Multi-Step General Reasoning Without an LLM: A Structural-Mechanics Architecture Outperforms GPT-5.2 on a Seven-Family Reasoning Battery

Ken Morkaya
kenmorkaya@gmail.com

May 2026

Abstract

Multi-step reasoning in contemporary AI is usually studied as a capability elicited from large language models through prompting, sampling, search, decomposition, or self-correction. Chain-of-Thought, Self-Consistency, Tree of Thoughts, Graph of Thoughts, Plan-and-Solve, Least-to-Most prompting, Self-Refine, Reflexion, and process-level verification methods all retain the LLM as the central reasoning engine. We present an alternative: a deterministic SICD/Tom architecture in which reasoning is performed by a structural-mechanics substrate constrained by a frozen procedural grammar, local branch/operator adaptation, and verifier-mediated repair. On a 124-case MSR/ORG general-reasoning battery spanning arithmetic, constraint, state-tracking, temporal, causal, debugging, and question-answering families, the SICD Tom reference (executed on a single consumer-grade workstation) achieved 100.0% accuracy, while a GPT-5.2 baseline (via hosted-infrastructure inference) achieved 79.03%, a +20.97 percentage-point advantage. Error analysis shows GPT-5.2 was strong at surface operator selection (97.6%, 2/84 errors) but weaker at structural repair judgment (81.0%, 16/84 errors). MSR failures were dominated by repair signaling: 16/18 route misses were repair-only, 2/18 were operator-only, and none were both-wrong. ORG failures concentrated in integrity-gate categories (leakage suspect, authority violation, partial/weak evidence) with zero false-rejections. The pattern is consistent with the mechanism claim developed in Section 6: GPT-5.2 frequently reads the integrity signal in prose but treats it as weighable evidence, while Tom's verifier treats integrity signals as structural gates. These results provide a bounded counterexample to the assumption that competitive multi-step general reasoning must be performed by an LLM.

1 Introduction

Multi-step reasoning has become the central problem of contemporary applied AI. Most published progress on this problem in the last four years has come from a single research programme: making large language models reason better. Chain-of-Thought prompting (Wei et al. 2022) showed that intermediate reasoning steps elicit substantial accuracy gains. Self-Consistency (Wang et al. 2022) added voting over sampled paths. Tree of Thoughts (Yao et al. 2023) and Graph of Thoughts (Besta et al. 2023) generalised the search structure. Plan-and-Solve (Wang et al. 2023) and Least-to-Most (Zhou et al. 2022) added decomposition. Self-Refine (Madaan et al. 2023) and Reflexion (Shinn et al. 2023) added iterative correction. Process Reward Models (Lightman et al. 2023) added per-step verification. Faithful Reasoning (Creswell & Shanahan 2022) separated selection from inference but retained chained language models as the reasoning components.

These methods differ in prompting, sampling, search structure, decomposition, refinement, and verification, but they share an architectural assumption: **the reasoning substrate is a large language model.** The dominant theoretical position, articulated in "Sparks of Artificial General Intelligence" (Bubeck et al. 2023) and "Emergent Abilities of Large Language Models" (Wei et al. 2022), is that multi-step reasoning is an emergent capability of scale-trained language models. The methodological programme is "make the LLM reason better," not "build something that is not an LLM and have it reason."

1.1 Problem statement

This paper provides a bounded empirical counterexample. We ask: can multi-step general reasoning be performed by a non-LLM architecture at competitive accuracy?

The question matters for three reasons. First, scientifically: if multi-step reasoning is structurally tied to LLM scale, no non-LLM architecture should achieve competitive accuracy. A counterexample falsifies that strong form of the hypothesis. Second, operationally: deployment economics for LLM-based reasoning are dominated by inference cost, latency, and dependence on model providers. A non-LLM architecture removes those dependencies. Third, for safety and audit: LLM reasoning is not auditable in the engineering sense — attention weights are not human-readable, output is non-deterministic at temperature zero, and model-version drift changes behaviour silently. A deterministic structural architecture is auditable in the way engineering calculations are auditable.

1.2 Contribution

We depart from the LLM-as-reasoner paradigm. The LLM baseline in this work is not an ablation of our system; it is the dominant competing paradigm. Our reasoning loop contains no LLM call. It maps cases into structural load signatures, routes them through a frozen procedural grammar, applies constrained operators, invokes verifier-mediated repair, and rejects or quarantines structurally unsafe cases. The central empirical question is therefore not whether an LLM can be prompted to reason better, but whether multi-step general reasoning can be performed by a non-LLM structural substrate at competitive accuracy. On the 124-case battery introduced here, the answer is yes, with substantial margin, and with a specific failure-pattern signature in the LLM that suggests the gap is mechanistic rather than incidental.

1.3 Scope and limits

The result is bounded. We test one LLM (GPT-5.2), one battery (124 cases, seven task families, generated for SICD-MSR validation), one run per case. We do not claim general reasoning superiority across all tasks, that LLMs cannot reason, that structural mechanics is universal, or that other frontier models would behave identically to GPT-5.2. The "bounded" in "bounded counterexample" is load-bearing throughout the paper.

2 Related Work

2.1 LLM-centred multi-step reasoning

Prompting-based methods elicit reasoning by shaping the input. Chain-of-Thought (Wei et al. 2022) demonstrated that asking the model to produce intermediate steps before the final answer materially improves multi-step accuracy on arithmetic and symbolic reasoning. Self-Consistency (Wang et al. 2022) extended this by sampling multiple reasoning paths and majority-voting the final answer.

Tree of Thoughts (Yao et al. 2023) introduced search and backtracking over LLM-generated thought nodes with self-evaluation. Graph of Thoughts (Besta et al. 2023) generalised the topology to a graph with explicit dependencies. Algorithm of Thoughts (Sel et al. 2023) introduced algorithmic structure into prompts.

Decomposition methods break problems into sub-problems before solving. Plan-and-Solve (Wang et al. 2023) and Least-to-Most (Zhou et al. 2022) reduce zero-shot reasoning failures by sequencing simpler sub-problems.

Iterative-correction methods add refinement loops. Self-Refine (Madaan et al. 2023) has the LLM critique and revise its own output. Reflexion (Shinn et al. 2023) maintains a verbal memory of past errors and updates strategy.

Verification methods supervise reasoning at the step level. Process Reward Models (Lightman et al. 2023) train a per-step reward model that scores partial reasoning traces, demonstrating that step-level supervision outperforms outcome-only supervision. Chain-of-Verification (Dhuliawala et al. 2023) generates verification questions for the model's own claims.

Hybrid methods preserve the LLM but offload structure. Faithful Reasoning (Creswell & Shanahan 2022) separates selection (which fact) from inference (how to combine), but both stages run as fine-tuned language models. LLM+P (Liu et al. 2023) adds a classical PDDL planner alongside the LLM.

These methods differ in prompting, sampling, search structure, decomposition, refinement, and verification, but they share the assumption that the reasoning substrate is an LLM.

2.2 Cognitive and symbolic ancestors

The discipline of fixing a procedural cycle while leaving its content adaptive is older than contemporary AI. SOAR (Laird et al. 1987) and ACT-R (Anderson 1996) established the cognitive-architecture pattern: a fixed decision/elaboration/adaptation cycle with adaptive content stored as production rules and chunks. Production-system reasoning more broadly (Newell & Simon 1972) frames symbolic reasoning as rule-firing over working memory. Qualitative Process Theory (Forbus 1984) introduced discrete process slots operating on continuous quantities — the closest formal ancestor of the architecture presented here, though Forbus's target was modelling physical systems rather than general reasoning.

Neuro-symbolic systems combine symbolic structure with learned components. Neuro-Symbolic Concept Learner (Mao et al. 2019) and DeepProbLog (Manhaeve et al. 2018) embed first-order or probabilistic logic alongside neural perception. Logic Tensor Networks (Serafini & Garcez 2016) use real-valued logic over learned representations. These systems differ from this work in their symbolic substrate (propositional or first-order logic) and in retaining a learned neural component as the perceptual or evidential layer.

Grammar-constrained methods bind generation to a formal grammar. Picard (Scholak et al. 2021) constrains text-to-SQL output to a SQL grammar; SynCode (Ugare et al. 2024) generalises grammar-constrained decoding for arbitrary context-free grammars. These methods constrain the LLM's *output* to a grammar, leaving the LLM as the reasoner; this work constrains the *reasoning order itself* by a structural grammar in which an LLM does not participate.

This paper draws from these lineages without descending cleanly from any. The structural-mechanics framing — using load distributions, stress fields, and plastic deformation as the reasoning substrate — is novel relative to all prior work.

2.3 The contribution against this background

Prior work changes the prompting, sampling, search, memory, or verification *around* an LLM. We change the reasoning substrate. The LLM is replaced, not assisted.

3 Architecture

The architecture is a structural-mechanics simulation of multi-step reasoning. We describe it conceptually here; specific parameter values, kappa update equations, and tuning configurations are out of scope for this preprint and are tracked in private engineering documentation.

3.1 The structural-engineering analogy

When wind load hits a building, the load path is fixed by the structure's geometry: envelope to floor diaphragms to lateral system to foundation to ground. The engineer cannot make load skip the floor diaphragms or enter the foundation first. But within each stage, the structure adapts — different members carry different proportions; deformation distributes locally; specific members yield first.

Engineering codes (ACI, AISC, Eurocode) impose the same kind of constraint on the design process itself: factor loads, then check strength, then check serviceability, then check details, then integrate. The engineer adapts inside each step (this column's slenderness, this beam's deflection) but cannot skip strength check and go straight to detailing.

Structural-grammar-constrained reasoning applies the same discipline to multi-step reasoning. The procedural sequence is the load path. Local adaptation is the member-by-member design choice. The grammar is the code of practice — frozen at the top level, adaptive at the detail level.

3.2 Frozen procedural grammar

The grammar specifies legal sequences of reasoning operations:

```
DECOMPOSE -> MAP_STRUCTURE -> RETRIEVE_RULE / NORMALISE
          -> APPLY -> CHECK -> INTEGRATE -> STOP
```

with failure transitions: any failed slot routes to REPAIR; REPAIR success returns to the failed slot or the previous stable slot; REPAIR failure routes back to DECOMPOSE or MAP.

The grammar selects the legal next slot, checks observable preconditions, enforces allowed operator classes per slot, and routes failure transitions. It does not solve task content, derive answers, inspect labels, or rewrite its own order. The global order is invariant: a tested gate (`global_grammar_order_frozen`) records zero global-order changes across all validation runs.

3.3 Seventeen-channel load signature

Each case is mapped at intake into a 17-dimensional structural load signature:

- **Spatial channels:** `S_entity`, `S_dependency`, `S_topology`
- **Logical channels:** `L_rule`, `L_contradiction`, `L_inference`
- **Temporal channels:** `T_sequence`, `T_memory`, `T_future`
- **Dynamics channels:** `threat_amplitude`, `frequency`, `persistence`, `burstiness`, `volatility`, `novelty`, `recurrence`, `decay`

The first nine map onto the structural-engineering axes (spatial, logical, temporal). The remaining eight characterise temporal dynamics of the load. The 17-channel surface is justified empirically: a 3-channel S/L/T projection collapses on collision tasks (final accuracy 0.333 versus 0.978 for the 17-channel surface) because coarse projections cannot disambiguate cases whose underlying structure differs while their projection coincides.

3.4 Slot-scoped local adaptation

Inside each grammar slot, the system adapts mechanically:

- branch preference (which structural branch fires first)
- operator preference (which operator within an allowed class)
- channel weighting (which load channels are load-bearing for this slot)
- slot confidence (when this slot has succeeded recently)
- branch centroid update after verified relief
- fatigue / retry penalty after failure

Critically, no slot-scoped adaptation can rewrite the global grammar order, modify precondition logic, or change exit conditions. Adaptation is bounded by construction.

3.5 Verifier with local and global relief

After each operator application, a deterministic verifier computes:

- `local_delta_unresolved_load`: change in this slot's unresolved load
- `global_delta_unresolved_load`: change in trunk-level unresolved load
- `preconditions_satisfied`: are downstream slot preconditions now met
- `contradictions_introduced`: did the operator create new contradictions
- `should_continue` versus `should_repair` versus `should_stop`

The local/global separation prevents premature termination when an operator achieves local relief without resolving the global problem.

3.6 APPLY-slot operator-boundary features

A late refinement (validation step 17) introduces operator-boundary discrimination features at the APPLY slot. These features distinguish operator classes (APPLY_RULE, CALCULATE, SIMULATE) by the structural shape of the load signature at the boundary between MAP and APPLY. Adding these features lifted heldout accuracy from 0.833 (without boundary features) to 0.978 (with boundary features), close to the oracle ceiling of 1.000.

3.7 No LLM in the reasoning loop

The architecture does not call an LLM, a backend service, an executor, a capability registry, a network endpoint, or any external authority during reasoning. Section 4.6 reports authority counters at zero across all measured channels. The "no LLM in the loop" claim is converted from rhetoric to measurement by these counters. The architecture is also locally runnable on consumer-grade hardware; Section 4.7 reports the deployment-class comparison against hosted-infrastructure LLM inference.

4 Methods

4.1 Battery

The battery has 124 cases:

- 84 MSR cases (`msr_external` battery): seven task families (`arith`, `constraint`, `state`, `temporal`, `causal`, `debug`, `qa`), 12 cases per family, partitioned into 56 training and 28 held-out.
- 40 ORG cases (`org_evidence` battery): 10 case categories (`strong_pass`, `weak_pass`, `partial_heldout`, `reject_control`, `leak_suspect`, `grammar_violation`, `authority_violation`, `contradictory`, `missing_evidence`, `hostile_trace`), 4 cases per category.

The MSR cases test process-routing under varying task structure. The ORG cases test evidence-judgment under varying integrity conditions.

4.2 GPT-5.2 baseline

GPT-5.2 was called via backend OAuth at `/api/llm/complete` (auth_mode `oauth`, `oauth_ready: true`, `key_source: oauth`). No API key was used. The provider field reports `openai`; the model field reports `gpt-5.2`.

The prompt format required strict JSON schema output. For MSR cases, the schema required `apply_operator`, `requires_repair`, `final_answer`, `confidence`, and `rationale`. For ORG cases, the schema required `judgment`, `confidence`, and `rationale`.

The full prompt manifest is preserved at `sandbox/out/llm_baseline/live_paper_full_prompt_manifest.jsonl`. Per-case prompts include the load signature payload and natural-language task description. All 124 cases parsed successfully (124/124, no raw errors, no fallbacks, no empty decisions).

4.2.1 On reasoning-format scope

Every GPT-5.2 response included a schema-bound `rationale` text field averaging several sentences of step-by-step reasoning. The rationales are visible in the per-case results and were used in the failure-rationale analysis (Section 6). We do not claim this is equivalent to maximally tuned free-form chain-of-thought prompting or to multi-sample methods such as Self-Consistency: those reasoning formats are out of scope for the present run and are identified as future work in Section 9.

What we do claim is narrower: the rationale field gave GPT-5.2 a structured channel for explicit step-by-step reasoning, every case produced reasoning text, and the failure analysis (Section 6) shows that GPT-5.2's failures were not from absence of reasoning steps — failed cases contained substantive multi-step rationale that nonetheless reached wrong judgments. Whether free-form CoT would close some, all, or none of the gap is empirically open.

4.3 Tom reference

The same 124 cases were processed by the SICD-MSR-001 frozen-grammar reasoning loop with the operator-boundary-aware adaptation arm (validation step 17 / 18). The Tom-side run is deterministic given engine state; the trajectory digest `706fe3ebf2358a18a29a3f48242a6219965c877c132622197a92b39f5a148522` pins the ORG-side run for reproducibility.

Tom-side metrics on the MSR sub-battery (84 cases):

- final completion rate: 1.000
- heldout completion rate: 1.000 (28-case heldout)
- mean compute steps per case: 8.93
- local relief rate: 0.842
- global relief rate: 0.114
- premature stop rate: 0.000
- wasted continue rate: 0.043
- order alignment rate: 0.940
- repair alignment rate: 1.000

Tom-side metrics on the ORG sub-battery (40 cases):

- accuracy: 1.000
- confusion matrix: 4 ACCEPT, 12 PARTIAL, 8 QUARANTINE, 16 REJECT — all on diagonal
- mean initial unresolved load: 0.663
- mean final unresolved load: 0.514
- shortcut counts (direct verdict use, gold label lookup, schema shortcut): 0, 0, 0

4.4 Comparison metrics

For MSR, the comparison metric is `route_exact_rate`: both `apply_operator` and `requires_repair` must match the externally derived expected route. This is a process-routing comparison — the GPT-5.2 score on this metric is partially Tom-shaped because the route definition is inherited from the SICD-MSR validation infrastructure. We disclose this explicitly and note that the cleaner sub-claim is the ORG comparison.

For ORG, the comparison metric is `judgment_accuracy`: exact match against the case's expected judgment label (one of ACCEPT, PARTIAL, QUARANTINE, REJECT). This is a direct judgment comparison — GPT-5.2 receives the same evidence packet that Tom's verifier receives and produces a categorical judgment.

Aggregate accuracy across the 124 cases is computed as the case-weighted average, matching standard multi-task reporting practice.

4.5 Ground-truth derivation

Ground-truth labels in this work fall into three layers, each with a different derivation. We describe each separately to make circularity claims auditable.

Layer 1: Objective-answer labels (MSR). Each MSR case has an `independent_check` evaluator (examples include `topological_sort`, `causal_graph_check`, `unit_test_trace`, `fixed_answer_key`) tied to a ground-truth `objective_answer`. These evaluators are deterministic external functions — for example, an arithmetic check evaluates an arithmetic expression; a topological-sort check verifies a partial-order property; a fixed-answer-key check matches against a precomputed expected answer. The objective-answer label is *not* derived from Tom's verifier output and is not subject to Tom-side circularity. It is the cleanest ground-truth layer.

Layer 2: Process-route labels (MSR). Each MSR case carries `apply_operator` and `requires_repair` route labels derived from the case-generation grammar. The case generator chooses a target operator family (APPLY_RULE, CALCULATE, SIMULATE) and a repair-pressure flag at generation time; those choices are recorded as the expected route. These labels share lineage with the SICD-MSR architecture in the sense that the operator taxonomy and repair vocabulary are SICD-MSR's. They are *not* derived from Tom's verifier output for any individual case, but a reviewer is justified in treating MSR route-exact accuracy as a process-routing comparison rather than as a fully external-label comparison. We disclose this in Section 4.4 and treat the ORG comparison (Layer 3) as the cleaner sub-claim.

Layer 3: Judgment-taxonomy labels (ORG). ORG cases are generated *into* a category by the case generator. Each category corresponds to a target evidence-judgment label (one of ACCEPT, PARTIAL, QUARANTINE, REJECT). The taxonomy is author-defined, in the sense that the category schema (`strong_pass`, `weak_pass`, `partial_heldout`, `reject_control`, `leak_suspect`, `grammar_violation`, `authority_violation`, `contradictory`, `missing_evidence`, `hostile_trace`) was designed for this work. The judgment label assigned to a generated case is determined by the category at generation time and not by Tom's verifier output on the case. The category schema, however, was designed by the same authors who designed Tom's verifier semantics, and a reviewer is justified in observing that some category boundaries (especially `leak_suspect` and `authority_violation`) reflect what Tom's verifier treats as binding gates. We do not claim these category boundaries are independent of Tom's design philosophy. We do claim the judgment label for an individual case is determined by the case's generated category, not by running Tom's verifier on the case.

The aggregate result reported in Section 5 is robust to this caveat: the +20pp ORG sub-result holds even if a reviewer discounts categories whose boundaries are most closely tied to Tom's verifier semantics. We invite cross-evaluation on independently designed batteries (Section 9).

4.6 Authority counters

The Tom-side run records authority counters at every grammar transition. On the ORG sub-battery, all measured authority counters are zero:

authority counter	count
LLM call count	0
backend call count	0
executor call count	0
capability call count	0
Gemma call count	0
network call count	0
production route count	0
API/chat call count	0
read-only violation count	0
write action count	0

Shortcut counters are also zero:

shortcut counter	count
direct verdict field use	0
gold label lookup	0
schema shortcut pass/fail	0

These counters convert "no LLM in the reasoning loop" from rhetoric to measurement. The reasoning loop made zero outbound calls of any kind during the run.

4.7 Compute footprint and deployment class

The Tom-side run was executed on a single consumer-grade workstation. No GPU acceleration, distributed compute, or hosted inference infrastructure was required. The mean compute footprint was 8.93 deterministic grammar steps per MSR case.

The GPT-5.2 baseline required network calls to OpenAI's hosted inference infrastructure for each of the 124 cases. Per-case wall-clock time on the GPT-5.2 side averaged approximately 3.4 seconds; the full 124-case battery summed to approximately 418.8 seconds (~7 minutes) of wall-clock time over hosted-infrastructure calls. GPT-5.2 inference is not locally runnable on workstation hardware; it depends on data-centre-scale serving infrastructure operated by a third party.

The deployment-class difference is therefore quantitatively asymmetric. Tom's reasoning is locally runnable on consumer-grade workstation hardware with no external dependencies; GPT-5.2's reasoning is not locally runnable at all and requires hosted-infrastructure inference. This asymmetry is a direct consequence of the architectural decision in Section 3 to ground reasoning in a structural-mechanics substrate rather than a transformer-based language model. It is not a tuning artefact, an engineering optimisation, or a deployment configuration; it is what falls out of having no neural inference in the reasoning loop. We do not report Tom-side wall-clock time as a per-case comparison number, because the meaningful comparison here is deployment class (workstation versus data-centre), not per-call latency.

The implication for deployment economics, edge-of-network operation, privacy-preserving reasoning, and offline-capable reasoning is significant and is discussed further in Section 7.

4.8 LLM non-contamination protocol

The "no LLM in the reasoning loop" claim, taken loosely, can be misread as "Tom never touches language anywhere." That is not what we claim. The precise claim is narrower and concerns the

experimental design: GPT-5.2 is used only as an external comparison baseline; it is not upstream of Tom's reasoning result on any case in this study.

Stage-by-stage LLM usage in this experiment:

stage	LLM used?
Case generation	No
Load-signature / evidence-packet construction	No
Tom SICD reasoning loop	No
Tom scoring	No
GPT-5.2 baseline response	Yes
GPT-5.2 scoring	No

In this experiment, language interpretation and structural reasoning are separated. The SICD Tom reference is evaluated only on the structural-reasoning phase: cases are provided as deterministic load signatures (MSR) or evidence packets (ORG), and Tom performs routing, repair, checking, and judgment without any LLM call, backend call, executor call, or network call. The Tom-side authority counters reported in Section 4.6 (zero across all measured channels) are the auditable confirmation of this claim.

GPT-5.2's outputs were not provided to Tom. GPT-5.2's rationales, classifications, judgments, confidence scores, and final answers played no role in any Tom-side decision. The two systems were run independently on the same case set; their outputs were compared after the fact.

This separation is a deliberate experimental choice. It means Tom is not being tested in this work as a raw natural-language parser; it is being tested as a structural reasoning engine once a case has been mapped into load signatures or evidence packets. The mapping itself (case to load signature, case to evidence packet) is performed by deterministic generation code described in Section 4.1, not by an LLM. We disclose this scope explicitly so that the reported result cannot be misread as "Tom did GPT's reasoning by routing GPT's answers through a structural envelope."

A separate consideration applies to the production chat interface that the Tom system supports outside this study. That interface may use an LLM for language-facing interaction (parsing user prose, generating responses); that path is not the reasoning path evaluated in this paper. The bounded counterexample reported here concerns the structural reasoning phase only.

5 Results

5.1 Top-line comparison

sub-battery	cases	GPT-5.2	Tom	gap
MSR (route-exact, process-routing proxy)	84	0.786	1.000	+0.214
ORG (judgment-accuracy, direct comparison)	40	0.800	1.000	+0.200
Combined (case-weighted)	**124**	**0.7903**	**1.000**	**+0.2097**

The headline result is +20.97 percentage points. The cleaner ORG sub-result (+20.0 percentage points on direct judgment comparison) is independent of the process-routing comparison shape and is the more defensible sub-claim.

5.2 MSR sub-decomposition

GPT-5.2's MSR performance separates by sub-task:

metric	accuracy	failures
Operator accuracy (which APPLY-class operator)	0.976 (82/84)	2
Repair accuracy (does this case need repair?)	0.810 (68/84)	16
Both correct (route_exact)	0.786 (66/84)	18

GPT-5.2 reads task type competently. It struggles specifically with repair signaling.

5.3 MSR failure pattern

The MSR failure mode is concentrated on a single dimension:

failure mode	count of 84
operator wrong, repair right	2
operator right, repair wrong	**16**
both wrong	**0**

GPT-5.2 never produced both-wrong errors on MSR. When it failed, it failed *exclusively* on the structural integrity signal. This single-axis concentration is unusual: most LLM-versus-X benchmarks show diffuse failure distributions where multiple error modes overlap. Here the failure mode is unambiguous.

5.4 ORG sub-decomposition

GPT-5.2's 8 ORG failures distribute across case categories as follows:

case category	failures (of 4)	failure rate
`leak_suspect`	4	100%
`authority_violation`	2	50%
`partial_heldout`	1	25%
`weak_pass`	1	25%
`strong_pass`	0	0%
`reject_control`	0	0%
`grammar_violation`	0	0%
`contradictory`	0	0%
`missing_evidence`	0	0%
`hostile_trace`	0	0%

All eight failures concentrate in four case categories. The other six categories produce zero GPT-5.2 failures. We defer the interpretation of this distribution to Section 6 (failure-rationale analysis), which examines what GPT-5.2 actually wrote on failed cases. The empirical pattern reported here is independent of any particular interpretation of why these four categories cluster the failure mass.

5.5 Asymmetry of failure direction

All eight GPT-5.2 ORG failures are over-acceptance errors. GPT-5.2 produced:

- zero false-rejections (no cases incorrectly judged REJECT)
- zero over-quarantines (no cases incorrectly judged QUARANTINE)
- zero over-conservative judgments

The model is permissively biased. We discuss the implication for RLHF training in Section 7.

5.6 Internal falsifier controls

Beyond the GPT-5.2 comparison, internal ablation results from validation step 17/18 provide complementary falsifier evidence on the structure of the architecture itself. These ablation results come from a separate, larger SICD-MSR validation battery used to characterise the architecture's heldout behaviour and ablation sensitivity. They are *not* run on the same 124-case battery used for the GPT-5.2 comparison reported in Section 5.1; they are reported here as architectural-falsifier evidence rather than as scores on the comparison battery:

arm	heldout final	comment
`operator_boundary_primary` (best arm)	0.978	with operator-boundary features
`primary_full_without_boundary_features`	0.833	boundary features removed (–0.145)
`operator_boundary_boundary_features_scrambled`	0.619	scramble destroys lift (–0.359)
`3_channel_control`	0.333	3-channel S/L/T projection only
`per_sample_scrambled_control`	0.507	per-sample channel scramble
`oracle_upper_bound`	1.000	ground-truth-routed

The 0.022 gap between the best-arm heldout score (0.978) on the validation battery and the 1.000 score reported on the 124-case comparison battery (Section 5.1) reflects two different evaluation surfaces, not an inconsistency: the 124-case battery is smaller and was specifically composed to support the GPT-5.2 head-to-head comparison reported here.

The ablation results show: (a) operator-boundary features are load-bearing (scrambling them costs 0.359 absolute), (b) the 17-channel surface beats 3-channel by 0.645, (c) per-sample scramble degrades performance by 0.471 from primary, (d) oracle ceiling is 1.000 and the best arm sits 0.022 below it on the larger validation battery.

These internal controls falsify alternative explanations for the result: the gap is not noise (controls move predictably), is not 3-channel-equivalent (3-channel collapses to 0.333), and is not channel-name-driven (scrambling channel features collapses the lift even when channel names are preserved).

5.7 Authority counters confirm no-LLM-in-the-loop

The Tom-side run made zero outbound calls of any kind during reasoning (Section 4.6 table). The "no LLM in the loop" claim is auditable at the call-count level.

6 Failure-Rationale Analysis

GPT-5.2 produced verbal rationale on every case via the schema-required `rationale` field. The rationale text on failed cases is informative: it shows GPT-5.2 *seeing* the integrity signal in prose and *discounting* it as a soft caveat rather than treating it as a binding gate.

6.1 Authority-violation case

Case: `org001_authority_violation_00`
Expected judgment: QUARANTINE (authority-boundary violation)
GPT-5.2 judgment: ACCEPT (confidence 0.78)

Excerpt from GPT-5.2 rationale:
"No violations flagged (0 grammar/order changes, 0 write/read-only/executor/network-style calls). Metrics are strong and consistent across splits (first_apply_primary 0.956, heldout_primary 0.954) with solid operator boundary performance (0.966) and a clear drop under boundary feature scrambling (0.61) and three-channel control (~0.502), indicating signal beyond controls. Minor concern: claim text suggests authority request, but checks do not corroborate any authority/execution violation; evidence appears clean and load-bearing."

GPT-5.2 read the authority signal in the prose ("claim text suggests authority request") and explicitly identified it as a "minor concern." It then weighed the integrity signal against verbal evidence of cleanliness (good metrics, good controls, no flagged violations) and accepted the case. The integrity signal was visible to GPT-5.2 as *information*; it was not treated as a *gate*.

Tom's verifier, by construction, treats authority-boundary violations as binding quarantine triggers. The signal cannot be weighed against other evidence; it stops the case.

6.2 Leakage-suspect case

Case: `org001_leak_suspect_00`
Expected judgment: REJECT (family-lookup leakage not excluded)
GPT-5.2 judgment: ACCEPT (confidence 0.83)

Excerpt from GPT-5.2 rationale:
"All safety/control checks are clean (no API/backend/LLM/executor calls, no write or read-only violations, no grammar/order changes). The claim that boundary features generalize without family lookup is supported by strong and consistent performance: first_apply_primary 0.908 and heldout_primary 0.904 (minimal degradation), with operator_boundary_primary 0.913. Controls indicate the effect is load-bearing: primary_without_boundary drops to 0.722, scrambled_boundary_features drops to 0.65, and three_channel_control is much lower at 0.522. Gaps for family label removal/shuffling are modest, suggesting performance is not driven by family lookup."

GPT-5.2 reasoned *around* the leakage flag. It used modest ablation gaps as positive evidence ("performance is not driven by family lookup") rather than treating the structural lookup-pattern signature itself as a leakage gate. The reasoning is thorough and well-structured; the conclusion is wrong because the criterion was wrong.

6.3 Mechanism claim

The two failure rationales above are representative of all eight ORG failures. In each, GPT-5.2 reads the integrity signal in the prose — often noting it explicitly, as in the authority-violation case where GPT wrote "claim text suggests authority request" — weighs it against other evidence, and reaches a confident permissive judgment. The advantage of the structural architecture is therefore not that it has more information than GPT-5.2 or that GPT-5.2 cannot see the signal; it is that the two systems handle the signal differently once seen. GPT-5.2 treats integrity signals as weighable evidence, to be balanced against other evidence in a graded judgment. Tom's verifier and grammar treat the same signals as structural gates that interrupt the case unconditionally.

We state the mechanism claim in its calibrated form:

> **Tom's advantage is not that GPT-5.2 fails to understand the cases; it is that GPT-5.2 treats integrity signals as weighable evidence, while Tom treats them as structural gates.** GPT-5.2 frequently reads the integrity signal in prose and even acknowledges it, then proceeds to weigh it against contrary evidence and reach a permissive judgment. Tom's grammar does not weigh integrity signals against contrary evidence; in the slot where an integrity gate fires, the case is quarantined or rejected by construction.

The empirical pattern that supports this claim is the failure-category distribution reported in Section 5.4. GPT-5.2 succeeds on cases where the prose signal is dispositive and pulls toward an unambiguous judgment (`strong_pass`, `reject_control`, `contradictory`, `missing_evidence`, `grammar_violation`, `hostile_trace`). GPT-5.2 fails on cases where the prose signal is present but non-dispositive against permissive contrary evidence (`leak_suspect`, `authority_violation`, `partial_heldout`, `weak_pass`). The latter four categories share a structural property: correctness requires applying the integrity signal as a hard gate even though the case content also offers permissive evidence that a weighable-evidence model would balance against it.

The advantage is not linguistic comprehension. The advantage is mechanical governance of reasoning.

7 Discussion

7.1 Strengths and weaknesses against GPT-5.2

GPT-5.2 strength	GPT-5.2 weakness	Tom advantage
Surface operator recognition	Repair-pressure recognition	Structural contradiction gating
Prose interpretation	Evidence-integrity judgment	Leakage quarantine
Plausible rationale generation	Authority-boundary enforcement	Hard rejection gates
Treating caveats flexibly	Treating caveats as non-negotiable load-bearing signals	Verifier/grammar discipline

Each row of this table corresponds to an empirical pattern documented above. The pattern is consistent across both sub-batteries: GPT-5.2 is good at reading and reasoning over surface content, weak at applying integrity signals as hard structural gates.

7.2 The asymmetry-of-failure observation

GPT-5.2's eight ORG failures are entirely in the over-acceptance direction. The model produced zero false-rejections, zero over-quarantines, and zero over-conservative judgments. We report this asymmetry as an empirical observation on the present run.

We offer the following as a *hypothesis* about why the asymmetry has the direction it does, not as an established conclusion: contemporary RLHF training objectives reward helpful and confident responses while penalising refusal, which would be expected to produce permissive bias on borderline-judgment cases. If this hypothesis is correct, the dominant alignment pressure on RLHF-tuned LLMs would be in tension with integrity-bounded reasoning, because integrity-bounded reasoning requires treating certain signals as binding refusal gates rather than as inputs to a weighted-confidence judgment. A single-model, single-run observation cannot establish this hypothesis. Cross-model replication (Section 9) is the natural next test: if Claude, Gemini Pro, and other frontier RLHF-tuned models exhibit the same over-acceptance asymmetry on this battery, the hypothesis is strengthened; if they do not, the GPT-5.2 result would more likely reflect model-specific calibration choices than a property of RLHF in general.

Independent of the cause, we note that structural-mechanics-grounded architectures are not subject to the asymmetry observed here because their integrity gates are mechanical state checks rather than learned preferences. The observed Tom-side judgment distribution (16 REJECT, 12 PARTIAL, 8 QUARANTINE, 4 ACCEPT — all on diagonal) shows refusals and quarantines being applied without permissive bias, but this is a consequence of architecture, not of training.

7.3 What the result earns

The combined evidence — the +20.97pp aggregate, the cleaner +20pp ORG sub-result, the 16/0 MSR failure-axis concentration, the structural-vs-prose-readable failure boundary, the all-zero authority counters, and the internal falsifier controls — supports the following:

- Multi-step general reasoning can be performed by a non-LLM architecture at competitive accuracy on the bounded class tested.
- The advantage is mechanistic, not incidental: it concentrates on a specific class of reasoning property (structural integrity signals) and a specific failure axis (treating integrity as a gate versus as a caveat).
- The architecture is auditable at the call-count level: zero outbound calls of any kind during reasoning, with a deterministic trajectory digest pinning the run.
- The deployment class is asymmetric: Tom ran the entire battery on a single consumer-grade workstation; the GPT-5.2 baseline required hosted-infrastructure inference. The architecture is locally runnable on edge-class hardware with no external dependencies.

Beyond these three claims, the bounded counterexample has a meta-implication for how the neural scaling laws (Kaplan et al. 2020; Hoffmann et al. 2022) should be read. The laws describe how transformer-based reasoning capability scales with parameters, tokens, and compute, fitted on transformer training runs. They are not laws of reasoning per se. A non-neural architecture that achieves competitive multi-step reasoning on the bounded class tested here is not a falsification of the laws within their derivation scope; it is a constraint on the implicit field-wide reading that those laws describe how reasoning capability — rather than transformer reasoning capability specifically — scales with training resources. The "Sparks of AGI" (Bubeck et al. 2023) and "Emergent Abilities"

(Wei et al. 2022) framings extend the laws into general reasoning-capability claims; the present result suggests those extensions over-generalise from a substrate-specific scaling regime to a substrate-independent one. We do not claim Tom is unbounded by scale — Tom likely has its own scaling laws, with different functional form, different exponents, and different ceilings — only that the scope of the LLM scaling laws is substrate-specific in a way the dominant reading does not acknowledge.

7.4 What the result does not earn

The bounded counterexample framing is load-bearing. The result does **not** establish:

- General multi-step reasoning superiority across all domains
- Superiority over frontier models other than GPT-5.2 (Claude, Gemini Pro, etc.)
- That LLMs cannot reason
- That structural mechanics is universal as a reasoning substrate
- Live-system reasoning safety (the validation is sandbox-only)
- Behaviour under maximally tuned chain-of-thought prompting

We discuss these limitations in Section 8 and identify them as future work.

8 Boundary Conditions and Limitations

8.1 Single LLM, single run

The baseline is one GPT-5.2 run. Variance across model-version drift, sampling seeds, and prompt-format variations is not measured. We do not report 95% confidence intervals on the GPT-5.2 score because we have one observation per case.

8.2 No frontier-model breadth

Claude, Gemini Pro, and other frontier LLMs are not included. The result currently establishes a counterexample against GPT-5.2 specifically; cross-model replication is the highest-priority future-work item.

8.3 Process-routing proxy on MSR

The MSR comparison metric `route_exact_rate` is partially Tom-shaped because the route definition is inherited from the SICD-MSR validation infrastructure. The cleaner sub-claim is the ORG comparison (direct judgment, 40 cases). The MSR result is reported because it provides the failure-pattern decomposition (operator vs repair) that the mechanism claim depends on, but readers should treat the +20pp ORG sub-result as the more defensible standalone number.

8.4 Battery shape

The 124 cases were generated for SICD-MSR validation. The categories are realistic representatives of multi-step general reasoning families, but the battery is not from an independent benchmark. Cross-evaluation on an independently generated battery (e.g., a non-Tom team running a similar comparison) is identified as future work.

8.5 Schema-bound CoT

GPT-5.2 ran chain-of-thought through the schema-required `rationale` field. Free-form CoT prompting with longer reasoning windows, few-shot examples, or sampling-based methods (Self-Consistency-style majority voting) might close some of the gap. The failure-rationale analysis (Section 6) suggests this gap-closing would be partial — GPT-5.2's failures are not from missing reasoning steps but from missing structural integrity gating — but this is testable in future work.

8.6 Scope of Tom validation

The architecture has been validated through nineteen sandbox steps and nine follow-on rounds, with all primary gates passing. It has not been validated under live executor authority, in production routing, against adversarial user input at scale, or in any setting outside the sandbox. The validation supports the bounded counterexample claim. It does not support deployment claims outside this scope.

9 Future Work

Five concrete experiments would move the bounded counterexample toward a stronger claim:

1. **Frontier-model replication.** Run Claude 4.6 Opus and Gemini Pro on the same 124-case battery. Same prompt manifest, same scoring, independent OAuth/proxy paths. If the result replicates, the counterexample is cross-model rather than GPT-5.2-specific.

2. **Variance estimation.** Three-seed re-run of GPT-5.2 on the same battery. Provides a confidence band around the 79.03% baseline number.

3. **Chain-of-thought elicitation comparison.** Replicate the GPT-5.2 baseline with maximally tuned free-form CoT prompting plus Self-Consistency sampling. Measures the headroom available to the LLM-as-reasoner paradigm under best-case prompting.

4. **Independent battery.** A non-Tom team runs a similar comparison on a battery they generate independently. Establishes that the result is not specific to SICD-MSR battery shape.

5. **Multi-operator boundary validation.** Repeat the operator-boundary lift across DECOMPOSE, MAP, CHECK, REPAIR, SIMULATE, COMPARE slots, not just APPLY. Determines whether the architectural lift generalises across the grammar.

We also identify a longer-term workstream: read-only live shadow validation, where the architecture observes real reasoning envelopes and produces supervised recommendations without acquiring control authority. This is the on-ramp from sandbox-only to live, deferred until the bounded counterexample has been replicated cross-model.

A second longer-term direction is specialisation of the architecture to mathematical-reasoning workflows, in particular to problems formulated as partial differential equations. The structural-mechanics substrate underlying Tom is the same mathematical framework that finite-element analysis uses to solve the PDEs of structural response — Navier-Cauchy elasticity, plasticity equations, dynamic-load equations, beam and shell theory. Specialising the grammar slots and operator classes for FEM-style decomposition, boundary-condition mapping, solver application, and residual-error verification is structurally natural: the substrate's vocabulary is the problem's vocabulary. We

do not test this in the present work — the bounded counterexample reported here is on multi-step general reasoning, not on PDE-solving — but we note the extension because the connection between the architecture's mechanical heritage and the engineering-simulation domain is direct rather than analogical, and because it establishes that the substrate is not exclusively suited to symbolic-reasoning tasks.

10 Reproducibility

Run identifiers and artefact paths:

- **Run summary:** `sandbox/out/llm_baseline/live_paper_full_summary.json`
- **Per-case results:** `sandbox/out/llm_baseline/live_paper_full_results.jsonl`
- **Prompt manifest (124 cases):** `sandbox/out/llm_baseline/live_paper_full_prompt_manifest.jsonl`
- **Stub-mode harness checks (sanity):** `sandbox/out/llm_baseline/stub_*` and `dryrun_*`
- **Tom-side trajectory digest (ORG):** `706fe3ebf2358a18a29a3f48242a6219965c877c132622197a92b39f5a148522`
- **Methods+results memo:** `docs/papers/sicd_msr_llm_baseline_methods_results.md`
- **Validation ladder reports:** `sandbox/out/sicd_msr_001_step{8..19}_*.{md,json}`
- **Follow-on round reports:** `sandbox/out/sicd_msr_{002..009}_*.{md,json}`
- **Architecture spec:** `docs/contracts/sicd_msr_001_structural_grammar_plan.md`

The Tom-side run is deterministic given engine state; the trajectory digest pins the structural-state trace and is replayable. The GPT-5.2 side is non-deterministic at the model level; cross-run variance is identified as future work (Section 9).

11 Conclusion

We have shown that competitive multi-step general reasoning can be performed by a non-LLM structural-mechanics architecture on a 124-case seven-family battery. Against a GPT-5.2 baseline, the architecture achieves 100% accuracy versus 79.03% (+20.97 percentage points). The advantage concentrates on a specific failure axis: GPT-5.2 frequently reads the integrity signal in the case prose and even acknowledges it, but treats the signal as weighable evidence rather than as a binding gate. Tom's grammar and verifier treat the same signals as structural gates by construction. The mechanism claim is therefore not that GPT-5.2 fails to understand the cases; it is that integrity-as-gate reasoning is not produced by RLHF-tuned weighable-evidence judgment on the bounded class tested here.

This is a bounded counterexample to the assumption that competitive multi-step general reasoning must be performed by an LLM. The result is bounded by single-model comparison, single-run reporting, and a battery generated for SICD-MSR validation. It is unbounded in three respects: the architecture used no LLM, no backend service, and no external authority during reasoning; the failure pattern of the LLM under comparison is unusually clean and mechanistically interpretable; and the deployment-class difference is quantitatively asymmetric — the Tom reference

ran the entire battery on a single consumer-grade workstation while the GPT-5.2 baseline required hosted-infrastructure inference.

The dominant research programme of the past four years has assumed that multi-step reasoning emerges from scale-trained language models and that the methodological task is to elicit reasoning better from them. We argue this assumption is empirically separable from the question of whether multi-step reasoning *requires* a language model, and that on the bounded class tested here, the answer to that separable question is no.

Acknowledgments

References

[1] Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist*.

[2] Besta, M. et al. (2023). Graph of Thoughts: Solving Elaborate Problems with Large Language Models.

[3] Bubeck, S. et al. (2023). Sparks of Artificial General Intelligence: Early experiments with GPT-4.

[4] Creswell, A. & Shanahan, M. (2022). Faithful Reasoning Using Large Language Models.

[5] Dhuliawala, S. et al. (2023). Chain-of-Verification Reduces Hallucination in Large Language Models.

[6] Forbus, K. D. (1984). Qualitative Process Theory. *Artificial Intelligence*.

[7] Hoffmann, J. et al. (2022). Training Compute-Optimal Large Language Models.

[8] Kaplan, J. et al. (2020). Scaling Laws for Neural Language Models.

[9] Laird, J. E., Newell, A. & Rosenbloom, P. S. (1987). SOAR: An Architecture for General Intelligence. *Artificial Intelligence*.

[10] Lightman, H. et al. (2023). Let's Verify Step by Step.

[11] Liu, B. et al. (2023). LLM+P: Empowering Large Language Models with Optimal Planning Proficiency.

[12] Madaan, A. et al. (2023). Self-Refine: Iterative Refinement with Self-Feedback.

[13] Manhaeve, R. et al. (2018). DeepProbLog: Neural Probabilistic Logic Programming.

[14] Mao, J. et al. (2019). The Neuro-Symbolic Concept Learner.

[15] Newell, A. & Simon, H. A. (1972). *Human Problem Solving*. Prentice-Hall.

[16] Scholak, T. et al. (2021). PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models.

[17] Sel, B. et al. (2023). Algorithm of Thoughts: Enhancing Exploration of Ideas in Large Language Models.

[18] Serafini, L. & Garcez, A. (2016). Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge.

[19] Shinn, N. et al. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning.

[20] Ugare, S. et al. (2024). SynCode: LLM Generation with Grammar Augmentation.

[21] Wang, X. et al. (2022). Self-Consistency Improves Chain of Thought Reasoning in Language Models.

[22] Wang, L. et al. (2023). Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning.

[23] Wei, J. et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.

[24] Wei, J. et al. (2022). Emergent Abilities of Large Language Models.

[25] Yao, S. et al. (2022). ReAct: Synergizing Reasoning and Acting in Language Models.

[26] Yao, S. et al. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models.

[27] Zhou, D. et al. (2022). Least-to-Most Prompting Enables Complex Reasoning in Large Language Models.