When Atomic Constraint Application Does Not Compose: Diagnosing Constraint Composition Failure in LLMs

Anonymous authors
Paper under double-blind review

Abstract

Recall does not establish that a model can apply a constraint. We define **application-conditioned residual Constraint Composition Failure (CCF)**: failure to construct a feasible joint assignment after exact rule representation and isolated application have been demonstrated. Our gate exhaustively tests active operator inputs using the joint task's canonical context and complete-assignment interface.

In 50-seed Matched Constraint Ring confirmations, residual failure at L2/L4 is .740/.935 for Qwen3.7-Max and .762/.867 for SenseNova-6.7-Flash-Lite. On seeds admitted at both lengths, Qwen increases by 21.7 points (paired 95% CI [6.5,37.0], $p = .0129$), whereas SenseNova's 7.9-point increase is inconclusive ([-7.9,23.7], $p = .5078$). A matched final-only repair instruction yields no reliable reduction. Repository-grounded evidence is model–family specific: DeepSeek-V4-Flash fails on 17/32 Updo and 4/24 Pebble control-success strict runs, whereas a jointly frozen Qwen extension succeeds on all 50 Updo runs but fails all 37 eligible Pebble runs. Deadlock and CodePlan audits delimit adjacent application, evaluation, and ordinary reliability failures. CCF is therefore a controlled diagnosis of an atomic-to-joint capability gap, not a universal depth law, model-wide deficit, software-task prevalence estimate, or renamed combinatorial explosion.

1 Introduction

Large language models (LLMs) increasingly generate plans and programs whose validity depends on interacting constraints. Their failures are often attributed to forgetting or instruction count, yet a model can restate a rule and still violate it (Kruthof, 2026). A stricter diagnosis must distinguish unavailable rules, failed atomic application, and failed coordination of applicable rules.

We study the third possibility by separating *rule representation*, *isolated application*, and *joint composition*. An invalid joint output after the relevant atomic capabilities have been established is **Constraint Composition Failure (CCF)**. CCF applies only to feasible, interacting systems; it is not a label for every violation or a fixed-depth law.

We ask four questions: **RQ1**, does joint failure remain after context- and interface-matched atomic application succeeds? **RQ2**, is that residual associated with a longer tested composition, and does the association replicate? **RQ3**, does the gap also occur in repository-grounded component orchestration? **RQ4**, can a single-call self-repair instruction reliably close it?

Our main testbed is **Matched Constraint Ring**. Seeded lookup operators over opaque symbols form a closed ring with one feasible assignment. Gate 1 checks exact representation; an application gate exhaustively checks every active operator–input pair without exposing the solution. Each isolated call uses the joint call's canonical context and complete-assignment contract. Only instances passing both gates enter the residual denominator.

Separately registered 50-seed confirmations replicate residual CCF in Qwen3.7-Max and SenseNova-6.7-Flash-Lite. Qwen shows a significant L2-to-L4 increase; SenseNova's increase is

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

inconclusive. Because constraints, decisions, gate calls, and prompt length co-vary, length is a composition dose, not a topology-only intervention, and differing within-model significance does not establish between-model heterogeneity.

We then reuse the frozen gates and Direct outcomes for one new, final-interface-matched propose–check–revise call per instance. Rescues coexist with regressions and no paired interval excludes zero. This supports only the narrow claim that final-only self-checking is not reliable repair.

Two separately frozen code families test the same atomic-to-joint distinction with executable repository-grounded components. DeepSeek-V4-Flash exhibits residual failures in both Updo (17/32) and Pebble (4/24). A jointly frozen Qwen3.7-Max extension supplies an informative contrast: Updo has no residual failure in 50 eligible runs, whereas Pebble fails in all 37 eligible runs. The positive and null families are reported together. These are repeated generations from fixed task families, not prevalence estimates over software tasks.

DeadlockGate, CodePlan, and PatchGate serve as boundary audits. They show, respectively, an application-to-evaluation bottleneck, no post-application residual event, and a case where verified helpers remove the free-composition gap. These negative boundaries prevent ordinary coding difficulty from being relabeled as CCF.

The paper makes three contributions:

1. We formalize the atomic-to-joint capability gap with matched residual estimands for exact assignments and paired executable component orchestration.
2. We adapt capability-gated evaluation (Yu et al., 2026) to interacting constraints with an exhaustive context/interface-matched gate and frozen two-model confirmations.
3. We test a matched self-repair intervention, characterize model–family-specific residual and null boundaries in two executable repository-grounded families, and use negative code audits to delimit adjacent bottlenecks.

2 Related Work

2.1 Forgetting and Constraint Following

One common explanation for complex LLM failures is that models lose access to relevant information as context length or task complexity increases. Prior work shows that language models underuse information placed in the middle of long contexts (Liu et al., 2024) and that increasing input length can reduce reasoning performance even when the task itself is unchanged (Levy et al., 2024). A robot-programming symposium study similarly attributes some execution errors to models being "forgetful" of key prompt information (Chen & Huang, 2024). FollowBench observes that instruction-following accuracy declines as constraints become more numerous and fine-grained (Jiang et al., 2024). IFEval provides reproducible checks for objectively verifiable instructions (Zhou et al., 2023), while MulDimIF organizes code-verifiable constraints across patterns, categories, and difficulty levels (Ye et al., 2026).

DRIFTBENCH directly documents a recall–adherence dissociation under multi-turn ideation pressure: models may restate initial constraints while producing proposals that violate them (Kruthof, 2026). CCF addresses a complementary setting without intentionally competing follow-up instructions: constraint interactions are controlled and failure is scored against an exact joint feasible set. Table 1 summarizes the positioning.

Two concurrent preprints sharpen, but do not anchor, our positioning. Composition Collapse filters factual chains through atomic-stability and sub-question gates (Yu et al., 2026); CombEval dynamically generates solver-verified combinatorial counting problems (Zhou et al., 2026). CCF neither claims priority over their composition observations nor evaluates counting: it diagnoses exact feasible-assignment failures under shared constraints and sequential commitments. Our core claims rely on our experiments and established literature; these preprints are cited as adjacent work.

Our gate hierarchy explicitly adopts the capability-conditioned principle exemplified by Composition Collapse: a joint failure is diagnostically meaningful only after the relevant atomic capabilities have been established. The contribution is therefore not gating per se. We adapt it to provided, in-

Table 1: CCF compared with related concepts. It diagnoses exact joint failures after demonstrated atomic application.

Concept	Main focus	Relation to CCF
Compositionality gap (Press et al., 2023)	Single-hop competence does not ensure multi-hop answers.	CCF is diagnostic rather than only descriptive: it measures representation and application, then conditions joint failure on the corresponding gate.
FollowBench (Jiang et al., 2024)	Instruction following degrades as constraints accumulate.	CCF separates access, isolated application, and exact joint satisfaction.
DRIFTBENCH (Kruthof, 2026)	Multi-turn pressure yields recall–adherence dissociation.	CCF controls constraint interaction and exact joint feasibility without competing follow-up instructions.
Attention Satisfies (Yuksekgonul et al., 2024)	Factual errors through a CSP and attention lens.	CCF extends the CSP view to multi-decision outputs where token access need not imply feasible joint assignments.
Composition Collapse (Yu et al., 2026)	Double-gates stable facts and isolated sub-question accuracy before scoring factual-chain composition.	CCF studies provided interacting constraints, exact feasible assignments, and sequential interventions.

teracting constraints by separating exact operator representation from isolated application, matching context and output interfaces in the strict control, scoring against an exact feasible set, and locating prefix collapse. These operations distinguish CCF from factual-chain gating.

2.2 Compositional Reasoning and Constraint Satisfaction

LLMs often struggle with systematic composition: they may acquire local rules or facts yet fail when those elements must be combined in new or adversarial configurations. Faith and Fate formalizes multi-step tasks as computation graphs and finds rapid degradation as compositional complexity grows (Dziri et al., 2023). Studies of compositional relation reasoning (Zhao & Zhang, 2024) and mathematical trap problems (Zhao et al., 2024) likewise show that component knowledge does not guarantee successful composition.

Constraint-oriented benchmarks make this issue more explicit. COLLIE systematically constructs constrained text generation tasks with compositional structure (Yao et al., 2024), while Attention Satisfies casts factual generation as a constraint satisfaction problem (Yuksekgonul et al., 2024). UltraBench reports that high per-constraint accuracy can coexist with poor all-constraint satisfaction under dense multi-attribute instructions (Yun et al., 2025). These benchmarks establish aggregate constraint-following gaps; CCF studies a stricter diagnostic signature by conditioning joint failure on demonstrated atomic application.

2.3 Repository-Level Code Generation

Repository-level coding benchmarks such as SWE-bench evaluate whether a model can edit an existing codebase to resolve a real issue (Jimenez et al., 2024). End-to-end patch success is important but aggregates repository navigation, specification interpretation, transport, build, regression preservation, atomic implementation, and cross-component coordination. Our coding evidence asks a narrower diagnostic question. It freezes model-generated components only after executable atomic tests, then compares matched coordinators with and without cross-component state dependence. This does not replace end-to-end software benchmarks; it isolates where an otherwise valid component set ceases to compose.

2.4 Search, State Maintenance, and Planning

Standard autoregressive decoding follows a single left-to-right path, which gives it limited ability to revise early decisions. Tree-of-Thoughts explicitly replaces single-path generation with branching, evaluation, and backtracking over intermediate states (Yao et al., 2023).

Realistic planning benchmarks expose a related gap. TravelPlanner shows that language agents struggle to satisfy multiple travel constraints jointly (Xie et al., 2024), and a long-horizon planning study reports that models struggle to analyze plans and provide accurate feedback for refinement (Chen et al., 2024). These results support the motivation for CCF: the bottleneck is not merely local parsing of constraints, but maintaining global consistency across a growing decision trajectory. Additional theoretical and architectural context is provided in Appendix G.

3 Diagnostic Framework

3.1 Feasibility and Residual CCF

Given input X, interacting constraints $C = \{c_1, \ldots, c_n\}$, and output space $\mathcal{Y}(X)$, define

$$\mathcal{F}(X, C) = \{Y \in \mathcal{Y}(X) : Y \models C\}.$$

We study feasible tasks, $\mathcal{F} \neq \varnothing$, in which at least two constraints share a variable or dependency, $\mathcal{I}(C) = 1$. Recall- conditioned probes motivate the problem, but recall does not establish rule use.

3.2 Application and Strict Matched Gates

Following capability-conditioned evaluation (Yu et al., 2026), let

$$G_1 = \{\text{all active rules are represented exactly}\},$$
$$G_2 = \{\text{all registered isolated applications are correct}\}.$$

The application gate is $G_{\text{app}} = G_1 \cap G_2$. A stricter matched control also requires M: isolated and joint stages receive the same canonical context and output contract. Thus $G_{\text{strict}} = G_{\text{app}} \cap M$, and

$$\mathcal{E}_{\text{RCCF}} = G_{\text{strict}} \cap \{\mathcal{I}(C) = 1\} \cap \{\mathcal{F} \neq \varnothing\} \cap \{Y \notin \mathcal{F}\}.$$

Rates condition on infrastructure-eligible admitted instances; malformed completed outputs count as failures. The matched Ring gate exhausts every active source value using the joint assignment interface. Boundary tasks use separately defined application gates and are not pooled with this estimand.

Identification scope. This estimand identifies a behavioral conditional failure rate under the registered Ring distribution: exact table recovery and every registered one-step application have succeeded, yet the complete joint assignment fails. It does not identify the causal excess due to interaction relative to an equally specified non-interactive task, nor does it separate start selection, multi-step propagation, state maintenance, closure, decoding order, or an internal representation as mechanisms. Those operations remain part of the measured joint-construction gap.

Figure 1 summarizes the diagnostic sequence and the separation between admission and joint construction.

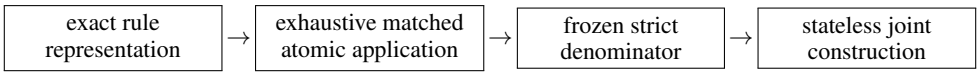

Figure 1: Matched residual diagnosis. Gate outputs determine admission but are never exposed to the joint call; a completed invalid joint output after the strict gate is residual CCF.

3.3 Paired Repository-Grounded Composition

Executable code introduces transport, build, and regression failure modes that are absent from exact assignment tasks. Let G_{atomic} denote that every generated component passes its registered executable tests, G_{id} exact source-identity preservation, and G_{pair} matched coordinator interfaces and non-joint checks. For an independent control outcome J_{ctrl} and an interacting outcome J_{int}, define

$$G_{\text{code}} = G_{\text{atomic}} \cap G_{\text{id}} \cap G_{\text{pair}} \cap J_{\text{ctrl}}, \qquad \mathcal{E}^{\text{code}}_{\text{RCCF}} = G_{\text{code}} \cap \neg J_{\text{int}}.$$

Conditioning on control success excludes ordinary coordinator and interface failure; changing only cross-component dependence isolates orchestration more narrowly. Because repeated runs may share a task family or component bundle, this estimand does not by itself support independent-task prevalence.

3.4 Sequential Collapse

For an autoregressive commitment $Y_{1:t}$, define

$$\mathcal{F}_t(Y_{1:t}) = \{Y' \in \mathcal{F}(X, C) : Y'_{1:t} = Y_{1:t}\}.$$

The **collapse step** is the first t for which $|\mathcal{F}_{t-1}| > 0$ but $|\mathcal{F}_t| = 0$. It localizes when an emitted prefix eliminates every joint solution; it does not claim that the model internally represents $\mathcal{F}_t$.

3.5 Metrics

We report gate rates, Wilson 95% intervals, paired bootstrap intervals, and exact McNemar tests for independent Ring instances. Length analyses use only seeds admitted at both lengths. The repair condition exposes no reasoning trace and estimates instruction effectiveness, not reasoning faithfulness. Fixed-family code repetitions are reported as counts without confidence intervals or population-prevalence claims.

4 Experimental Design

4.1 Matched Constraint Ring

Each seed generates sixteen deterministic non-identity permutations over five opaque symbols. For length L, the first L operators form a closed ring,

$$X_1 \xrightarrow{O_1} X_2 \xrightarrow{O_2} \cdots \xrightarrow{O_{L-1}} X_L \xrightarrow{O_L} X_1.$$

A rejection sampler retains inventories with one fixed point, hence one feasible assignment; L2 and L4 share a nested inventory per seed.

This unique-solution construction yields two analytic calibrations without model calls. Uniformly sampling a complete assignment succeeds with probability 5^{-L} (.040 at L2 and .0016 at L4). A stronger policy that samples X_1 uniformly and then propagates every operator exactly succeeds with probability $1/5 = .20$, because exactly one start closes the ring. These oracle-defined policies are descriptive baselines, not matched non-interactive controls or estimands of an interaction effect.

Gate 1 uses three paraphrases and requires exact recovery of every active five-entry lookup table. The **matched application gate** tests all $5L$ active operator–input pairs. Every isolated call receives the joint variables, options, active tables, constraints, and complete-assignment JSON contract. One source is pinned; its target must be computed while fixed fillers remain unchanged. Gate outputs are never shown to the stateless joint call.

4.2 Frozen Confirmations and Repair

Outcome-independent discovery uses seeds 56000–56009; confirmation uses reserved seeds 57000–57049. Qwen3.7-Max and SenseNova-6.7-Flash-Lite separately passed registered admission thresholds, after which each Direct confirmation froze 50 paired seeds at L2 and L4. Selection uses identity, seed, length, context/contract hashes, and infrastructure status, never residual outcome.

For **final-only self-repair**, one new joint call per registered instance privately proposes, checks, and revises an assignment while returning the same final JSON contract as Direct. It receives no gate answers or oracle feedback and uses the same pretreatment strict denominator.

All calls use $T = 0$. Cache keys include provider, endpoint, model, prompt hash, seed, temperature, and output budget; logs record raw-output hashes, latency, cache status, and token usage. Wilson intervals use admitted denominators. Length and repair comparisons use 20,000-resample paired bootstrap intervals and exact McNemar tests. Ring length is a dose variable, not a topology-only causal contrast.

4.3 Repository-Grounded Paired Composition

Here *repository-grounded* means that each family freezes a repository URL and base commit, an upstream DeepSWE task and its hashes, a container image, and typed interfaces and executable

contracts derived from that task. Each row nevertheless runs generated components and a coordinator in a purpose-built package harness; it is not a fresh end-to-end edit of an independently sampled repository.

The first two frozen replications use SenseNova-hosted DeepSeek-V4-Flash. Updo uses seeds 80200–80249 and independently generates availability, latency, and cooldown components. Pebble uses seeds 81300–81349 and generates durability advance, wait resolution, and durable-callback components. A bundle is admitted only after deterministic transport, build, regression, and semantic tests; its exact sources are injected read-only into both matched coordinator prompts.

Updo contrasts independent availability/latency lanes with non-commutative shared-state coordination. Pebble contrasts three pre-event snapshots with a shared lifecycle in which durability is visible to waiting, wait status is visible to callback recording, and the complete callback state feeds the next event. Within each family, the declarative prompts share components, signature, output type, writable scope, and hidden traces, and neither gives a procedural call order. Common-strict admission requires atomic and exact-source gates plus both coordinator non-joint gates; residual admission additionally requires control success. These are registered repeated generations of two selected task families, not independent repositories.

After ten-seed Qwen3.7-Max qualifications were observed, both families were jointly registered for fresh 50-seed extensions rather than selecting only the positive family. Updo uses seeds 80500–80549 and Pebble 81600–81649 with the same task manifests, prompts, scorer, and gate logic. This extension decision is outcome-aware, although the fresh outcomes did not affect its plan. We therefore report complete raw funnels and paired transitions, but no independent-instance intervals, pooled family rate, software prevalence, or cross-model significance.

4.4 Boundary Audits

DeadlockGate tests all 20 helper applications before safe selection; CodePlan replays programs against 13 rule and three alias contrasts; PatchGate compares free composition with verified-helper composition. These and legacy SCPP, Travel, and IRCG probes distinguish adjacent failure layers but do not replace the matched Ring confirmation or the paired code estimands. The appendices document full protocols and stopped pilots; the anonymous artifact contains the explicitly designated frozen claim set, not every historical probe.

For topology attribution, we also constructed a factorized Ring comparator without cross-module dependencies while matching modules, function and wiring counts, the complete output interface, canonical search budget, and lexical prompt multiset. This matches observable task specification, not latent model compute. Its empirical viability and a subsequent serialization crossover are reported only as exploratory measurement-boundary audits.

5 Results

5.1 Matched Atomic Application Does Not Ensure Joint Satisfaction

The two registered confirmations provide the strongest CCF evidence (Figure 2). Gate 1 passes on all 200 condition records. Qwen's matched gate admits 50/50 at L2 and 46/50 at L4; residual CCF is 37/50=.740 (Wilson 95% CI [.604,.841]) and 43/46=.935 [.825,.978]. SenseNova admits 42/50 and 45/50, with residual CCF 32/42=.762 [.615,.865] and 39/45=.867 [.738,.937]. Failure thus replicates after identical canonical context, complete-assignment interfaces, and exhaustive application over every active source value.

Equivalently, joint success is 13/50=.260 and 3/46=.065 for Qwen, and 10/42=.238 and 6/45=.133 for SenseNova. For calibration, uniform complete assignment succeeds at .040/.0016 for L2/L4, whereas exact propagation from a uniformly sampled start succeeds at .200 at either length. All four model cells exceed uniform assignment guessing; L2 success is slightly above the random-start propagation baseline, while both L4 cells are below it. These are descriptive reference policies: they neither identify an interaction effect nor attribute failure to a particular joint suboperation.

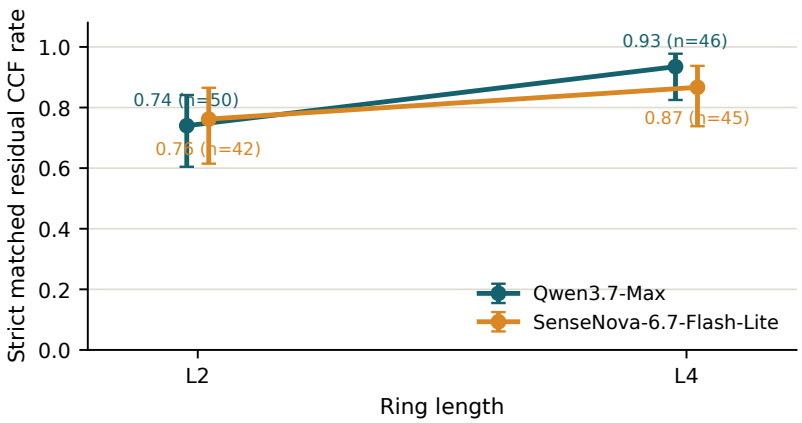

Figure 2: Two-model matched confirmation on 50 seeds. Error bars are Wilson 95% intervals; n is the admitted denominator.

5.2 Positive Length Association Is Confirmed Only for Qwen

Among seeds strict at both lengths, Qwen's L2/L4 residual is .717/.935 ($n = 46$), giving $\Delta = +.217$ (paired bootstrap 95% CI [.065,.370], exact McNemar $p = .0129$). SenseNova is .789/.868 ($n = 38$), $\Delta = +.079$ [-.079,.237], $p = .5078$. Residual CCF therefore replicates, while the positive length association is confirmed for Qwen but not SenseNova. Because length also changes the number of decisions, constraints, and tokens, this is not a topology-only effect. No between-model interaction test was registered; different within-model significance outcomes do not establish effect heterogeneity.

5.3 Final-Only Self-Repair Is Not Reliable

Table 2 shows that the matched intervention produces no reliable reduction: all four repair intervals include zero and all exact tests have $p \geq .5$. Qwen L2 rescues five failures but breaks four successes; SenseNova rescues one and breaks two. At L4, Qwen rescues two with no regression, while SenseNova rescues four and breaks two. These transitions yield only small net changes, and residual failure after repair remains .720/.891 for Qwen and .786/.822 for SenseNova. The instruction changes some decoding paths without reliably restoring joint feasibility. Because reasoning is hidden, we do not infer that checks were faithfully executed. Visible-trace and explicit-enumeration diagnostics appear in Appendices C and B.

Table 2: Matched paired comparisons. CI is the paired bootstrap 95% interval for $\Delta = \text{right} - \text{left}$.

Model	Contrast	n	Left	Right	Δ [CI], p
Qwen	L2 → L4	46	.717	.935	+.217 [.065,.370], .0129
SenseNova	L2 → L4	38	.789	.868	+.079 [-.079,.237], .5078
Qwen	L2 Direct → Repair	50	.740	.720	-.020 [-.140,.100], 1.00
Qwen	L4 Direct → Repair	46	.935	.891	-.043 [-.109,.000], .5000
SenseNova	L2 Direct → Repair	42	.762	.786	+.024 [-.048,.119], 1.00
SenseNova	L4 Direct → Repair	45	.867	.822	-.044 [-.156,.067], .6875

5.4 Repository-Grounded Coordination Is Model–Family Specific

The two code families supply a deliberately narrower evidence chain (Table 3). With DeepSeek, Updo leaves 32 control-success common-strict pairs and 17 fail non-commutative coordination; Pebble leaves 24 residual-eligible pairs and four fail shared-lifecycle coordination. Qwen sharply separates the families: all 50 Updo pairs pass both conditions, whereas all 37 control-success Pebble pairs fail the interactive condition.

Table 3: Repository-grounded fixed-family replications. Counts are repeated generations, not independent repositories; D/Q denote DeepSeek/Qwen.

Model	Family	Atomic	Common strict	Control / interact.	Residual
D	Updo	50/50	32/50	32/32 / 15/32	17/32
D	Pebble	47/50	28/50	24/28 / 22/28	4/24
Q	Updo	50/50	50/50	50/50 / 50/50	0/50
Q	Pebble	49/50	49/50	37/49 / 0/49	37/37

DeepSeek Updo failures exhibit missing availability-to-latency dependencies or stale latency views after cooldown merging; its Pebble failures exhibit partial shared-state returns or lost serial visibility/first-error precedence. For Qwen Pebble, all 49 strict interactive outputs violate the explicit requirement that all returned lanes equal the terminal shared state. Twelve also fail the matched control and are not residual events; the other 37 form the residual numerator and denominator. Qwen Updo's 50/50 success is the mandatory null boundary, not discarded evidence. This contrast supports model–task-family specificity, not a pooled software-task prevalence estimate or a model-wide composition deficit.

An exploratory GLM-5.2 qualification is directionally consistent (Updo 7/10; Pebble 9/9 residual among control-success strict runs), but remains an $n = 10$ qualification rather than a third confirmation (Appendix D).

5.5 Boundary Audits Distinguish Adjacent Failures

Table 4 shows why difficult code outputs are not automatically CCF. DeadlockGate admits 95/100 records after all-candidate helper application, but its nested safety-evaluation gate admits none; the observed selection errors therefore locate an application-to-evaluation bottleneck. In CodePlan, only 9/36/52 SenseNova/DeepSeek/Qwen programs pass a conservative rule-and-alias audit, and none then fails the original joint cases. PatchGate's verified-helper condition succeeds 10/10, showing that sufficient scaffolding can remove rather than expose a composition gap.

Table 4: Code audits locate different failure layers. "Post-gate" uses the strongest registered denominator for each audit.

Audit	Gate admitted	Post-gate failure	Diagnostic role
DeadlockGate	95/100 app.; 0/100 eval.	63/95 selection	Application-to-evaluation boundary
CodePlan S/D/Q	9/36/52	0/9, 0/36, 0/52	Ordinary reliability/application boundary
PatchGate helper	10/10	0/10	Verified scaffold removes the free-composition gap

A partitioned-cycle Ring discovery reaches a both-condition failure floor and identifies no connectivity effect. A later module-gated comparator removes cross-module dependencies under matched observable task specification: all ten instances pass its complete-interface module gate, but the cross-module cycle succeeds on 9/10 while the factorized local-loop comparator succeeds on only 1/10. A subsequent 2-by-2 serialization–topology crossover passes exact module and wiring gates on all ten instances but yields only 1/40 successful joint cells. Thus we constructed a formally matched no-cross-module comparator but not a viable empirical success baseline; neither audit identifies a connectivity, topology, or serialization-alignment effect. Full stopped-control results appear in Appendix E.

6 Discussion and Limitations

What the gap reveals. Rule availability, isolated application, and joint coordination are distinct capability levels. Recall, code coverage, and differently formatted atomic quizzes may overstate transferable competence. Our contribution is not gating itself, but its exhaustive context/interface-matched operationalization for interacting constraints.

Why final-only checking is insufficient for reliable repair. Next-token generation commits to one prefix whereas feasibility is global. Detecting a violation supplies negative information but not

a closing trajectory. Final-only repair reshuffles successes without reliable net gain. This does not imply that external search, symbolic solvers, or multi-call backtracking are ineffective (Yao et al., 2023); they can preserve alternatives that single-call continuation cannot.

What the code ladder reveals. Code generation is not one capability. End-to-end failure can arise at patch transport, build, regression preservation, atomic semantics, component identity, or interactive coordination. The negative audits show both sides of this distinction: natural repository tasks often fail before a conservative application gate, whereas verified helpers can make composition trivial. Updo and Pebble occupy the informative middle. Their components and matched controls are executable, yet declarative non-commutative or shared-lifecycle state threading can remain unreliable. Qwen's complete Updo success alongside complete residual Pebble failure shows that the gap is neither an inevitable consequence of this scaffold nor a uniform property of one model. CCF is therefore most useful as a model–task-layer diagnosis, not as a new name for low patch success.

Scope and negative controls. CCF is not renamed combinatorial explosion or a fixed-depth law. Ring instances are tiny and exactly solvable; length changes constraints, decisions, and prompt size together, so only an association is identified. The confirmations cover two hosted models and a synthetic task. Updo and Pebble add two frozen repository-grounded families under DeepSeek and an outcome-aware Qwen extension, but not an open-weight model or a broad natural-software benchmark. Repeated generations within a family are not independent repositories, and the Qwen family contrast supports specificity rather than prevalence. A positive length association is confirmed only for Qwen; SenseNova is inconclusive, and this difference in significance does not establish heterogeneity. Hosted $T = 0$ inference may remain nondeterministic, although the core statistics use distinct instances.

Deadlock, CodePlan, and PatchGate prevent overgeneralization: they respectively locate an evaluation bottleneck, a zero-residual post-application boundary, and a scaffolded success boundary. Updo and Pebble demonstrate that residual failure can also occur with verified code components, while Qwen Updo shows that it need not occur under the same design. The evidence remains limited to two selected frozen task families. Autoregressive commitment, representation interference, and low-dimensional geometry are possible explanations, not mechanisms established here. A partitioned-cycle discovery also hits the failure floor in both conditions. A stronger module-gated design constructs a same-observable-workload comparator without cross-module dependencies, but that comparator succeeds on only 1/10 instances; the follow-up serialization crossover likewise reaches a one-success-in-forty joint floor. These audits do not identify a topology-only effect or an internal topology prior and are not expanded post hoc. Ring residual CCF is established, but its excess over equally specified non-interactive Ring work remains unresolved. This limitation is specific to Ring topology attribution: Updo and Pebble use their own empirically executable matched controls.

7 Conclusion

Exact rule representation and context/interface-matched isolated application do not ensure joint satisfaction: two models retain high residual failure on closed constraint rings. Repository-grounded evidence under DeepSeek and Qwen is model–family specific: both families yield nonzero residuals for DeepSeek, while Qwen completely succeeds on Updo and completely fails on residual-eligible Pebble runs. Qwen shows a significant positive L2-to-L4 association whereas SenseNova is inconclusive; we infer no cross-model heterogeneity from that difference in significance. A matched final-only self-repair instruction does not reliably close the gap in either model; Deadlock, CodePlan, and PatchGate delimit rather than inflate the claim. The code results establish fixed-family existence, not software-task prevalence. Reliable systems should preserve feasible alternatives through explicit state, search, or solvers.

AI Use Statement

Generative AI tools assisted with research-methodology feedback, protocol and schema drafting, software implementation, data and statistical auditing, interpretation of experimental outputs, literature search and summarization, paper outlining, drafting, editing, translation, and artifact preparation. They did not autonomously authorize confirmatory runs or determine which outcomes entered

the paper. The authors froze confirmatory protocols before execution, inspected the underlying records, ran and reviewed tests and statistical analyses, checked claims and citations against the cited sources, and manually reviewed all AI-assisted text and code. Hosted-model calls that constitute the reported experiments are separately documented as part of the methodology. The authors take responsibility for the paper's final text, claims, analyses, and artifacts.

Reproducibility Statement

Appendices A–H specify the task, matched gates, prompts, frozen denominators, statistical tests, boundary audits, cache identities, and provider-stability checks. The anonymous supplementary artifact contains the registered analysis code, schemas, tests, preregistration records, selected trial rows, summary tables, and a SHA-256 manifest; Appendix I describes its evidence roles and exclusions. Raw credentials and hosted-response logs are excluded for security and anonymity. All main-text numerical results and the appendix audits registered in the machine-readable claim ledger can be checked offline against the frozen release; historical exploratory tables are retained for context but are outside that selected claim set.

References

Juo-Tung Chen and Chien-Ming Huang. Forgetful large language models: Lessons learned from using LLMs in robot programming. *Proceedings of the AAAI Symposium Series*, 2(1):508–513, 2024. doi: 10.1609/aaaiss.v2i1.27721. URL https://ojs.aaai.org/index.php/AAAI-SS/article/view/27721. Proceedings of the 2023 AAAI Fall Symposia.

Yanan Chen, Ali Pesaranghader, Tanmana Sadhu, and Dong Hoon Yi. Can we rely on LLM agents to draft long-horizon plans? let's take TravelPlanner as an example, 2024. URL https://arxiv.org/abs/2408.06318. arXiv preprint.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. The neural data router: Adaptive control flow in transformers improves systematic generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KBQP4A_J1K. ICLR 2022 poster; virtual venue: https://dev.iclr.cc/virtual/2022/poster/6053.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang (Lorraine) Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70293–70332. Curran Associates, Inc., 2023. doi: 10.52202/075280-3081. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/deb3c28192f979302c157cb653c15e90-Paper-Conference.pdf.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 41340–41391. Curran Associates, Inc., 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Paper-Conference.pdf.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.257. URL https://aclanthology.org/2024.acl-long.257/.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf. OpenReview: https://openreview.net/forum?id=VTF8yNQM66.

Garvin Kruthof. Models recall what they violate: Constraint adherence in multi-turn LLM ideation, 2026. URL https://arxiv.org/abs/2604.28031. arXiv preprint.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15339–15353, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.818. URL https://aclanthology.org/2024.acl-long.818/. Code and data: https://github.com/alonj/Same-Task-More-Tokens.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Will Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth transformers. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 95315–95339. Curran Associates, Inc., 2025. doi: 10.52202/085713-3186. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/88dd7aa6979e352fda7c4952ca8eac59-Paper-Conference.pdf.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54590–54613. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xie24j.html. Code and data: https://github.com/youngerous/TravelPlanner.

Wenkai Yang, Shuming Ma, Yankai Lin, and Furu Wei. Towards thinking-optimal scaling of test-time compute for llm reasoning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 43605–43631. Curran Associates, Inc., 2025. doi: 10.52202/085713-1452. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3e22bea3b170f4c2aebb9c48d98ae64d-Paper-Conference.pdf.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 11809–11822. Curran Associates, Inc., 2023. doi: 10.52202/075280-0517. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf. Code and prompts: https://github.com/princeton-nlp/tree-of-thought-llm.

Shunyu Yao, Howard Chen, Austin Hanjie, Runzhe Yang, and Karthik Narasimhan. COLLIE: Systematic construction of constrained text generation tasks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 38595–38606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/a77eadda332b6d4a9ae1e0e4024555f2-Paper-Conference.pdf. OpenReview: https://openreview.net/forum?id=kxgSlyirUZ; code and data: https://github.com/princeton-nlp/Collie.

Junjie Ye, Caishuang Huang, Zhuohan Chen, Wenjie Fu, Chenyuan Yang, Leyi Yang, Yilong Wu, Peng Wang, Meng Zhou, Xiaolong Yang, Tao Gui, Qi Zhang, Zhongchao Shi, Jianping Fan, and Xuanjing Huang. MulDimIF: A multi-dimensional constraint framework for evaluating and improving instruction following in large language models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 2078–2104, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.99. URL https://aclanthology.org/2026.findings-acl.99/.

Zhe Yu, Wenpeng Xing, Yunzhao Wei, Jie Chen, Hongzhi Wang, Xuyang Teng, and Meng Han. Composition collapse: Stable factual knowledge does not imply compositional reasoning, 2026. URL https://arxiv.org/abs/2605.26789. arXiv preprint.

Mert Yuksekgonul, Varun Chandrasekaran, Erik Jones, Suriya Gunasekar, Ranjita Naik, Hamid Palangi, Ece Kamar, and Besmira Nushi. Attention satisfies: A constraint-satisfaction lens on factual errors of language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32097–32121, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/880ee6e23b6ee548ebcccafe0ef0f513-Paper-Conference.pdf. OpenReview: https://openreview.net/forum?id=gfFVATffPd; arXiv: https://arxiv.org/abs/2309.15098; code: https://github.com/microsoft/mechanistic-error-probe.

Longfei Yun, Letian Peng, and Jingbo Shang. ULTRABENCH: Benchmarking LLMs under extreme fine-grained text generation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15438–15453, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.835. URL https://aclanthology.org/2025.findings-emnlp.835/.

Jinman Zhao and Xueyan Zhang. Large Language Model is not a (Multilingual) Compositional Relation Reasoner. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wLQ3I0F1oj. COLM 2024; accepted paper, submission 407; PDF: https://openreview.net/pdf?id=wLQ3I0F1oj.

Jun Zhao, Jingqi Tong, Yurong Mou, Ming Zhang, Qi Zhang, and Xuanjing Huang. Exploring the compositional deficiency of large language models in mathematical reasoning through trap problems. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16361–16376, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.915. URL https://aclanthology.org/2024.emnlp-main.915/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911. arXiv preprint; code and data: https://github.com/google-research/google-research/tree/master/instruction_following_eval.

Yuxu Zhou, Ondřej Kuželka, Yuyi Wang, Yuanhong Wang, and Yi Chang. CombEval: A framework for evaluating combinatorial counting in large language models, 2026. URL https://arxiv.org/abs/2606.19788. arXiv preprint; code: https://github.com/YuxuZhou-CN/combination-problem-generation.

APPENDIX

A MATCHED CONSTRAINT RING PROTOCOL

A.1 TASK CONSTRUCTION AND ESTIMAND

Each seed fixes sixteen five-symbol permutation operators independently of model responses. A rejection sampler requires every registered prefix composition to have exactly one fixed point. L2 and L4 therefore share a nested inventory, and each registered task has one feasible assignment. For an instance with feasible set $\mathcal{F}(C)$, joint success is $J \Leftrightarrow Y \in \mathcal{F}(C)$. Residual CCF is evaluated only when the representation and matched application gates both pass and no infrastructure failure occurs.

A.2 REPRESENTATION AND MATCHED APPLICATION GATES

Gate 1 uses three paraphrases that request canonical active lookup tables; every entry must match in every response. The matched application gate issues $5L$ complete-assignment calls. Each call receives byte-identical canonical instance context and the same JSON output contract as the joint call. The audited source variable is pinned, the adjacent target must equal the operator output, and deterministic filler variables must be copied unchanged. All active inputs must pass. Context, contract, and prompt hashes are checked per record, and no gate answer is exposed to the stateless joint call.

A.3 JOINT OUTPUT AND DETERMINISTIC SCORING

Direct requests only $\{$`"X1":"...",...,"XL":"..."`$\}$. The scorer accepts any member of the feasible set rather than a single serialized gold answer. Model-caused parse or format violations count as joint failures; network, provider, or evaluator failures are excluded as infrastructure events. Failure types are parse failure, transition failure, nonclosing trajectory, and joint success. Collapse step is computed by enumerating oracle-feasible completions after each output-order commitment.

A.4 REGISTERED INTERVENTION CONTRACTS

Final-only CoT-Repair receives the same instance context and returns the same assignment interface as Direct. It asks the model to check and revise privately, but exposes neither scorer feedback nor a gold state. The treatment therefore tests instruction effectiveness, not chain-of-thought faithfulness.

Enumerate-Starts instead requires a `trajectories` object with exactly one entry for each of the five X_1 symbols. Every entry contains the fully propagated assignment and a Boolean `closes`; the response also gives `selected_start` and the final assignment. Trace pass requires all five assignments and closure labels to match the oracle, while selection pass requires the unique gold start and assignment. Full prompt text, output schemas, and hashes are included in the anonymous artifact described in Appendix I.

A.5 METHODOLOGICAL LINEAGE

The principle of conditioning composition on demonstrated atomic competence follows prior capability-gated evaluation (Yu et al., 2026). Ring adds exact table representation, source-complete isolated application, context/interface matching, closed feasible-assignment scoring, paired length analysis, and prefix-collapse diagnosis. The contribution is this operationalization for interacting constraints, not gating in the abstract.

B CONFIRMATORY CONSTRAINT RING RESULTS

ADRs 0030/0033 separately freeze Qwen3.7-Max and SenseNova-6.7-Flash-Lite, Direct decoding, protocol `context_interface_matched_v1_2`, scorer `constraint_ring_matched_scorer_v1_2`, seeds 57000–57049, and L2/L4. Their 100-row selected CSV SHA-256 values are, respectively:

c914d5c16095eccab23cee5dcb2db188ee064f4528d4d13adf5c69bd6aa95326

and dba27b7c3cef7a0b43c34693216d7a8ecfa7b2631ce831035946e8642a7ffc14. Selection uses no gate or residual outcome. Table 5 reports the frozen outcomes.

Table 5: Two-model matched-gate confirmations (Wilson 95% CI).

Model	L	Gate 1	Matched gate	Strict n	Residual [CI]
Qwen	2	50/50	50/50	50	37/50=.740 [.604,.841]
Qwen	4	50/50	46/50	46	43/46=.935 [.825,.978]
SenseNova	2	50/50	42/50	42	32/42=.762 [.615,.865]
SenseNova	4	50/50	45/50	45	39/45=.867 [.738,.937]

All 200 context and output-contract hash audits pass, with no infrastructure failures. For Qwen's 46 jointly strict pairs, L2/L4 residual failure is .717/.935, $\Delta = +.217$, paired bootstrap 95% CI [.065,.370], and exact McNemar $p = .0129$ ($n_{01} = 12, n_{10} = 2$). For SenseNova's 38 pairs it is .789/.868, $\Delta = +.079$ [-.079,.237], $p = .5078$ ($n_{01} = 6, n_{10} = 3$). The association is positive in both models, but only Qwen's within-model interval excludes zero. Because no between-model interaction test was registered, these results do not establish effect heterogeneity and are not presented as a universal length law or a topology-only causal effect.

Strict-gate failures remain visible in the audit funnel but never enter the residual numerator or denominator. The complete selected rows, denominator identities, and source hashes are frozen separately from discovery and legacy broad-gate records.

C Interventions and Diagnostic Cases

C.1 Matched final-only self-repair

ADRs 0039/0040 reuse the frozen SenseNova/Qwen gates and Direct records, call every registered joint instance once, and preserve the final assignment interface. Both datasets contain exactly 100 unique model–length–seed rows, 100 distinct prompt hashes, and zero infrastructure or parse failures. Table 6 reports the paired transitions.

Table 6: Final-only repair on frozen matched-gate denominators. F→S and S→F are paired rescues and regressions.

Model	L	n	Direct	Repair	F→S/S→F	Δ [CI], p
Qwen	2	50	.740	.720	5/4	-.020 [-.140,.100], 1.00
Qwen	4	46	.935	.891	2/0	-.043 [-.109,.000], .5000
SenseNova	2	42	.762	.786	1/2	+.024 [-.048,.119], 1.00
SenseNova	4	45	.867	.822	4/2	-.044 [-.156,.067], .6875

Repair rescues some failures but also breaks previously correct answers, and no paired comparison is significant. The matched treatment therefore does not reliably reduce residual CCF.

C.2 Explicit-enumeration control

The separate control pairs Direct and enumeration on the same seeds and requires all five start-symbol trajectories, their closure labels, and a selected final assignment. It uses `active_exhaustive_v1_1`, the active-only strict gate, and seeds 31000–31049. It is not the core `context_interface_matched_v1_2` confirmation on seeds 57000–57049; accordingly, its denominators neither match nor pool with Figure 2. Table 7 reports this separate control.

No exact McNemar comparison is below .05, and `correct_trace_joint_failure_n`=0 in all conditions. Requesting enumeration does not reliably solve the task, but the experiment does not show failure after a correctly executed exhaustive procedure.

Table 7: Explicit-enumeration elicitation. Trace/Select denote exact five-start enumeration and correct selection among strict-gate instances.

Model	L	Residual [95% CI]	Trace	Select
SenseNova	2	37/50=.740 [.604,.841]	10/50	13/50
SenseNova	4	45/47=.957 [.858,.988]	0/47	2/47
Qwen3.7-Max	2	37/50=.740 [.604,.841]	12/50	13/50
Qwen3.7-Max	4	40/45=.889 [.765,.952]	1/45	5/45

C.3 Visible-trace diagnostic cases

The primary matched intervention is final-only and has no visible trace. The following cases come from a separately frozen broad-gate trace audit and are explanatory rather than part of the matched estimand. Table 8 contrasts one failed and one successful repair.

Table 8: SenseNova L4 cases with complete and correct initial checks.

	Seed 31002: failed repair	Seed 31007: successful repair
Initial	$(P2, R5, M3, M3)$; C1–C4 false	$(R5, P2, M3, M3)$; C1–C4 false
Revision	$(K7, P2, R5, Q8)$	$(P2, P2, R5, K7)$
Result	C1/C2 pass; C3/C4 fail	C1–C4 pass
Diagnosis	Correct; all four variables revised	Correct; three revised

For seed 31002, the prefix follows $K7 \xrightarrow{O_1} P2 \xrightarrow{O_2} R5$, but $O_3(R5) = P2$, not $Q8$, and $O_4(Q8) = R5$, not $K7$. The revision splices incompatible partial trajectories and the first commitment leaves no feasible completion. Seed 31007 instead propagates $P2 \xrightarrow{O_1} P2 \xrightarrow{O_2} R5 \xrightarrow{O_3} K7 \xrightarrow{O_4} P2$, matching the oracle. Both traces diagnose all violations, but only one constructs a coherent cycle.

D Repository-Grounded Fixed-Family Evidence

These experiments test whether the same application-to-composition gap can be observed in executable repository code. They are deliberately reported below the two-model Ring confirmation because each row is a repeated generation from one fixed task family rather than an independent software task.

D.1 Verified-component design

For every fresh seed, the model implements atomic components independently. Only components passing transport, writable-scope, build, regression, and semantic tests are frozen and injected read-only into matched coordinator prompts. Control and interactive coordinators receive identical interfaces and verified source. The control keeps state lanes independent; the interactive condition introduces non-commutative visibility or feedback. Both prompts state declarative postconditions rather than a call sequence. Residual CCF is counted only among common-strict pairs whose control coordinator succeeds.

D.2 Updo temporal-policy replication

The Updo task combines independently verified availability and latency components. Its interactive coordinator must feed the updated availability state into latency processing and return a consistent merged terminal state. Table 9 reports the registered funnel.

Strict exclusions remain outside the residual numerator: four control outputs fail transport, while the interactive condition has four transport failures and ten build or atomic-regression failures. Of the 17 admitted semantic failures, ten feed `AdvanceLatency` from a stale independent lane rather than the output of `AdvanceAvailability`; seven use the correct dependency but return a latency lane copied before the cooldown timestamp is merged. Seeds 80203 and 80209 are representative cases.

Table 9: Updo declarative-only v1.2 fixed-family replication. Counts are descriptive repeated generations, not independent software tasks.

Audit stage	Count
Atomic component admission	50/50
Raw control strict / successful	46/50 / 46/46
Raw interactive strict / successful	36/50 / 19/36
Common strict denominator	32/50
Control / interactive success on common strict	32/32 / 15/32
Residual CCF on common strict	17/32 (.531)
Unique verified-component bundles	45/50
Infrastructure failures	0/50

Four component bundles recur with identical interactive prompt hashes but different outputs and outcomes. These repeats are a hosted-backend/seed stability audit, not independent task evidence. A procedural predecessor had only 1/20 residual failures, suggesting that an explicit call order may mask coordination errors; because protocol and seeds differ, this is not a causal scaffold comparison.

D.3 Pebble durability replication

Pebble independently freezes the same component-to-coordinator design. O1 advances durability, O2 resolves waiting, and O3 records durable callbacks. The interaction requires same-event durability-to-wait visibility, wait-to-callback visibility, and complete callback-state feedback. Table 10 reports the registered funnel.

Table 10: Pebble v1.1 fixed-family confirmation on fresh seeds 81300–81349. Strict exclusions are not residual failures.

Audit stage	Count
O1 / O2 / O3 atomic admission	50/50 / 50/50 / 47/50
Complete / unique admitted bundles	47/50 / 43/47
Raw control strict / successful	47/47 / 41/47
Raw interactive strict / successful	28/47 / 22/28
Common strict denominator	28/50
Control / interactive success on common strict	24/28 / 22/28
Residual CCF among control-success common strict	4/24 (.167)
Infrastructure failures	0/50

The four residual seeds are 81300, 81327, 81332, and 81342, with four different bundles and outputs. Three coordinators fail to feed back and return the complete callback state consistently; one attempts feedback but loses serial visibility or first-error precedence. The discovery run contains the same two verifier-level failure classes.

Gate attrition remains material. Three O3 generations fail their atomic build gate. Of 47 complete bundles, 19 interactive coordinators fail the strict build/regression gate, mostly by drifting from the declared `sharedState` identifier. A broader funnel would count 21 interactive failures among 41 atomically admitted raw control successes, but 17 are strict exclusions and only four are semantic residuals. We report both layers without pooling them.

D.4 Qwen joint fixed-family extension

After the ten-seed qualification outcomes were observed, Updo and Pebble were jointly committed to fresh Qwen3.7-Max 50-seed extensions. The fresh slices retain the parent prompts, task manifests, scorer, and gate logic. Reporting both families is mandatory; the extension is not described as outcome-blind. Table 11 reports both extensions.

Pebble seed 81629 fails atomic O1, explaining 248 rather than 250 completed calls. Twelve further seeds fail the matched control and are both-condition failures, not residual events. All 49 strict interactive coordinators fail the same registered assertion: the returned durability, wait, and callback lanes do not all equal the terminal shared state, despite that postcondition being explicit in the declarative prompt. The 37 control-success cases are residual CCF. In contrast, every Qwen

Table 11: Qwen fixed-family extensions on fresh seeds. The Pebble atomic exclusion skips both coordinator calls by the registered protocol.

Audit stage	Updo	Pebble
Atomic admission	50/50	49/50
Raw control strict / success	50/50 / 50/50	49/49 / 37/49
Common strict denominator	50/50	49/50
Control / interactive success	50/50 / 50/50	37/49 / 0/49
Residual CCF	0/50	37/37
Unique verified-component bundles	39/50	37/49
Infrastructure failures	0/50	0/50

Updo control and interactive coordinator passes. A zero-call post-run freeze binds 4,580 whitelisted evidence artifacts with bundle hash `87eaffbb...1979b49e`.

D.5 Exploratory GLM-5.2 qualification

Fresh ten-seed qualifications under audited non-thinking GLM-5.2 inference admit all atomic and paired bundles. Updo has ten control successes and seven interactive failures (7/10 residual). Pebble has nine control successes and all nine fail interactively (9/9 residual); the remaining pair is a control failure with interactive success. These results show protocol feasibility and provide a third-model signal, but they remain qualification-only: no independent-instance intervals, cross-model test, or confirmation claim is authorized.

D.6 Scope of the code evidence

Updo and Pebble establish repository-grounded fixed-family residual and null boundaries under two 50-seed model slices, plus exploratory qualification in a third model. They do not estimate prevalence over software tasks, justify independent-instance confidence intervals, or support a pooled cross-model rate. Their role is external-validity and discriminant support for the behavior isolated more cleanly by Constraint Ring.

E Boundary Audits

The following audits separate residual composition failure from adjacent failures in evaluation, rule application, code transport, and task elicitation. Null or non-admitted results are retained rather than relabeled as CCF.

E.1 DeadlockGate: application-to-evaluation failure

Each DeadlockGate worker has a directed two-lock acquisition edge; selecting `atomic_pair` removes its blocking edge. Both paired conditions contain 16 workers, six eligible workers, budget three, all 20 candidate triples, eight safe triples, and random-safe probability .4. Independent matched calls must return every candidate and its complete 16-worker helper mapping. The application gate requires all 20 mappings; the nested evaluation gate also requires all 20 safety labels. Table 12 reports both interaction levels.

Table 12: SenseNova DeadlockGate v2.1 confirmation, seeds 66000–66049.

Condition	Repr.	G_{app20}	G_{eval20}	Admitted	Failure [95% CI]
Low	47/50	47/50	0/50	47	.660 [.517,.778]
High	49/50	48/50	0/50	48	.667 [.525,.783]

Candidate identities and mappings pass on 2000/2000 calls, and helper semantics on 1999/2000, but every response labels its candidate safe. Thus no instance enters the all-label evaluation denominator. The 63/95 downstream selection failures diagnose an application-to-evaluation/selection break, not failure after successful exhaustive safety evaluation.

E.2 CodePlan: static coverage is not application

At depth 5, SenseNova, DeepSeek, and Qwen3.7-Max reach baseline CCSR .110, .360, and .530 while implementation coverage is .998, .988, and 1.000. The offline audit replaces token coverage with 13 canonical and three alias near-miss application contrasts. Table 13 reports the resulting funnel.

Table 13: Depth-5 baseline application audit (100 programs/model).

Model	Core	Alias	Strict	Joint	CCF-A
SenseNova	9	11	9	11	0/9
DeepSeek	57	41	36	36	0/36
Qwen3.7-Max	52	53	52	53	0/52

No strict-pass program fails the original joint cases. CodePlan therefore supports a reliability gap and an important negative boundary, but not application-conditioned residual CCF. A representative Qwen program and SenseNova seed 7082 both check the global budget before slot cost is added, pruning feasible completions. The strict audit correctly classifies this as a C1 application error rather than residual composition failure.

Scaffolded repair raises SenseNova and Qwen CCSR by +.860 and +.470 (paired $p = .0001$); DeepSeek changes by only +.040 with a bootstrap interval crossing zero. Scaffolding can improve local execution and exact satisfaction, but only when translated into correct control flow.

E.3 PatchGate and stopped repository canaries

PatchGate's verified-helper condition succeeds on 10/10 instances, yielding no residual failure after component identity is enforced. Separately, an Ofetch circuit-breaker canary passes transport, scope, application, build, and regression on all 50 calls but admits 0/10 complete five-operation bundles because one purported atomic gate mixes unreachable states and several mechanisms. No joint call was authorized. These outcomes bound the construct instead of being pooled with residual CCF.

E.4 Partitioned-cycle floor

A registered SenseNova L4 discovery compares one global four-cycle with two local two-cycles while matching operators, constraints, assignment interface, candidate space, one feasible assignment, and 20 matched lookup calls. Table 14 reports the failure floor.

Table 14: Partitioned-cycle discovery. Both conditions are at the failure floor.

Condition	Matched/strict	Residual [95% CI]
Global four-cycle	10/10	10/10=1.00 [.722,1]
Two local two-cycles	10/10	10/10=1.00 [.722,1]

All ten pairs are both-failures. The result supports residual CCF under cyclic composition but cannot identify an effect of global connectivity. The planned confirmation was stopped rather than retuning difficulty after discovery.

E.5 Matched non-interaction controls remain empirically unresolved

The first disjoint-self-loop control was construct-invalid because one condition exposed conflicting source and audit inputs. The partitioned-cycle repair above removed that interface conflict but retained two local cycles. We therefore constructed a namespace-disjoint module-gated comparator with four immutable modules. The cross condition uses one four-module cycle; the factorized condition uses four module-local loops with no cross-module dependence. Both conditions retain four function applications, four wiring equalities, the same eight-field output, the same 5^4 canonical search budget, and the same lexical prompt multiset. "Matched" here refers to this observable task specification, not equal internal FLOPs or search. Table 15 summarizes the stopped controls.

Table 15: Exploratory Ring topology-boundary audits. Calls are repeated conditions on ten admitted instances, not independent samples.

Audit	Strict admission	Joint success	Identification result
Module-gated cross cycle	10/10	9/10	Paired comparator contrast
Module-gated local loops	10/10	1/10	Control is not a viable success baseline
Crossover aligned cells	10/10	1/20	One event drives all descriptive contrasts
Crossover misaligned cells	10/10	0/20	Alignment is not identified

In the module-gated audit, all 200 Stage-A module applications pass both the target update and seven-field preservation checks. The paired Stage-B transitions are nine local-failure/cross-success, one local-success/cross-failure, and no ties. Although the direction is striking, the local condition also changes fixed-point decomposition and visible solution structure; the result is therefore not evidence that cross-module interaction helps or that the model has an internal topology prior.

The crossover holds the Hamiltonian-cycle class fixed and crosses two topologies with two visible serialization orders. Stage A passes all module and wiring items on all ten instances, yet the four cells produce successes of 1/10, 0/10, 0/10, and 0/10. The reported alignment contrast (.05), topology main effect (.05), and serialization main effect (.05) all arise from the same single successful cell. The experiment is at a joint-failure floor and cannot separate those effects. Confirmation and further prompt calibration were stopped. Collectively, these audits show that a formally matched no-cross-module comparator was constructed, but a usable non-interactive Ring baseline was not obtained.

F Supporting and Exploratory Probes

This appendix groups the historical probes that motivated the strict matched protocol but do not carry the paper's primary residual-CCF claim. Fixed-prompt repetitions are stability diagnostics rather than independent instances; Travel and IRCG test transfer or feasibility under different task designs. These historical tables are retained for methodological context but are not part of the anonymous artifact's selected offline claim set.

F.1 Fixed-Prompt SCPP Results

SCPP is retained as the motivating fixed-prompt mechanism probe. Its repeated calls are not independent task instances and carry no population confidence intervals. Figures 3 and 4 summarize the depth/recall and repair diagnostics.

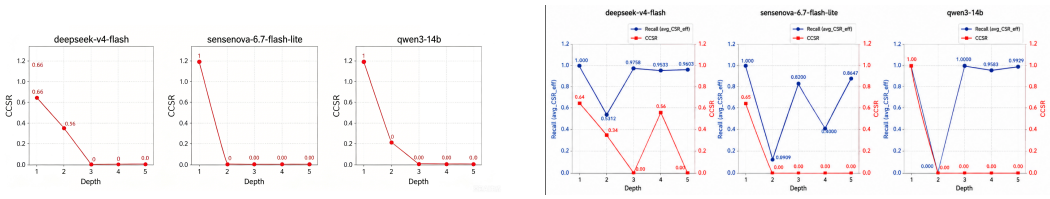

Figure 3: SCPP fixed-prompt stability: high-depth CCSR degrades while recall remains near one.

F.1.1 CID and SCPP Task Construction

This subsection provides additional details for the constraint interaction density (CID) measure and the controlled task construction used in the main experiments.

Given a constraint set $C = \{c_1, c_2, \ldots, c_n\}$, we construct a constraint interaction graph $G = (V, E)$, where each node corresponds to one constraint. An edge is added between c_i and c_j when the two constraints share at least one dependent decision variable:

$$Var(c_i) \cap Var(c_j) \neq \emptyset.$$

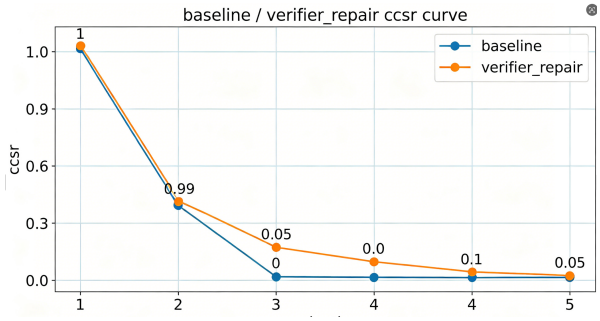

Figure 4: Verifier-Repair helps some SCPP conditions but does not reliably restore high-depth performance.

CID is then computed as

$$CID = \frac{2|E|}{|V|(|V|-1)}.$$

In SCPP, increasing depth introduces additional decision variables and activates more cross-variable constraints. This creates a controlled gradient from shallow, weakly coupled instances to deeper, more tightly coupled instances. The main text reports the aggregate results; this appendix is reserved for full instance-generation rules and additional CID statistics.

F.1.2 Legacy Prompt Variants

This subsection summarizes prompt variants used in the historical probes; the exact matched Ring contracts are specified in Appendix A and released in the artifact. The variants below are used for the reported experiments. All main experiments use deterministic decoding with temperature $T = 0.0$; stochastic sampling is used only in the supplementary Search condition.

Baseline. The model is given the task description, candidate options, and full constraint set, then asked to output the final solution directly.

Recall-then-Answer. The model is first asked to restate the constraints, then produce the final answer conditioned on the restated constraint list.

Verifier-Repair. The model generates an initial answer. A verifier returns violated constraints, and the model is asked to revise the answer.

CoT-Verify and CoT-Repair. The model externalizes intermediate constraint states before producing the final answer. CoT-Repair additionally asks the model to identify conflicting variables and repair local decisions before finalizing the solution.

F.1.3 Fixed-Prompt Reasoning Strategies

This subsection gives the operational definitions of the reasoning strategies summarized in Section 3.

Baseline. The baseline uses standard autoregressive generation:

$$Y = f_\theta(X, C).$$

The model directly produces the final answer without explicit verification or additional computation.

Recall-then-Answer. The model first reconstructs the constraint set $\hat{C}$, then generates the final solution. This tests whether externally restating the constraints is sufficient to improve composition.

Verifier-Repair. Verifier-Repair follows a generate-check-repair loop:

1. Generate an initial solution.
2. Verify constraint violations.
3. Return violated constraints.
4. Regenerate a revised solution.

The verifier identifies violated constraints but does not provide a backtracking trajectory or alternative candidate states.

CoT-Verify. CoT-Verify asks the model to write an explicit intermediate constraint checking trace before producing the final answer. We treat this trace as an external scratchpad rather than direct access to hidden reasoning.

CoT-Repair. CoT-Repair extends CoT-Verify by asking the model to detect conflicts, identify conflicting variables, modify local decisions, and then produce the final solution.

Search. The Search condition samples multiple candidate solutions at temperature $T = 0.7$ and selects the best candidate using the verification function. This condition tests whether multi-trajectory exploration mitigates single-path commitment.

Full fixed-prompt mitigation results. Figure 5 compares Baseline, CoT-Verify, and CoT-Repair. Table 16 reports the high-depth Search comparison. These repeated calls characterize fixed-prompt stability rather than task-instance uncertainty.

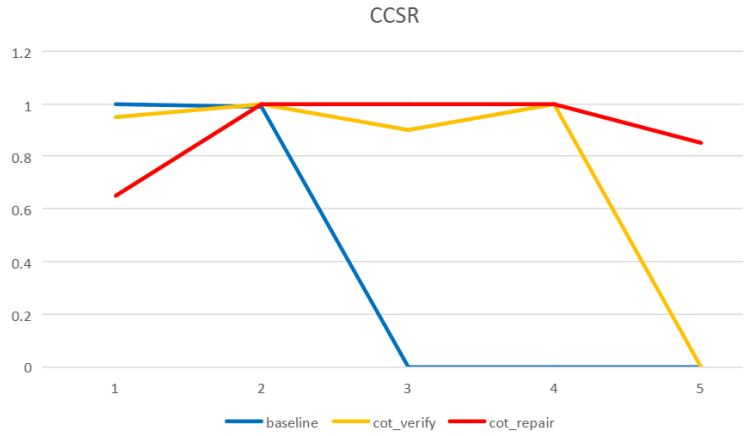

Figure 5: CoT-Repair is the strongest SCPP fixed-prompt mitigation; this task-specific result does not imply a reliable Ring benefit.

Table 16: High-depth SCPP mitigation summary for SenseNova. Values are fixed-prompt stability estimates, not task-instance CIs.

Depth	Baseline	Search	CoT-Repair
4	0.60	0.80	1.00
5	0.00	0.10	0.85

F.1.4 Historical Large-Model Fixed-Prompt Pilot

To probe whether the pattern persists for a stronger model, we ran a small fixed-prompt pilot with Qwen3.7-Max on the high-depth SCPP conditions. This pilot is not treated as an independent-sample statistical benchmark; it is a deterministic scale check under the same diagnostic protocol. Table 17 reports the fixed-prompt outcomes.

Table 17: Qwen3.7-Max fixed-prompt pilot on high-depth SCPP conditions ($n = 5$ repeated calls per condition).

Depth	Mode	CCSR	FAA	CSR_{eff}	Gap	Recall
4	Baseline	1.00	1.00	1.00	0.000	–
4	Recall-then-Answer	1.00	1.00	1.00	0.000	1.000
4	CoT-Repair	1.00	1.00	1.00	0.000	–
5	Baseline	0.00	1.00	1.00	0.125	–
5	Recall-then-Answer	0.00	1.00	1.00	0.125	0.875
5	CoT-Repair	0.80	1.00	0.914	0.014	–

Qwen3.7-Max solves depth 4 under all tested modes, but at depth 5 both Baseline and Recall-then-Answer drop to 0.00 CCSR despite full format validity and $CSR_{\text{eff}} = 1.00$. CoT-Repair recovers CCSR to 0.80 on this one fixed prompt. This historical pilot motivated the intervention but is not independent-instance evidence: the later 50-seed Constraint Ring analysis finds no reliable CoT-Repair benefit.

F.2 Travel Task Results

This appendix contains extended travel-domain results and supplemental analyses omitted from the main paper for space. RSC2P-Travel evaluates whether CCF generalizes beyond controlled symbolic planning to semantically rich constraints such as attraction category, dining restrictions, budget, opening hours, transportation, and cross-variable compatibility.

We treat Travel as an external-validity check rather than primary evidence. The source-complete Constraint Ring experiment provides the primary gated evidence, while Travel tests whether related reliability patterns appear with semantic constraints. For Travel, the raw logs include pilot runs and follow-up runs with different sample counts. To avoid mixing protocols, the canonical summary file `results/results_ccf_travel.csv` keeps only comparable $n = 20$ runs: the DeepSeek baseline/recall/search block and the June 15 SenseNova block. The $n = 2$ pilot rows and $n = 50$ follow-up rows are not averaged into this summary. Table 18 reports the curated high-depth block.

Table 18: Curated RSC2P-Travel results at high depth ($n = 20$).

Model	Strategy	Depth 4 CCSR	Depth 5 CCSR
DeepSeek-v4	Baseline	0.50	0.00
DeepSeek-v4	Recall-then-Answer	0.80	0.05
DeepSeek-v4	Verifier-Search	0.75	0.00
SenseNova-6.7	Baseline	0.05	0.00
SenseNova-6.7	Recall-then-Answer	0.15	0.05
SenseNova-6.7	CoT-Repair	1.00	1.00
SenseNova-6.7	Verifier-Repair	0.05	0.00
SenseNova-6.7	Verifier-Search	0.95	0.45

These results show the same qualitative pattern as SCPP. First, formatting is not the bottleneck: FAA remains 1.00 and parse-error rate remains 0.00 for the curated high-depth Travel rows, while CCSR can collapse. Second, constraint access does not guarantee composition. For SenseNova, Recall-then-Answer reports perfect recall at depths 4 and 5, but CCSR remains only 0.15 and 0.05, respectively. Third, mitigation depends on whether the method changes the search trajectory. In this Travel block, CoT-Repair fully recovers SenseNova at depths 4 and 5, and Verifier-Search substantially improves depth 4. Verifier-Repair, however, mirrors the baseline. This supports the interpretation that final violation feedback alone is often insufficient: the model needs either explicit conflict-state management or multi-candidate exploration to avoid committing to an invalid constraint combination.

F.2.1 Semantic vs. Symbolic Representation Ablation

To analyze whether CCF depends on natural-language semantics, we compare Semantic and Symbolic variants of RSC2P-Travel. The Semantic condition uses natural-language labels such as "historical site" or "Beijing cuisine." The Symbolic condition replaces these labels with abstract identifiers while preserving the same constraint structure and feasible solution space. Figure 6 compares the two representations.

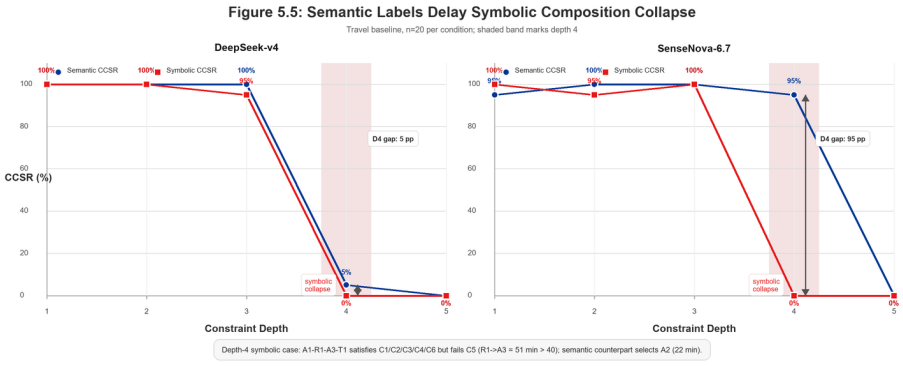

Figure 6: CCSR under Semantic and Symbolic travel representations. The shaded band marks depth 4, where symbolic labels produce a sharp composition collapse despite perfect format validity.

The figure shows that symbolic abstraction does not remove CCF. At depth 4, SenseNova drops from 0.95 CCSR in the Semantic condition to 0.00 in the Symbolic condition, while FAA remains 1.00 and the parse-error rate remains 0.00. DeepSeek is also difficult at depth 4, but symbolic labels still reduce effective constraint satisfaction: CSR_{eff} falls from 0.825 to 0.500. Thus, the failure is not caused by malformed outputs; it reflects a loss of stable joint satisfaction under abstract labels.

A representative depth-4 Symbolic failure illustrates the mechanism. Two trials produce the same plan, A1-R1-A3-T1. This plan satisfies the category, dining, budget, opening-time, and total-distance constraints: A1 is a valid morning attraction, R1 is a valid restaurant, A3 is a valid afternoon attraction, and the total cost is $60 + 80 + 20 + 5 = 165 \leq 200$. However, it fails the travel-time constraint. The A1$\rightarrow$R1 leg is short, 1.0 grid unit or 10 minutes, but the R1$\rightarrow$A3 leg is 5.1 grid units or 51 minutes, exceeding the 40-minute limit. A matched Semantic success instead selects A2 for the afternoon attraction, yielding a 22-minute post-lunch travel leg. The repeated symbolic output suggests that the failure is systematic: abstract labels remove useful spatial priors, and the model converges on a locally plausible but globally invalid combination.

Overall, semantic labels can help by activating priors about likely locations and compatibility, but they do not eliminate the underlying bottleneck. CCF remains visible whenever the model must jointly compose type, budget, temporal, spatial, and transportation constraints.

F.3 Exploratory IRCG Pilot

Interaction-Rewired Constraint Graph (IRCG) was designed to hold an operator inventory approximately fixed while changing dependency structure. We retain it as exploratory rather than confirmatory because early protocols made Gate 2 too difficult, producing an empty or very small strict-gate denominator.

Under the revised `connected_unicyclic_v1_2` protocol, Qwen3.7- Max at depth 2 passes Gate 1 on 10/10 instances and Gate 2 on 7/10 instances; item and operator accuracies are .98 and .94. All seven strict-pass instances fail the joint task, yielding residual CCF 7/7 (Wilson 95% CI [.646,1.000]). This pilot is compatible with the Ring result but is not used in any primary claim: the denominator is small, only one model is represented, and the intended paired rewiring contrast has not yet been confirmed at adequate gate pass rates.

G Theoretical Context

This appendix provides theoretical context for CCF. The main paper uses only the minimal definitions needed for the empirical claims; here we make the connection to constraint satisfaction, sequential commitment, and architectural limits more explicit.

G.1 CSP Context, Not a Complexity Claim

Constraint spaces can grow combinatorially: with n variables and d values per variable, the naive space is d^n. This is background, not the CCF contribution. CCF does not claim that every evaluated instance is computationally hard or that autoregressive models fail because they cannot enumerate an exponential space. Indeed, each audited depth-5 CodePlan hidden case has only 162 tuples, one feasible solution, and a maximum per-instance median exhaustive runtime of 4.01 ms on our host. The diagnostic question is whether a generated procedure preserves global feasibility despite accessible rules, even when a conventional enumerator solves the concrete instance easily.

This CSP perspective is aligned with Attention Satisfies, which models factual generation as constraint satisfaction and links attention to constraint tokens with factual correctness (Yuksekgonul et al., 2024). CCF extends that lens from factual queries to multi-decision generation: accessing constraint tokens is necessary, but not sufficient, for maintaining a feasible joint assignment.

This also separates CCF from CombEval, a concurrent preprint that evaluates solver-verified combinatorial counting over typed objects (Zhou et al., 2026). CombEval asks for the number of valid structures; CCF asks for one exact feasible assignment and conditions the interpretation on access or isolated application. We use both it and the concurrent Composition Collapse preprint for positioning, not as premises for our empirical claims.

G.2 Irreversible Sequential Commitment

Autoregressive generation differs from classical constraint solving in that each early decision becomes part of the subsequent context. A solver can maintain candidate states and backtrack when a partial assignment becomes infeasible. A single autoregressive rollout instead commits to a sequence of decisions without native state revision.

This creates an irreversible commitment mechanism. A locally plausible early choice can shrink the feasible continuation set for later variables. If the rollout crosses into a prefix with no feasible completion, later tokens may still mention the correct rules but cannot recover a globally valid assignment without restarting or searching.

G.3 Feasible-Space Collapse

Let $\mathcal{F}_t(Y_{1:t}; X, C)$ denote the feasible continuations consistent with a partial prefix. Feasible-space collapse occurs at the first step t for which

$$|\mathcal{F}_{t-1}| > 0 \quad \text{and} \quad |\mathcal{F}_t| = 0.$$

The collapse index $\kappa(Y)$ in Section 3.1 records when this happens along a generated trajectory. This index is not an assumption about the model's internal computation; it is an external diagnostic that connects observed invalid outputs to the structure of the task's feasible space.

Under CCF, a model may satisfy $\mathcal{R}_\tau$, meaning that it can recover the constraints, while still generating a trajectory whose prefix collapses the feasible continuation set. This explains why Recall or CSR can remain high while CCSR falls.

G.4 Lightweight Lemmas and Observation

Lemma 1: Recall is not sufficient for feasible decoding. Perfect recall does not imply that greedy single-path decoding finds a feasible assignment. Consider first-step choices a, b with $p(a \mid X, C) > p(b \mid X, C)$, while the constraints permit completions only after b. A model may reproduce every constraint yet greedily select a, for which $\mathcal{F}_1(a) = \emptyset$. Thus constraint availability and feasible prefix selection are distinct.

Lemma 2: Continuation-only repair cannot undo collapse. If $|\mathcal{F}_t| = 0$, then by definition no suffix preserving $Y_{1:t}$ satisfies all constraints. Hence a verifier may detect the violation, but recovery requires revising an earlier commitment, restarting, or exploring another prefix.

Observation: Search and scaffolding can delay commitment. If a proposal independently reaches a feasible prefix with probability $q > 0$, then K branches contain at least one feasible prefix with probability $1 - (1 - q)^K$. A sound verifier can exploit this coverage; scaffolding can instead test feasibility before commitment. This is not a universal guarantee: gains depend on proposal mass, verifier quality, and whether the model operationalizes the scaffold, explaining variation across tasks and models.

G.5 Transformer Depth and Routing

Architectural work suggests that reliable composition may require more than parameter scaling. The Neural Data Router shows that explicit routing can improve systematic generalization (Csordás et al., 2022). Expressivity results also indicate that log-depth Transformers can be strictly more powerful than shallower alternatives on certain computations (Merrill & Sabharwal, 2025). These results do not prove CCF, but they support the broader claim that constraint composition depends on computational structure, not only on stored knowledge.

G.6 Search and Latent Reasoning

Tree-of-Thoughts improves reasoning by branching, evaluating, and backtracking over intermediate states (Yao et al., 2023). From the CCF perspective, search helps because it can recover from a collapsed prefix by exploring alternative commitments. Recent test-time scaling work further suggests that longer reasoning traces are not always beneficial (Yang et al., 2025). Latent-reasoning approaches such as recurrent depth perform iterative computation in hidden state rather than relying only on token-level traces (Geiping et al., 2025). Together, these approaches add branching search or iterative computation beyond a single autoregressive rollout.

H Reproducibility, Freezes, and Stopping Decisions

H.1 Core matched confirmations

The strongest construct-validity controls are frozen separately from the legacy 400-row Ring set. ADR/RFC-0030 and 0033 select Qwen3.7-Max and SenseNova-6.7-Flash-Lite, respectively, with Direct decoding, seeds 57000–57049, and L2/L4 after outcome-independent discovery admission. Each freeze verifies exactly 100 rows, 50 paired operator inventories, explicit protocol/scorer identities, identical context and output-contract hashes, and zero infrastructure failures. The audit classifies both slices as strict matched residual CCF evidence while prohibiting a topology-only interpretation.

H.2 Hosted-model provenance boundary

Registrations and call records preserve provider and requested model IDs, endpoint identity, call time, prompt and raw-output hashes, seed, temperature, cache status, latency, and returned token accounting. This supports exact prompt/scorer/result replay and evidence verification. It does not archive the providers' serving weights or guarantee that a future request with the same model ID reaches the historical checkpoint; a durable system fingerprint was not available for the core hosted confirmations. Exact future regeneration of their raw outputs is therefore outside the reproducibility claim.

H.3 Matched repair and code-family freezes

ADRs/RFCs 0039 and 0040 reuse the two immutable confirmation gates for a joint-only, final-interface-matched repair intervention. Each model has 100 unique registered rows, zero infrastructure failures, and an outcome-independent denominator. The SenseNova/Qwen trial hashes

are `b27e9c2f...91662302` and `6bec2a5d...1c20e5d`; caches, logs, results, and analysis identities are provider-separated, and the repair prompt is byte-identical across models.

The Updo and Pebble fixed-family replications separately freeze component source, component semantic tests, coordinator prompts, common-strict denominators, and post-run evidence manifests. Updo hashes 1,759 whitelisted artifacts and Pebble 1,739 without additional hosted calls. Their registrations explicitly forbid independent-task prevalence, confidence-interval, cross-model, or paper-core claims.

ADR/RFC-0105 jointly commit both Qwen fixed-family extensions after their ten-seed qualification outcomes were known. ADR/RFC-0106 then freeze the two completed fresh slices together with zero hosted calls: 100 trials, 300 capability entries, 498 call records, and 754,640 tokens. All 4,580 allowlisted evidence files replay under bundle hash `87eaffbb...1979b49e`. Joint reporting preserves Updo's 0/50 null boundary alongside Pebble's 37/37 residual result. Separate GLM-5.2 ten-seed code qualifications remain exploratory and are not promoted to confirmation.

H.4 Outcome-blind model qualification stops

Two L2 qualifications tested whether additional GLM models could enter the same strict denominator. Both use audited non-thinking inference, ten fresh instances, zero infrastructure/protocol failures, and a preregistered matched-gate promotion threshold of .50. GLM-5.1 and GLM-5.2 pass Gate 1 on 10/10 and attain matched-item accuracy 89/100 and 91/100, but their all-items matched gates admit only 3/10 and 4/10. Neither run authorizes L4. Residual failures in 3/3 and 3/4 admitted instances are descriptive capability-boundary cases, not third-model confirmations. ADR/RFC-0100 freeze the stop without new hosted calls.

H.5 CodePlan replay and application audit

Table 13 uses baseline seeds 7000–7099 for every model. The paired comparison uses shared seeds 7000–7099 for DeepSeek and Qwen3.7-Max, and 7030–7129 for SenseNova, matching its available scaffolded block. Consequently, SenseNova baseline is .110 in the primary block but .140 on the paired seed set; pilot/debug rows outside these sets are excluded.

The offline script rebuilds the instances, verifies prompt/code and hidden-case hashes, and replays saved programs without new LLM calls. All 300 baseline records match the original scorer. Thirteen canonical and three alias near-miss contrasts produce Core/Alias/Strict counts 9/11/9 for SenseNova, 57/41/36 for DeepSeek, and 52/53/52 for Qwen. No strict-pass baseline fails the original joint cases. For scaffolded code, SenseNova and Qwen replay at strict/joint 100/100. DeepSeek replays at 40/99 after one historical scorer-inconsistent record is excluded rather than silently corrected.

H.6 Hosted-provider stability audit

Repeated deterministic SCPP calls distinguish fixed-prompt stability from instance-level evidence. In the July 14 audit, SenseNova depth-5 baseline has 19 logged calls but one prompt hash; at $T = 0$, two raw-output hashes produce both valid and invalid completions. Temperature zero therefore does not imply bitwise-identical hosted inference. The logs cannot identify whether backend routing, numerical nondeterminism, model revision, or tie-breaking causes the variation, so these repetitions are provider/stability evidence rather than independent samples. The primary Ring and code results instead use generated instances and frozen selection manifests.

I Anonymous Submission Artifact

ADR/RFC-0053 freeze the original paper-facing artifact; additive evidence supplements preserve that freeze while extending the audited claim set without changing any scientific outcome. The release contains a generated allowlist of registered experiment and analysis code, JSON Schemas, focused tests, preregistration records, and the selected primary, intervention, and boundary result tables. Every file is recorded with its release path, source path, role, byte size, and SHA-256 in a schema-validated manifest.

The artifact separates evidence by role. The two model-specific matched Ring confirmations are `primary`; the final-only repair records are `paired_intervention`; DeadlockGate v2.1, CodePlan's application audit, PatchGate's verified-helper boundary, the partitioned-cycle discovery, and the Updo and Pebble fixed-family replications are `boundary`. The stopped GLM qualifications are a separate exploratory capability boundary and do not enter primary estimates. Each code-family subset includes all 50 sanitized trial records, its aggregate and post-run manifests, and selected representative generated coordinators, but excludes raw hosted responses. The Qwen fixed-family subset additionally includes the joint post-run freeze that binds the complete Updo null and Pebble positive slices; neither family may be selected away. GLM-5.2 code-family qualifications are labeled exploratory rather than merged into the 50-seed table. Boundary results are not pooled with the primary residual estimand. Offline result files with redundant local absolute paths are omitted when an equivalent registered table or report is included. Historical SCPP, Travel, and IRCG development tables are documented in the appendix but intentionally excluded from the selected offline claim set.

The allowlist excludes API credentials, local configuration, raw hosted-model logs, caches, rate-limit ledgers, local usernames and workspace paths, unrelated operational canaries, development-only discovery output, manuscript author metadata, and repository history. Packaging performs source-to-copy hash checks, JSON-Schema validation, and a local-path/credential scan. It makes no hosted calls and never modifies source results.