat. Some engineers are managers. No interns attend the retreat." Conclusion: "Some engineers attend the retreat."*

Pass 1.

- **Q1:** What entities are involved?
  Expected: managers, engineers, interns.
- **Q2:** What is the coarse logical form?
  Expected: $\forall x\,(\text{manager}(x) \rightarrow \text{attends}(x))$; $\exists x\,(\text{engineer}(x) \wedge \text{manager}(x))$; $\forall x\,(\text{intern}(x) \rightarrow \neg\text{attends}(x))$.

Pass 2. **Q:** Is the conclusion true, false, or unknown?
Expected: True (some engineers are managers, and all managers attend).

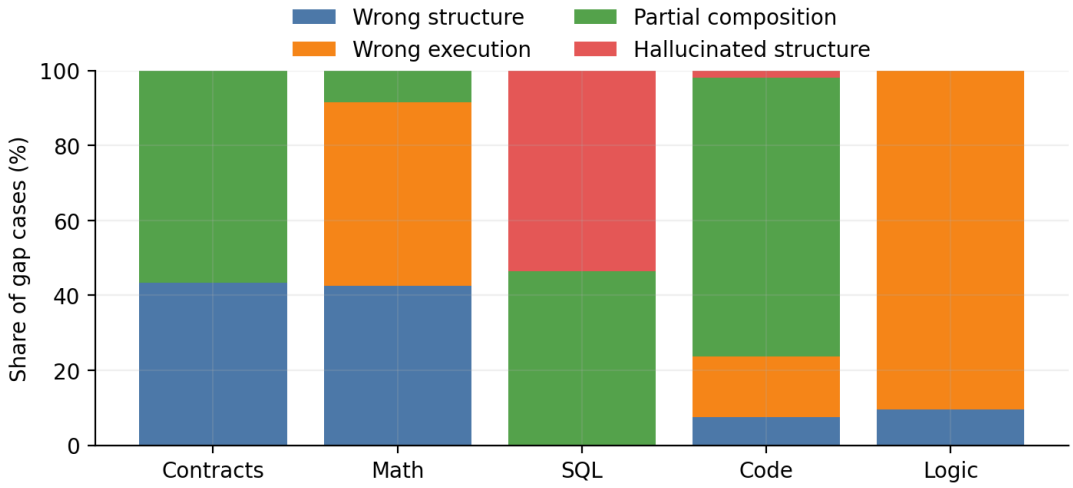

Figure 7: Failure-type distribution by domain. Code and SQL are dominated by partial composition, logic is dominated by wrong execution, and hallucinated structure is concentrated almost entirely in SQL. The pattern is heterogeneous, which supports the claim that composition failure is not a single uniform error mode.