

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

Shravani Hariprasad^{1*}, Mohsin Ali Mohammed^{2*}, Vinija Jain³, Aman Chadha⁴

¹Hatch, Boston, Massachusetts, USA ²Telaeris, Inc., San Diego, California, USA

³Meta, Menlo Park, California, USA ⁴Apple, Cupertino, California, USA

*Equal contribution.

Correspondence: hariprasadshravani@gmail.com

Abstract

Large language models often recover the local pieces of a problem yet still fail to assemble them into a correct global solution. We study this *compositional gap* in settings where the underlying structure is implicit and must be inferred from the input, using a two-pass framework that first verifies piece-level understanding and then measures composition on the same example via $P(\text{composed wrong} \mid \text{all pieces correct})$. We evaluate task formulations in five domains (contracts, math, SQL, code, and logic) and four model families; across the gold-anchored domains, larger conditional gaps are associated with settings where relevant structure is less directly exposed. A controlled intervention with correct and incorrect structural hints indicates heterogeneous bottlenecks: hints substantially reduce the gap in contracts and SQL, have no aggregate effect in math, and have no reliable effect in logic. Structure-aware prompting does not repair the gap and often harms smaller models. We conclude that composition failure is not a single phenomenon, and that diagnosing the dominant bottleneck in each task is the practical output of our framework.

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; that is, prior work asks whether models can combine explicitly

provided subproblems. We ask whether models can infer the structure of those subproblems when it is not given. Many practical tasks do not present the relevant pieces as labeled sub-questions: 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 and apply the way those pieces should compose?

We make two claims of different strengths and keep them separate throughout. The first is descriptive: across our four gold-anchored task formulations, larger conditional gaps are associated with settings where the relevant structure is less directly exposed in the input. This association is correlational, since the domains also differ in dataset, format, output space, and scoring. The second claim is diagnostic, and it carries the main weight of the paper: the *source* of the gap differs by domain. A within-example intervention with correct and incorrect structural hints locates the bottleneck: correct hints substantially reduce the gap in contracts and SQL while wrong hints sharply increase failure there, pointing to structure inference, whereas the same intervention has no aggregate effect in math and no reliable effect in logic, pointing to execution-dominated or mixed bottlenecks.

Structure-aware prompting, tested as a follow-up diagnostic, shows a capability-dependent pattern: asking smaller open models to infer and reuse their own structural analysis often hurts them (most visibly in SQL and logic), while GPT-4o shows only modest effects in either direction; prompting alone does not repair composition failure.

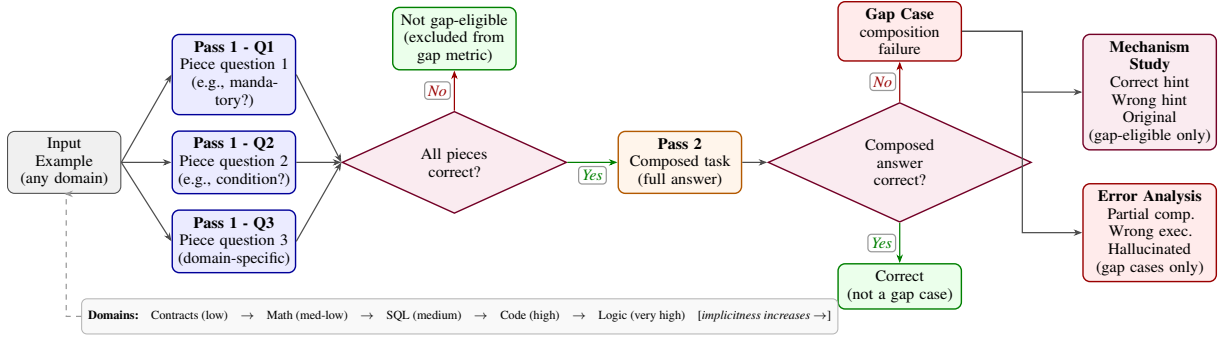


Figure 1: **Two-pass evaluation framework.** Each input example is evaluated in two sequential passes. **Pass 1** probes piece-level understanding via 2–3 independent sub-questions, each asked in its own prompt (e.g., modality and conditionality for contracts; tables, filters, and aggregations for SQL). 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-eligible examples are the subject of the mechanism study (§6); gap cases (examples that pass Pass 1 but fail Pass 2) are the subject of the manual error analysis. The five evaluation domains are ordered by structural implicitness (bottom strip).

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 document a descriptive cross-domain gradient from low-implicitness contracts to high-implicitness logic across the three full-scale model families, with stated caveats about cross-domain comparability.
- We show with a controlled hint study that the bottleneck behind the gap is heterogeneous and domain-specific, and that structure-aware prompting is a capability-dependent diagnostic rather than a 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 on realistic data; and Dziri et al. (2023) 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 ours. 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) show that models answer multi-hop questions without solving their sub-questions, and Saparov and He (2023) show that models take valid individual deduction steps yet fail to plan multi-step proofs. 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 structure-dependent generalization, Kim et al. (2022) find that pretraining exposure inflates apparent compositional generalization, structural generalization remaining poor once exposure is controlled, and Wu et al. (2024) show that performance collapses on counterfactual task variants, indicat-

ing reliance on memorized defaults rather than abstract task structure. Our mechanism experiment (correct versus incorrect structural hints) directly operationalizes whether structure induction is the bottleneck.

Our evaluation draws on Spider (Yu et al., 2018), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), FOLIO (Han et al., 2024), and a contract corpus we assembled in the spirit of CUAD (Hendrycks et al., 2021); these settings differ in how much compositional structure is directly exposed in the input.

3 Framework

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 counts only examples where the model has already demonstrated local understanding of all required pieces, then asks whether the composed answer is still wrong. Unlike aggregate piece-vs.-task accuracy differences, it 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. 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 (Figure 1).

One property of this design deserves emphasis. Pass 1 measures *prompted* recognition: each piece question includes the complete original input and targets one ingredient. The composed task requires *unprompted* deployment of the same information during solving. Our metric therefore captures structure inference and deployment, not composition in isolation, and the hint intervention in §6 probes exactly this distinction by surfacing structure at composition time.

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 constraint type (in the released corpus, mandatory versus discretionary). 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 asks two yes/no questions, whether the best solution needs iteration and whether it needs recursion; 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 G.

3.4 Composed-Task Scoring

The Pass-2 answer is scored against gold with a domain-appropriate criterion, and the criteria are not identical across domains. For **contracts**, the predicted constraint type must match the gold label. For **math**, the extracted final answer must match the gold answer exactly. For **logic**, the predicted true/false/unknown label must match gold. For **SQL**, the generated query is mapped to a structural template (simple, join, nested, set operation, multiple aggregation) by a rule-based classifier, and the prediction is scored correct when its template matches the template of the gold query. This criterion targets the object we study, the structure of the composed answer, but it is stricter than execution accuracy: a semantically equivalent query built on a different template counts as a composition failure. The SQL gap should therefore be read as structural disagreement with the gold composition rather than as query incorrectness, and the hint effects in §6 are internally valid because every condition is scored with the same criterion.

The code protocol. Code is the exception in both passes, and we document it in full because it changes what the code numbers mean. In Pass 1 the model’s structural answers (does this problem need a loop, does it need recursion) were accepted as given rather than scored against gold-derived structure, so all 164 HumanEval problems are gap-eligible for every model. In Pass 2 the generated program is checked for *consistency with the model’s own Pass-1 answers*: code is scored cor-

rect when its control flow matches what the model itself said the problem requires. The code row is therefore a *plan-consistency* measure, not functional correctness, and it is not gold-anchored the way the other four domains are. A model can write a perfectly working program and still be scored as a gap case because the implementation deviates from its stated plan; GPT-4o does this frequently, which is what its 90.2% code cell reflects. We therefore exclude code from every cross-domain claim and from the analyses in §6 and §7; its numbers are reported for completeness with this caveat wherever they appear.

3.5 Pass-1 Scoring

Pass-1 answers are scored with deliberately lenient domain-specific heuristics. For **contracts**, the gold modality and condition answers are derived from the clause’s constraint-type label and compared as exact matches; because every evaluated clause is labeled mandatory or discretionary with no conditional label (the corpus file also holds non-constraint and unlabeled rows, which the runner skips), the gold condition answer is “no” throughout. For **SQL**, extracted tables are correct when at least half of the gold query’s tables are recovered, and filters and aggregations are scored on presence or absence relative to the gold query. For **math**, quantities are correct when at least 70% of the gold numbers are recovered, and operations when at least one gold operation keyword is present. For **logic**, entities are correct at 50% overlap with the gold entity set, and the coarse logical form must match a rule-derived logic type for the example (from the released classifier). (Code Pass-1 answers were accepted as given rather than scored, §3.4; for code, the audit below therefore judges whether the accepted answers were in fact correct.) These heuristics err toward admitting examples. 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 reported gap overestimates the true conditional gap. Appendix B reports a worst-case sensitivity analysis under these audited rates; the

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. We do not claim this ordering is absolute; it reflects the degree to which the relation among pieces is explicitly recoverable from surface form versus requiring latent task-structure inference, for our specific task formulations.

qualitative pattern survives for every model family.

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, whose algorithmic patterns form a closed set) versus open-ended (as in logic, where the valid proof structure depends on the specific premises). 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 must increasingly be inferred. Implicitness is a property of how each task is formulated, not of a domain as such; the ordering in Table 1 is

a coarse operationalization for this study.

4.2 Datasets and Scale

For **contracts**, we evaluate 2,000 clauses from a corpus we assembled from public-sector agreements (aviation, transit, healthcare, and building services). Constraint-type labels come from a combined process: an OpenAI model (GPT-4o) proposed a label and written rationale for each clause, and the authors defined the label scheme and reviewed and revised the proposed labels (see Limitations). The corpus, including every rationale, ships with our release. For **math**, we use the complete GSM8K test set (1,319 problems). For **SQL**, we use the complete Spider development set (1,034 examples). For **code**, we use all 164 problems from HumanEval. For **logic**, we use the 204-example FOLIO validation split.¹ Appendix A (Table 6) reports, for each domain and model, the number of gap-eligible examples, i.e., examples where all Pass-1 pieces were answered correctly.

4.3 Models

We evaluate GPT-4o, DeepSeek, Qwen2.5-7B-Instruct, and Llama-3.1-8B-Instruct (hereafter Qwen-7B and Llama-8B). Each configuration was run twice with identical prompts (§5.1); Table 2 reports the first run. GPT-4o and DeepSeek were accessed through their public APIs in April 2026 under the provider aliases gpt-4o and deepseek-chat; the aliases do not expose snapshot identifiers, a reproducibility limitation we note. The open models are Qwen/Qwen2.5-7B-Instruct and meta-llama/Llama-3.1-8B-Instruct. All outputs were generated with deterministic decoding where supported.

One scale difference matters for interpretation. GPT-4o, Qwen-7B, and Llama-8B were run on the full datasets in every domain. The DeepSeek baseline runs are 50-example pilot subsets in all five domains; its repeat run was full-scale for math and code only (§5.1). We retain DeepSeek for completeness but exclude it from analyses needing stable per-domain estimates (eligible counts in Appendix A).

¹All results reported here were recomputed directly from the per-example logs; some values, including the per-domain sample counts and several mechanism cells, changed from earlier aggregations without changing the qualitative conclusions.

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 run-to-run and cross-domain comparisons. None of the 20 first-run versus repeat-run comparisons (four families by five domains) is significant. Of the 24 cross-domain comparisons among the four gold-anchored domains (four base configurations by six domain pairs), 22 are significant at $p < 0.05$; both non-significant pairs involve DeepSeek’s pilot-scale cells. Exact prompts, parsing and scoring scripts, per-example prediction logs, the error-taxonomy rubric, and the analysis code that regenerates every table derived from those logs are released.² Our primary comparisons are within-model across domains; between-model comparisons are secondary, since eligible subsets differ by model (Appendix C).

5 Results: Gap Sizes Across Task Formulations

Our first, descriptive result is a consistent ordering: under our task formulations, the composition gap increases as structure becomes less explicit, across model families. The conditional metric controls for overall task difficulty by restricting to examples where all pieces are correct, but it does not rule out dataset- or format-level differences, and the composed-task criteria themselves differ by domain (§3.4). We therefore present the gradient as an organizing observation, not a causal finding; the interventional evidence is in §6.

Table 2 and Figure 2 report the gaps. The ordering is consistent across the three full-scale model families. Contracts show gaps between 1.7% and 5.4%, math stays in the single or low double digits, SQL rises to 21.1–31.1%, and logic occupies the high end at 46.5–57.5%. Confidence intervals for GPT-4o are tight where samples are large: contracts 5.4% [4.1, 6.7] ($n=1114$), math 8.1% [6.6, 9.7] ($n=1282$), SQL 21.1% [18.4, 23.8] ($n=887$), logic 50.0% [37.5, 62.5] ($n=56$). Qwen and Llama show similarly stable patterns. Appendix C repeats the comparison on the shared subset of examples that are gap-eligible for all three full-scale families; the low-to-high ordering is unchanged for every family, so eligible-subset composition does

²<https://github.com/mohammedmohsinali/LLMCompositionGap>

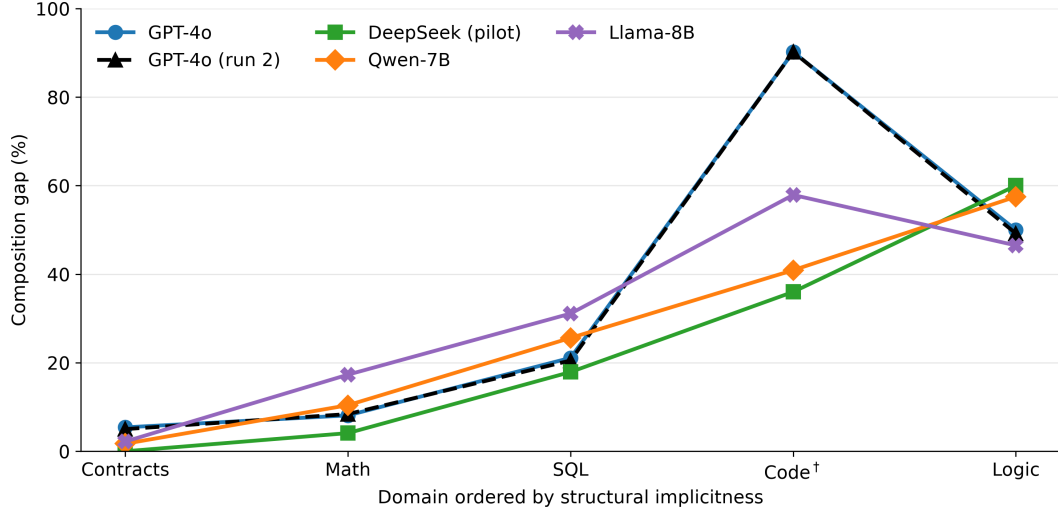


Figure 2: Composition gap as a function of structural implicitness across five domains. The broad pattern is shared by proprietary and open models: when the compositional structure is explicit, the gap is small; when the structure must be inferred, the gap rises substantially. GPT-4o’s two runs (§5.1) are drawn with different colors, line styles, and markers because their values nearly coincide. The code row (†) uses the plan-consistency protocol and is excluded from cross-domain claims; the DeepSeek line is its base configuration, which is pilot-scale ($n=50$ per domain).

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

Table 2: Composition gap (%) across five domains using $P(\text{composed wrong} \mid \text{all pieces correct})$. Gaps grow from low-implicitness contracts to high-implicitness logic across model families. †The code row uses the plan-consistency protocol (§3.4) rather than gold-anchored scoring; it is not comparable to the other rows and is excluded from all cross-domain claims. *DeepSeek cells are 50-example pilot runs in every domain; per-cell eligible counts are in Appendix A.

not drive the pattern.

5.1 Run-to-Run Stability, and a Note on Chain-of-Thought

Each configuration was run twice with identical prompts. The second run was intended as a Chain-of-Thought condition, but while preparing the code release we found that the CoT instruction was never applied (the configuration flag was defined but never read), so the two runs differ only in decoding nondeterminism. We therefore treat these runs only as repeat runs and report no Chain-of-Thought results. The repeat runs remain informative as a stability check: GPT-4o’s gaps reproduce within

0.8pp in every domain (5.4 vs. 5.0, 8.1 vs. 8.4, 21.1 vs. 20.5, 90.2 vs. 90.2, and 50.0 vs. 49.2 for contracts, math, SQL, code, and logic), none of the 20 first-run versus repeat-run comparisons is significant under a 10,000-permutation test, and Qwen-7B and Llama-8B reproduce their parsed predictions exactly under deterministic decoding. For DeepSeek the repeat run was full-scale for math (5.0%, $n=1308$) and code (31.1%, $n=164$), consistent with its pilot values (4.1% and 36.0%).

6 Mechanism: Locating the Bottleneck

To test where structure inference is the bottleneck, 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. Each condition is scored with the same composed-task

criterion as the baseline (§3.4). Code is excluded because its hint conditions were scored against gold-derived structure while its baseline uses plan-consistency (raw numbers in Appendix D).

The contracts and SQL results (Table 3, Figure 3) provide the cleanest interventional evidence, and the asymmetry is consistent within every full-scale family. In contracts, correct hints drive the gap to or near zero for each family (to 2.2% or below), while wrong hints inflate it by +18.7 to +64.5pp. In SQL, correct hints remove most of the gap (−15.7 to −18.8pp) and wrong hints again cause large degradation (+34.1 to +51.8pp). This asymmetry argues strongly against the simple explanation that “extra context helps”: the helpful signal is specifically *correct* structural information.

Math behaves differently. Correct hints move the gap by at most 0.8pp in any family, and in two of three families the movement is slightly upward. This is consistent with a setting where the model often already knows the structure but fails on execution, though the error analysis (§8) complicates this picture (§9). Logic also differs: correct hints produce effects between −6.3 and +2.8pp with no consistent sign, and wrong hints are essentially inert (−1.3 to +1.8pp). Structural misinformation that sharply damages contracts and SQL does almost nothing in logic, which is hard to reconcile with structure recovery as the primary bottleneck there and instead points at downstream reasoning execution.

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 gives the Stage-1 output as context and asks the model to solve the composed task with it.
- **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 full gap-eligible set for each model and domain (the same denominator as Table 2, with counts in Appendix A), so the analysis can detect harm as well as improvement. The metric is again the conditional gap, and a successful mitigation should reduce it relative to the *original* condition.

7.2 Results

We report mitigation results for GPT-4o, Qwen-7B, and Llama-8B; DeepSeek is excluded because of a configuration error in its mitigation execution environment (its baseline and mechanism results are unaffected). Code is excluded for the protocol reason given in §6: its mitigation conditions were scored against gold-derived structure while its baseline uses plan-consistency, so the apparent code effects are movements between two metrics, not treatment effects (raw numbers in Appendix E).

The pattern (Table 4) is capability-dependent, and mostly negative. For GPT-4o the effects are modest in both directions, the largest being CoT-structure *harming* logic by 12.5pp. For the smaller open models, self-derived structure often hurts: Qwen-7B’s SQL gap rises from 25.6% to 33.7% under self-structure, Llama-8B’s from 31.1% to 41.5%, and CoT-structure costs Llama-8B 31.0pp on logic. In the mechanism study, externally supplied *correct* structure removed most of the contracts and SQL gap for these same models; self-derived structure is evidently often wrong or distracting for weaker models. Structure-aware prompting is therefore a diagnostic lens, not a mitigation.

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 rubric is released; the per-example labels are not. Note that 215 of the 916 cases (23.5%) are code cases labeled under the plan-consistency protocol (§3.4), so the aggregate shares

Domain	Implicitness	Original	Correct Hint	Wrong Hint	Verdict
Contracts	Low	3.1%	0.7% (−2.4pp)	47.2% (+44.1pp)	Closed
Math	Med-Low	11.9%	12.3% (+0.4pp)	14.5% (+2.6pp)	No effect
SQL	Medium	25.9%	9.0% (−16.9pp)	69.0% (+43.1pp)	Largely closed
Logic	Very High	51.3%	50.2% (−1.1pp)	51.5% (+0.2pp)	No reliable effect

Table 3: Mechanism experiment, averaged over the three full-scale model families (GPT-4o, Qwen-7B, Llama-8B); per-model results, and DeepSeek’s pilot-scale results, are in Appendix D (Table 9). Correct structure hints substantially reduce the gap in contracts and SQL while wrong hints sharply increase failure there. Math and logic show no comparable sensitivity in either direction. Code is excluded for the protocol reason given in §6.

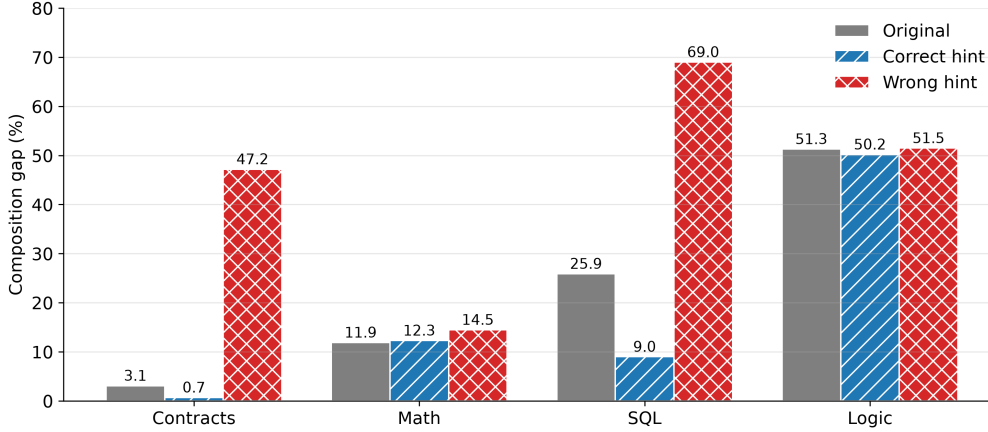


Figure 3: Mechanism experiment by domain, averaged over the three full-scale model families. Correct hints reduce the gap in contracts and SQL, whereas wrong hints substantially worsen performance in those domains. Math and logic show no comparable sensitivity, suggesting a different bottleneck there.

Domain	Orig	Self	CoT	Δ Self	Δ CoT
Contracts	5.4	3.1	3.2	−2.3	−2.2
Math	8.1	7.8	9.1	−0.3	+1.0
SQL	21.1	21.2	19.6	+0.1	−1.5
Logic	50.0	46.4	62.5	−3.6	+12.5

Table 4: Composition gap (%) for GPT-4o under structure-aware prompting, evaluated on the full gap-eligible set per domain. Effects are modest and mixed. Per-model tables for Qwen and Llama, and the excluded code rows, are in Appendix E.

mix gold-anchored and plan-consistency failures; the per-domain view in Appendix F is the safer reading.

The analysis yields four failure types (Table 5), listed by frequency:

- **Partial composition** (40.4%): the dominant failure mode; the model composes some but not all required pieces.
- **Correct structure, wrong execution** (23.7%): the model picks the right template but executes it incorrectly.
- **Hallucinated structure** (18.9%): the model

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 5: Manual error analysis of 916 gap cases. Partial composition is the single most common failure type: models often assemble only some of the pieces they had recovered. The 215 code cases were scored under the plan-consistency protocol (§3.4), so their labels describe plan/implementation mismatches rather than gold-anchored failures.

invents structural relations unsupported by the correctly recovered pieces.

- **Wrong structure selection** (17.0%): the model identifies the pieces but chooses the wrong structural template.

Partial composition dominates in code and contracts, while SQL splits between partial composition and hallucinated structure, the latter concentrated almost entirely there. The per-domain distributions differ in an important way between

math and logic: logic gap cases are almost entirely wrong execution (roughly 90%), whereas math splits nearly evenly between wrong structure (about 42%) and wrong execution (about 48%). Per-domain failure distributions are shown in Appendix F (Figure 6), and Appendix F also gives one worked gap case from the prediction logs for each failure type.

9 Discussion

Composition failure is not a single phenomenon, and it is not well explained by a uniform “reasoning is hard” account. The same model recovers all relevant pieces yet fails once it must infer how they relate, and the source of that failure differs by domain: in contracts and SQL the gap is largely a structure-inference and deployment problem, since supplying correct structure removes most of it and wrong structure is severely damaging, whereas in math and logic the gap survives correct structural information, so the bottleneck lies elsewhere.

Across the domains we study, current LLMs appear fragile in latent structure assembly; whether this extends beyond the five formulations evaluated remains an empirical question.

Reading the mechanism study against the error taxonomy. For logic, the two analyses agree: roughly 90% of logic gap cases are execution failures, and correspondingly neither correct nor wrong structural hints move the logic gap in a reliable direction. Structural misinformation being inert is exactly what an execution-dominated bottleneck predicts. For math, however, there is a genuine tension. About 42% of math gap cases carry a wrong-structure label, so if those labels captured a hint-repairable deficit, correct hints should have visibly reduced the math gap. They did not (at most 0.8pp movement in any family). At least four non-exclusive explanations are compatible with this outcome: the model may not comply with the hint at composition time; the hint may be too coarse, since the math hint restates the quantities and operations already verified in Pass 1 while many wrong-structure cases concern how those operations are composed, such as equation setup or operation order; the structural error may co-occur with a downstream execution failure that the hint cannot repair; or the taxonomy label may be ambiguous for cases at the structure/execution boundary. The granularity explanation seems most plausible given the hint’s construction; distinguishing these accounts

requires joining per-example taxonomy labels with hint outcomes, which awaits joining those labels to the released logs.

The results also connect to what current LLMs learn during pre-training: models acquire substantial local competence (extracting entities, recognizing operations, identifying conditions) without reliably acquiring the ability to infer how those pieces assemble into a global solution. The capability-dependent prompting pattern reinforces this reading, since externally supplied correct structure helps every model in contracts and SQL while self-derived structure helps none reliably and harms the weaker ones.

Two experiments follow directly: a within-domain manipulation that rewrites the same problems into explicit, partially explicit, and implicit variants with the gold decomposition fixed (SQL is the natural first target), which would test the implicitness association without cross-domain confounds; and finer-grained structural hints (compositional templates rather than ingredient restatements), which would test the granularity explanation for math directly.

10 Conclusion

We study the compositionality gap where structure is implicit rather than explicitly decomposed, using a two-pass framework and a conditional per-example metric. Descriptively, larger gaps are associated with task formulations whose structure is less directly exposed, a pattern that holds across the full-scale families and survives worst-case Pass-1 noise corrections. Diagnostically, and more importantly, a controlled hint intervention shows the gap does not have a single source: in contracts and SQL it behaves like a structure-inference and deployment failure, while in math and logic it does not. Structure-aware prompting does not repair it. The framework’s practical output is a diagnosis of which bottleneck dominates a task.

Limitations

Several limitations qualify our findings.

Prompted recognition versus unprompted deployment. Pass 1 measures whether the model can answer targeted piece questions given the full input; the composed task requires deploying the same information without targeted prompting. A model may pass Pass 1 yet fail to recognize the

pieces in context. Our construct is therefore structure inference and deployment, and the residual difficulty of separating deployment failures from general task difficulty remains.

Pass-1 scoring. Pass-1 scoring uses domain-specific matching heuristics (§3.5) that introduce noise, particularly in SQL. The manual audit that produced our false-positive estimates was performed by a single author over 250 judgments, without a second annotator (a second-annotator re-scoring was not completed, and the audit judgments are not released); the sensitivity analysis in Appendix B bounds the impact of the audited rates, but the rates themselves carry sampling and annotator uncertainty. For this noise to explain the cross-domain ordering it would need to be systematically larger in high-implicitness domains beyond what the audit found.

Contract labels are LLM-assisted. The contract corpus labels were proposed by GPT-4o with rationales and then reviewed and revised by the authors (§4.2); the release records the model’s rationales but not the authors’ revisions. Both contract passes score against these labels, so residual label noise propagates into the contracts gap, and because the proposals originated from a model related to GPT-4o, GPT-4o’s contracts results may partly reflect agreement with model-anchored labels (its contracts gap, 5.4%, is in fact the largest of the three full-scale families, so no such advantage is visible in the aggregate, but the concern applies to label quality itself). Contracts is the low-implicitness anchor of the descriptive gradient, so this qualifies that endpoint in particular; the hint asymmetry in contracts is within-model and less exposed to this concern. The corpus, with every rationale, is released so the labels can be audited or replaced.

Composed-task scoring differs by domain. The Pass-2 criterion is gold label match for contracts and logic, exact answer match for math, structural-template match for SQL, and plan-consistency for code (§3.4). The SQL criterion can score semantically valid queries as failures when they use a different template than gold, and the code row is not gold-anchored at all. This heterogeneity is a real limit on cross-domain comparability, which is one reason we restrict causal interpretation to the within-example hint intervention and exclude code from comparative claims. A functional-correctness re-evaluation of the code domain requires regener-

ating model outputs and is left to future work.

DeepSeek scale. DeepSeek baseline results are 50-example pilots in every domain (its contracts cell, for example, has only nine gap-eligible examples), and its mitigation runs failed due to an execution-environment configuration error. DeepSeek is accordingly excluded from averaged analyses and from the mitigation study, and no headline claim rests on it.

Model versioning. The proprietary models were accessed through provider aliases without recorded snapshot identifiers, so exact reproduction of the GPT-4o and DeepSeek runs may not be possible as providers update these aliases. The open models were loaded by repository name without a recorded revision hash.

Implicitness ordering. The domain ordering is author-defined based on the criteria in Table 1. We do not claim it is the only defensible one, and it applies to our specific task formulations rather than to domains in the abstract. A within-domain implicitness manipulation is the right test and has not yet been run.

References

- Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefer. 2024. [Graph of thoughts: Solving elaborate problems with large language models](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.
- 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, Heidy Khlaaf, et al. 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, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.
- Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. [Faith and fate: Limits of transformers on compositionality](#). In *Advances in Neural Information Processing Systems 36*.

- Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhen-ting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. [FOLIO: Natural language reasoning with first-order logic](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. [CUAD: An expert-annotated NLP dataset for legal contract review](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.
- Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. [Measuring compositional generalization: A comprehensive method on realistic data](#). In *Proceedings of the 8th International Conference on Learning Representations*.
- Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. [Decomposed prompting: A modular approach for solving complex tasks](#). In *Proceedings of the 11th International Conference on Learning Representations*.
- Najoung Kim, Tal Linzen, and Paul Smolensky. 2022. Uncontrolled lexical exposure leads to overestimation of compositional generalization in pretrained models. *arXiv preprint arXiv:2212.10769*.
- Brenden M. Lake and Marco Baroni. 2018. [Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks](#). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2873–2882. PMLR.
- Ofir Press, Muru 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*, pages 5687–5711, Singapore. Association for Computational Linguistics.
- Abulhair Saparov and He He. 2023. [Language models are greedy reasoners: A systematic formal analysis of chain-of-thought](#). In *Proceedings of the 11th International Conference on Learning Representations*.
- Yixuan Tang, Hwee Tou Ng, and Anthony K. H. Tung. 2021. [Do multi-hop question answering systems know how to answer the single-hop sub-questions?](#) In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3244–3249, Online. Association for Computational Linguistics.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. [MuSiQue: Multi-hop questions via single-hop question composition](#). *Transactions of the Association for Computational Linguistics*, 10:539–554.
- 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](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634, Toronto, Canada. Association for Computational Linguistics.
- 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 35*.
- Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2024. [Reasoning or reciting? Exploring the capabilities and limitations of language models through counterfactual tasks](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1819–1862, Mexico City, Mexico. Association for Computational Linguistics.
- 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 the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- 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 36*.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, 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 the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.
- 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*

A Dataset Statistics and Experiment Coverage

Table 6 reports the per-domain dataset sizes and the number of gap-eligible examples (all Pass-1 pieces correct) for each base model configuration. These eligible counts are the denominators for every gap rate in the paper, including the mechanism and mitigation experiments.

B Pass-1 False-Positive Sensitivity

Table 7 reports a worst-case correction of every cell of Table 2 under the audited per-domain Pass-1 false-positive rates from §3.5. Let g be the reported conditional gap and f the audited false-positive rate. Assuming every false positive failed Pass 2, the corrected gap is $\max\left(0, \frac{g-f}{1-f}\right)$. The broad low-to-high pattern survives for all four families, and strictly for three; DeepSeek’s contracts and math cells both floor at zero and tie. For Pass-1 noise to erase the pattern it would need to be substantially larger than audited: eliminating Llama-8B’s SQL-versus-math difference, for example, would require a SQL false-positive rate near 22%, twice the audited 11%.

Uniform contamination needs no separate table. For any uniform rate f , the correction above is the same monotone transformation applied to every cell of a configuration, so within-configuration orderings are preserved exactly, except that small gaps floor at zero (at $f=5\%$ this ties DeepSeek’s contracts and math cells at zero; at $f=10\text{--}15\%$ it also floors the contracts cells of the other families). Only domain-specific rates, the audited case shown here, can reorder domains.

C Shared Eligible-Subset Analysis

Because different models are eligible on different examples, cross-model comparisons could in principle reflect the composition of the eligible subsets rather than compositional ability. Our main comparisons are within-model, but as a check, Table 8 recomputes each full-scale family’s gap on the intersection of gap-eligible examples across GPT-4o, Qwen-7B, and Llama-8B (DeepSeek is omitted because the four-way intersection is too small for stable estimates). The low-to-high ordering is unchanged for every family. The shared logic subset

is small ($n=39$) and those values carry wide uncertainty.

D Per-Model Mechanism Results

Table 3 in the main text reports mechanism results averaged over the full-scale families. Table 9 below disaggregates by model, adds DeepSeek’s pilot-scale results, and reports the code rows excluded from the main analysis. All values were recomputed directly from the per-example prediction logs.

E Mitigation Detail: Per-Model Tables and Figures

This appendix reports the structure-aware prompting results for Qwen-7B (Table 10) and Llama-8B (Table 11), the excluded code rows for all three models (Table 12), and per-cell visual summaries (Figures 4 and 5). All conditions are evaluated on the full gap-eligible set per model and domain (Appendix A).

F Error Distribution by Domain and Worked Examples

Figure 6 shows the failure-type distribution by domain for the 916 manually labeled gap cases, and the worked examples below give one real gap case from the prediction logs for each failure type.

Each example below is taken verbatim from the GPT-4o prediction logs. In all four, every Pass-1 piece question was answered correctly.

Wrong structure selection (SQL, Spider).

Question: “What is the name and country of origin of every singer who has a song with the word ‘Hey’ in its title?” The gold query is a simple selection over the singer and song information. The model instead routed the query through concert participation:

```
SELECT DISTINCT S.name, S.country FROM
SINGER S JOIN SINGER_IN_CONCERT SIC ON
S.singer_id = SIC.singer_id JOIN CONCERT
C ON SIC.concert_id = C.concert_id WHERE
EXISTS (SELECT 1 FROM SONG SO WHERE
SO.singer_id = S.singer_id AND SO.title
LIKE '%Hey%');
```

The model identified the correct tables, filters, and aggregations in Pass 1, then selected a composition template that traverses relations the question never asks about.

Domain	Total n	GPT-4o eligible	DeepSeek eligible*	Qwen eligible	Llama eligible
Contracts	2000	1114	9	964	757
Math	1319	1282	49	1261	1295
SQL	1034	887	39	504	366
Code [†]	164	164	50	164	164
Logic	204	56	10	80	71

Table 6: Dataset sizes and gap-eligible counts per base model configuration. Eligible counts vary across models because different models pass different subsets of Pass-1 questions. *DeepSeek was evaluated on 50-example pilot subsets, so its eligible counts are out of 50, not out of the total. [†]Code Pass-1 answers were not scored against gold (§3.4), so all 164 problems are eligible for every full-scale model. Repeat runs (§5.1) have their own eligible sets: GPT-4o 1102/1293/889/164/61 (contracts/math/SQL/code/logic), DeepSeek 8/1308/39/164/10 (full-scale for math and code only); the Qwen-7B and Llama-8B repeat runs equal their first runs exactly.

Domain (f)	GPT-4o	DS*	Qwen-7B	Llama-8B
Contracts (4%)	1.5	0.0	0.0	0.0
Math (6%)	2.2	0.0	4.7	12.0
SQL (11%)	11.3	7.8	16.4	22.6
Code (14%) [†]	88.6	25.6	31.3	51.0
Logic (8%)	45.7	56.5	53.8	41.8

Table 7: Worst-case corrected gaps (%) under the audited per-domain Pass-1 false-positive rates, assuming every false positive failed Pass 2. [†]Plan-consistency protocol, excluded from cross-domain claims. *Pilot scale.

Domain	n_{shared}	GPT-4o	Qwen	Llama
Contracts	635	3.1	0.8	0.6
Math	1215	7.7	10.5	17.4
SQL	297	17.2	25.3	33.3
Code [†]	164	90.2	40.9	57.9
Logic	39	48.7	64.1	51.3

Table 8: Composition gap (%) on the shared subset of examples that are gap-eligible for all three full-scale families. [†]Plan-consistency protocol; code is shared trivially because all 164 examples are eligible for every model.

Correct structure, wrong execution (Math, GSM8K). *Question:* “Carlos is planting a lemon tree. The tree will cost \$90 to plant. Each year it will grow 7 lemons, which he can sell for \$1.50 each. It costs \$3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?” The model identified all quantities and operations correctly in Pass 1 and set up the right computation (net \$7.50 per year against a \$90 cost), but answered 12. At year 12 Carlos has only broken even ($12 \times 7.5 = 90$); earning money starts in year 13. The structure is right and a boundary step in the execution is wrong.

Partial composition (SQL, Spider). *Question:* “What is the average earnings of poker players with height higher than 200?” Height is stored in the people table, so the gold query joins poker players

to people. The model composed only part of the required structure:

```
SELECT      AVG(earnings)      AS
average_earnings FROM    POKER_PLAYER
WHERE height > 200;
```

It named both tables in Pass 1, then dropped the join and filtered on a column the poker-player table does not contain.

Hallucinated structure (SQL, Spider). *Question:* “What is the smallest weight of the car produced with 8 cylinders on 1974?” The gold query is a single-table aggregation over the car data. The model invented a relation to the model list:

```
SELECT      MIN(cd.weight)      AS
smallest_weight FROM    CARS_DATA cd
JOIN MODEL_LIST ml ON cd.model_id =
ml.model_id WHERE cd.cylinders = 8 AND
cd.year = 1974;
```

The join to MODEL_LIST is supported by nothing in the question; the model added a structural relation beyond the pieces it had itself recovered.

G Pass-1 and Pass-2 Examples

This appendix provides one example per domain, showing the Pass-1 (piece-level) and Pass-2 (composed) question formats and the expected answers, complementing the scoring definitions in §3.4. The inputs below are illustrative (constructed) rather than logged instances; Appendix F gives logged cases.

Domain	Model	n	Original	Correct Hint	Wrong Hint
Contracts	GPT-4o	1114	5.4%	2.2% (−3.2pp)	24.1% (+18.7pp)
	DeepSeek*	9	0.0%	0.0% (0.0pp)	33.3% (+33.3pp)
	Qwen-7B	964	1.7%	0.0% (−1.7pp)	66.2% (+64.5pp)
	Llama-8B	757	2.2%	0.0% (−2.2pp)	51.4% (+49.2pp)
Math	GPT-4o	1282	8.1%	8.6% (+0.5pp)	11.8% (+3.7pp)
	DeepSeek*	49	4.1%	2.0% (−2.1pp)	8.2% (+4.1pp)
	Qwen-7B	1261	10.4%	10.2% (−0.2pp)	10.4% (0.0pp)
	Llama-8B	1295	17.3%	18.1% (+0.8pp)	21.3% (+4.0pp)
SQL	GPT-4o	887	21.1%	4.8% (−16.3pp)	72.9% (+51.8pp)
	DeepSeek*	39	17.9%	0.0% (−17.9pp)	33.3% (+15.4pp)
	Qwen-7B	504	25.6%	9.9% (−15.7pp)	59.7% (+34.1pp)
	Llama-8B	366	31.1%	12.3% (−18.8pp)	74.3% (+43.2pp)
Logic	GPT-4o	56	50.0%	50.0% (0.0pp)	51.8% (+1.8pp)
	DeepSeek*	10	60.0%	50.0% (−10.0pp)	60.0% (0.0pp)
	Qwen-7B	80	57.5%	51.2% (−6.3pp)	56.2% (−1.3pp)
	Llama-8B	71	46.5%	49.3% (+2.8pp)	46.5% (0.0pp)
Code [†]	GPT-4o	164	90.2%	12.8%	37.8%
	DeepSeek*	50	36.0%	72.0%	88.0%
	Qwen-7B	164	40.9%	57.3%	84.8%
	Llama-8B	164	57.9%	50.6%	90.2%

Table 9: Per-model mechanism results, recomputed from the per-example prediction logs. The correct-vs.-wrong hint asymmetry in contracts and SQL is consistent across families; math and logic show no comparable sensitivity. *Pilot scale ($n \leq 50$ evaluated); DeepSeek deltas on these cells correspond to at most seven examples and should not be interpreted. [†]The code hint conditions were scored against gold-derived structure while the code original condition uses the plan-consistency protocol (§3.4); the code rows therefore compare two different metrics, do not measure a hint effect, and are excluded from all analysis. No deltas are shown for them for this reason.

Domain	Orig	Self	CoT	Δ Self	Δ CoT
Contracts	1.7	3.3	3.3	+1.6	+1.6
Math	10.4	10.1	15.4	−0.3	+5.0
SQL	25.6	33.7	31.3	+8.1	+5.7
Logic	57.5	55.0	63.7	−2.5	+6.2

Table 10: Composition gap (%) for Qwen-7B under structure-aware prompting. Self-derived structure hurts SQL substantially and helps nowhere meaningfully.

Domain	Orig	Self	CoT	Δ Self	Δ CoT
Contracts	2.2	0.5	1.5	−1.7	−0.7
Math	17.3	18.7	19.2	+1.4	+1.9
SQL	31.1	41.5	42.6	+10.4	+11.5
Logic	46.5	52.1	77.5	+5.6	+31.0

Table 11: Composition gap (%) for Llama-8B under structure-aware prompting. The harm pattern matches Qwen’s: SQL and logic get substantially worse when the model must derive and reuse its own structure.

G.1 Contracts (authors’ corpus)

Input. “*The Contractor shall submit a written progress report to the Authority each month.*”

Pass 1.

- **Q1:** Is the obligation mandatory, discretionary, or neither?

Model	Orig (plan-cons.)	Self	CoT
GPT-4o	90.2	18.9	21.3
Qwen-7B	40.9	76.8	81.1
Llama-8B	57.9	84.8	81.7

Table 12: Code-domain mitigation values, reported for completeness only. The original condition uses the plan-consistency protocol while the mitigation conditions were scored against gold-derived structure (§3.4), so these columns are on different metrics and the apparent swings (including GPT-4o’s large drop) are not treatment effects. No deltas are reported and no conclusions are drawn from this table.

Expected: Mandatory (“shall”).

- **Q2:** Does this clause contain a condition?
Expected: No.

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

Expected: HARD (a mandatory, unconditioned obligation). Every evaluated clause is HARD (mandatory) or SOFT (discretionary) (§3.5).

G.2 Math (GSM8K)

Input. “*A baker makes 12 loaves each morning and 8 loaves each evening. If he sells each loaf for*

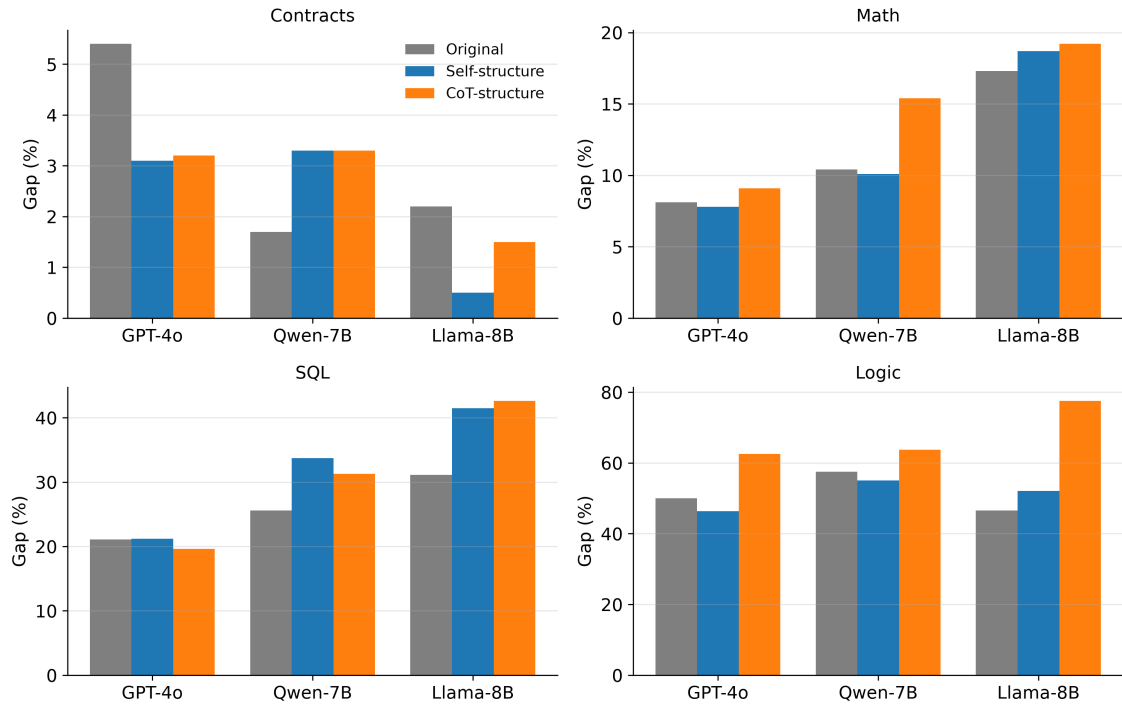


Figure 4: Composition gap by domain and mitigation condition for the three evaluated models, on the four gold-anchored domains. Structure-aware prompting has only modest effects for GPT-4o and is frequently harmful for the smaller open models, most visibly on SQL and logic.

\$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 ($\times 3$, $\times 5$).

Pass 2. Q: Solve the problem.

Expected: $(12 + 8) \times 3 \times 5 = \$300$.

G.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.

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;`
Scoring compares the structural template of the generated query (here, a join with grouped aggregation) to the gold query’s template (§3.4).

G.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?
Model answer (accepted as given, not gold-scored): Yes.
- **Q2:** Does this problem require recursion?
Model answer (accepted as given): No.

Pass 2. Q: Write the function.

Scored as: whether the generated code’s control flow (loop and recursion usage) matches the model’s own Pass-1 answers, i.e., plan-consistency, not functional correctness (§3.4).

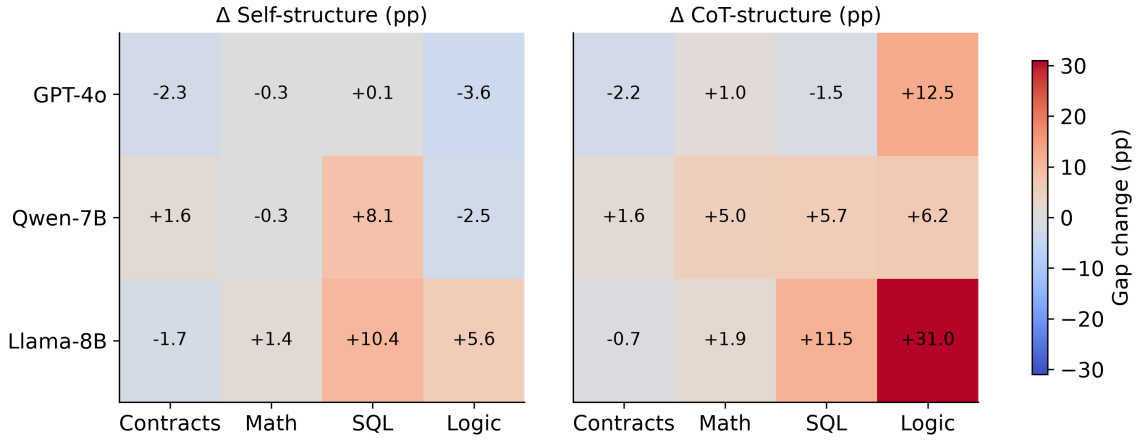


Figure 5: Per-cell mitigation effect (percentage-point change in composition gap; blue = reduced gap, red = increased gap) across 24 cells (3 models $\times$ 4 domains $\times$ 2 conditions). CoT-structure (right panel) is broadly more harmful than self-structure (left panel), with the largest single harm being CoT-structure on Llama logic (+31.0pp).

G.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).

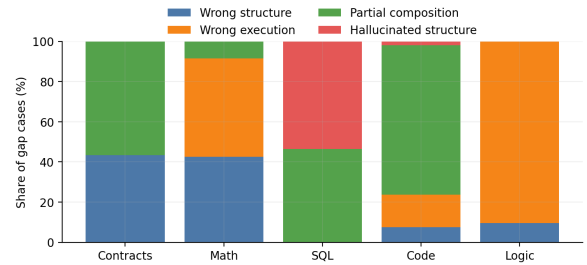


Figure 6: Failure-type distribution by domain. Code and contracts are dominated by partial composition, SQL splits between partial composition and hallucinated structure (the latter concentrated almost entirely in SQL), logic is dominated by wrong execution (roughly 90%), and math splits nearly evenly between wrong structure and wrong execution. The pattern is heterogeneous, which supports the claim that composition failure is not a single uniform error mode. Code cases are plan-consistency failures (§3.4).