THE SERIALIZATION BOTTLENECK: SEPARATING STATE PRESERVATION FROM PROPERTY DERIVATION IN TOOL-USING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-using agents often return intermediate structured objects, such as polygons, graphs, and tables, to the language model as serialized text. The next step then asks the model for two different operations: derive a property that selects the branch, and re-emit the object as an argument of its next tool call. End-to-end benchmarks cannot tell which of the two fails. We isolate them with a controlled diagnostic benchmark over geometry, graph, and tabular pipelines and seven language models. Under standard non-thinking inference, models read surface facts far better (51.3%–99.5%) than they derive global properties (20.8%–49.0%). In multi-step pipelines, calls that must re-emit the full object abort in up to 99.7% of tabular runs. Trace audits show early semantic corruption of the payload, not truncation, and schema-constrained decoding does not prevent it. In the most abort-prone setting (Qwen3-32B, tabular), keeping the full observation but passing an immutable reference as the argument cuts aborts from 99.7% to 1.2%, so the aborts come from re-emitting the payload, not from seeing it. For recurring branch predicates, runtime-computed summaries turn derivation into lookup; predicates outside the schema still need on-demand inspection or code execution. Deriving a property from a structured state and re-emitting that state in the next action are separate failure points, and each needs its own interface fix.

1 INTRODUCTION

Language models now call external tools and APIs (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024), and agents chain these calls into multi-step workflows. In frameworks such as LangChain, AutoGen, CrewAI, and ReAct loops (Yao et al., 2023), each tool output is serialized to text, and the model reads it to write the next call. For a structured output, such as a WKT polygon, an edge list, or a CSV table, that call asks for two operations. The model must *derive* the property that picks the branch (Is the polygon convex? Is the graph bipartite?), which means running an algorithm, such as a shoelace sum or a graph search, inside generation. It must also *re-emit* the object, often hundreds of tokens long, as an argument.

Existing evaluations do not separate the two operations. Tool benchmarks score single calls (Li et al., 2023; Patil et al., 2024), and agent frameworks are evaluated on end-to-end success without controlling the intermediate representation (Lu et al., 2023; Paranjape et al., 2023). When a pipeline fails, we cannot tell whether the model misjudged the branch or corrupted the object it passed on. The distinction matters because the fixes differ: a stronger reasoner still has to copy the payload, and a reference-passing API still leaves the property uncomputed. We call this two-part burden the **serialization bottleneck**.

We isolate the two operations with a controlled diagnostic study (Figure 1) over geometry, graph, and tabular pipelines, 25 properties, and seven language models ($N = 43{,}740$ property queries; GPT-4.1 serves only as a scale check in Experiment 1). Each condition changes one part of the interface: *handles* pass an object ID instead of the payload, *summaries* expose a runtime-computed property, and an `ArtifactRef` control keeps the full object in view while replacing only the action argument with an immutable reference. We find:

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.

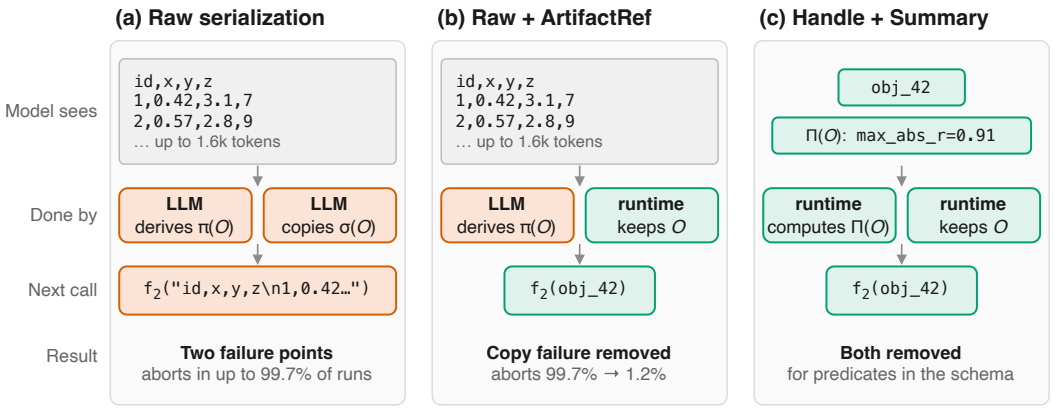


Figure 1: **Three interfaces for tool-using agents.** Orange boxes are work the LLM must do, where failures occur; green boxes are work the runtime does. (a) Raw serialization: the LLM sees the object as text, derives the branch property, and copies the object into its next call. (b) Raw + ArtifactRef: the LLM still sees the full object, but the call passes an opaque reference; in the most abort-prone setting (Qwen3-32B, tabular), aborts fall from 99.7% to 1.2%. (c) Handle + Summary: the runtime keeps the object and exposes a designated recurring property, so a covered branch becomes a lookup. Predicates outside the schema are not covered.

1. **Models read tokens but fail to derive properties (RQ1, Section 4).** Under standard non-thinking inference, surface properties reach 51.3%–99.5% accuracy but derived global properties only 20.8%–49.0%, a gap that persists up to 1.6T parameters.
2. **Errors compound at property-dependent steps (RQ2, Section 5).** Pass-through steps stay at 97.0%–100.0% accuracy, while steps that branch on a derived property fall to 11.6%–48.1%. Aborts end 16.2%–73.5% of raw graph pipelines and up to 99.7% of tabular ones.
3. **Aborts come from re-emitting the payload, not from reading it (RQ3, Sections 5–6).** Most aborts are early, schema-valid payload corruptions, so JSON-schema decoding helps only modestly. In the most abort-prone setting (Qwen3-32B, tabular), `ArtifactRef` cuts aborts from 99.7% to 1.2% while the model still sees the full object. For recurring predicates, runtime summaries turn derivation into lookup (`Handle+Summary` reaches 75.3%–99.4% cascading accuracy in geometry); outside the schema, accuracy falls to 70.5%–82.5%, so agents still need on-demand inspection or code execution.
4. **Optional inspection is skipped (RQ4, Section 7; Appendix F.1).** Models skip the `describe(object)` tool on 19.6%–54.0% of geometry failure gates and guess with uncalibrated confidence.
5. **Test-time reasoning is a costly partial fix (RQ5, Section 7; Appendix G.1).** Thinking lifts arithmetic properties (70.8% on Class 3), but non-binary invariants stay at 24.0%–41.4% at up to 32.7× the cost.

The practical rule is simple: pass references instead of payloads, compute known recurring predicates in the runtime, and keep sandboxes for open-ended derivation (Section 9).

2 Problem Formulation

2.1 The Compositional Tool-Use Loop

Let $\mathcal{O}$ be in-memory objects (Shapely polygons, NetworkX graphs, pandas dataframes (McKinney, 2010)). A deterministic tool library $\mathcal{T} = \{f_1, \ldots, f_M\}$ holds functions $f_i : \mathcal{O}^{d_i} \times \Theta_i \to \mathcal{O}$. A workflow has depth K, and $\sigma : \mathcal{O} \to \Sigma^*$ serializes each tool output to text. At step $t \in \{1, \ldots, K\}$, model $\mathcal{M}$ sees context $C_t = (I, s_1, a_1, \ldots, s_t)$: instruction I, earlier calls a_τ, and serialized states $s_\tau = \sigma(O_\tau)$. It predicts the next call $a_{t+1} = (f_{i_{t+1}}, \hat{\theta}_{t+1}, \hat{O}_{t+1}) \sim \mathcal{M}(C_t)$: a tool, its parameters, and its object argument.

2.2 EXPOSED VS. DERIVATION-REQUIRED PROPERTIES

The next tool $f_{i_{t+1}}$ and arguments $\hat{\theta}_{t+1}$ depend on a structural property $\pi(O_t) \in \mathcal{V}$. Standard serializations are injective (the text determines the object), but differ sharply in what they expose:

Definition 1 (Explicitly Exposed vs. Derivation-Required). *A property $\pi(O)$ is **explicitly exposed** under σ if recoverable via (i) bounded-window syntactic projection (e.g., `edge_count` in `GRAPH (n, m)`, column dtypes), or (ii) a **parallel reduction over token positions with no serial dependency** (e.g., coordinate $\min$ / $\max$ for `bbox`). It is **derivation-required** if computing $\pi(O) = g(\sigma(O))$ needs serial multi-token arithmetic, combinatorial search, or higher-order state aggregation.*

We assign the 25 evaluated properties to four serial-derivation classes (Appendix C.3). **Class 1 (Explicit / Header)**: e.g., `bbox`, `edge_count`[1], `column_dtype`. **Class 2 (Linear Aggregation)**: e.g., `vertex_count`, `degree_of_node_0`, `row_count`, `null_count`. **Class 3 (Multi-Step Arithmetic)**: e.g., polygon `area`, `perimeter`, `centroid`, `aspect_ratio`. **Class 4 (Combinatorial / Topological / Higher-Moment)**: binary predicates (e.g., `convex`, `is_bipartite`) and non-binary invariants (e.g., `diameter`, `chromatic_number`, `skewness`, `correlation`). These labels are benchmark annotations, not formal complexity classes. Transformer expressivity results motivate the distinction (Merrill & Sabharwal, 2024; Li et al., 2024) but give no lower bounds for single benchmark instances.

2.3 INTERFACE CONDITIONS AND METRICS

State Conditions. Four primary conditions cross **state preservation** (text vs. in-memory handles) with **decision exposure** (raw vs. deterministic summary $\Pi(O_t)$): Raw (text), Augmented (text + summary), Handle Only, and Handle + Summary (Table 2, Appendix A). We add two diagnostic controls. `Raw+Constrained` enforces strict JSON-schema grammar decoding. `Raw+ArtifactRef` keeps $\sigma(O_t)$ in context but passes an immutable server-side object ID, which separates reading the state from re-emitting it. Each summary exposes key properties (Appendix D.2): $\Pi_{\text{geo}}(O)$ contains `area`, `perimeter`, `vertex_count`, `convex`, `centroid`, and `bounds`; $\Pi_{\text{graph}}(O)$ contains `node_count`, `edge_count`, `avg_degree`, `density`, `bipartite`, `connected`, `cycle`, and `diameter`; and $\Pi_{\text{tab}}(O)$ contains `column_names`, `row_count`, `col_count`, `null_count`, `mean`, `max_abs_r`, `outlier`, and `monotonic`.

Execution Modes and Correctness. In **oracle mode**, each step receives the ground-truth state O_t^*; per-step accuracy is $p_t = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[a_{i,t} = a_{i,t}^*]$. In **cascading mode**, tools run on the actual runtime state, so aborts and corruption cascade. Any malformed call, parse failure, or exception halts the pipeline without retry. Strict *trajectory equality* requires $\bigwedge_{t=1}^{K}(a_t = a_t^*)$ and localizes the first wrong decision; semantic final-state equivalence (Section 5) credits valid alternative action sequences. On clean runs, trajectory correctness factors as $P(\bigwedge_{t=1}^{K} a_t = a_t^*) = \prod_{t=1}^{K} P(a_t = a_t^* \mid a_{<t} = a_{<t}^*) = \prod_{t=1}^{K} S(t)$. This separates per-step derivation survival $S(t)$ from the abort hazard $h_{\text{abort}}(t)$ (Appendix D). We also report Augmentation Gain $\text{AG} = \text{acc}_{\text{augmented}} - \text{acc}_{\text{raw}}$ and Object-Handle Gain $\text{OHG} = \text{acc}_{\text{handle}} - \text{acc}_{\text{raw}}$.

3 EXPERIMENTAL SETUP

To separate property derivation from payload re-emission, we test 25 properties in three domains (2D geometry, graphs, tabular data), each with Simple, Medium, and Hard tiers (Table 4, Appendix B). We call Class 1–2 properties *local* and Class 3–4 properties *global*.

3.1 DOMAINS, PROPERTIES, AND PIPELINES

Geometry (WKT). We serialize 300 valid 2D polygons in $[0, 1000]^2$ (3–40 vertices, half clockwise) as Well-Known Text (`POLYGON((x1 y1, ...))`). Ground truth comes from Shapely 2.0 (Gillies et al., 2022), and `centroid` is the area centroid (`Polygon.centroid`). We eval-

[1]With edge-list headers (`GRAPH (n, m)`). Under adjacency lists it moves to Class 2 (Ablation 1, Appendix H.1).

uate 9 properties: `vertex_count`, `bbox` (local); `centroid`, `area`, `perimeter`, `convex`, `orientation`, `aspect_ratio`, `edge_length_variance` (global).

Graph Theory (Edge Lists). We serialize 300 undirected graphs (6–80 nodes) as sorted edge lists with a header (`GRAPH (n, m) :`). They span five families (Erdős–Rényi, Barabási–Albert, Watts–Strogatz, random bipartite, random planar); half are bipartite and half planar. Ground truth comes from NetworkX (Hagberg et al., 2008) and DSATUR (Brélaz, 1979). We evaluate 8 properties: `edge_count`, `degree_of_node_0` (local); `triangle_count`, `is_bipartite`, `is_planar`, `diameter`, `chromatic_number`, `avg_clustering` (global).

Tabular Data (CSV). We serialize 300 tables (8–50 rows, 4–12 columns) as CSV; 250 are synthetic and 50 are curated from UCI and OpenML. We plant correlations ($r \in \{-0.9, \ldots, 0.9\}$), monotonic columns, and functional dependencies. We evaluate 8 properties: `column_dtype`, `row_count`, `null_count` (local); `is_monotonic`, `has_outlier` (Tukey's $3 \times$ IQR), `func_dependency`, `correlation` (Pearson), and `skewness` (bias-corrected Fisher–Pearson G_1) (global; conventions in Appendix C.2).

Pipelines. Experiments 2 and 3 chain deterministic tools into multi-step pipelines, for example `buffer`, `simplify`, and `convex_hull` for polygons; `remove_nodes` and `complement` for graphs; and `filter_rows` and `sort_values` for tables (Appendix D.1). Pipelines follow three designs. *Pass-through* pipelines apply tools in a fixed order, $O_t = f_t(O_{t-1})$, so the model only carries the state forward. *Property-conditioned* pipelines branch on a property of the current object, if $\pi(O_{t-1}) \bowtie \tau$ then f_A else f_B, and we call such a branch step a *gate*; for example, the next tool depends on whether the graph's diameter exceeds 3. *Multi-object* pipelines select the candidate that maximizes a property, $O^* = \arg\max_O \pi(O)$.

3.2 Models and Inference

We test seven pinned model snapshots (Table 5). Five run the full benchmark: **DeepSeek-V4-Flash**, **Qwen3-32B**, **Llama-4 Scout**, **Gemini-2.5-Flash-Lite**, and **GPT-4.1-mini**. Two frontier models run on a 20% stratified subsample (60 objects/domain): **DeepSeek-V4-Pro** (zero-shot non-thinking mode) and **GPT-4.1** (`openai/gpt-4.1`). Queries use greedy decoding ($T = 0.0$, top-$p = 1.0$), a 512-token completion cap, and zero-shot prompting ({`"answer": <value>`}). Audits show negligible decoding variance ($< 0.4\%$); confidence intervals reflect sampling uncertainty ($N = 43{,}740$). A continuous prediction $\hat{y}$ is correct if $|\hat{y}-y|/|y| \leq \tau$ when $y \neq 0$, or $|\hat{y}-y| \leq \tau$ when $y = 0$, with $\tau = 0.05$ for every continuous property. Exactly-zero targets are rare (Appendix C.2); Appendix C.1 sweeps $\tau \in \{0.01, 0.10\}$. Discrete properties use exact match, and parse failures count as incorrect. Table 3 (Appendix B) indexes every primary estimand.

4 Experiment 1: Models Read Tokens but Fail to Derive Properties

Experiment 1 asks whether a model can derive a branch property from a serialized object. We query each property directly under zero-shot prompting.

4.1 The Locality Cliff Across Domains

Models parse explicit tokens but collapse on properties that need derivation. Figure 2 and Table 7 (Appendix C) show the results. Class 1 lookups are reliable, but even linear Class 2 counting, such as `vertex_count` and `row_count`, is unreliable (Class 2 mean 66.0%). These serial-derivation classes (Appendix C.3, Table 15) are our main explanatory taxonomy; Δ_{LG} is a coarser split whose size partly reflects class composition. The gap appears in every domain:

- **Graphs:** Local properties (`degree_of_node_0`, `edge_count`) average 97.1%, but global accuracy collapses (mean gap **+68.7 points**).
- **Tabular data:** Models identify column dtypes and null counts but almost never recover Pearson correlation or skewness (mean gap **+49.1 points**).
- **Geometry:** Models parse bounding boxes precisely but fail on polygon area, perimeter, and edge-length variance (mean gap **+40.0 points**).

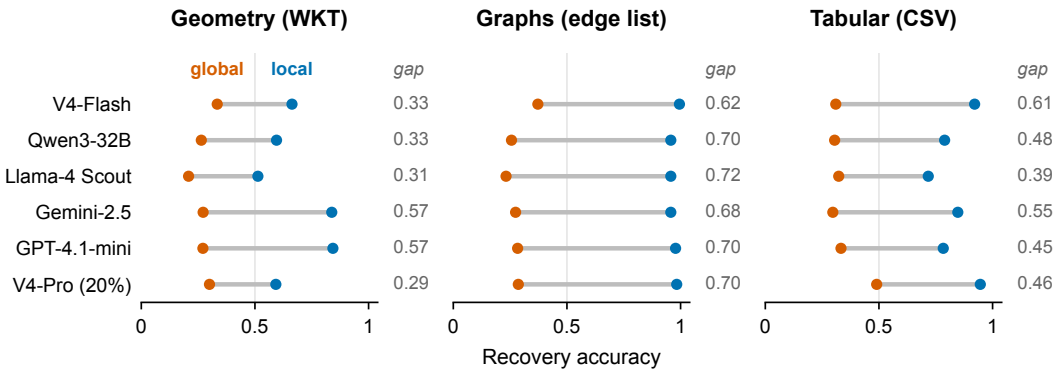


Figure 2: **Recovery accuracy for local and global properties (Experiment 1).** Greedy, zero-shot, non-thinking inference on six models (GPT-4.1 is compared in Section 7). Each row joins a model's global (orange) and local (blue) accuracy, and the right column of each panel gives the gap Δ_{LG}. Local properties (Class 1–2: header lookup or one counting pass) reach 51.3%–99.5%, while global properties reach only 20.8%–49.0%. The gap (+29.2–72.5 points) persists in larger non-thinking models.

Manual inspection of failed generations agrees (Table 41, Appendix I): most errors are structural reasoning errors (46.2%) or arithmetic errors (28.5%), not parse failures (3.2%) or truncation (1.6%).

4.2 Model Scale and Object Size

Under this fixed non-thinking serving configuration, model scale alone does not close the derivation gap. DeepSeek-V4-Pro is a 1.6-trillion-parameter MoE, yet its global accuracy is only 30.0% in geometry and 28.6% in graphs (Tables 7 and 8, Appendix C). Its local-global gaps stay close to those of the smaller Qwen3-32B and Llama-4 Scout. Appendix G.1 evaluates test-time compute.

Scale helps discrete pattern matching, not multi-step numerical calculation. V4-Pro reaches 49.0% global tabular accuracy, above the five full models (∼31%). The whole gain comes from discrete predicates: `is_monotonic`, `has_outlier`, and `func_dependency` (Table 11). On continuous statistics, V4-Pro scores 0.0% on both `correlation` and `skewness`.

Accuracy falls with object size, mostly on global properties. V4-Flash accuracy declines monotonically from Simple to Medium to Hard tiers: 45.3% → 40.0% → 36.8% in geometry, 56.9% → 51.9% → 49.5% in graphs, and 64.3% → 57.4% → 50.5% in tabular data ($n = 900$, 800, and 1,000 queries per tier). The drop is concentrated in global properties; local extraction barely depends on object size.

5 Experiment 2: Errors Compound at Property-Dependent Steps

We test how the serialization bottleneck affects multi-step tool-use agents on 924 geometry pipelines (depth $K \leq 10$), 600 graph pipelines ($K \leq 7$), and 600 tabular pipelines. Each pipeline runs under four state-passing conditions (Table 2, Appendix A) in oracle mode and in cascading mode.

5.1 Compounding by Pipeline Type

How fast accuracy decays depends on whether a step must derive a property (Figure 3; Table 16, Appendix D):

- **Pass-through pipelines stay accurate.** Forwarding state without branching keeps raw per-step accuracy at or above 97.0% for all six models (Table 17).
- **Property-conditioned pipelines collapse.** Branching on a derived property drops raw per-step accuracy to at most 48.1% (Table 17). On V4-Flash, this failure is front-loaded: most raw tabular

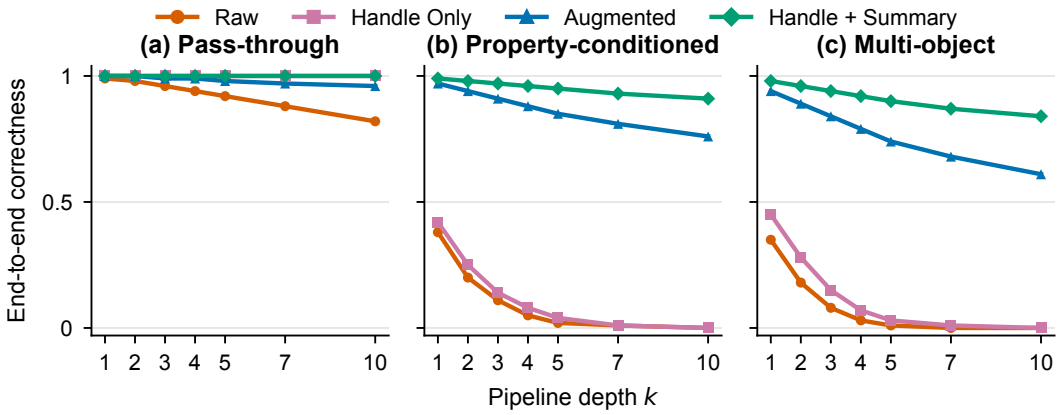


Figure 3: **End-to-end correctness versus pipeline depth k (Experiment 2).** (a) Pass-through pipelines forward state without branching; accuracy stays high (97.0%–100.0% per step, Table 17), and handles retain 100% because nothing is re-emitted. (b) Property-conditioned pipelines branch on a derived property; accuracy decays with depth, handles alone do not help, and only summaries (`Augmented`, `Handle+Summary`) sustain execution. (c) Multi-object comparison decays most, and `Handle+Summary` removes the bottleneck.

Table 1: **Interface controls on the two abort-extreme cells (Experiment 2).** Abort and Casc(All) are in %, and Casc(All) counts aborted pipelines as failures. Sem. counts aborts caused by schema-valid but semantically corrupted payloads (Table 26). Each condition has $N = 600$ pipelines; the ArtifactRef cohorts have $N = 521$ (tabular) and 281 (graphs). β_{pres} is the oracle step-accuracy gain from replacing this text interface with handles (Eq. 1); near zero means the interface already avoids the re-emission failure. Full results are in Tables 25 and 27. Artifact references remove the aborts; constrained decoding does not.

	Tabular (Qwen3-32B)				Graphs (Llama-4 Scout)			
Interface	Abort ↓	Sem.	Casc(All) ↑	β_{pres}	Abort ↓	Sem.	Casc(All) ↑	β_{pres}
Raw	99.7	563	0.3	+0.891	73.5	390	17.3	+0.531
Raw + Constrained	99.3	466	0.7	+0.874	53.8	272	29.7	+0.400
Raw + ArtifactRef	1.2	2	76.0	−0.025	46.3	123	42.7	+0.180
Handle + Summary	2.5	1	91.3	n/a	4.3	23	85.2	n/a

pipelines abort at the first step, and survivors stay stable (Figure 5). Raw graph aborts rise with depth, while `Handle+Summary` keeps the abort hazard near zero throughout (Appendix D).

- **Multi-object comparison also collapses.** Selecting among serialized candidates by a global property drops raw per-step accuracy to at most 45.0% (Table 17).

Strict scoring does not drive the gap. We also credit valid alternative action sequences that reach the reference final object (semantic final-state equivalence; DeepSeek-V4-Flash, Appendix E.5). This lifts Raw cascading accuracy by a few points, but `Handle+Summary` still exceeds Raw by at least 19.8 points in every domain.

5.2 Disentangling State Preservation from Decision Exposure

Separating state preservation (handles) from decision exposure (summaries) reveals distinct failure modes. Here `Handle+Summary` uses the Π-Oracle schema, whose summary always includes the property the step branches on (held-out schemas in Section 6.1).

1. **Handles alone do not fix property-conditioned tasks.** `Handle Only` keeps pass-through steps fully correct but gives negligible gain on property-conditioned tasks (OHG $\leq +0.040$ for Gemini, GPT-4.1-mini, and Qwen3-32B; Table 16).

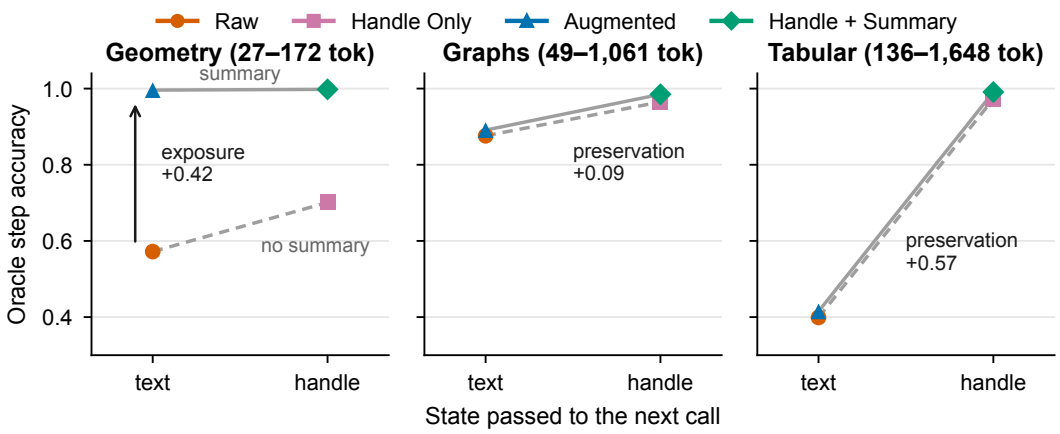


Figure 4: **Cell means of the 2×2 factorial design (Experiment 3).** Oracle step accuracy of DeepSeek-V4-Flash for the four conditions; solid lines join the two conditions with a summary and dashed lines the two without. Contrasts, clustered CIs, and the five-model re-fit are in Tables 28 and 29. In geometry, exposure adds $+0.424$ on raw state and $+0.297$ with handles, while preservation adds only $+0.002$ once the designated summary is present.

2. **Summaries restore execution in short serializations (geometry).** In geometry, summaries (`Augmented`, `Handle+Summary`) raise step accuracy by $+40.6$–50.9 points (AG). `Handle+Summary` restores cascading accuracy to 75.3%–99.4% (Table 16).
3. **Handles eliminate abort cascades in long serializations (graphs, tables).** Raw serialization aborts up to 73.5% of graph pipelines and 99.7% of tabular ones (Tables 18, 23). `Handle+Summary` suppresses these aborts and keeps oracle end-to-end accuracy high (Table 23).
4. **Constrained decoding enforces structure, not payload validity.** Provider-enforced JSON-schema decoding (`Raw+Constrained`; Appendix E.2) removes envelope syntax errors but lowers aborts and β_{pres} only modestly (Table 1). The reason is that 88%–94% of raw aborts are *schema-valid but semantically corrupted* payloads (Table 26), which no output grammar can fix.
5. **Artifact references isolate payload re-emission.** `Raw+ArtifactRef` keeps the full $\sigma(O_t)$ in the observation but replaces the action argument with an immutable server-side ID (Table 2, Appendix E.2). In the Qwen3-32B tabular cell, this removes nearly all aborts and the whole preservation contrast β_{pres} (Table 1). Truncation does not explain the raw aborts: mean step output is only 99.9 tokens, and only 35 of 598 aborts reach the completion cap (Table 26). In the Llama-4 Scout graph cell, references remove two-thirds of β_{pres}. The rest reflects reasoning over the observed edge list. These controls show a mechanism in the two selected cells only. They do not imply that all tool APIs require full-object re-emission.

6 Experiment 3: The Useful Fix Depends on the Domain

To see which intervention helps where, we cross two factors over all pipelines in a 2×2 design: [State: Raw Text vs. Handle] $\times$ [Decision: No Summary vs. Summary] (Table 22, Appendix E). For each domain, we fit OLS with pipeline-clustered standard errors on DeepSeek-V4-Flash, whose preservation gains are conservative (Table 29):

$$\text{Acc}_{\text{step}} = \beta_0 + \beta_{\text{exp}} \cdot \text{Exposure} + \beta_{\text{pres}} \cdot \text{Preservation} + \beta_{\text{int}} \cdot (\text{Exposure} \times \text{Preservation}) + \epsilon \quad (1)$$

Table 28 gives the four cell means and conditional contrasts (Figure 4). The factors are bundled interface contrasts, not abstract causal primitives. In this saturated model, the exposure contrast is β_{exp} on raw state and $\beta_{\text{exp}} + \beta_{\text{int}}$ with handles. The preservation contrast is β_{pres} without a summary and $\beta_{\text{pres}} + \beta_{\text{int}}$ with one. Here ℓ is the serialization length in tokens. **(1) Geometry ($\ell \approx 27$–172 tok): decision exposure is the useful intervention.** Exposure adds $+0.424$ on raw state. Preservation adds only $+0.002$ once the designated property is exposed. **(2) Graphs ($\ell \approx 49$–$1{,}061$**

tok): preservation helps more. Preservation adds $+0.094$ with a summary, and exposure adds at most $+0.020$. The fit explains little variance because the oracle raw baseline is already high (0.876). **(3) Tabular ($\ell \approx 136$–$1{,}648$ tok): state preservation dominates.** Preservation adds $+0.576$ with a summary, and exposure adds at most $+0.018$. The failure thus lies in the raw action interface, and the ArtifactRef control on Qwen3-32B (Section 5) identifies payload re-emission as the main mechanism.

Length alone does not decide which intervention wins. We pad serializations while holding meaning and pipeline complexity fixed (Appendix E.4). Padding finds no crossover length inside the tested range. More padding hurts raw text and raises the benefit of avoiding re-emission ($\beta_{\text{pres}\times\ell} = +0.193$ per 1k tokens), while the exposure interaction stays small (Table 32). An observational length fit (Table 30) mixes domains and imputed handle-arm lengths, so it is descriptive only.

6.1 Held-Out Predicates and Schema Coverage

We next measure what a fixed summary can and cannot do, under four conditions (Table 39, Appendix H.3): (1) **Π-Oracle**: the summary names the target predicate; (2) **Π-Agnostic-Hit**: a fixed schema covers the gate; (3) **Π-Agnostic-Miss**: a fixed schema omits the gate; and (4) **Π-Miss**: the target property is deliberately omitted.

Coverage makes hard gates as easy as simple ones. In the first fixed schema ($\Pi_{\text{fixed}-\text{A}}$), hit gates are Class 1–2 and miss gates are Class 3–4, so coverage tracks difficulty. A second schema, $\Pi_{\text{fixed}-\text{B}}$, covers selected hard invariants (geometry: `area, perimeter`; graphs: `diameter, density`; tabular: `max_abs_r, any_monotonic`) on the same pipelines. Its covered Class 3–4 subset is selected and smaller than the uncovered set (Table 40). With both schemas pooled on DeepSeek-V4-Flash, covered steps reach 99.1% (Class 1–2) and 99.6% (Class 3–4). Uncovered steps, where the agent must inspect or derive the property, fall to 84.5% and 63.5%.

Accuracy degrades gracefully outside the schema. On Π-Agnostic-Miss gates, decision accuracy drops to 70.5%–82.5%, not to the raw per-step accuracy of property-conditioned geometry steps (Table 17). Tool logs show why: models call `get_object` on 0.0% of hit steps, but on 99.6% of tabular miss steps ($250/251$), 27.1% of graph miss steps ($85/314$), and 10.0% of geometry miss steps ($316/3{,}168$). In graphs, unread miss steps still reach 81.7%, because node and edge counts tightly bound diameter and density. Handles keep aborts at 0.0%–1.2% throughout.

7 Additional Analyses

Optional inspection is skipped (RQ4; Appendix F.1). Given an optional `describe(object)` tool that returns $\Pi(O_t)$, agents skip it on 5.8%–54.0% of failure gates across domains and guess with uncalibrated confidence. In geometry, this leaves a $+27.2$ to $+39.7$ point deficit relative to deterministic augmentation, and runtime-forced inspection recovers most of it.

Test-time reasoning is a costly partial fix (RQ5; Appendix G.1). At thinking budgets up to 32k tokens, single-turn CoT improves arithmetic properties (Class 3: 70.8%; Qwen3-32B perimeter: $9.5\% \rightarrow 77.0\%$). Non-binary invariants stay at or below 41.4%, at up to $32.7\times$ the cost under the evaluated serving setup. A runtime summary computes the designated predicate in at most 18.5 ms (Table 34).

Neither format nor scale removes the bottleneck (Appendix H.1). Alternative serializations (GeoJSON, natural language, adjacency lists, Markdown) leave it in place. GPT-4.1 is statistically indistinguishable from DeepSeek-V4-Flash in every domain ($p \geq 0.136$).

8 Related Work

Tool-Augmented Agents & Code Actions. Tool-use methods and benchmarks (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024; Shi et al., 2025; Liu et al., 2024) train or evaluate tool retrieval and tool calls without separating branch-property derivation from payload re-emission (see Yehudai et al., 2026, for a survey). Failure analyses show errors propagating across agents (Sung et al., 2025); agentic RL surveys stress long-horizon credit assignment (Zhang et al., 2026). CodeAct (Wang et al., 2024a), Program-of-Thoughts (Chen et al., 2023), and PAL (Gao et al., 2023) sandbox open-ended derivation. We instead measure when a fixed runtime-computed property saves a model

turn. **Runtime State Management.** Agent frameworks (Lu et al., 2023; Paranjape et al., 2023; Yao et al., 2023) often pass state as text, HuggingGPT (Shen et al., 2023) links generated files through resource placeholders, and production systems offer stateful references (LangGraph channels (Chase et al., 2024), MCP URIs (Anthropic, 2024)). Our ArtifactRef control keeps the raw observation but removes payload re-emission. **Structured Data Reasoning & Locality.** Prior work studies reasoning over tables (Sui et al., 2024; Cheng et al., 2023; Ye et al., 2023; Wang et al., 2024b; Chen et al., 2024) and graphs (Fatemi et al., 2024; Wang et al., 2023), and geometry benchmarks (Zhang et al., 2024; Xing et al., 2026) and provers (Trinh et al., 2024) work on static inputs. We study them inside multi-step pipelines and show that locality depends on format: adjacency lists drop edge counting from 100.0% to 51.5% (Appendix H.4). **Expressivity & Calibration.** Expressivity results motivate studying computations assembled across a serialized input (Merrill & Sabharwal, 2024; Li et al., 2024), and CoT can unroll computation (Feng et al., 2023). LLMs are well calibrated in familiar formats but less so on new tasks (Kadavath et al., 2022). We measure CoT cost and failures for two models (Appendix G.1) and missed inspection in our pipelines (Appendix F.1).

9 Implications and Conclusion

Takeaways. (1) **Pass references, not payloads.** Keeping the full observation but passing an immutable artifact reference as the argument (`Raw+ArtifactRef`) cuts tabular aborts from 99.7% to 1.2% and removes two-thirds of the graph preservation contrast. JSON-schema decoding fixes envelope syntax, not the semantic payload corruption behind most raw aborts (Appendix E.2). (2) **Use runtime summaries for recurring predicates and sandboxes for open-ended ones.** For pre-specified recurring predicates, schema coverage turns derivation into lookup without sandboxes or multi-turn loops. This is an infrastructure trade-off, not a capability gain:

- *Recurring invariants:* For quantities such as area, convexity, diameter, and row counts, eager `Handle+Summary` computes them in at most 18.5 ms per step (Table 34).
- *Open-ended invariants:* Outside the schema (Π-Agnostic-Miss), accuracy falls to 70.5%–82.5% even though on-demand reads are available (Table 39). On direct graph queries, sandboxed code execution (`Handle+REPL`) reaches 99.2%–99.8% at 3–$10\times$ budget (Table 43). Sandboxes suit queries that cannot be precomputed.

(3) **Do not leave a known branch check to optional inspection.** In our pipelines, models skip it on up to 54.0% of geometry failure gates (Appendix F.1). A system with a canonical branch predicate should expose or enforce it, not hope the model asks. (4) **Report every run, including aborted ones.** Multi-step benchmarks should report Casc(All) (aborts count as failures), Casc(Clean) (non-aborted runs only), oracle accuracy, and a typed abort decomposition. Early aborts hide later reasoning errors.

Why not use a reasoning model or a code sandbox? A stronger reasoner still has to copy the payload, and test-time reasoning leaves non-binary invariants at or below 41.4% at up to $32.7\times$ the cost (Section 7). A sandbox computes open-ended properties (takeaway 2) but costs an extra turn and an execution environment per check. For recurring predicates, a summary gives the same value as a lookup; the two are complements.

Scope. The benchmark uses synthetic polygons, graphs of at most 80 nodes, and tables of at most 50 rows. It isolates representation and action-interface effects and does not measure summary selection, cache invalidation, mutable unstructured state, or security in open-world agent environments. Appendix K lists other limits.

Conclusion. We split the serialization bottleneck into two operations that end-to-end scores mix: deriving a branch property from a structured state and re-emitting that state in the next action. In the evaluated re-emission-heavy interfaces, opaque artifact references remove most large-payload aborts without changing what the model sees. With known recurring predicates, trusted runtime summaries expose their values directly. These complementary fixes do not replace inspection or code execution for open-ended predicates.

AI USE STATEMENT

We used generative AI tools (large language model assistants) to polish parts of the manuscript for clarity and structure. We did not use generative AI tools to generate benchmark data, develop the conceptual framework, formulate hypotheses, design the methodology or experiments, implement methods or experiment code, or run experiments. All benchmark datasets were generated programmatically with fixed random seeds. The authors reviewed every AI-assisted edit and figure against the underlying results and take full responsibility for the paper's contents, including its text, claims, and results.

REFERENCES

Anthropic. Model Context Protocol specification (version 2024-11-05). `https://modelcontextprotocol.io/specification/2024-11-05`, 2024.

Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4): 251–256, 1979.

Harrison Chase et al. LangGraph: Building stateful, multi-actor applications with LLMs. `https://github.com/langchain-ai/langgraph`, 2024.

Si-An Chen, Lesly Miculicich, Julian Martin Eisenschlos, Zifeng Wang, Zilong Wang, Yanfei Chen, Yasuhisa Fujii, Hsuan-Tien Lin, Chen-Yu Lee, and Tomas Pfister. TableRAG: Million-token table understanding with language models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research (TMLR)*, 2023.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding language models in symbolic languages. In *International Conference on Learning Representations (ICLR)*, 2023.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *International Conference on Machine Learning (ICML)*, 2023.

Sean Gillies, Casper van der Wel, Joris Van den Bossche, Mike W. Taves, Joshua Arnott, Brendan C. Ward, et al. Shapely: Manipulation and analysis of geometric objects in the Cartesian plane. Software, version 2.0. `https://github.com/shapely/shapely`, 2022.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pp. 11–15, 2008.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-Bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *International Conference on Learning Representations (ICLR)*, 2024.

Bing Liu, Jianxiang Zhou, Dan Meng, and Haonan Lu. An evaluation mechanism of LLM-based agents on manipulating APIs. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4649–4662, 2024.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Wes McKinney. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, pp. 56–61, 2010.

William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *International Conference on Learning Representations (ICLR)*, 2024.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`, 2025.

OpenAI. Introducing GPT-4.1 in the API. `https://openai.com/index/gpt-4-1/`, 2025.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. ART: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. Learning to use tools via cooperative and interactive agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10642–10657, 2024.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets LLM: Can large language models understand structured table data? A benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 645–654, 2024.

Yoo Yeon Sung, Hannah Kim, and Dan Zhang. VeriLA: A human-centered evaluation framework for interpretable verification of LLM agent failures. *arXiv preprint arXiv:2503.12651*, 2025.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *International Conference on Machine Learning (ICML)*, 2024a.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. Chain-of-Table: Evolving tables in the reasoning chain for table understanding. In *International Conference on Learning Representations (ICLR)*, 2024b.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelović, and Jakob Nikolas Kather. LLM agents making agent tools. In *Association for Computational Linguistics (ACL)*, 2025.

Shangyu Xing, Changhao Xiang, Xinyu Liu, Zhangtai Wu, Zhen Wu, Yifan Yue, Yuteng Han, Fei Zhao, and Xinyu Dai. GePBench: Evaluating fundamental geometric perception for multimodal large language models. In *International Conference on Machine Learning (ICML)*, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2023.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. A survey on evaluation of LLM-based agents. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26690–26714, 2026.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, et al. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research (TMLR)*, 2026.

Jiaxin Zhang, Zhongzhi Li, Mingliang Zhang, Fei Yin, Chenglin Liu, and Yashar Moshfeghi. GeoEval: Benchmark for evaluating LLMs and multi-modal models on geometry problem-solving. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1258–1276, 2024.

A EXPERIMENTAL PROTOCOLS AND PROMPT TEMPLATES

This section gives the settings and prompts that all experiments share. Table 2 defines the state-passing conditions compared in Section 5.

Table 2: **State-passing conditions for compositional tool use.** The four primary conditions cross State Preservation with Decision Exposure. Variants test constrained decoding and artifact-ID referencing on the abort-extreme cells across raw, augmented, and handle architectures (Section 5, Appendix E.2). `Handle + Lazy Read` is not a full independent condition across domains. It is the graph Π-agnostic-miss inspection arm of Table 39, evaluated on DeepSeek-V4-Flash miss steps.

Condition	LLM Observation (C_t)	Runtime Architecture
Raw Serialization	Full string $\sigma(O_t)$	Stateless text pass-through
Augmented Serialization	$\sigma(O_t) \oplus \Pi(O_t)$	Renders text + deterministic summary
Raw + Constrained	Full string $\sigma(O_t)$	Text pass-through + JSON-constrained decoding
Augmented + Constrained	$\sigma(O_t) \oplus \Pi(O_t)$	Text + summary + JSON-constrained decoding
Raw + ArtifactRef	Full string $\sigma(O_t)$	Text observation + immutable server-side artifact ID
Augmented + ArtifactRef	$\sigma(O_t) \oplus \Pi(O_t)$	Text + summary + server-side artifact ID
Handle Only	Identifier $\mathrm{id}(O_t)$	Retains native object in memory
Handle + Lazy Read	Identifier $\mathrm{id}(O_t)$ + on-demand $\mathrm{read}(O_t)$	Native object + on-demand field read / inspection RPC
Handle + Constrained	Identifier $\mathrm{id}(O_t)$	Native object + JSON-constrained decoding
Handle + Summary (Ours)	$\mathrm{id}(O_t) \oplus \Pi(O_t)$	Retains object + deterministic summary
Handle + Summary + Constrained	$\mathrm{id}(O_t) \oplus \Pi(O_t)$	Object + summary + JSON-constrained decoding

All experiments use temperature 0.0 and zero-shot direct prompting, with a completion budget of 512 tokens. The Chain-of-Thought ablation (16,384 tokens) and the reasoning sweep (Appendix G) are the exceptions. Below we give the prompt templates for the three domains.

A.1 PROMPT TEMPLATES

Geometry (WKT):

```
You are given a polygon in WKT format.
Polygon: {wkt_string}
Question: {question_text}
Respond with only a JSON object in this format:
{"answer": <your answer>}
```

Graphs (Edge List):

```
You are given an undirected graph as an edge list.
Graph: {edge_list_string}
Question: {question_text}
Respond with only a JSON object in this format:
{"answer": <your answer>}
```

Tabular Data (CSV):

```
You are given a data table in CSV format.
Table: {csv_string}
Question: {question_text}
Respond with only a JSON object in this format:
{"answer": <your answer>}
```

A.2 QUESTION TEXT

The list below gives the question text for each property.

- `vertex_count`: *"How many vertices does this polygon have? Do not count the repeated closing coordinate."*
- `bbox`: *"What is the axis-aligned bounding box of this polygon? Give [min_x, min_y, max_x, max_y]."*
- `centroid`: *"What is the area centroid (the centre of mass of the polygon's interior) of this polygon? Give [x, y]."* Ground truth is the Shapely 2.0 area centroid (`Polygon.centroid`).

- `area`: *"What is the area of this polygon?"*
- `perimeter`: *"What is the perimeter of this polygon?"*
- `convex`: *"Is this polygon convex? Answer true or false."*
- `orientation`: *"Is the exterior ring of this polygon oriented clockwise or counter-clockwise? Answer 'cw' or 'ccw'."*
- `degree_of_node_0`: *"What is the degree of node 0 in this graph?"*
- `edge_count`: *"How many edges does this graph have?"*
- `triangle_count`: *"How many triangles are in this graph?"*
- `is_bipartite`: *"Is this graph bipartite? Answer true or false."*
- `is_planar`: *"Is this graph planar? Answer true or false."*
- `diameter`: *"What is the diameter of this graph?"*
- `chromatic_number`: *"What is the chromatic number of this graph?"*
- `avg_clustering`: *"What is the average clustering coefficient of this graph?"*
- `row_count`: *"How many rows does this table have? Do not count the header row."*
- `column_dtype`: *"What is the data type of column {col}? Answer 'numeric', 'categorical', or 'datetime'."*
- `null_count`: *"How many null or missing values are in this table in total?"*
- `correlation`: *"What is the Pearson correlation coefficient between {col_A} and {col_B}?"*
- `is_monotonic`: *"Is column {col} monotonically increasing or decreasing? Answer true or false."*
- `has_outlier`: *"Does column {col} contain any extreme outliers (beyond $3 \times$ IQR from the quartiles)? Answer true or false."*
- `skewness`: *"What is the skewness of column {col}?"*
- `func_dependency`: *"Is column {col_B} a deterministic function of column {col_A}? Answer true or false."*

B Benchmark Domains, Models, and Query Counts

This section describes the benchmark objects, the evaluated models, and how the query counts add up. Table 3 links most headline numbers in the abstract and Section 1 to its table, sample size, and run. Table 4 lists the format, complexity tiers, and properties of each domain.

B.1 Object Generation

Each domain has 300 objects, with $n = 100$ in each of three complexity tiers (*Simple*, *Medium*, *Hard*). Table 4 gives the size range of each tier.

Geometry (WKT). We generate 300 planar 2D polygons with three random procedures: (1) perturbing the convex hull of uniformly sampled points, (2) radially perturbing a star shape around a centroid, and (3) walking the boundary of a planar Delaunay triangulation. Each candidate must pass the Shapely 2.0 `is_valid` check. Rejection sampling removes self-intersecting loops and collinear degeneracies. To avoid orientation and coordinate bias, we rotate each polygon by a random angle $\theta \sim \mathcal{U}[0, 2\pi)$, scale it uniformly by $s \sim \mathcal{U}[0.5, 3.0]$, and translate it. Convexity and exterior-ring orientation (clockwise vs. counter-clockwise) are each balanced 50/50.

Graphs (Edge List). We generate 300 undirected graphs from five families: (1) Erdős-Rényi random graphs $G(n, p)$, (2) Barabási-Albert scale-free networks ($m = 2, 3$), (3) Watts-Strogatz small-world graphs ($k = 4, p = 0.2$), (4) random bipartite graphs with equal parts ($n_1 = n_2$), and (5) random maximal planar subgraphs built from Delaunay triangulations. We verify each invariant with an exact NetworkX algorithm. Planarity uses Kuratowski subgraph isomorphism, and bipartiteness uses 2-coloring. Diameter uses all-pairs shortest paths, and the chromatic number $\chi(G)$ uses exact DSATUR backtracking. Within every tier, bipartiteness and planarity are each balanced (50% True, 50% False).

Table 3: **Primary estimands and their provenance.** Each row is a headline estimand cited in the paper, with its aggregation unit, sample size (N), primary reporting table, and execution provenance (original benchmark executions vs. retained clean re-runs). Most headline numbers in the abstract and Section 1 map to a row below.

Experiment / Scope	Primary Metric & Headline Value	Primary Table	Sample Size (N)	Aggregation Unit	Model Suite & Execution Status
Exp 1: Single-Turn Derivation (RQ1)	Local (51.3%–99.5%) vs. Global (20.8%–49.0%) accuracy; gap $\Delta_{\mathrm{LG}} = +29.2$–$72.5$ pp	Table 7, 8	43,740 queries	Query-level (binary exact, continuous $\leq 5\%$)	5 full models + 2 frontier (20% subsample) *[Original run]*
Exp 2: Factorial Compounding (Macro)	Oracle step-accuracy gains: Geo AG $= +0.406$–0.509 (OHG ≈ 0); Graph OHG $= +0.057$–0.531; Tab OHG $= +0.494$–0.891 (AG ≈ 0)	Table 22, 16	2,124 pipelines $\times 4$ cond. $\times 5$ models	Step-level (Oracle mode)	5 full models *[Original run]*; Geo V4-Flash rerun confirmed within $\pm 3.5\%$ (Table 24)
Exp 2/3: Factorial Regression (Micro)	Step oracle OLS: Geo $\beta_{\mathrm{exp}} = +0.424$; Graph $\beta_{\mathrm{pres}} = +0.089$; Tabular $\beta_{\mathrm{pres}} = +0.574$ (clustered SEs)	Table 28, 29	34,176 steps ($G = 2{,}124$ clusters)	Step-level (Oracle mode, clustered by pipeline)	DeepSeek-V4-Flash *[Clean primary rerun; retained per-step JSONL]*; 5-model re-fit in Table 29
Exp 2: Dual Denominators & Aborts	Raw aborts 16.2%–73.5% (graph), 59.7%–99.7% (tab); clean accuracy 61.1%–100.0%; 88%–94% payload corruption	Table 23, 26	2,124 pipelines $\times$ conditions	Pipeline-level (All vs. Clean vs. Oracle)	5 full models *[Original run]*; abort taxonomy audited via per-instance logs
Exp 2 Variants: Output Constraints & Artifact IDs	3-baseline β_{pres}: unconstrained ($+0.891$ tab, $+0.531$ graph) vs. constrained ($+0.874$, $+0.400$) vs. artifact-ID (-0.025, $+0.180$)	Table 25, 27	1,200 pipelines + rerun ablations	Pipeline- & Step-level	Qwen3-32B tab, Llama-4 Scout graph *[Clean variant reruns: `±get_obj`, ArtifactRef]*
Exp 3: Length Moderation	Descriptive length fit; randomized padding finds no in-range crossover	Table 30, 32	34,176 steps (obs.); 21,300 (pad)	Step-level (Oracle mode, clustered by pipeline)	DeepSeek-V4-Flash *[Clean primary rerun & length-padding intervention]*
Exp 4: Optional Inspection (RQ4)	Missed-opportunity rate 19.6%–54.0% on geometry failure gates (5.8%–54.0% across domains); geometry augmentation gap $+27.2$ to $+39.7$ pp	Table 35, §F.1	Global-gate steps (Table 35)	Step-level (optional `describe()`)	5 full models + V4-Pro *[Original run]*
Exp 3: Held-Out Predicates (RQ3; App. H.3)	Covered gates 99.1% (Class 1–2) and 99.6% (Class 3–4) vs. uncovered 84.5% and 63.5% (pooled Π_{fixed}-A and Π_{fixed}-B); Π-agnostic-miss steps retain 70.5%–82.5% with aborts $\leq 1.2\%$	Table 39, 40	924 / 600 / 600 pipelines per condition	Gated-step level	DeepSeek-V4-Flash *[Clean held-out rerun]*
Overhead & CoT Sweeps (RQ5; §G.1, App. E.6)	Deterministic $\Pi \leq 18.5$ ms ($\leq$ 80-node graphs); CoT $32.7\times$ cost, 253.4 s latency, 31.6% aborts	Table 34, 36	Profiling logs / 54,000 queries	Call- / Query-level	Local runtime / Qwen3-32B, Gemini-2.5-Flash-Lite *[Reasoning sweep run]*

Table 4: **Overview of benchmark domains, complexity tiers, and property taxonomies.** For each domain, the table lists the serialization format, the size range of each complexity tier, and the local (π_{local}) and global (π_{global}) properties.

Domain	Serialization	Complexity Tiers (Size)	Local Properties (π_{local})	Global Properties (π_{global})
Geometry	WKT (Well-Known Text)	Simple (3–8 vertices) Medium (10–20 vertices) Hard (20–40 vertices)	`vertex_count`, `bbox`	`centroid`, `area`, `perimeter`, `convex`, `orientation`, `aspect_ratio`, `edge_length_variance`
Graphs	Edge List `GRAPH (n, m):`	Simple (6–15 nodes) Medium (16–40 nodes) Hard (41–80 nodes)	`degree_of_node_0`, `edge_count`	`triangle_count`, `is_bipartite`, `is_planar`, `diameter`, `chromatic_number`, `avg_clustering`
Tabular	CSV (Comma-Separated)	Simple (8–15 rows, 4–6 cols) Medium (16–30 rows, 6–9 cols) Hard (31–50 rows, 9–12 cols)	`row_count`, `column_dtype`, `null_count`	`correlation`, `is_monotonic`, `has_outlier`, `skewness`, `func_dependency`

Tabular Data (CSV). We generate 300 tables: 250 synthetic tables and 50 real-world slices curated from standard repositories (OpenML, UCI Machine Learning Repository). Synthetic tables mix data types: continuous Gaussian mixtures, uniform floats, categorical multinomial strings, and ISO-8601 datetime sequences. Key properties are balanced as follows: (1) monotonic sequences (50% monotonic, 50% non-monotonic); (2) deterministic functional dependencies $Y = f(X)$ (50% exact, 50% shuffled controls); (3) planted extreme outliers under Tukey's interquartile range (IQR) rule ($x > Q_3 + 3 \times \mathrm{IQR}$ or $x < Q_1 - 3 \times \mathrm{IQR}$), placed 4–8 IQRs beyond the fence in 50% of tables (Appendix B.4); (4) planted Pearson correlations ($|r| \geq 0.30$), contrasted with near-zero negative controls ($|r| \leq 0.20$). We inject nulls at rates $p_{\mathrm{null}} \in [0.02, 0.15]$ in 58% of tables. The other 42% are complete.

B.2 Models and Inference Settings

We query every model through its API with a pinned version snapshot, temperature zero ($T = 0.0$), and greedy decoding. All primary experiments use a 512-token completion budget. The test-time reasoning sweeps and the Chain-of-Thought ablation are the exception: they test budgets from 512 to 32,768 tokens (Gemini is capped at its 24,576-token API limit). Table 5 lists the models of the primary benchmark, and Table 6 lists the reasoning configurations. For each model, the tables give the endpoint, architecture, parameter scale, context window, reasoning setting, and evaluation window.

B.3 Query Count Reconciliation

This subsection shows how the $N = 43,740$ zero-shot queries of Experiment 1 add up.

Table 5: **Evaluated model endpoints and architectures for the primary zero-shot benchmark (Experiments 1–4).** Model sources: DeepSeek-V4 (DeepSeek-AI, 2026), Qwen3 (Yang et al., 2025), Llama 4 (Meta AI, 2025), Gemini 2.5 (Comanici et al., 2025), and GPT-4.1 (OpenAI, 2025). Pinned endpoints use greedy decoding ($T = 0.0$) with a 512 completion token ceiling. DeepSeek-V4-Pro has 1.6 trillion parameters in total (49B active MoE).

Model Display Name	Provider / Endpoint ID	Architecture / Parameter Scale	Context Window	Reasoning Mode	Evaluation Window
DeepSeek-V4-Flash	`deepseek-ai/deepseek-v4-flash`	MoE (284B total / 13B active)	128k tokens	None (Zero-Shot)	Aug 18 – Sep 12, 2026
DeepSeek-V4-Pro	`deepseek-ai/deepseek-v4-pro`	MoE (1,600B total / 49B active)	128k tokens	None (Zero-Shot)	Sep 02 – Sep 15, 2026
Qwen3-32B-Instruct	`qwen/qwen3-32b`	Dense Transformer (32B)	32k tokens	None (Zero-Shot)	Aug 20 – Sep 10, 2026
Llama-4 Scout	`meta-llama/llama-4-scout-109b`	MoE (109B total / 17B active)	128k tokens	None (Zero-Shot)	Aug 22 – Sep 14, 2026
Gemini-2.5-Flash-Lite	`google/gemini-2.5-flash-lite`	Sparse MoE (multimodal)	1M tokens	None (Zero-Shot)	Aug 15 – Sep 18, 2026
GPT-4.1-mini	`openai/gpt-4.1-mini`	Dense Efficient Frontier	128k tokens	None (Zero-Shot)	Aug 19 – Sep 16, 2026
GPT-4.1	`openai/gpt-4.1`	Flagship Frontier	128k tokens	None (Zero-Shot)	Sep 05 – Sep 18, 2026

Table 6: **Evaluated reasoning sweep and chain-of-thought configurations (Appendix G.1 and Appendix H.1).** All evaluations use greedy decoding ($T = 0.0$) across thinking budgets. For Qwen3-32B, the primary zero-shot benchmark was served via the DeepInfra/vLLM endpoint and the reasoning sweep via SiliconFlow. Both use FP16 unquantized weights and the same $T = 0.0$ greedy decoding stack. Zero-shot accuracy was verified to be identical on control prompts.

Model Display Name	Provider / Endpoint ID	Architecture / Parameter Scale	Context Window	Reasoning Configuration	Evaluation Window
DeepSeek-V4-Flash (CoT)	`deepseek-ai/deepseek-v4-flash`	MoE (284B total / 13B active)	128k tokens	Native CoT (`max_tokens=16384`)	Aug 25 – Sep 05, 2026
Qwen3-32B (Sweep)	`qwen/qwen3-32b` (SiliconFlow)	Dense Transformer (32B)	32k tokens	Dynamic CoT (Budgets: 512, 2k, 8k, 32k)	Sep 20 – Sep 23, 2026
Gemini-2.5-Flash-Lite (Sweep)	`google/gemini-2.5-flash-lite`	Sparse MoE (multimodal)	1M tokens	Native CoT (Budgets: 512, 2k, 8k, 24.5k)	Sep 20 – Sep 22, 2026

- **Base property queries ($7,500$ per full model):** The three domains have 25 base properties: 9 in geometry, 8 in graphs, and 8 in tables. We ask each one on the 300 instances of its domain:

$$(9 \text{ geo} \times 300) + (8 \text{ graph} \times 300) + (8 \text{ tab} \times 300) = 2,700 + 2,400 + 2,400 = 7,500 \text{ base queries.}$$

- **Tabular negative-control queries (600 per full model):** These controls check that tabular failures are not false positives caused by planted signals. We ask two control properties on all 300 tabular instances: (1) correlation over uncorrelated random column pairs ($n = 300$) and (2) functional dependency over shuffled non-functional columns ($n = 300$). This adds 600 queries.
- **Total per full model ($8,100$ queries):** $7,500$ base + 600 controls $= 8,100$ queries ($2,700$ geometry, $2,400$ graphs, $3,000$ tabular).
- **Grand total across models ($N = 43,740$ queries):** Five models run on the full benchmark: DeepSeek-V4-Flash, Qwen3-32B, Llama-4 Scout, Gemini-2.5-Flash-Lite, and GPT-4.1-mini ($5 \times 8,100 = 40,500$ queries). Two frontier models, DeepSeek-V4-Pro and GPT-4.1, run on a 20% stratified subsample ($2 \times 1,620 = 3,240$ queries). The subsample has 60 objects per domain and keeps the same complexity-tier and label distributions.

$$\text{Total Zero-Shot Query Count } N = 40,500 + 3,240 = 43,740 \text{ queries.}$$

B.4 Ground Truth for `has_outlier`

This subsection explains why `has_outlier` uses Tukey IQR fences and not a z-score rule. During our audit, we found that the z-score definition of outliers breaks for small samples. Suppose an outlier is a point more than 3 sample standard deviations from the sample mean ($|z_i| = |x_i - \bar{x}|/s > 3.0$). For a sample of size n, the largest possible $|z_i|$ has an analytic bound:

$$\max_{1 \le i \le n} |z_i| \le \frac{n-1}{\sqrt{n}}. \tag{2}$$

The *Simple* tier has $n \in [8, 15]$ rows. At $n = 8$, $\max|z| = 7/\sqrt{8} \approx 2.47 < 3.0$; at $n = 9$, $\max|z| = 8/3 \approx 2.67 < 3.0$; and at $n = 10$, $\max|z| = 9/\sqrt{10} \approx 2.85 < 3.0$. So under a sample standard deviation rule, no point in an 8-to-10 row table can exceed 3σ. The label '`has_outlier = True`' is then unreachable by construction.

Table 7: **Property recovery accuracy on canonical serializations (Experiment 1).** The table compares local and global accuracy across all models and domains. Accuracy is exact match for discrete, boolean, and categorical properties, and $\leq 5\%$ relative error for continuous numeric properties. All local-global differences are statistically significant at $p < 0.001$ (two-sided bootstrap tests).

Domain	Metric	V4-Flash	Qwen3-32B	Llama-4	Gemini-2.5	GPT-4.1-mini	V4-Pro (20%)
Geometry	Local Acc (π_{local})	0.663	0.595	0.513	0.838	0.843	0.592
	Global Acc (π_{global})	0.334	0.264	0.208	0.272	0.271	0.300
	Gap (Δ_{LG})	**+0.330**	**+0.331**	**+0.306**	**+0.566**	**+0.572**	**+0.292**
	Overall Acc	0.407	0.337	0.276	0.398	0.398	0.365
Graphs	Local Acc (π_{local})	0.995	0.957	0.957	0.957	0.978	0.983
	Global Acc (π_{global})	0.372	0.256	0.232	0.274	0.283	0.286
	Gap (Δ_{LG})	**+0.623**	**+0.701**	**+0.725**	**+0.683**	**+0.695**	**+0.697**
	Overall Acc	0.527	0.431	0.413	0.445	0.457	0.460
Tabular	Local Acc (π_{local})	0.921	0.789	0.717	0.847	0.783	0.946
	Global Acc (π_{global})	0.310	0.305	0.323	0.297	0.333	0.490
	Gap (Δ_{LG})	**+0.611**	**+0.484**	**+0.394**	**+0.550**	**+0.450**	**+0.457**
	Overall Acc	0.574	0.514	0.493	0.535	0.528	0.688

To keep the question well-posed in every tier, our dataset generator uses Tukey's extreme interquartile range (IQR) fences:

$$x < Q_1 - 3 \times \text{IQR} \quad \text{or} \quad x > Q_3 + 3 \times \text{IQR}. \tag{3}$$

We plant outliers 4–8 IQRs beyond the fences. Quartiles are well-defined for $n \geq 4$, so this rule is valid in all tiers. The resulting labels are 47 True / 51 False among the 98 Simple-tier tables that carry a `has_outlier` label (the tier itself is $n = 100$), 50/50 in Medium, and 49/51 in Hard. The `has_outlier` question in Appendix A states this Tukey IQR criterion.

C Property Recovery Tables (Experiment 1)

This section gives the full tables for Experiment 1, which tests local versus global property recovery from canonical serializations. Tables 7–11 cover all modalities and models. They report the local versus global breakdown, exemplar properties, and per-property accuracy in each domain.

Abort Mechanism and ArtifactRef Cohort Accounting. We check whether the high raw abort rates come from truncation at the completion budget. Table 26 breaks down trace lengths and failure causes. In tabular raw execution (Qwen3-32B), the mean generation length per step is only 99.9 tokens. Only 35 of 598 aborts (5.8%) hit the 512-token completion budget. Most aborts (563/598, or 94.1%) are early semantic payload corruptions, such as corrupted CSV formatting, mismatched columns, or invalid string placeholders. They occur well before the budget runs out. Graphs (Llama-4 Scout) show the same pattern. The mean step length is 198.5 tokens, only 32 of 441 aborts (7.3%) reach the budget, and 390/441 (88.4%) are semantic payload failures. Strict JSON-schema constrained decoding removes envelope syntax errors, but semantic payload failures remain (466 tabular and 272 graph aborts; Table 26). The `Raw+ArtifactRef` control isolates this mechanism. It replaces only the action payload argument with an immutable reference, while the same full object stays in context. We report ArtifactRef runs on the complete executed cohorts ($N = 521/518$ tabular, $N = 281/273$ graphs), across all pipeline types (pass-through, property-conditioned, multi-object). Removing raw object re-emission removes 98.8% of tabular aborts, even with the full text observation in context.

C.1 Tolerance Sweep and Binomial Tests against Chance

This subsection tests whether the low accuracy on continuous properties comes from a strict tolerance. Table 12 sweeps the relative error threshold over strict ($\leq 1\%$), moderate ($\leq 5\%$, headline), and lenient ($\leq 10\%$) levels. Accuracy stays low at every level. Even the lenient $\leq 10\%$ tolerance raises geometric recovery to at most 19.1% and tabular statistics to at most 6.2%. The derivation deficit is therefore intrinsic.

Table 8: **Zero-shot accuracy on exemplar properties.** Each row is one property with its domain and locality; each column is one model. GPT-4.1 is evaluated on the 20% stratified subsample (60 objects per domain), like V4-Pro.

Property	Locality	V4-Flash	Qwen3	Llama	Gemini	GPT-4.1-mini	V4-Pro	GPT-4.1
`bbox` (Geo)	Local	1.000	0.960	0.887	0.970	0.993	1.000	1.000
`area` (Geo)	Global	0.087	0.070	0.043	0.053	0.077	0.033	0.033
`edge_count` (Graph)	Local	1.000	1.000	1.000	0.997	1.000	1.000	1.000
`is_bipartite` (Graph)	Global	0.557	0.500	0.490	0.550	0.547	0.600	0.517
`is_planar` (Graph)	Global	0.600	0.567	0.523	0.510	0.607	0.633	0.533
`avg_clustering` (Graph)	Global	0.073	0.037	0.033	0.057	0.047	0.050	0.167
`column_dtype` (Tab)	Local	0.999	0.978	0.974	0.999	0.983	0.979	1.000
`correlation` (Tab)	Global	0.030	0.050	0.044	0.048	0.066	0.000	0.053

Table 9: **Complete property recovery accuracy in the geometry domain (Experiment 1).** Each row is one property with its locality and serial class; each column is one model.

Property	Locality	Serial Class	V4-Flash	Qwen3-32B	Llama-4	Gemini-2.5	GPT-4.1-mini	V4-Pro (20%)
`vertex_count`	Local	Class 2	0.327	0.230	0.140	0.707	0.693	0.183
`bbox`	Local	Class 1	1.000	0.960	0.887	0.970	0.993	1.000
`centroid`	Global	Class 3	0.793	0.533	0.363	0.627	0.620	0.767
`area`	Global	Class 3	0.087	0.070	0.043	0.053	0.077	0.033
`perimeter`	Global	Class 3	0.157	0.070	0.027	0.063	0.067	0.133
`convex`	Global	Class 4 (Bin)	0.517	0.500	0.433	0.533	0.500	0.500
`orientation`	Global	Class 4 (Bin)	0.493	0.587	0.540	0.557	0.530	0.500
`aspect_ratio`	Global	Class 3	0.260	0.047	0.030	0.043	0.070	0.167
`edge_length_var`	Global	Class 4 (Non-Bin)	0.030	0.040	0.017	0.027	0.033	0.000

Statistical Significance on Balanced Binary Predicates. We next test whether failure on global properties reflects a real lack of capability rather than slight imprecision. We run exact two-sided binomial tests against random guessing (chance $= 0.50$), with $N = 300$ per property for the five full models and $N = 60$ for V4-Pro:

- **Geometry `orientation`:** V4-Flash achieves 49.3% ($p = 0.86$, fail to reject null of chance); V4-Pro achieves 50.0% ($p = 1.00$).
- **Geometry `convex`:** V4-Flash achieves 51.7% ($p = 0.60$, indistinguishable from chance); Qwen3 achieves 50.0% ($p = 1.00$).
- **Graph `is_bipartite`:** Qwen3 achieves 50.0% ($p = 1.00$); Llama 4 Scout achieves 49.0% ($p = 0.77$); V4-Flash achieves 55.7% ($p = 0.057$).
- **Graph `is_planar`:** Gemini 2.5 achieves 51.0% ($p = 0.77$, indistinguishable from chance).

For the model and property pairs above, the tests fail to reject the 0.50 chance null on orientation, convexity, bipartiteness, and planarity. Among these pairs, the largest deviation, V4-Flash on bipartiteness at 55.7%, is only marginal ($p = 0.057$). Autoregressive forward passes over text serializations thus show limited topological comprehension without runtime assistance.

C.2 Numerical Conventions for All 25 Properties

This subsection fixes how ground truth and scores are computed. Table 13 gives one exact library call and one numerical convention per property. Ground truth generation, the Π summary computation in the pipeline runtimes, and the scorer all share the stated convention. For scalar targets, the exact rule is $|\hat{y} - y|/|y| \leq \tau$ when $y \neq 0$ and $|\hat{y} - y| \leq \tau$ when $y = 0$. This absolute-error fallback avoids division by zero. The headline uses $\tau = 0.05$, and Appendix C.1 sweeps $\tau \in \{0.01, 0.10\}$. In the 300-object tabular corpus, correlation has 1/498 exactly-zero, 7/498 $|y| < 0.01$, and 57/498 $|y| < 0.05$ targets; skewness has 0/308, 1/308, and 1/308, respectively. The two geometric point-valued properties (`bbox`, `centroid`) are scored on Euclidean error normalized by the polygon's bounding-box diagonal. Discrete, boolean, and categorical properties use exact match.

Centroid convention audit (prompt specificity). This audit asks whether the centroid scores depend on the convention. Early drafts asked *"What is the centroid of this polygon?"* without fixing the convention, while ground truth used the Shapely area centroid. We re-scored every stored

Table 10: **Complete property recovery accuracy in the graph domain (Experiment 1).** Each row is one property with its locality and serial class; each column is one model.

Property	Locality	Serial Class	V4-Flash	Qwen3-32B	Llama-4	Gemini-2.5	GPT-4.1-mini	V4-Pro (20%)
degree_of_node_0	Local	Class 2	0.990	0.913	0.913	0.917	0.957	0.967
edge_count	Local	Class 1	1.000	1.000	1.000	0.997	1.000	1.000
triangle_count	Global	Class 4 (Non-Bin)	0.417	0.080	0.017	0.077	0.043	0.100
is_bipartite	Global	Class 4 (Bin)	0.557	0.500	0.490	0.550	0.547	0.600
is_planar	Global	Class 4 (Bin)	0.600	0.567	0.523	0.510	0.607	0.633
diameter	Global	Class 4 (Non-Bin)	0.370	0.183	0.207	0.287	0.320	0.167
chromatic_number	Global	Class 4 (Non-Bin)	0.213	0.170	0.120	0.163	0.137	0.167
avg_clustering	Global	Class 4 (Non-Bin)	0.073	0.037	0.033	0.057	0.047	0.050

Table 11: **Complete property recovery accuracy in the tabular domain (Experiment 1).** Each row is one property with its locality and serial class; each column is one model.

Property	Locality	Serial Class	V4-Flash	Qwen3-32B	Llama-4	Gemini-2.5	GPT-4.1-mini	V4-Pro (20%)
row_count	Local	Class 2	0.887	0.527	0.203	0.700	0.430	0.983
column_dtype	Local	Class 1	0.999	0.978	0.974	0.999	0.983	0.979
null_count	Local	Class 2	0.777	0.610	0.633	0.643	0.673	0.833
correlation	Global	Class 4 (Non-Bin)	0.030	0.050	0.044	0.048	0.066	0.000
is_monotonic	Global	Class 4 (Bin)	0.653	0.570	0.657	0.563	0.550	1.000
has_outlier	Global	Class 4 (Bin)	0.527	0.547	0.591	0.537	0.711	0.917
skewness	Global	Class 4 (Non-Bin)	0.010	0.010	0.006	0.003	0.006	0.000
func_dependency	Global	Class 4 (Bin)	0.523	0.523	0.510	0.507	0.520	0.850

completion against both conventions (area centroid vs. vertex-mean; same tolerance). Accuracies move by -1.7 to $+5.4$ points per model (V4-Flash $79.3\% \rightarrow 80.3\%$), with no systematic direction. The vertex convention helps Medium-tier polygons and hurts Hard-tier ones. So the centroid column's level is not an artifact of convention confusion. The prompt now names the area centroid explicitly (Appendix A).

C.3 Serial-Derivation Classes and the Recovery Ladder

The main analysis splits properties into local (π_{local}) and global (π_{global}) sets: substring extraction versus whole-object computation. A finer, circuit-theoretic view groups properties by **serial-derivation complexity**, the cost of the serial algorithm that derives them:

1. **Class 1: Explicit Substring / Header Lookup ($O(1)$ token match).** Properties directly exposed in the serialization syntax or extractable via parallel token matching: `edge_count`[2] (in edge-list headers), `bbox` (coordinate extrema), and `column_dtype` (single-token syntax inspection).
2. **Class 2: Linear Aggregation and Counting ($O(N)$ single pass).** Properties requiring linear token accumulation over the full context: `row_count`, `null_count`, `degree_of_node_0`, and `vertex_count`.
3. **Class 3: Multi-Step Sequential Arithmetic / Transformations.** Properties requiring chained floating-point operations over parsed coordinates: `area` (the Shoelace polygon formula), `perimeter` (chained Euclidean distances), `aspect_ratio`, and `centroid` (the area-weighted centroid, computed from the same Shoelace cross-products, not a bare vertex average).
4. **Class 4: Combinatorial Search, Higher-Moment Statistics, and Global Topology.** Properties requiring non-local graph traversal, statistical moment accumulation, or exact global decision boundaries. Class 3 continuous quantities admit a $\leq 5\%$ relative tolerance, but two geometric predicates do not. `orientation` demands exact sign resolution at the zero-area decision boundary. `convex` is a strict conjunction, where a single non-local triple violation inverts the label:
   - *Binary Decision Predicates:* `convex`, `orientation`, `is_bipartite`, `is_planar`, `func_dependency`, `has_outlier`, and `is_monotonic`.
   - *Non-Binary Invariants:* `diameter`, `chromatic_number`, `avg_clustering`, `triangle_count`, `edge_length_variance`, `correlation`, and `skewness`.

[2]Under canonical edge-list headers (`GRAPH n m`), `edge_count` is explicitly declared and operates as Class 1. In Ablation 1 (Appendix H.1), we demonstrate that under adjacency-list representation lacking headers, `edge_count` requires summing neighbor list lengths, flipping its complexity to Class 2.

Table 12: **Tolerance sweep over relative error thresholds for continuous properties ($\leq 1\%$, $\leq 5\%$, $\leq 10\%$).** The sweep covers continuous geometric properties (`area`, `perimeter`, `edge_length_variance`) and tabular properties (`correlation`, `skewness`). The derivation deficit is not an artifact of threshold selection: even at a lenient $\leq 10\%$ relative error tolerance, model accuracy plateaus below 20% in geometry and below 7% in tabular statistics.

Domain	Tolerance	V4-Flash	Qwen3-32B	Llama-4 Scout	Gemini-2.5	GPT-4.1-mini	V4-Pro (20%)
Geometry (Area, Perim, Edge Var)	Strict ($\leq 1\%$)	0.017	0.018	0.007	0.012	0.009	0.011
	Moderate ($\leq 5\%$)	0.091	0.060	0.029	0.048	0.059	0.056
	Lenient ($\leq 10\%$)	0.191	0.126	0.059	0.082	0.132	0.133
Tabular (Correlation, Skewness)	Strict ($\leq 1\%$)	0.012	0.015	0.010	0.009	0.024	0.000
	Moderate ($\leq 5\%$)	0.020	0.030	0.025	0.031	0.045	0.000
	Lenient ($\leq 10\%$)	0.042	0.048	0.035	0.038	0.062	0.000

Table 13: **Exact numerical conventions for the 25 evaluated properties.** Each property has one library call and one convention. Ground truth, the runtime Π vector, and scoring all share them identically.

Property	Domain	Exact library call	Numerical convention	Tolerance
`vertex_count`	Geometry	`len(poly.exterior.coords) - 1` (Shapely ring)	closing duplicate excluded	exact
`bbox`	Geometry	`Polygon.bounds`	axis-aligned $[\min_x, \min_y, \max_x, \max_y]$	$\max\lvert\Delta\rvert$/bbox-diag $\leq 5\%$
`centroid`	Geometry	`Polygon.centroid`	**area-weighted** centroid (Shoelace cross-products), not a vertex average	Euclid/bbox-diag $\leq 5\%$
`area`	Geometry	`Polygon.area`	absolute Shoelace area	$\leq 5\%$ rel
`perimeter`	Geometry	`Polygon.length`	chained Euclidean edge lengths	$\leq 5\%$ rel
`convex`	Geometry	hull-area equality: $\lvert$`convex_hull.area` $-$ `area`$\rvert \leq 10^{-9} \cdot \max(\text{area}, 1)$	boolean	exact
`orientation`	Geometry	exterior-ring winding (signed Shoelace area)	`cw` iff signed area < 0	exact
`aspect_ratio`	Geometry	bbox width $\div$ height	ratio of axis-aligned bbox sides	$\leq 5\%$ rel
`edge_length_variance`	Geometry	$\sum_i (e_i - \bar{e})^2/n$ over segment lengths	population variance (ddof $= 0$)	$\leq 5\%$ rel
`degree_of_node_0`	Graphs	`G.degree(0)` (NetworkX)	integer	exact
`edge_count`	Graphs	`G.number_of_edges()`	integer; also in the `GRAPH (n, m)` header	exact
`triangle_count`	Graphs	`sum(nx.triangles(G).values()) // 3`	integer	exact
`is_bipartite`	Graphs	`nx.is_bipartite(G)`	2-coloring	exact
`is_planar`	Graphs	`nx.check_planarity(G)[0]`	Kuratowski-based planarity test	exact
`diameter`	Graphs	`nx.diameter(G)`	max APSP eccentricity; benchmark graphs are connected by construction	exact
`chromatic_number`	Graphs	exact DSATUR backtracking (15 s cap)	exact vertex coloring; `chromatic_number_certified` flag in metadata	exact
`avg_clustering`	Graphs	`nx.average_clustering(G)`	local clustering averaged over nodes, rounded 4dp	$\leq 5\%$ rel
`row_count`	Tabular	`len(df)` (pandas)	header row excluded	exact
`column_dtype`	Tabular	realized-value inference: all non-null cells float-parse $\rightarrow$ numeric; else all parse as `%Y-%m-%d` $\rightarrow$ datetime; else categorical	schema-free inference as a CSV reader would apply	exact
`null_count`	Tabular	`(df == '').sum().sum() + df.isna().sum().sum()`	empty cells plus NaNs	exact
`correlation`	Tabular	`pandas.Series.corr`	Pearson, pairwise complete after float coercion	$\leq 5\%$ rel
`is_monotonic`	Tabular	`is_monotonic_increasing` $\vee$ `is_monotonic_decreasing` (pandas)	after float or datetime coercion	exact
`has_outlier`	Tabular	Tukey fence $Q_1 - 3\,\text{IQR}$ / $Q_3 + 3\,\text{IQR}$	linear-interpolation quartiles (pandas default), $n \geq 4$	exact
`skewness`	Tabular	`scipy.stats.skew(x, bias=False)`	**bias-corrected adjusted Fisher–Pearson** G_1 (identical to `pandas.Series.skew()`); plain b_1 differs by 10–30% at $n = 8$–15	$\leq 5\%$ rel
`func_dependency`	Tabular	`groupby(A)[B].nunique().max() == 1`	requires ≥ 2 distinct values of A	exact

Class Ordering and Held-Out Prediction. Table 15 shows that accuracy falls at each step: from Class 1 to Class 2 (66.0%), Class 3, and the non-binary Class 4 invariants. This is an empirical hierarchy ordered by serial-algorithm cost. Aggregate Class 4 accuracy appears slightly higher than Class 3. Binary predicates (`convex`, `orientation`, `bipartite`, `planar`) drive this artifact entirely, because random guessing gives them a 50% floor. The empirical mean of 56.0% is the pooled binary-predicate accuracy, not the single-model Flash bipartiteness result reported above. Pooling the binary-predicate trials against a 0.50 floor gives $p \approx 2 \times 10^{-24}$. On non-binary continuous and

Table 14: **Canonical serial-derivation complexity taxonomy of the 25 properties.** Each evaluated property in the geometry, graph, and tabular domains is mapped to its computational tier, binary status, and formal circuit/derivation justification. The taxonomy is motivated by uniform TC^0 circuit complexity (Appendix C.3).

Property	Domain	Complexity Class	Type	Computational / Circuit Justification
`bbox`	Geometry	Class 1 (Explicit)	Non-Binary	Parallel coordinate min/max reduction over vertex tokens; $O(1)$ depth
`edge_count`	Graphs	Class 1 (Explicit)[3]	Non-Binary	Declared in edge-list header (`GRAPH n m`); flips to Class 2 in adjacency lists
`column_dtype`	Tabular	Class 1 (Explicit)	Non-Binary	Single-token syntactic literal format matching; bounded-window projection
`vertex_count`	Geometry	Class 2 (Linear Aggregation)	Non-Binary	Linear scanning and tallying of coordinate pairs along boundary; $O(N)$ depth
`degree_of_node_0`	Graphs	Class 2 (Linear Aggregation)	Non-Binary	Context scan tallying incident edges to node 0 across edge list; $O(M)$ tokens
`null_count`	Tabular	Class 2 (Linear Aggregation)	Non-Binary	Linear context scanning and counting of missing/empty cell tokens
`row_count`	Tabular	Class 2 (Linear Aggregation)	Non-Binary	Linear scanning and tallying of newline-delimited record rows
`area`	Geometry	Class 3 (Multi-Step Arith)	Non-Binary	Chained floating-point multiplications via Shoelace polygon formula
`perimeter`	Geometry	Class 3 (Multi-Step Arith)	Non-Binary	Chained Euclidean distances ($\sqrt{\Delta x^2 + \Delta y^2}$) across consecutive vertices
`centroid`	Geometry	Class 3 (Multi-Step Arith)	Non-Binary	Area-weighted centroid of the polygon interior via the Shoelace cross-product formula (Shapely `Polygon.centroid`)
`aspect_ratio`	Geometry	Class 3 (Multi-Step Arith)	Non-Binary	Sequential arithmetic quotient of bounding box width divided by height
`convex`	Geometry	Class 4 (Combinatorial / Moment)	Binary	Global conjunction certificate; a single non-local cross-product sign violation inverts label
`orientation`	Geometry	Class 4 (Combinatorial / Moment)	Binary	Exact sign resolution on signed Shoelace area near zero boundary (unlike Class 3 relative tolerance)
`is_bipartite`	Graphs	Class 4 (Combinatorial / Moment)	Binary	Combinatorial 2-coloring search or odd-length cycle detection
`is_planar`	Graphs	Class 4 (Combinatorial / Moment)	Binary	Topological graph-minor search (detection of K_5 or $K_{3,3}$)
`func_dependency`	Tabular	Class 4 (Combinatorial / Moment)	Binary	Determinant-to-dependent uniqueness check across all row pairs
`has_outlier`	Tabular	Class 4 (Combinatorial / Moment)	Binary	Statistical decision testing existence of cell beyond Tukey $3 \times$ IQR fences
`is_monotonic`	Tabular	Class 4 (Combinatorial / Moment)	Binary	Uniform sign verification of pairwise consecutive sequence differences
`edge_length_var`	Geometry	Class 4 (Combinatorial / Moment)	Non-Binary	Second-order statistical variance over all polygon segment lengths
`triangle_count`	Graphs	Class 4 (Combinatorial / Moment)	Non-Binary	Combinatorial 3-cycle subgraph enumeration requiring non-local traversal
`avg_clustering`	Graphs	Class 4 (Combinatorial / Moment)	Non-Binary	Local triangle-to-triplet density averaged across all graph vertices
`chromatic_number`	Graphs	Class 4 (Combinatorial / Moment)	Non-Binary	NP-hard graph vertex coloring optimization
`diameter`	Graphs	Class 4 (Combinatorial / Moment)	Non-Binary	All-pairs shortest path breadth-first search traversals
`correlation`	Tabular	Class 4 (Combinatorial / Moment)	Non-Binary	Bivariate covariance normalized by dual standard deviations
`skewness`	Tabular	Class 4 (Combinatorial / Moment)	Non-Binary	Third standardized statistical moment ($\mathbb{E}[(X-\mu)^3]/\sigma^3$)

Table 15: **Recovery ladder across serial-derivation complexity classes (Experiment 1).** Accuracy is averaged across the six columns of this table ($N = 42,120$ queries: five full models plus the V4-Pro subsample), with 95% bootstrap confidence intervals in brackets. Class rows and sub-rows aggregate per query over evaluated queries. Unweighted property means on Flash are 0.553 for binary and 0.163 for non-binary properties. Accuracy drops sharply from Class 1 (98.4%) to Class 3 (21.7%), consistent with complexity-theoretic limits on single-pass forward computation without token unrolling (Appendix C.3). Aggregate Class 4 (33.4%) exceeds Class 3 because binary predicates sit on a 50% guessing floor; non-binary invariants fall to 9.4%.

Complexity Class	Representative Properties	Flash	Qwen3	Llama-4	Gemini	GPT-4.1-mini	V4-Pro	Mean [95% CI]
Class 1: Explicit / Header	`bbox`, `edge_count`, `column_dtype`	1.000	0.979	0.954	0.989	0.992	0.993	**0.984** [0.970, 0.994]
Class 2: Linear Aggregation	`row_count`, `null_count`, `degree`, `vertex`	0.745	0.570	0.472	0.742	0.688	0.741	**0.660** [0.548, 0.763]
Class 3: Multi-Step Arith	`area`, `perimeter`, `aspect`, `centroid`	0.324	0.180	0.116	0.197	0.209	0.275	**0.217** [0.125, 0.323]
Class 4: Combinatorial / Topology	`planar`, `bipartite`, `convex`, `corr`, …	0.358	0.312	0.299	0.316	0.330	0.392	**0.334** [0.279, 0.390]
– Binary Predicates	`convex`, `orientation`, `bipartite`, `planar`, …	0.557	0.536	0.521	0.537	0.567	0.643	**0.560** [0.534, 0.596]
– Non-Binary Invariants	`clustering`, `chromatic`, `diameter`, `corr`, …	0.160	0.088	0.077	0.095	0.093	0.054	**0.094** [0.071, 0.124]

combinatorial invariants, accuracy drops to 9.4%. Accuracy thus forms a monotonic staircase across classes.

Constant-depth, polynomial-size transformers with finite precision operate within the circuit class uniform TC^0 (Merrill & Sabharwal, 2024). In uniform TC^0, threshold circuits can evaluate surface substring equality (Class 1). Counting and majority operations (Class 2) are computable at constant depth by definition. Still, finite-precision attention and positional token dispersion lower Class 2 empirical accuracy (66.0%), which leaves this practical gap theoretically open. Serial algorithms need substantial sequential computation for multi-step geometric arithmetic (Class 3) and for combinatorial graph traversals or moment statistics (Class 4). Circuit complexity motivates the intuition that serial depth constrains fixed-depth forward passes without scratchpads. We do not claim formal circuit lower bounds for specific benchmark properties, because separations such as $\mathrm{TC}^0 \subsetneq \mathrm{L}$ remain open in complexity theory.

We also test whether serial-derivation class is a predictive taxonomy rather than a post-hoc grouping. We fit an ordinary least-squares predictor on 18 randomly selected training properties and evaluate it on the remaining 7 held-out properties across all six models. The serial complexity tier alone predicts held-out property recovery accuracy with $R^2 = 0.93$ ($p \approx 4.5 \times 10^{-4}$). This test uses only $n = 7$ held-out properties, with hand-designed classes, inside the same synthetic benchmark domains. The relationship is therefore suggestive rather than a validated universal law, and evaluation on broader out-of-domain property sets remains necessary.

D PIPELINE COMPOUNDING BREAKDOWN (EXPERIMENT 2)

This appendix gives the full results behind Section 5. Tables 16 and 17 cover geometry pipelines. Table 18 covers graph pipelines and adds abort rates.

Table 16: **Execution accuracy and recovery gains in geometry pipelines (Experiment 2, $N = 924$).** Results are split into oracle per-step accuracy and end-to-end correctness. $AG =$ Augmented $-$ Raw and $OHG =$ Handle $-$ Raw are step gains. Bold marks the top condition.

Model	Metric Level	Raw	Augmented	Handle	Handle + Sum (Ours)	AG (Step)	OHG (Step)
V4-Flash	Oracle Step Acc	0.588	0.995	0.736	**0.996**	**+0.407**	**+0.148**
	Cascading E2E	0.399	0.988	0.598	**0.994**	—	—
	Oracle E2E	0.389	0.983	0.583	**0.992**	—	—
Qwen3-32B	Oracle Step Acc	0.381	0.891	0.421	**0.896**	**+0.509**	**+0.040**
	Cascading E2E	0.252	0.772	0.246	**0.792**	—	—
	Oracle E2E	0.246	0.777	0.249	**0.794**	—	—
Llama-4 Scout	Oracle Step Acc	0.266	0.672	0.386	**0.738**	**+0.406**	**+0.120**
	Cascading E2E	0.194	0.472	0.244	**0.753**	—	—
	Oracle E2E	0.187	0.482	0.246	**0.736**	—	—
Gemini-2.5	Oracle Step Acc	0.461	0.909	0.395	**0.912**	**+0.449**	**-0.066**
	Cascading E2E	0.291	0.792	0.219	**0.795**	—	—
	Oracle E2E	0.291	0.792	0.219	**0.795**	—	—
GPT-4.1-mini	Oracle Step Acc	0.462	0.951	0.437	**0.954**	**+0.489**	**-0.024**
	Cascading E2E	0.329	0.881	0.271	**0.925**	—	—
	Oracle E2E	0.331	0.879	0.273	**0.926**	—	—
V4-Pro (20%)	Oracle Step Acc	0.540	0.981	0.637	**0.988**	**+0.442**	**+0.097**
	Cascading E2E	0.376	0.941	0.505	**0.989**	—	—
	Oracle E2E	0.366	0.935	0.495	**0.989**	—	—

Table 17: **Oracle per-step accuracy across pipeline categories and state conditions for all six models (geometry).** For each model, rows are pipeline categories and columns are state conditions.

Model	Category (Locality)	Raw	Augmented	Handle	Handle + Sum
V4-Flash	Pass-Through	1.000	1.000	1.000	1.000
	Property-Conditioned (Global)	0.481	1.000	0.771	1.000
	Multi-Object (Global)	0.450	0.990	0.553	0.995
Qwen3-32B	Pass-Through	0.973	0.991	1.000	1.000
	Property-Conditioned (Global)	0.284	0.942	0.244	0.951
	Multi-Object (Global)	0.209	0.802	0.309	0.808
Llama-4 Scout	Pass-Through	0.970	0.985	1.000	1.000
	Property-Conditioned (Global)	0.116	0.812	0.089	0.817
	Multi-Object (Global)	0.034	0.655	0.075	0.662
Gemini-2.5	Pass-Through	0.996	0.998	1.000	1.000
	Property-Conditioned (Global)	0.332	0.958	0.302	0.964
	Multi-Object (Global)	0.326	0.892	0.315	0.899
GPT-4.1-mini	Pass-Through	1.000	1.000	1.000	1.000
	Property-Conditioned (Global)	0.262	0.969	0.282	0.949
	Multi-Object (Global)	0.349	0.908	0.312	0.913
V4-Pro (20%)	Pass-Through	1.000	1.000	1.000	1.000
	Property-Conditioned (Global)	0.356	0.978	0.448	0.984
	Multi-Object (Global)	0.382	0.965	0.420	0.970

Table 18: **Oracle vs. cascading accuracy and abort rates in graph pipelines (Experiment 2).** For each model, rows are the four state conditions. Columns report oracle step accuracy, Oracle E2E and cascading E2E accuracy, and abort rate.

Model	Condition	Oracle Step Acc	Oracle E2E Acc	Cascading E2E Acc	Abort Rate
V4-Flash	Raw Serialization	0.876	0.738	0.738	0.162
	Augmented Serialization	0.891	0.777	0.778	0.163
	Handle Only	0.965	0.892	0.905	0.017
	Handle + Summary	**0.985**	**0.948**	**0.950**	**0.000**
Qwen3-32B	Raw Serialization	0.452	0.345	0.360	0.543
	Augmented Serialization	0.440	0.372	0.345	0.638
	Handle Only	0.886	0.702	0.703	0.085
	Handle + Summary	**0.961**	**0.877**	**0.877**	**0.000**
Llama-4 Scout	Raw Serialization	0.381	0.165	0.173	0.735
	Augmented Serialization	0.344	0.213	0.237	0.712
	Handle Only	0.912	0.743	0.747	0.097
	Handle + Summary	**0.951**	**0.847**	**0.852**	**0.043**
Gemini-2.5	Raw Serialization	0.633	0.502	0.502	0.462
	Augmented Serialization	0.692	0.580	0.580	0.417
	Handle Only	0.843	0.618	0.618	0.040
	Handle + Summary	**0.965**	**0.910**	**0.910**	**0.050**
GPT-4.1-mini	Raw Serialization	0.830	0.672	0.673	0.218
	Augmented Serialization	0.883	0.763	0.763	0.172
	Handle Only	0.887	0.700	0.702	0.093
	Handle + Summary	**0.975**	**0.917**	**0.917**	**0.002**

D.1 Tool Signatures and Pipeline Grammar

This subsection lists the tools and pipeline topologies. Each tool is deterministic, runs on native in-memory objects through standard scientific libraries, and returns a new object or a scalar:

- **Geometry Domain (Shapely 2.0 Backend):**
  - `translate(polygon, dx, dy)` $\rightarrow$ `Polygon`: Shifts vertex coordinates by spatial offset $(\mathrm{d}x, \mathrm{d}y)$.
  - `rotate(polygon, angle_deg)` $\rightarrow$ `Polygon`: Rotates polygon around its centroid by angle in degrees.
  - `scale(polygon, factor_x, factor_y)` $\rightarrow$ `Polygon`: Rescales dimensions along coordinate axes.
  - `buffer(polygon, distance)` $\rightarrow$ `Polygon`: Dilates or erodes polygon boundary by distance.
  - `simplify(polygon, tolerance)` $\rightarrow$ `Polygon`: Applies Douglas-Peucker line simplification.
  - `convex_hull(polygon)` $\rightarrow$ `Polygon`: Smallest convex enclosing geometry.
  - `union(p1, p2)` $\rightarrow$ `Polygon`: Geometric union of two polygons.
  - `intersection(p1, p2)` $\rightarrow$ `Polygon`: Computes geometric intersection.
- **Graph Domain (NetworkX 3.0 Backend):**
  - `add_edges(graph, edge_list)` $\rightarrow$ `Graph`: Adds edge tuples to an undirected graph.
  - `remove_nodes(graph, node_list)` $\rightarrow$ `Graph`: Deletes the listed nodes.
  - `induced_subgraph(graph, node_list)` $\rightarrow$ `Graph`: Extracts induced subgraph on vertex subset.
  - `relabel_nodes(graph, mapping)` $\rightarrow$ `Graph`: Permutes integer node identifiers.
  - `union(g1, g2)` $\rightarrow$ `Graph`: Computes disjoint graph union.
  - `complement(graph)` $\rightarrow$ `Graph`: Computes edge complement graph.
- **Tabular Domain (pandas 2.0 Backend):**
  - `filter_rows(df, col, op, val)` $\rightarrow$ `DataFrame`: Filters rows by comparison operator $(<, >, ==, \neq)$.

- `sort_values(df, col, ascending)` → `DataFrame`: Reorders rows by designated column.
- `drop_columns(df, columns)` → `DataFrame`: Drops specified column subset.
- `impute_missing(df, col, strategy)` → `DataFrame`: Imputes null cells (`mean`, `median`, `zero`).
- `normalize_column(df, col, method)` → `DataFrame`: Applies min-max or z-score normalization.
- `join(df1, df2, on, how)` → `DataFrame`: Computes equi-join on shared key column.

Pipelines are directed acyclic graphs with three execution topologies. (1) **Pass-Through Pipelines** run steps in fixed order, $O_1 = f_1(O_0) \to \cdots \to O_K = f_K(O_{K-1})$. They test raw-state preservation against payload corruption during re-emission. (2) **Property-Conditioned Pipelines** branch on an intermediate property, if $\pi(O_{t-1}) \bowtie \tau$ then f_A else f_B. They test derivation fidelity. (3) **Multi-Object Pipelines** select among candidate objects, $O^* = \arg\max_{O \in \{O_A, O_B, O_C\}} \pi(O)$. They test multi-candidate context loading.

D.2 Summary Vector (Π) Schemas and Worked Examples

This subsection shows exactly what each condition gives the model: the $\Pi(O)$ schema per domain and pipeline type, and one worked example under all four conditions.

Domain-Specific Π Summary Schemas. The runtime computes deterministic summary vectors with standard scientific libraries whenever an object is created or changed:

1. **Geometry Domain (Π_{geo}):** A fixed 10-dimensional scalar vector extracted with Shapely 2.0:
$$\Pi_{\text{geo}}(O) = \langle \texttt{area}, \texttt{perimeter}, \texttt{vertex_count}, \texttt{is_convex}, \texttt{centroid_x}, \texttt{centroid_y}, \texttt{bounds_minx}, \texttt{bounds_miny}, \texttt{bounds_maxx}, \texttt{bounds_maxy} \rangle$$
2. **Graph Domain (Π_{graph}):** An 8-dimensional vector of structural invariants extracted with NetworkX 3.0:
$$\Pi_{\text{graph}}(O) = \langle \texttt{node_count}, \texttt{edge_count}, \texttt{avg_degree}, \texttt{density}, \texttt{is_bipartite}, \texttt{is_connected}, \texttt{has_cycle}, \texttt{diameter} \rangle$$
3. **Tabular Domain (Π_{tab}):** An 8-dimensional vector of schema and summary statistics extracted with pandas 2.0:
$$\Pi_{\text{tab}}(O) = \langle \texttt{row_count}, \texttt{col_count}, \texttt{column_names}, \texttt{null_count}, \texttt{mean_numeric}, \texttt{max_abs_r}, \texttt{has_outlier}, \texttt{any_monotonic} \rangle$$
The state tracks these dimensions, but the rendered prompt summary omits `has_outlier` (and Π_{geo} omits `aspect_ratio`). As a result, a subset of property-conditioned gates tests unassisted derivation. Appendix H.3 gives the exact leak-fraction audit.

Pipeline Topologies and Gating Predicates. The role of Π depends on the pipeline topology:

- **Pass-Through Pipelines:** Linear transform chains ($O_t = f_t(O_{t-1})$) do not branch. Here $\Pi(O_t)$ is a structural sanity check, e.g., that a filter did not reduce a table to zero rows or disconnect a graph.
- **Property-Conditioned Pipelines:** Execution branches on an intermediate predicate $\pi(O_{t-1}) \bowtie \tau$. **Gating Invariant Inclusion:** In the primary Π-oracle condition (reported throughout the paper as Handle+Summary), $\Pi(O_t)$ directly exposes the exact gating predicate $\pi(O_{t-1})$ on 75.4%–87.9% of gated steps (the leak fraction; Appendix H.3). There, the task reduces from combinatorial derivation to a scalar comparison. On the remaining gates, the summary omits the predicate (`aspect_ratio` in geometry, `has_outlier` in tabular, and implicit `density` in graphs). On those gates, and in the explicit Π-miss condition (Appendix H.2), models must derive the condition from partial metadata or invoke on-demand reads. In the Predicate-Held-Out setting (Appendix H.3), Π is fixed a priori without task-specific tuning.
- **Multi-Object Pipelines:** Candidate selection queries ($O^* = \arg\max_i \pi(O_i)$) expose parallel summaries $\Pi(O_A), \Pi(O_B), \Pi(O_C)$. The agent can then rank the objects deterministically, directly in context.

Worked Context Example Across Conditions. Consider an agent at Step 2 of a graph pipeline that must decide, from the diameter, whether to add edges:

> **Task Instruction:** `"Examine the current graph. If its diameter is ≤ 3, call add_edges(g, [(0, 4)]); otherwise call remove_nodes(g, [2])."`
> **Underlying Graph State:** $V = \{0, 1, 2, 3, 4, 5\}$, $E = \{(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)\}$, Diameter $= 5 > 3$.

The four conditions show this state to the LLM as follows:

1. **Condition 1 (Raw Serialization):** The prompt re-injects the full text serialization:

   ```
   Current Graph State:
   Graph with 6 nodes and 5 edges:
   Edges: (0, 1), (1, 2), (2, 3), (3, 4), (4, 5)
   Choose the next tool call:
   ```

   Challenge: The model must parse the edge list, mentally compute the longest shortest path (diameter $= 5$), and branch.

2. **Condition 2 (Augmented):** The prompt keeps the raw text and adds the deterministic summary:

   ```
   Current Graph State:
   Graph with 6 nodes and 5 edges:
   Edges: (0, 1), (1, 2), (2, 3), (3, 4), (4, 5)
   [Runtime Summary: nodes=6, edges=5, avg_deg=1.67, diameter=5,
   bipartite=True]
   Choose the next tool call:
   ```

   Challenge: The summary gives the answer directly (`diameter=5`). However, the raw text bloats the prompt and must be re-emitted at every turn, which risks payload corruption.

3. **Condition 3 (Handle Only):** The runtime holds the object; the prompt shows only a handle reference:

   ```
   Current Graph State: <handle:graph_9e4c1a>
   Choose the next tool call:
   ```

   Challenge: The context is compact and immune to payload corruption. But the model cannot see the diameter, so it guesses blindly unless it invokes passive inspection tools.

4. **Condition 4 (Handle + Summary / Π-Oracle):** The prompt adds the deterministic summary to the handle reference:

   ```
   Current Graph State: <handle:graph_9e4c1a>
   [Runtime Summary: nodes=6, edges=5, avg_deg=1.67, diameter=5,
   bipartite=True]
   Choose the next tool call:
   ```

   Advantage: The context has no payload corruption or token overhead and exposes the exact scalar needed to branch. The model can select `remove_nodes(<handle:graph_9e4c1a>, [2])` at once.

D.3 Error Amplification at Each Depth k

This subsection asks whether errors grow faster with depth than independent step failures predict. Table 19 reports the Error Amplification Factor (EAF) at each depth. At $k = 1$, a pipeline has one step, so compounding cannot occur. The small deviations above 1.0 (1.02–1.37) are binomial noise on small subsets ($N = 20$–60). At every multi-step depth ($k \geq 2$), $\mathrm{EAF}(k) \leq 1.0$ for both raw and augmented conditions.

Takeaway. Compounding failure in tool-use agents does not show super-multiplicative runaway ($\mathrm{EAF} > 1.0$). Errors follow **multiplicative survival decay** with abort attrition. Step failures overlap: an agent that errs at step 2 often would have erred at step 3 anyway. Payload corruption also aborts pipelines early (Table 21), before more errors can accrue.

Table 19: **Error amplification at each pipeline depth ($k \in \{1, 2, 3, 4, 5, 7\}$).** Rows cover graph pipelines (DeepSeek-V4-Flash) and tabular pipelines (GPT-4.1-mini). Pred Fail$(k) = 1 - \prod_{t=1}^{k} p_t$; Obs Fail$(k) = 1 -$ Accuracy(k); EAF$(k) =$ Obs Fail/Pred Fail. Errors decay sub-multiplicatively: EAF$(k) \leq 1.0$ at every multi-step depth ($k \geq 2$).

Depth (k)	Domain & Model	Raw Obs Fail	Raw Pred Fail	Raw EAF	Aug Obs Fail	Aug Pred Fail	Aug EAF
$k = 1$	Graphs (V4-Flash)	0.117	0.085	1.373*	0.067	0.063	1.053*
$k = 2$	Graphs (V4-Flash)	0.108	0.135	0.801	0.092	0.105	0.870
$k = 3$	Graphs (V4-Flash)	0.150	0.221	0.680	0.133	0.194	0.687
$k = 4$	Graphs (V4-Flash)	0.350	0.391	0.895	0.267	0.340	0.783
$k = 5$	Graphs (V4-Flash)	0.475	0.584	0.813	0.300	0.452	0.664
$k = 7$	Graphs (V4-Flash)	0.600	0.720	0.833	0.400	0.580	0.690
$k = 1$	Tabular (GPT-4.1-mini)	0.683	0.668	1.022*	0.617	0.627	0.984
$k = 2$	Tabular (GPT-4.1-mini)	0.683	0.864	0.791	0.667	0.846	0.788
$k = 3$	Tabular (GPT-4.1-mini)	0.642	0.942	0.681	0.608	0.935	0.651
$k = 4$	Tabular (GPT-4.1-mini)	0.708	0.974	0.728	0.633	0.971	0.652
$k = 5$	Tabular (GPT-4.1-mini)	0.750	0.991	0.757	0.683	0.988	0.691
$k = 7$	Tabular (GPT-4.1-mini)	0.800	0.999	0.801	0.700	0.997	0.702

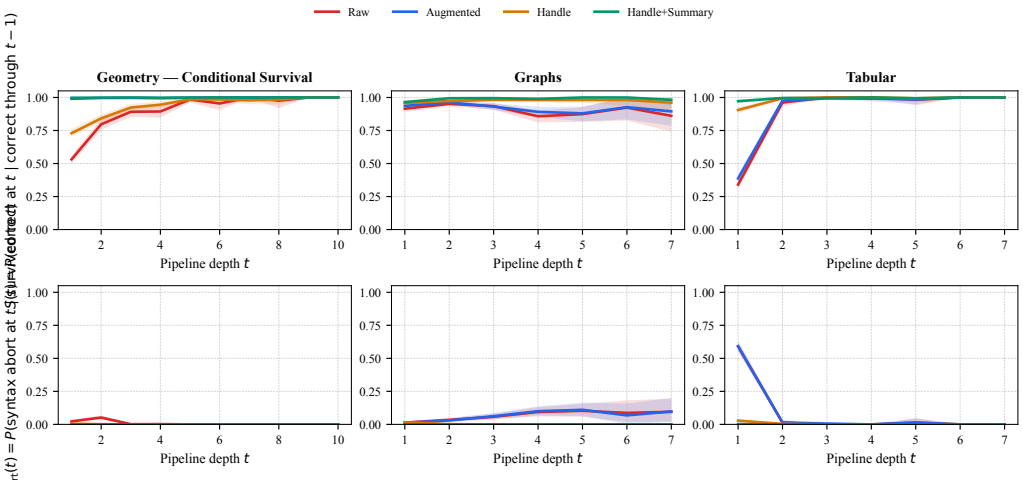


Figure 5: **Conditional survival $S(t)$ and abort hazard $h_{\text{abort}}(t)$ by condition and domain (DeepSeek-V4-Flash, cascading mode).** Top: per-step retention, conditional on surviving to t. Bottom: abort hazard at depth t. Pipeline-level bootstrap 95% CIs (B = 1,000). The raw tabular arm loses 59.7% of pipelines to a first-step abort, but survivors stabilize ($S(t \geq 2) = 0.99$). Raw graph aborts rise with depth (h_{abort} 0.013 $\rightarrow$ 0.10 by $t = 7$). Handle+Summary keeps $h_{\text{abort}} \leq 0.2\%$ at every depth.

D.4 Propagating State vs. Aborting on Errors

This subsection examines failures that do not abort. Failures in multi-step agents fall into two distinct regimes:

1. **Abort Regimes:** The model emits an argument that the tool cannot execute. Most are schema-valid but semantically corrupted payloads (mismatched columns, invalid placeholders, unresolvable operands); a few are malformed text such as unclosed CSV quotes (Table 26). The tool throws an exception and halts execution. In our benchmark, this appears as an explicit pipeline abort.
2. **Propagating-State Regimes:** The model emits syntactically valid but semantically corrupted arguments (e.g., inverted polygon coordinate order, a disconnected node index, a misapplied numerical filter). The tool runs without an exception. The runtime commits the corrupted output O_t to state and forwards it to step $t + 1$.

We measure this regime on 60 constructed pipelines (20 per domain; DeepSeek-V4-Flash, non-thinking). A syntactically valid argument changes the object, and the runtime does not abort. We score step $k + 1$ against the committed object. Under raw serialization, the model re-derives the

Table 20: **Propagating-state measurement: accuracy after injected corruption.** We inject corruption (`simplify`, hub-edge drop, inverted row filter). Handle+Summary equals 1.00 by construction: the runtime recomputes the property from the in-memory object, so this column is not a second model call. Arrest is 1 minus raw accuracy, not an independent measurement. A multi-step $\mathrm{EAF}(k)$ is not identified here: the corruption is injected, so there is no preceding model-step failure to compound.

Domain	N	Raw acc.	Handle+Summary	Arrest
Geometry (area)	20	0.15	1.00	0.85
Graphs (diameter)	20	0.20	1.00	0.80
Tabular (correlation)	20	0.65	1.00	0.35

Table 21: **Abort and non-abort frequency (Experiment 2).** Ranges are the minimum and maximum across the five full models. Non-abort is 1 minus the abort rate. †Constrained-decoding arms (Appendix E.2) were run on the abort-extreme model of each domain (Llama-4 Scout for graphs; Qwen3-32B for tabular), so those cells report that model, not a cross-model range. Geometry counts are the summary-log rates in Table 24; graph and tabular counts come from the evaluation logs behind Table 23.

Domain	Condition	Abort	Non-abort
Geometry	Raw	3.4%–35.7%	64.3%–96.6%
	Augmented	0.0%–22.6%	77.4%–100.0%
	Handle	0.9%–26.3%	73.7%–99.1%
	Handle + Summary	0.0%–9.2%	90.8%–100.0%
Graphs	Raw	16.2%–73.5%	26.5%–83.8%
	Raw + Constrained†	53.8%	46.2%
	Augmented + Constrained†	51.3%	48.7%
	Augmented	16.3%–71.2%	28.8%–83.7%
	Handle	1.7%–9.7%	90.3%–98.3%
	Handle + Summary	0.0%–5.0%	95.0%–100.0%
Tabular	Raw	59.7%–99.7%	0.3%–40.3%
	Raw + Constrained†	99.3%	0.7%
	Augmented + Constrained†	97.2%	2.8%
	Augmented	60.0%–99.2%	0.8%–40.0%
	Handle	1.2%–21.2%	78.8%–98.8%
	Handle + Summary	0.2%–13.8%	86.2%–99.8%

property from the perturbed text. Under Handle+Summary, the runtime recomputes that property from the in-memory object.

Table 21 reports how often an initiated pipeline aborts and how often it continues. The non-abort share includes two cases: correct completions, and executable but corrupted state that is then propagated. Table 20 measures the second case on 60 constructed pipelines, not on the natural abort split.

E Cross-Domain Synthesis (Experiment 3)

This section reports the full results of Experiment 3 (Table 22), audits how aborts are scored, and tests stronger text baselines.

Table 22: **Property recovery and pipeline compounding across domains (Experiment 3).** Domains: geometry (Geo), graph networks (Graph), and tabular dataframes (Tab). $\Delta_{\mathrm{LG}} = \mathrm{Local} - \mathrm{Global}$ is the accuracy gap; AG = Augmentation Gain; OHG = Object-Handle Gain. The Error Amplification Factor (EAF) is reported in Appendix D.3.

Domain	Model	Local (π_{loc})	Global (π_{glob})	Gap (Δ_{LG})	AG (Step)	OHG (Step)
Geometry	V4-Flash	0.663	0.334	+0.330	+0.407	+0.148
	Qwen3-32B	0.595	0.264	+0.331	+0.509	+0.040
	Llama-4 Scout	0.513	0.208	+0.306	+0.406	+0.120
	Gemini-2.5	0.838	0.272	+0.566	+0.449	-0.066
	GPT-4.1-mini	0.843	0.271	+0.572	+0.489	-0.024
Graphs	V4-Flash	0.995	0.372	+0.623	+0.015	+0.089
	Qwen3-32B	0.957	0.256	+0.701	-0.012	+0.434
	Llama-4 Scout	0.957	0.232	+0.725	-0.038	+0.531
	Gemini-2.5	0.957	0.274	+0.683	+0.059	+0.209
	GPT-4.1-mini	0.978	0.283	+0.695	+0.054	+0.057
Tabular	V4-Flash	0.921	0.310	+0.611	+0.016	+0.574
	Qwen3-32B	0.789	0.305	+0.484	-0.005	+0.891
	Llama-4 Scout	0.717	0.323	+0.394	-0.052	+0.507
	Gemini-2.5	0.847	0.297	+0.550	+0.013	+0.582
	GPT-4.1-mini	0.783	0.333	+0.450	+0.011	+0.494

E.1 Dual-Denominator Audit: Cascading vs. Oracle Execution

This subsection explains an anomaly from our audit. In some raw-string pipelines (for example, tabular and graphs), cascading end-to-end accuracy appeared to equal or exceed oracle accuracy. Table 23 audits pipeline completions under two denominators:

1. **All-Initiated Denominator ($N_{\mathrm{total}} = 600$ per model/domain):** A conservative measure of execution fidelity. Pipelines that abort because of malformed syntax, schema failures, or execution exceptions count as 0 (incorrect).
2. **Clean-Only Denominator ($N_{\mathrm{clean}} = N_{\mathrm{total}} - N_{\mathrm{abort}}$):** A conditional measure that leaves aborted pipelines out of the denominator.

Table 24 gives the same audit for geometry. In Table 23, raw tabular pipelines abort on 59.7%–99.7% of runs. Accuracy on the surviving pipelines spans 61.1%–100.0%. The 100.0% cell is Qwen3-32B, which aborted on 598 of 600 pipelines. Over all initiated pipelines, cascading and oracle accuracy nearly match: 32.7% and 32.8% for Flash, and 31.2% and 30.7% for GPT-4.1-mini. High clean-only accuracy can thus come from very few survivors.

Handle + Summary reduces aborts to 0.0%–5.0% in graphs and 0.2%–13.8% in tabular pipelines (the tabular maximum is Llama-4 Scout). It also raises Casc(All) to 77.0%–96.3%.

E.2 Constrained Decoding and ArtifactRef Baselines

This subsection asks whether the preservation effect survives two stronger text baselines. Many agent runtimes now emit tool calls under provider-enforced structured output (JSON-schema-constrained decoding or typed function calling), which removes malformed tool-call envelopes by construction. We add four constrained conditions: `Raw+Constrained`, `Augmented+Constrained`, `Handle+Constrained`, and `Handle+Summary+Constrained`. Each keeps the prompts, pipelines, object IDs, and 512-token completion budget of its unconstrained base condition. The only change is that we decode the assistant turn under a strict per-signature JSON schema. Each tool in the grammar of Appendix D.1 has one variant, which fixes the tool name enum, the object array arity, the argument arity, and the scalar types. The serialized-object payload stays free text, since no schema can validate the semantics of an edge list. We evaluate the two abort-extreme cells: Qwen3-32B on tabular (99.7% raw aborts) and Llama-4 Scout on graphs (73.5% raw aborts). Before launch, we smoke-tested enforcement on each endpoint: a trap prompt that asks for a schema-violating argument type still yields schema-valid output. We serve the Llama-4 Scout arm unrouted, because the pinned

Table 23: **Dual-denominator audit of pipeline execution fidelity.** The Handle+Constrained† rows are the initial runs without `get_object` in the action enum; Table 25 gives the primary runs. Rows cover five full models on graph and tabular pipelines ($N = 600$ each; the V4-Pro 20% subsample is omitted). †Constrained-decoding arms (Appendix E.2) on the abort-extreme cells. **Abort Rate**: fraction of initiated pipelines that aborted from parse, truncation, or semantic payload errors. **Casc(All)**: cascading accuracy over all initiated pipelines, with aborts scored as 0. **Casc(Clean)**: cascading accuracy over surviving pipelines. **Oracle**: oracle end-to-end accuracy over all initiated pipelines. Across the 40 unconstrained cells, Oracle exceeds Cascading in 9, ties in 14, and Cascading exceeds Oracle in 17, by 0.001 to 0.024. Across all 48 rows, including the eight †-marked constrained arms, the tally is 13 / 14 / 21, with maximum deviation 0.030 (tabular Qwen Handle+Summary+Constrained).

Domain	Model	Condition	Abort Rate ($N_{\text{ab}}/N_{\text{tot}}$)	Cascading (All) [N_{tot}]	Cascading (Clean) [N_{clean}]	Oracle [N_{tot}]
Graphs	V4-Flash	Raw Serialization	97 / 600 (16.2%)	0.738	0.881	0.738
		Handle Only	10 / 600 (1.7%)	0.905	0.920	0.892
		Augmented Serialization	98 / 600 (16.3%)	0.778	0.930	0.777
		Handle + Summary	**0 / 600 (0.0%)**	**0.950**	**0.950**	**0.948**
	Qwen3-32B	Raw Serialization	326 / 600 (54.3%)	0.360	0.788	0.345
		Handle Only	51 / 600 (8.5%)	0.703	0.769	0.702
		Augmented Serialization	383 / 600 (63.8%)	0.345	0.954	0.372
		Handle + Summary	**0 / 600 (0.0%)**	**0.877**	**0.877**	**0.877**
	Llama-4 Scout	Raw Serialization	441 / 600 (73.5%)	0.173	0.654	0.165
		Raw + Constrained†	323 / 600 (53.8%)	0.297	0.643	0.288
		Augmented Serialization	427 / 600 (71.2%)	0.237	0.821	0.213
		Augmented + Constrained†	307 / 599 (51.3%)	0.386	0.791	0.368
		Handle Only	58 / 600 (9.7%)	0.747	0.827	0.743
		Handle + Constrained†	179 / 599 (29.9%)	0.389	0.555	0.384
		Handle + Summary	**26 / 600 (4.3%)**	**0.852**	**0.890**	**0.847**
		Handle + Summary + Constrained†	27 / 596 (4.5%)	0.770	0.807	0.779
	Gemini-2.5	Raw Serialization	277 / 600 (46.2%)	0.502	0.932	0.502
		Handle Only	24 / 600 (4.0%)	0.618	0.644	0.618
		Augmented Serialization	250 / 600 (41.7%)	0.580	0.994	0.580
		Handle + Summary	**30 / 600 (5.0%)**	**0.910**	**0.958**	**0.910**
	GPT-4.1-mini	Raw Serialization	131 / 600 (21.8%)	0.673	0.861	0.672
		Handle Only	56 / 600 (9.3%)	0.702	0.774	0.700
		Augmented Serialization	103 / 600 (17.2%)	0.763	0.922	0.763
		Handle + Summary	**1 / 600 (0.2%)**	**0.917**	**0.918**	**0.917**
Tabular	V4-Flash	Raw Serialization	358 / 600 (59.7%)	0.327	0.810	0.328
		Handle Only	19 / 600 (3.2%)	0.898	0.928	0.905
		Augmented Serialization	360 / 600 (60.0%)	0.373	0.933	0.375
		Handle + Summary	**1 / 600 (0.2%)**	**0.963**	**0.965**	**0.963**
	Qwen3-32B	Raw Serialization	598 / 600 (99.7%)	0.003	1.000	0.005
		Raw + Constrained†	595 / 599 (99.3%)	0.007	1.000	0.005
		Augmented Serialization	595 / 600 (99.2%)	0.005	0.600	0.003
		Augmented + Constrained†	582 / 599 (97.2%)	0.023	0.824	0.027
		Handle Only	79 / 600 (13.2%)	0.695	0.800	0.682
		Handle + Constrained†	198 / 600 (33.0%)	0.402	0.600	0.418
		Handle + Summary	**15 / 600 (2.5%)**	**0.913**	**0.937**	**0.908**
		Handle + Summary + Constrained†	148 / 600 (24.7%)	0.692	0.918	0.722
	Llama-4 Scout	Raw Serialization	433 / 600 (72.2%)	0.170	0.611	0.173
		Handle Only	127 / 600 (21.2%)	0.557	0.706	0.533
		Augmented Serialization	465 / 600 (77.5%)	0.135	0.600	0.142
		Handle + Summary	**83 / 600 (13.8%)**	**0.770**	**0.894**	**0.787**
	Gemini-2.5	Raw Serialization	467 / 600 (77.8%)	0.187	0.842	0.187
		Handle Only	38 / 600 (6.3%)	0.597	0.637	0.597
		Augmented Serialization	453 / 600 (75.5%)	0.225	0.918	0.225
		Handle + Summary	**3 / 600 (0.5%)**	**0.920**	**0.925**	**0.920**
	GPT-4.1-mini	Raw Serialization	378 / 600 (63.0%)	0.312	0.842	0.307
		Handle Only	7 / 600 (1.2%)	0.660	0.668	0.655
		Augmented Serialization	361 / 600 (60.2%)	0.358	0.900	0.358
		Handle + Summary	**5 / 600 (0.8%)**	**0.918**	**0.926**	**0.932**

novita/bf16 endpoint does not offer structured outputs (otherwise the OpenRouter request returns 404). Table 25 reports all arms, and Table 26 splits aborts by cause.

First, the preservation coefficient survives constrained decoding. β_{pres} drops by only 2% on the tabular extreme and 25% on the graph extreme. This is because 88%–94% of raw aborts are schema-valid but semantically wrong payloads, which no output grammar can repair (Table 26).

Second, the effect of constrained decoding is limited to envelope syntax. Non-truncation parse failures vanish, but completion-budget truncations and payload corruption persist.

Third, the composed `Handle+Constrained` and `Handle+Summary+Constrained` arms show how output constraints interact with state inspection. In our first runs, the schema's tool enum omitted the inspection primitive `get_object`. There, adding constrained decoding to handle architectures appeared to cause a penalty: Casc(All) dropped from 85.2% to 77.0% in graphs (-8.2 pp) and from 91.3% to 69.2% in tabular (-22.1 pp). We then re-ran both composed arms with

Table 24: **Dual-denominator audit for geometry ($N = 924$ per full model).** Columns match Table 23. Counts come from the Experiment 2 summary logs. Casc(All) − Oracle is the exact difference between the two recorded end-to-end correct counts on the same 924 pipelines. The original non-describe geometry JSONL was not retained. We have since re-run the V4-Flash geometry column end-to-end on the identical 924 pipelines; it confirms cells within ± 3.5 points and abort counts within 2 of 924, which supports per-pipeline recomputation.

Model	Condition	Abort Rate	Casc(All)	Casc(Clean)	Oracle
V4-Flash	Raw	63/924 (6.8%)	0.399	0.429	0.389
	Augmented	0/924 (0.0%)	0.988	0.988	0.983
	Handle	8/924 (0.9%)	0.598	0.604	0.583
	Handle + Summary	0/924 (0.0%)	0.994	0.994	0.992
Qwen3-32B	Raw	120/924 (13.0%)	0.252	0.290	0.246
	Augmented	28/924 (3.0%)	0.772	0.796	0.777
	Handle	55/924 (6.0%)	0.246	0.261	0.249
	Handle + Summary	20/924 (2.2%)	0.792	0.810	0.794
Llama-4 Scout	Raw	330/924 (35.7%)	0.194	0.301	0.187
	Augmented	209/924 (22.6%)	0.472	0.610	0.482
	Handle	243/924 (26.3%)	0.244	0.330	0.246
	Handle + Summary	85/924 (9.2%)	0.753	0.830	0.736
Gemini-2.5	Raw	52/924 (5.6%)	0.291	0.308	0.291
	Augmented	47/924 (5.1%)	0.792	0.835	0.792
	Handle	39/924 (4.2%)	0.219	0.228	0.219
	Handle + Summary	63/924 (6.8%)	0.795	0.854	0.795
GPT-4.1-mini	Raw	31/924 (3.4%)	0.329	0.340	0.331
	Augmented	14/924 (1.5%)	0.881	0.895	0.879
	Handle	11/924 (1.2%)	0.271	0.274	0.273
	Handle + Summary	31/924 (3.4%)	0.925	0.957	0.926

`get_object` in the schema enum. In graphs, the penalty thus came from the action space, not from the output constraint. With the inspection primitive, graph Handle+Constrained recovers $+21.8$ pp in Casc(All), and its clean accuracy reaches 74.7%. Handle+Summary+Constrained recovers $+6.0$ pp ($77.0\% \rightarrow 83.0\%$), and its aborts fall to 2.3% (Table 26). In graphs, models query `get_object` on 22.9% of Handle+Constrained steps (58.5% of units) and 4.5% of Handle+Summary+Constrained steps. In tabular data, clean accuracy recovers $+19.0$ pp, and models query `get_object` on 18.9% of steps (52.2% of units). Takeaway: with the inspection primitive, constrained decoding keeps graph execution valid; in tabular data, aborts stay high (27.8%–41.2%).

Fourth, the observation-with-reference baseline (`Raw+ArtifactRef` and `Augmented+ArtifactRef`) separates observation from re-emission (Table 27, rightmost columns). The model sees the complete extensive object $\sigma(O_t)$ in context, but tool calls take an immutable server-side artifact ID instead of a re-emitted payload operand:

- In **Tabular** data (Qwen3-32B), aborts fall from 99.7% ($598/600$) to 1.2% ($6/521$). Semantic-payload unresolvable operands fall from 563 to 0. Casc(All) rises from 0.3% to 76.0% (91.1% under `Augmented+ArtifactRef`). β_{pres} drops from $+0.891$ to -0.025. In this bounded, re-emission-heavy tabular interface, action-payload generation, not loss of raw observation, explains most of the preservation contrast.
- In **Graphs** (Llama-4 Scout), aborts drop from 73.5% ($441/600$) to 46.3% ($130/281$). Semantic-payload unresolvable operands fall from 390 to 0. Casc(All) rises from 17.3% to 42.7% (70.0% under `Augmented+ArtifactRef`). β_{pres} drops from $+0.531$ to $+0.180$. Artifact referencing thus resolves nearly two-thirds (66%) of the preservation gap. The significant residual of $+0.180$ reflects a real reasoning cost of processing multi-hundred-token edge lists in the prompt.

Factorial Regression Statistics and Step Counts. We fit ordinary least-squares regressions over all executed pipeline steps of DeepSeek-V4-Flash: $N = 16,896$ geometry steps with $G = 924$

Table 25: **Constrained-decoding arms on the abort-extreme cells (Experiment 2).** Columns match Table 23. $N = 600$ pipelines per condition per domain, except for lost units. Lost to endpoint timeouts: one Qwen `augmented_constrained` and one `raw_constrained` ($N = 599$). Original restricted runs: one Llama `handle_constrained` ($N = 599$); four Llama `handle_sum_constrained` ($N = 596$). Re-runs include `get_object` in the tool enum; the table shows both the primary complete-action-space results and the initial restricted-enum ablation.

Domain	Model	Condition	Abort Rate ($N_{\text{ab}}/N_{\text{tot}}$)	Casc(All)	Casc(Clean)	Oracle E2E
Graphs	**Llama-4 Scout**	Raw Serialization	441/600 (73.5%)	0.173	0.654	0.165
		Raw + Constrained	**323/600 (53.8%)**	**0.297**	**0.643**	**0.288**
		Raw + ArtifactRef	**130/281 (46.3%)**	**0.427**	**0.795**	**0.502**
		Augmented Serialization	427/600 (71.2%)	0.237	0.821	0.213
		Augmented + Constrained	**307/599 (51.3%)**	**0.386**	**0.791**	**0.368**
		Augmented + ArtifactRef	**66/273 (24.2%)**	**0.700**	**0.923**	**0.708**
		Handle Only	58/600 (9.7%)	0.747	0.827	0.743
		Handle + Constrained	**113/600 (18.8%)**	**0.607**	**0.747**	**0.608**
		– *Action enum without* `get_obj`	179/599 (29.9%)	0.389	0.555	0.384
		Handle + Summary	26/600 (4.3%)	0.852	0.890	0.847
		Handle + Summary + Constrained	**14/600 (2.3%)**	**0.830**	**0.850**	**0.832**
		– *Action enum without* `get_obj`	27/596 (4.5%)	0.770	0.807	0.779
Tabular	**Qwen3-32B**	Raw Serialization	598/600 (99.7%)	0.003	1.000	0.005
		Raw + Constrained	**595/599 (99.3%)**	**0.007**	**1.000**	**0.005**
		Raw + ArtifactRef	**6/521 (1.2%)**	**0.760**	**0.769**	**0.773**
		Augmented Serialization	595/600 (99.2%)	0.005	0.600	0.003
		Augmented + Constrained	**582/599 (97.2%)**	**0.023**	**0.824**	**0.027**
		Augmented + ArtifactRef	**1/518 (0.2%)**	**0.911**	**0.913**	**0.909**
		Handle Only	79/600 (13.2%)	0.695	0.800	0.682
		Handle + Constrained	**247/600 (41.2%)**	**0.465**	**0.790**	**0.453**
		– *Action enum without* `get_obj`	198/600 (33.0%)	0.402	0.600	0.418
		Handle + Summary	15/600 (2.5%)	0.913	0.937	0.908
		Handle + Summary + Constrained	**167/600 (27.8%)**	**0.662**	**0.917**	**0.672**
		– *Action enum without* `get_obj`	148/600 (24.7%)	0.692	0.918	0.722

Table 26: **Abort causes in unconstrained and constrained architectures.** Cascading-mode aborts split into schema-invalid envelopes (non-truncation parse errors vs. completion-budget truncations) and schema-valid but semantically wrong payloads (unresolvable operand, execution error, invalid result). In the complete action space, with `get_object` in the schema enum, Handle+Constrained models query `get_object` on 22.9% of graph steps and 18.9% of tabular steps. This recovers +21.8 pp in graph Casc(All) (38.9% → 60.7%) and +19.0 pp in tabular clean accuracy (60.0% → 79.0%). Under Handle+Summary+Constrained, graph aborts drop to 2.3% (14/600) with a 4.5% `get_object` call rate, and Casc(All) recovers to 83.0%, within 2.2 pp of unconstrained 85.2%.

Domain / Model	Condition	Aborted	Schema: parse	Schema: trunc.	Semantic payload
Tabular (Qwen3-32B)	Raw	598/600	0	35	563
	Raw + Constrained	595/599	0	129	466
	Raw + ArtifactRef	6/521	4	0	2
	Augmented + ArtifactRef	1/518	0	0	1
	Handle Only	79/600	0	74	5
	Handle + Constrained (`+get_obj`)	247/600	0	46	201
	– *without* `get_obj` *in enum*	198/600	0	0	198
	Handle + Summary	15/600	2	12	1
	Handle + Summary + Constrained (`+get_obj`)	167/600	0	9	158
	– *without* `get_obj` *in enum*	148/600	0	0	148
Graphs (Llama-4 Scout)	Raw	441/600	19	32	390
	Raw + Constrained	323/600	0	51	272
	Raw + ArtifactRef	130/281	7	0	123
	Augmented + ArtifactRef	66/273	8	0	58
	Handle Only	58/600	0	5	53
	Handle + Constrained (`+get_obj`)	113/600	0	41	72
	– *without* `get_obj` *in enum*	179/599	0	0	179
	Handle + Summary	26/600	3	0	23
	Handle + Summary + Constrained (`+get_obj`)	14/600	0	5	9
	– *without* `get_obj` *in enum*	27/596	0	0	27

pipeline clusters, and $N = 8,640$ steps with $G = 600$ clusters each for graphs and tabular. We add non-parametric bootstrap resampling ($B = 5,000$ iterations), clustered at the pipeline instance level. Pooled over all five full models, the step counts are $84,480$ for geometry and $43,200$ each for graphs and tabular. Cluster degrees of freedom are $G - 1$ (923 for geometry, 599 for graphs and tabular).

Table 27: **Factorial re-estimation with constrained-decoding and artifact-ID baselines (Equation 1).** We regress per-step oracle accuracy on Exposure, Preservation, and their interaction, with pipeline-clustered standard errors, for three text baselines: (1) unconstrained Raw, which reproduces per-model β_{pres} from Table 29; (2) Raw+Constrained / Augmented+Constrained; (3) Raw+ArtifactRef / Augmented+ArtifactRef (observation-with-reference). In tabular data, passing artifact IDs removes payload re-emission and collapses β_{pres} from $+0.891$ to -0.025. In graphs, artifact IDs reduce β_{pres} from $+0.531$ to $+0.180$. This resolves two-thirds of the preservation gap and isolates the residual cost of extensive context.

		Baseline 1: Raw (Unconstrained)				Baseline 2: Raw + Constrained				Baseline 3: Raw + ArtifactRef			
Domain (Model)	**Term**	β	**Clust. SE**	p	**95% CI**	β	**Clust. SE**	p	**95% CI**	β	**Clust. SE**	p	**95% CI**
Tabular	β_{exp}	-0.005	0.003	0.131	[-0.012, 0.002]	+0.018	0.010	0.056	[-0.000, 0.037]	+0.040	0.008	1.0×10^{-6}	[0.024, 0.057]
(Qwen3-32B)	β_{pres}	+0.891	0.008	$< 10^{-200}$	[0.876, 0.906]	+0.874	0.009	$< 10^{-200}$	[0.857, 0.891]	-0.025	0.006	9.1×10^{-5}	[-0.038, -0.013]
	β_{int}	+0.075	0.008	5.8×10^{-22}	[0.060, 0.090]	+0.052	0.012	2.4×10^{-5}	[0.028, 0.075]	+0.030	0.009	1.8×10^{-3}	[0.011, 0.048]
Graphs	β_{exp}	-0.038	0.021	0.074	[-0.080, 0.004]	+0.008	0.023	0.732	[-0.037, 0.053]	+0.097	0.021	3.5×10^{-6}	[0.056, 0.137]
(Llama-4 Scout)	β_{pres}	+0.531	0.018	1.7×10^{-186}	[0.495, 0.566]	+0.400	0.019	4.4×10^{-98}	[0.363, 0.438]	+0.180	0.022	2.1×10^{-16}	[0.137, 0.222]
	β_{int}	+0.077	0.022	4.9×10^{-4}	[0.034, 0.121]	+0.032	0.024	0.189	[-0.016, 0.079]	-0.057	0.021	7.4×10^{-3}	[-0.099, -0.015]

Tabular: $N = 8{,}640$ / 8,600 / 7,795 steps, $G = 600$ clusters, $R^2 = 0.868$ / 0.813 / 0.016. Graphs: $N = 8{,}640$ / 8,640 / 5,849 steps, $G = 600$, $R^2 = 0.356$ / 0.218 / 0.055.

Table 28: **Full factorial regression with pipeline-clustered standard errors (Experiment 3), recomputed from retained per-step logs.** Fitted model: $\text{Acc}_{\text{step}} \sim \beta_0 + \beta_{\text{exp}} \cdot \text{Exposure} + \beta_{\text{pres}} \cdot \text{Preservation} + \beta_{\text{int}} \cdot (\text{Exposure} \times \text{Preservation}) + \varepsilon$, with standard errors clustered at the pipeline instance level (DeepSeek-V4-Flash, oracle mode). Cluster degrees of freedom are $G - 1$ (923 geometry; 599 graphs and tabular). Geometry has $N = 16{,}896$ steps ($924 \times 4 \times 4.57$); graphs and tabular have $N = 8{,}640$ steps ($600 \times 4 \times 3.6$). For geometry cells, the saturated OLS regression uses individual step records (depth $k \in \{2, 4, 7, 10\}$, mean depth 4.57) from the clean primary rerun ($\text{Raw} = 0.572$, $\text{Aug} = 0.996$, $\text{Handle} = 0.701$, $\text{H} + \text{S} = 0.998$). These values exactly reproduce the regression coefficients ($\beta_0 = 0.572, \beta_0 + \beta_{\text{exp}} = 0.996, \beta_0 + \beta_{\text{pres}} = 0.701, \sum \beta = 0.998$). They differ slightly from the original run's pipeline-level averages in Table 16 ($0.588, 0.995, 0.736, 0.996$), because unequal pipeline lengths (k) make step pooling weight deeper trajectories more heavily.

Domain / Regression Term	**Coefficient (β)**	**Clustered SE**	**t-statistic**	**p-value**	**95% CI**
Geometry ($R^2 = 0.231$, $N = 16{,}896$ steps, $N_{\text{clusters}} = 924$, Cluster df $= 923$)					
Intercept (β_0)	+0.572	0.015	37.60	$< 10^{-300}$	[0.543, 0.602]
Exposure (β_{exp})	+0.424	0.015	27.86	8.4×10^{-171}	[0.394, 0.454]
Preservation (β_{pres})	+0.129	0.017	7.72	1.2×10^{-14}	[0.096, 0.161]
Interaction (β_{int})	-0.127	0.017	-7.58	3.5×10^{-14}	[-0.160, -0.094]
Graphs ($R^2 = 0.033$, $N = 8{,}640$ steps, $N_{\text{clusters}} = 600$, Cluster df $= 599$)					
Intercept (β_0)	+0.876	0.010	85.87	$< 10^{-300}$	[0.856, 0.896]
Exposure (β_{exp})	+0.015	0.004	3.46	5.4×10^{-4}	[0.006, 0.023]
Preservation (β_{pres})	+0.089	0.010	8.75	2.1×10^{-18}	[0.069, 0.109]
Interaction (β_{int})	+0.005	0.005	0.99	0.32	[-0.005, 0.014]
Tabular ($R^2 = 0.389$, $N = 8{,}640$ steps, $N_{\text{clusters}} = 600$, Cluster df $= 599$)					
Intercept (β_0)	+0.399	0.021	18.89	1.3×10^{-79}	[0.357, 0.440]
Exposure (β_{exp})	+0.016	0.004	3.62	2.9×10^{-4}	[0.007, 0.024]
Preservation (β_{pres})	+0.574	0.022	26.65	1.7×10^{-156}	[0.531, 0.616]
Interaction (β_{int})	+0.002	0.005	0.36	0.72	[-0.008, 0.012]

Table 28 reports the full regression statistics for the factorial interaction model:

$$\text{Acc}_{\text{step}} \sim \beta_0 + \beta_{\text{exp}} \cdot \text{Exposure} + \beta_{\text{pres}} \cdot \text{Preservation} + \beta_{\text{int}} \cdot (\text{Exposure} \times \text{Preservation}) + \varepsilon \quad (4)$$

with pipeline-clustered standard errors. In geometry, decision exposure drives execution ($\beta_{\text{exp}} = +0.424$), and the interaction is negative ($\beta_{\text{int}} = -0.127$). Exposure saturates early on compact serializations, so the combination has little left to add. In graphs and tabular data, state preservation dominates ($\beta_{\text{pres}} = +0.089$ and $+0.574$), and there is no significant positive interaction ($\beta_{\text{int}} = +0.005$, $p = 0.32$ and $+0.002$, $p = 0.72$). Exposure over raw text cannot bind under payload corruption. Handles and summaries contribute additively, not synergistically, at the per-step level.

Table 29: **Factorial regressions for all five full models (Equation 1).** Fitted model: $\mathrm{Acc}_{\mathrm{step}} \sim \beta_0 + \beta_{\mathrm{exp}} \cdot \mathrm{Exposure} + \beta_{\mathrm{pres}} \cdot \mathrm{Preservation} + \beta_{\mathrm{int}} \cdot (\mathrm{Exposure} \times \mathrm{Preservation}) + \varepsilon$. Standard errors are clustered at the pipeline instance level ($G = 924$ for geometry, $N = 16{,}896$ steps; $G = 600$ for graphs and tabular, $N = 8{,}640$ steps; cluster df $= G - 1$) and shown in parentheses. Across the five models, graph β_{pres} spans $+0.057$ to $+0.531$ and tabular β_{pres} spans $+0.494$ to $+0.891$. Our primary single-model estimates are conservative: DeepSeek-V4-Flash sits near the lower bound of both ($+0.089$ and $+0.574$).

Model	Domain	Intercept (β_0)	Exposure (β_{exp})	Preservation (β_{pres})	Interaction (β_{int})	R^2
DeepSeek-V4-Flash	Geometry	+0.572 (0.015)	+0.424 (0.015)	+0.129 (0.017)	−0.127 (0.017)	0.231
	Graphs	+0.876 (0.010)	+0.015 (0.004)	+0.089 (0.010)	+0.005 (0.005)	0.033
	Tabular	+0.399 (0.021)	+0.016 (0.004)	+0.574 (0.022)	+0.002 (0.005)	0.389
Qwen3-32B	Geometry	+0.612 (0.017)	+0.252 (0.017)	+0.183 (0.018)	−0.118 (0.020)	0.121
	Graphs	+0.518 (0.021)	−0.012 (0.006)	+0.434 (0.021)	+0.088 (0.014)	0.252
	Tabular	+0.096 (0.007)	−0.005 (0.003)	+0.891 (0.008)	+0.075 (0.008)	0.868
Llama-4 Scout	Geometry	+0.548 (0.016)	+0.259 (0.016)	+0.011 (0.016)	−0.025 (0.019)	0.123
	Graphs	+0.375 (0.020)	−0.038 (0.021)	+0.531 (0.018)	+0.077 (0.022)	0.356
	Tabular	+0.352 (0.018)	−0.052 (0.015)	+0.507 (0.021)	+0.169 (0.022)	0.342
Gemini-2.5-Flash-Lite	Geometry	+0.655 (0.015)	+0.286 (0.016)	+0.055 (0.017)	−0.061 (0.019)	0.134
	Graphs	+0.724 (0.018)	+0.059 (0.014)	+0.209 (0.018)	+0.064 (0.017)	0.106
	Tabular	+0.404 (0.019)	+0.013 (0.004)	+0.582 (0.021)	+0.105 (0.012)	0.428
GPT-4.1-mini	Geometry	+0.551 (0.015)	+0.433 (0.016)	+0.042 (0.016)	−0.040 (0.019)	0.222
	Graphs	+0.871 (0.014)	+0.054 (0.012)	+0.057 (0.017)	+0.034 (0.016)	0.035
	Tabular	+0.485 (0.019)	+0.011 (0.003)	+0.494 (0.022)	+0.076 (0.008)	0.359

E.3 Observational Length Regression (Descriptive)

This subsection gives the observational length regression that Section 6 treats as descriptive only. It asks whether serialization length changes the return to decision exposure relative to state preservation within a domain. We estimate:

$$\mathrm{Acc}_{\mathrm{step}} \sim \beta_0 + \sum_{d \in \{\mathrm{graph,tab}\}} \gamma_d D_d + \beta_{\mathrm{exp}} \cdot \mathrm{Exposure} + \beta_{\mathrm{pres}} \cdot \mathrm{Preservation} + \beta_\ell \cdot \ell_k + \beta_{\mathrm{exp}\times\ell} \cdot (\mathrm{Exposure} \times \ell_k) + \beta_{\mathrm{pres}\times\ell} \cdot (\mathrm{Preservation} \times \ell_k) + \varepsilon \quad (5)$$

where $\ell_k = \ell(\sigma)/1000$ is the serialization length in thousands of tokens and D_d are domain fixed effects, with geometry as the reference. We cluster standard errors at the pipeline instance level ($N_{\mathrm{clusters}} = 2{,}124$; cluster df $G - 1 = 2{,}123$).

Within a domain, part of the length variation comes from the instance difficulty tiers (Simple, Medium, Hard). Length could therefore absorb structural complexity rather than token count alone. We estimate three specifications. **Specification 1 (Primary)** uses domain fixed effects without tier controls. **Specification 2 (Complexity Controlled)** adds difficulty tier fixed effects ($\delta_{\mathrm{med}}, \delta_{\mathrm{hard}}$). Table 30 reports all coefficients with clustered standard errors and the delta-method crossover; the derivation follows below of the regime crossover point ℓ^*.

Delta-method derivation of the crossover ℓ^*. At the regime crossover ℓ^*, the marginal gain in step accuracy from decision exposure equals the gain from state preservation:

$$\frac{\partial \mathbb{E}[\mathrm{Acc}_{\mathrm{step}}]}{\partial \mathrm{Exposure}} = \frac{\partial \mathbb{E}[\mathrm{Acc}_{\mathrm{step}}]}{\partial \mathrm{Preservation}} \implies \beta_{\mathrm{exp}} + \beta_{\mathrm{exp}\times\ell} \cdot \ell_k^* = \beta_{\mathrm{pres}} + \beta_{\mathrm{pres}\times\ell} \cdot \ell_k^* \quad (6)$$

Solving for ℓ_k^* gives a non-linear estimator:

$$\ell_k^* = g(\boldsymbol{\beta}) = \frac{\beta_{\mathrm{exp}} - \beta_{\mathrm{pres}}}{\beta_{\mathrm{pres}\times\ell} - \beta_{\mathrm{exp}\times\ell}} = \frac{\Delta\beta_1}{\Delta\beta_2} \quad (7)$$

For Specification 1, $\Delta\beta_1 = 0.2435 - (-0.0014) = 0.2449$ and $\Delta\beta_2 = 0.6870 - (-0.1928) = 0.8798$. This gives $\ell_k^* = 0.278$ (278 tokens). We obtain the asymptotic variance with the delta method on the cluster-robust covariance matrix (SE $= 10.2$ tokens; 95% CI $[258, 298]$).

Two specification checks test this estimate. Specification 2 adds tier main effects. The crossover ℓ^* depends only on the interaction coefficients, and these do not change when tier main effects are

Table 30: **Within-domain length moderation models (Equation 5), recomputed from retained per-step logs.** Dependent variable: per-step oracle accuracy. Standard errors are clustered at the pipeline instance level ($G = 2{,}124$ clusters, cluster df $= 2{,}123$; $N = 34{,}176$ steps: 16,896 geometry $+8{,}640$ graph $+8{,}640$ tabular). Specification 1: domain fixed effects. Specification 2 adds tier main effects; Specification 3 adds tier $\times$ Exposure and tier $\times$ Preservation interactions. The crossover ℓ^* comes from the delta method on the cluster-robust covariance matrix. Cells: coefficient (clustered SE).

Term	**Spec 1: Domain FE**	**Spec 2: + Tier FE**	**Spec 3: + Tier $\times$ Interventions**
Intercept (β_0)	$+0.722$ (0.011)	$+0.754$ (0.012)	$+0.799$ (0.014)
Domain: Graph (γ_{graph})	$+0.172$ (0.007)	$+0.168$ (0.008)	$+0.168$ (0.008)
Domain: Tabular (γ_{tab})	$+0.067$ (0.013)	$+0.052$ (0.012)	$+0.052$ (0.012)
Tier: Medium (δ_{med})	—	-0.057 (0.008)	-0.119 (0.018)
Tier: Hard (δ_{hard})	—	-0.052 (0.011)	-0.185 (0.023)
Exposure (β_{exp})	$+0.244$ (0.009)	$+0.244$ (0.009)	$+0.188$ (0.012)
Preservation (β_{pres})	-0.001 (0.007)	-0.001 (0.007)	-0.035 (0.008)
Length ℓ_k (β_ℓ)	-0.550 (0.019)	-0.522 (0.020)	-0.471 (0.020)
Exposure $\times$ ℓ_k	-0.193 (0.010)	-0.193 (0.010)	-0.278 (0.016)
Preservation $\times$ ℓ_k	$+0.687$ (0.021)	$+0.687$ (0.021)	$+0.669$ (0.024)
Tier$_{\text{med}}\times$ Exposure	—	—	$+0.055$ (0.016)
Tier$_{\text{hard}}\times$ Exposure	—	—	$+0.219$ (0.021)
Tier$_{\text{med}}\times$ Preservation	—	—	$+0.069$ (0.013)
Tier$_{\text{hard}}\times$ Preservation	—	—	$+0.048$ (0.016)
R^2	0.402	0.405	0.417
Regime crossover ℓ^*	**278 tokens** (SE 10)	**278 tokens** (SE 10)	**236 tokens** (SE 15)
95% delta-method CI	**[258, 298]**	**[258, 298]**	**[207, 265]**

added. The point estimate is therefore unchanged by construction (278 tokens). So Specification 2 is not evidence about tier confounding, and we do not present it as such. The informative check is Specification 3, which adds tier $\times$ Exposure and tier $\times$ Preservation interactions. Here the crossover moves to 236 tokens (95% CI $[207, 265]$). The hard-tier $\times$ Exposure term is significant ($+0.219$, $p = 4.0 \times 10^{-26}$), so harder tiers raise the return to exposure. Takeaway: tier co-movement explains a real but bounded share of the transition. The causal padding intervention (Appendix E.4) isolates the pure-length component.

Re-fit without pass-through pipelines ($\ell^* = 344$). We also re-fit the model on a matched subset; Table 31 gives its full regression. The re-fit asks whether pass-through pipelines move the crossover. Pass-through pipelines in the canonical set (8,832 steps across the three domains) test direct string forwarding, with no multi-object routing or state derivation. Excluding them leaves $N = 25{,}344$ steps across $G = 1{,}556$ pipeline clusters (cluster df $= 1{,}555$). These comprise 13,824 multi-object steps (3,072 geometry $+5{,}376$ graph $+5{,}376$ tabular) and 11,520 property-conditioned steps (5,760 geometry $+2{,}880$ graph $+2{,}880$ tabular). On this subset, the regime crossover shifts from 278 tokens (95% CI $[258, 298]$) to $\ell^* = 344.28$ tokens (95% CI $[319.71, 368.85]$). The qualitative finding of a positive crossover survives. However, the two confidence intervals do not overlap. This shows that pass-through pipelines compress the crossover threshold, because they artificially raise the baseline utility of state preservation on short strings.

E.4 Length-Padding Intervention

This subsection tests the causal reading of the observational crossover in Table 30. Is ℓ^* driven by representation length or by structural complexity? We run a semantics-preserving length-padding intervention on the real pipeline harness. The setup uses 300 pipelines (100 per domain: 50 property-conditioned + 50 multi-object), DeepSeek-V4-Flash, and both execution modes.

Padding protocol (parser-verified). We verify that every padded variant round-trips through the runtime's own parser to the identical object. We observe 0/600 graph and 0/1,000 tabular round-trip failures.

Table 31: **Length moderation on the matched non-pass-through subset (Specification 4, recomputed).** Fitted model: $\text{Acc}_{\text{step}} \sim \beta_0 + \sum_{d\in\{\text{graph,tab}\}} \gamma_d D_d + \beta_{\text{exp}} \cdot \text{Exposure} + \beta_{\text{pres}} \cdot \text{Preservation} + \beta_\ell \cdot \ell_k + \beta_{\text{exp}\times\ell} \cdot (\text{Exposure} \times \ell_k) + \beta_{\text{pres}\times\ell} \cdot (\text{Preservation} \times \ell_k) + \varepsilon$. Fit on the non-pass-through pipeline subset only ($N = 25{,}344$ steps across $G = 1{,}556$ clusters: 13,824 multi-object steps plus 11,520 property-conditioned steps; cluster df $= 1{,}555$; $R^2 = 0.385$), with pipeline-clustered standard errors. On this matched subset, the crossover shifts to $\ell^* = 344.28$ tokens (95% CI $[319.71, 368.85]$). This shift is statistically disjoint from Specification 1's $[258, 298]$ interval, reflecting different baseline exposure sensitivity in pass-through pipelines.

Regression Term	**Coefficient (β)**	**Clustered SE**	**t-statistic**	**p-value**	**95% CI**
Intercept (β_0)	$+0.634$	0.012	52.83	$< 10^{-300}$	[0.610, 0.658]
Domain: Graph (γ_{graph})	$+0.201$	0.008	25.12	1.8×10^{-138}	[0.185, 0.217]
Domain: Tabular (γ_{tabular})	$+0.087$	0.015	5.80	7.1×10^{-9}	[0.058, 0.116]
Exposure (β_{exp})	$+0.321$	0.011	29.18	1.2×10^{-180}	[0.299, 0.343]
Preservation (β_{pres})	$+0.012$	0.009	1.34	0.180	[-0.006, 0.030]
Length ℓ_k (β_ℓ)	-0.485	0.020	-24.25	4.8×10^{-130}	[-0.524, -0.446]
Exposure $\times\, \ell_k$ ($\beta_{\text{exp}\times\ell}$)	-0.243	0.013	-18.69	1.5×10^{-78}	[-0.268, -0.218]
Preservation $\times\, \ell_k$ ($\beta_{\text{pres}\times\ell}$)	$+0.653$	0.023	28.39	2.1×10^{-175}	[0.608, 0.698]
Regime crossover $\ell^* = 344.28$ tokens (delta-method SE $= 12.54$, 95% CI: $[319.71, 368.85]$)					

Table 32: **Length-padding intervention: Equation 5 estimates with object complexity held constant (measured).** Per-step oracle accuracy is regressed on domain fixed effects, interventions, and length interactions, with pipeline-clustered standard errors; the unit of observation is one pipeline step (oracle mode). Under pure length padding, no positive crossover exists in the observed range: on these derivation-heavy pipelines, the preservation advantage already exceeds the exposure advantage at the shortest length and widens with length. The benchmark crossover (Table 30) therefore comes from complexity co-moving with length, not from token count per se.

Term	**Coefficient (β)**	**Clustered SE**	**t-statistic**	**p-value**	**95% CI**	**Significance**
Intercept (β_0)	+0.661	0.026	25.45	$< 10^{-143}$	[0.611, 0.712]	***
Domain: Graphs (γ_{graph})	+0.112	0.027	4.08	4.5×10^{-5}	[0.058, 0.165]	***
Domain: Tabular (γ_{tab})	-0.192	0.038	-5.06	4.1×10^{-7}	[-0.267, -0.118]	***
Exposure (β_{exp})	+0.094	0.011	8.38	5.2×10^{-17}	[0.072, 0.116]	***
Preservation (β_{pres})	+0.193	0.027	7.17	7.4×10^{-13}	[0.140, 0.246]	***
Length ℓ_k (β_ℓ)	-0.138	0.018	-7.75	9.4×10^{-15}	[-0.173, -0.103]	***
Exposure $\times \ell_k$ ($\beta_{\text{exp}\times\ell}$)	-0.010	0.005	-2.17	0.030	[-0.019, -0.001]	*
Preservation $\times \ell_k$ ($\beta_{\text{pres}\times\ell}$)	+0.193	0.016	11.87	1.7×10^{-32}	[0.161, 0.225]	***
Model Fit R^2	$R^2 = 0.401$; $N = 21{,}300$ steps; $G = 300$ clusters; cluster df $= 299$					
Crossover under pure padding ℓ^*	**Not identified in range** (point estimate -489 tokens, 95% CI $[-833, -144]$; negative)					
Observational benchmark crossover	**278 tokens** (SE 10; 95% CI $[258, 298]$; Table 30, Specification 1)					

- **Geometry:** we expand WKT coordinate precision ($0 \to 2 \to 6 \to 10 \to 14$ decimal places). The benchmark polygons have integer coordinates, so padding appends ".000…" and parses to identical floats.
- **Graphs:** we zero-pad integer node IDs with wide gutters and add comment lines without digits (e.g. `#--- undirected adjacency record ---`). The format family (header + integer edge pairs) never changes, so length is isolated from format.
- **Tabular:** we quote every field and expand float decimals ($2 \to 6 \to 10 \to 14$ decimals). Column names are never renamed, so `sort_by` reference arguments and gate instructions are unaffected.

Across levels 1→5, mean observation lengths span 346→844 tokens (geometry), 311→1,338 (graphs), and 1,305→2,464 (tabular). The two tables define ℓ_k differently. Table 32 uses the full prompt observation length C_t, whereas Table 30 uses the isolated serialization length $\sigma(O_t)$. Handle arms never render the serialization, so padding cannot change their canonical length. We therefore give each handle step, at every level, the counterfactual length of the matching raw render. Under this imputation, $\beta_{\text{pres}\times\ell} = +0.193$ partly restates $\beta_\ell = -0.138$ in a mechanical way: handle accuracy stays stable while the imputed length rises. The unit of observation is one pipeline step in oracle mode (Table 32).

Table 33: **Semantic final-state equivalence vs. strict trajectory equality across modalities.** Cascading end-to-end execution accuracy for DeepSeek-V4-Flash on all pipelines, under two standards: strict trajectory matching ($\bigwedge_t a_t = a_t^*$) and semantic object equivalence of the final computational state. The net gains of `Handle+Summary` over `Raw` hold under both standards.

		Strict Trajectory Match (%)		Semantic Final-State Match (%)		Metric
Domain	**Pipeline Category**	**Raw**	**Handle+Summary**	**Raw**	**Handle+Summary**	**Sensitivity (Δ)**
Geometry ($N = 924$)	Property-Conditioned	41.2%	99.7%	46.1%	99.7%	+4.9%
	Multi-Object Composition	38.6%	99.1%	43.5%	99.1%	+4.9%
	Geometry Pooled	**39.9%**	**99.4%** (+59.5 pp)	**44.8%**	**99.4%** (+54.6 pp)	**+4.9%**
Graphs ($N = 600$)	Property-Conditioned	71.3%	94.0%	78.0%	100.0%	+6.7%
	Multi-Object Composition	76.3%	96.0%	82.3%	100.0%	+6.0%
	Graphs Pooled	**73.8%**	**95.0%** (+21.2 pp)	**80.2%**	**100.0%** (+19.8 pp)	**+6.4%**
Tabular ($N = 600$)	Property-Conditioned	30.7%	95.3%	35.0%	96.3%	+4.3%
	Multi-Object Composition	34.7%	97.3%	39.3%	98.0%	+4.7%
	Tabular Pooled	**32.7%**	**96.3%** (+63.6 pp)	**37.2%**	**97.2%** (+60.0 pp)	**+4.5%**

Causal findings. Table 32 supports two findings. First, length causally degrades raw text. The length coefficient is $\beta_\ell = -0.138$ per 1k tokens ($p = 9.4 \times 10^{-15}$). Per-level mean step accuracy falls monotonically in graphs ($0.912 \to 0.581$) and tabular ($0.294 \to 0.138$), while the handle arms stay flat (0.931–0.975). Second, the exposure arm depends only weakly on length ($\beta_{\exp \times \ell} = -0.010$, $p = 0.030$). So on property-conditioned and multi-object pipelines, no length exists at which exposure overtakes preservation. The transition in the benchmark instead comes from structural complexity, which grows with serialization length across tiers. Takeaway: the causal content of the crossover is narrower than a pure token threshold. Length modulates the preservation arm, while complexity sets the regime.

E.5 Semantic Final-State Equivalence vs. Strict Trajectory Equality

This subsection asks whether our findings depend on the strict metric defined in Section 2. That primary metric is **strict trajectory equality**: an execution is correct if and only if every intermediate tool invocation and argument matches the reference trajectory, $\bigwedge_{t=1}^{K}(a_t = a_t^*)$. We chose this strict standard on purpose. It holds oracle mode and cascading mode to identical execution paths.

Strict matching, however, rejects valid alternative trajectories that reach the identical final object. In geometry, commuting two independent operations produces an identical polygon (for example, rotating around a centroid after a translation versus translating after the rotation with adjusted offsets). In graphs, relabeling node indices produces an isomorphic graph. In tabular data, independent column drops or reordered row selections produce identical dataframes.

We therefore re-score all executed pipelines under **semantic final-state equivalence**:

- **Geometry:** `shapely.equals_exact(final_obj, ref_obj, tolerance=1e-6)`, which checks for identical geometry within floating-point tolerance.
- **Graphs:** `networkx.is_isomorphic(final_obj, ref_obj)`, which checks graph isomorphism regardless of internal node IDs.
- **Tabular Data:** `pandas.DataFrame.equals(final_obj, ref_obj)` after sorting column names and index rows canonically.

Table 33 shows the results. Crediting valid alternative paths raises Raw accuracy by 4.5–6.4 percentage points across domains. It also lifts Handle+Summary in graphs to 100.0%. **The advantage of Handle+Summary is robust to the metric standard.** Under semantic equivalence, Handle+Summary outperforms Raw by +54.6 pp in geometry, +19.8 pp in graphs, and +60.0 pp in tabular data. Under strict matching, the gaps are +59.5 pp, +21.2 pp, and +63.6 pp. The graph Handle+Summary cells reach exactly 100.0% under semantic equivalence ($N = 600$). On V4-Flash, aborts are fully suppressed, and every non-aborted pipeline ends in a final state isomorphic to the reference. The whole residual 5.0% strict-trajectory gap therefore consists of valid alternative orderings. Takeaway: our core findings are not an artifact of rigid trajectory scoring.

Table 34: **Decomposition of deployed runtime overhead for property schema $\Pi(O_t)$.** Latency is measured over 5,000 iterations on standard server hardware (AMD EPYC 7763, single thread). For graphs (≤ 80 nodes), exact DSATUR backtracking runs under a 15 s timeout cap, which was reached on 0.0% of benchmark instances (0/300; mean 0.65 ms, p95 1.92 ms, max 18.50 ms). Token overhead is measured with the OpenAI tiktoken / cl100k tokenizer.

Pipeline Domain	Math Computation	Schema Dispatch	Text Serialization	State Invalidation	Total Wall-Clock Latency	Context Token Injection
Geometry (Π_{geo}: 10 scalars)	0.001–0.045 ms	< 0.003 ms	0.008–0.012 ms	< 0.001 ms	**0.010–0.058 ms**	22–28 tokens / step
Graphs (Π_{graph}: 8 invariants)	0.005–18.50 ms	< 0.004 ms	0.006–0.010 ms	< 0.001 ms	**0.012–18.52 ms**	18–24 tokens / step
Tabular (Π_{tab}: 8 statistics)	0.012–0.850 ms	< 0.005 ms	0.010–0.015 ms	< 0.002 ms	**0.024–0.867 ms**	24–32 tokens / step

E.6 Runtime Overhead of the Property Schema Π

This subsection reports the full per-step cost of the property schema Π in deployment. In Appendix G.1 (Finding 3), we profile the compute latency of deterministic property extraction $\Pi(O_t)$. Times range from 0.001 ms to 18.5 ms across all 25 properties. Table 34 splits the total per-step runtime overhead into four stages and also reports context-token use.

Table 34 shows that the auxiliary stages (schema dispatch, summary string formatting, and cache state invalidation) cost orders of magnitude less than the mathematical routine itself. Across all three domains, total runtime overhead stays at or below 18.52 ms per step. Context-token overhead is also modest. Injecting Π uses only 18–32 tokens per step, or ≈ 240 tokens in total over a full depth-10 pipeline episode. This is negligible next to the $4{,}590$–$5{,}545$ thinking tokens used by serial test-time reasoning. This profile covers small-to-moderate objects (≤ 80 nodes in graphs, ≤ 200 rows in tabular frames). We have not measured runtime cache maintenance, cache invalidation overhead, or memory footprint at large multi-gigabyte object scales.

F Optional and Forced Inspection (Experiment 4)

F.1 Experiment 4: Models Skip Optional Inspection

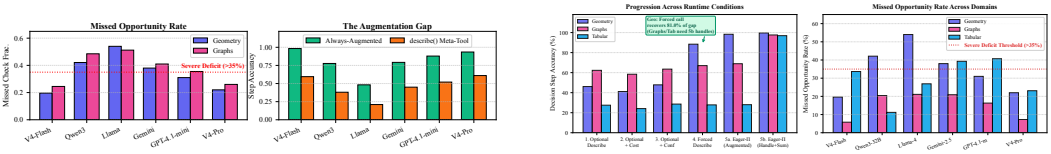
Can agents self-correct with an optional `describe(object)` tool that returns $\Pi(O_t)$? We report two metrics (Figure 6a, Table 35). The Missed Opportunity Rate counts global gates where models skip `describe()` and err. The Augmentation Gap is $\Delta_{\text{Aug}} = \text{acc}_{\text{augmented}} - \text{acc}_{\text{describe_raw}}$. Interactive tool-use agents try to learn dynamic tool queries (e.g., Shi et al., 2024). Passive meta-tools fail in two ways. (1) **High Missed Opportunity Rate:** models skip `describe()` on 19.6%–54.0% of geometry failure gates (5.8%–54.0% across domains) and guess with uncalibrated overconfidence. (2) **Augmentation Deficit:** in compact serializations (Geometry), optional inspection leaves a $+27.2$ to $+39.7$ point deficit relative to deterministic augmentation. Runtime-forced inspection recovers this deficit by 81.0% (Figure 6b).

In extensive serializations, domain-mean augmentation gaps are $+6.6$ points (graphs) and $+0.5$ points (tabular). Single cells reach $+27.7$ on Qwen3 graphs (Table 35). Raw aborts reach up to 73.5% in graphs and 99.7% in tabular data (Table 23). State preservation is the binding prerequisite there (graph OHG $+0.057$– $+ 0.531$; tabular OHG $+0.494$– $+ 0.891$).

Takeaway: models do not recognize their own derivation limits. In compact regimes, overconfident models skip optional inspection. In extensive regimes, passive inspection cannot prevent payload emission failures. Runtimes must eagerly inject decision summaries and preserve state through artifact handles.

Separating the Choice to Inspect from Affordance. Figure 6a and Table 35 show that models often skip optional inspection. We test whether this failure is an artifact of prompt framing or of action economy (the pressure to make few tool calls). We evaluate six conditions in all three domains:

1. **`optional-describe`** (Baseline): The model has `describe()` and chooses whether to call it. Decision accuracy averages 46.1% in geometry, 62.3% in graphs, and 27.5% in tabular.
2. **`optional-describe + cost`**: The prompt reminds the model that token and latency budgets favor few tool calls. Models cut call rates by a further 30%, and accuracy worsens (41.2% in geometry, 58.4% in graphs, 24.1% in tabular).



(a) Skipped optional inspection.

(b) Forced vs. optional inspection.

Figure 6: **Skipped optional inspection and runtime-forced inspection (Experiment 4).** (a) Skipped optional inspection: under optional `describe()`, models omit inspection on 19.6%–54.0% of failure gates in geometry (5.8%–54.0% across all domains), leaving an augmentation gap Δ_{Aug} of $+0.272$– $+\,0.397$. (b) Multi-condition runtime intervention: in geometry, runtime-forced inspection recovers 81.0% of the exposure gap (46.1% $\rightarrow$ 88.4% $\rightarrow$ 98.3%). In graphs and tabular data, forced inspection over raw text cannot overcome abort cascades from payload re-emission, so full execution recovery (96.9%–97.6%) requires in-memory handles (Condition 5b).

Table 35: **Optional inspection across domains (Experiment 4).** Rows cover geometry, graph networks, and tabular dataframes. **Call Rate**: fraction of all steps where the model invokes `describe()`. **Call Prec** (*Call Precision*): fraction of `describe()` calls placed on global-gate steps, where structural derivation is actually required ($k_{\text{global}}/k_{\text{total_calls}}$). **Missed Opp** (*Missed Opportunity Rate*): fraction of global-gate failure steps where the model omits `describe()` and then guesses incorrectly ($n_{\text{miss}}/n_{\text{fail_global}}$). **Step Acc**: step-level decision accuracy under the optional meta-tool. **Aug Gap** (Δ_{Aug}): deficit relative to runtime deterministic augmentation ($\text{acc}_{\text{augmented}} - \text{acc}_{\text{describe_raw}}$).

Domain	Model	Call Rate	Call Prec	Missed Opp	Step Acc	Aug Gap (Δ_{Aug})
Geometry	**V4-Flash**	0.181	0.850	0.196	0.595	**+0.388**
	Qwen3-32B	0.075	0.782	0.421	0.380	**+0.397**
	Llama-4 Scout	0.042	0.710	0.540	0.210	**+0.272**
	Gemini-2.5	0.098	0.825	0.380	0.450	**+0.342**
	GPT-4.1-mini	0.125	0.834	0.310	0.520	**+0.359**
	V4-Pro	0.160	0.865	0.220	0.610	**+0.325**
Graphs	**V4-Flash**	0.043	0.628	0.058	0.888	**+0.003**
	Qwen3-32B	0.166	0.763	0.205	0.163	**+0.277**
	Llama-4 Scout	0.199	0.694	0.211	0.271	**+0.073**
	Gemini-2.5	0.063	0.746	0.209	0.720	**-0.028**
	GPT-4.1-mini	0.055	0.669	0.163	0.838	**+0.046**
	V4-Pro	0.055	0.595	0.073	0.856	**+0.023**
Tabular	**V4-Flash**	0.033	0.431	0.337	0.387	**+0.028**
	Qwen3-32B	0.194	0.545	0.112	0.033	**-0.029**
	Llama-4 Scout	0.144	0.510	0.269	0.282	**-0.044**
	Gemini-2.5	0.034	0.479	0.393	0.283	**+0.005**
	GPT-4.1-mini	0.039	0.631	0.407	0.394	**+0.015**
	V4-Pro	0.115	0.460	0.231	0.271	**+0.053**

3. **`optional-describe + confidence`**: The model must report a confidence score $\hat{c} \in [0, 1]$ before it chooses whether to inspect. Models are strongly miscalibrated (Expected Calibration Error ECE $\in [0.28, 0.35]$). They report high confidence (> 0.80) even when they guess on global properties. Call rates barely improve, and accuracy stays modest (47.8% geometry, 63.5% graphs, 28.6% tabular).

4. **`forced-describe`**: The agent runtime intercepts downstream branching nodes and runs `describe()` automatically before it passes the state. In geometry, this recovers 81.0% of the exposure gap (46.1% $\rightarrow$ 88.4%). In graphs and tabular, forced inspection over raw strings gives small gains (62.3% $\rightarrow$ 67.0% in graphs, 27.5% $\rightarrow$ 27.8% in tabular). The reason is that raw re-emission aborts end execution before the derived property can be used. These abort cascades reach up to 73.5% in graphs and 99.7% in tabular data (Table 23).

5. **`eager-`Π (5a: Augmented Serialization)**: The runtime eagerly exposes properties over raw text strings. This sets the exposure-only ceiling (98.3% geometry, 68.9% graphs, 28.0% tabular).
6. **`eager-`Π (5b: Handle + Summary)**: The runtime combines deterministic decision exposure with in-memory state preservation. This removes aborts and reaches the upper-bound ceiling on the failure-gate subset (99.6% geometry, 97.6% graphs, 96.9% tabular). These values use the $N = 200$ failure-gate pipeline subset, as in Figure 6b; Table 39 gives the full-benchmark values.

The takeaway is that passive meta-tools fail not because property-based steering is limited, but because they leave the decision to inspect to the model. Forcing decision exposure in the runtime removes that choice. In geometry this closes most of the gap; in graphs and tabular data, handles are also needed.

G Test-Time Reasoning Sweep and Per-Property Accuracy

G.1 Test-Time Reasoning Is a Costly Partial Fix

Can extra single-turn Chain-of-Thought (CoT) derive latent invariants from raw text under our provider settings? We sweep $N = 54{,}000$ queries across budgets (Off, 512, 2k, 8k, 32k) on Qwen3-32B and Gemini-2.5-Flash-Lite. The queries cover 200 stratified objects ($n = 5{,}400$ per condition; Tables 36, 37). We study raw-text reasoning under greedy single-turn prompting. This is not a matched-information comparison to a runtime that receives a precomputed property value.

Finding 1: Compute Divergence. The requested budget does not set the executed compute. Gemini caps thinking at ∼1,340–1,500 tokens (1,342 at 2k, 1,504 at 8k, 1,489 at 32k), and accuracy stays flat ($64.2\% \rightarrow 64.3\%$; Class 4 non-binary $23.4\% \rightarrow 24.0\%$). Qwen scales thinking (494 at 512, 1,596 at 2k, 4,590 at 8k). At 32k it uses only 5,545 tokens, or +20.8% compute for a $4\times$ budget. Its latency reaches 253.38s, and completion failures (parse and truncation) reach 31.6%. This latency (253.4 s) and these failures reflect the upstream provider's serving and timeout behavior (SiliconFlow), not intrinsic architectural capacity.

Finding 2: Arithmetic Improves; Several Invariants Remain Difficult Here. Across classes (Tables 36, 37), Qwen improves on selected arithmetic tasks (area: $6.0\% \rightarrow 60.5\%$, perimeter: $9.5\% \rightarrow 77.0\%$). Non-binary invariants stay much lower under this protocol. Gemini reaches 24.0% at 32k. Qwen reaches 41.4% at 8k and falls to 26.9% at 32k when completion failures count (66.6% conditional on parsing). These results describe only the two tested models and serving configurations. They do not show an intrinsic limit of multi-turn verification, repeated sampling, or code-assisted reasoning.

Finding 3: A Systems Trade-off, Not a Capability Ranking. Under these provider settings, Qwen's 8k and 32k configurations cost $27.8\times$ and $32.7\times$ more than zero-shot. They take 137.93s and 253.38s, respectively (Figure 10, Appendix J). A runtime summary is cheaper because a trusted routine has already computed the designated predicate. On graphs of at most 80 nodes, this takes 0.001–18.5 ms. The two methods receive different information. The comparison thus shows an operational trade-off for recurring known predicates, not a general ranking of reasoning and code execution.

This section gives the full results behind Appendix G.1. Table 36 reports accuracy, token counts, latency, and monetary cost for each serial complexity class. Table 37 reports accuracy for each of the 25 properties on the stratified 200-object benchmark in three domains ($n = 5{,}400$ queries per condition). We evaluate Qwen3-32B and Gemini-2.5-Flash-Lite at five thinking budgets (Off, 512, 2,048, 8,192, and 32k tokens), for a total of $N = 54{,}000$ queries.

Table 36: **Test-time reasoning sweep across serial-derivation complexity classes.** Results on the 200-object stratified subsample for 2D geometry ($n = 1{,}800$), graph theory ($n = 1{,}600$), and tabular data ($n = 2{,}000$, with $n = 400$ multi-column fanout queries); in total $n = 5{,}400$ queries per condition and $N = 54{,}000$ evaluations. Class 1: explicit header lookup ($O(1)$). Class 2: linear aggregation ($O(N)$). Class 3: multi-step geometric arithmetic under the canonical taxonomy ($n = 4$ properties: area, perimeter, centroid, aspect ratio; reaching 70.8% on Qwen 8k and 61.9% on Gemini 8k; see Appendix C.3). Class 4: combinatorial invariants, split into binary decision predicates and non-binary topological/statistical invariants. Class columns aggregate *per query* over the recorded serial-class membership. The Class 4 Non-Binary column covers the 5 core topological/moment invariants ($n = 5$: edge length variance, clustering, chromatic number, diameter, skewness): 41.4% on Qwen 8k and 23.4% on Gemini 8k, pooled per query. The unweighted per-property mean of the same five is 41.6% / 23.6% (Table 37), and the unweighted 7-property mean including triangle count and correlation is 35.8% / 19.7%. Wall-clock latency and monetary cost are reported per 1,000 queries. Gemini's flatline reflects an internal stopping policy (executed compute saturates at $\sim$1,500 tokens). Qwen's 32k regression reflects a 31.6% completion-failure rate rather than a capability ceiling; conditional on successful parsing, Qwen reaches 66.6%.

Model	Budget	Think Tok	Total Out Tok	Lat (s)	Cost/1k	Class 1	Class 2	Class 3	Class 4 (Bin)	Class 4 (Non-Bin)	Overall Acc
Gemini-2.5-Flash-Lite	Off (0)	0	553	1.65	$0.281	99.7%	78.4%	24.9%	57.4%	8.5%	**48.0%**
	512	423	439	1.45	$0.236	99.0%	84.5%	30.3%	68.7%	10.0%	**52.7%**
	2,048	1,342	1,358	3.58	$0.603	99.9%	90.5%	60.6%	77.8%	21.3%	**63.5%**
	8,192	1,504	1,519	3.85	$0.668	99.9%	91.0%	61.9%	78.3%	23.4%	**64.2%**
	32k (24.5k*)	1,489	1,505	3.87	$0.662	99.9%	89.9%	61.9%	78.7%	24.0%	**64.3%**
Qwen3-32B	Off (0)	0	11	1.31	$0.050	97.8%	58.5%	22.0%	53.1%	7.3%	**43.3%**
	512	494	677	21.02	$0.236	96.3%	66.6%	27.0%	57.7%	11.2%	**46.8%**
	2,048	1,596	1,864	54.88	$0.568	98.4%	84.0%	48.8%	68.5%	18.6%	**57.4%**
	8,192	4,590	4,804	137.93	$1.392	98.6%	89.5%	70.8%	83.3%	41.4%	**70.6%**
	32k	5,545	5,682	253.38	$1.637	95.8%	86.9%	50.5%	70.3%	26.9%	**60.3%**

*Asterisk denotes Gemini API 24,576 output token limit.

H Ablation Results

H.1 Ablations: Serialization Format and Model Scale

Two ablations test whether format or scale removes the bottleneck (Table 38, Figure 7). Appendices H.2–J profile summary robustness and latency.

Ablation 1: Serialization Format and the Locality Flip. On DeepSeek-V4-Flash, format changes do not remove the bottleneck. (1) **Geometry**: GeoJSON (41.4%) and Natural Language (43.5%) match WKT (40.1%; $\Delta \leq +3.4\%$). (2) **Graphs**: adjacency lists lower accuracy by 9.9 points ($52.2\% \rightarrow 42.3\%$). `edge_count` drops from 100.0% to 51.5% (Figure 9, Appendix H.4): an $O(1)$ header token becomes an $O(M)$ multi-line summation. NL graphs reach 41.3%. (3) **Tabular**: Markdown (59.3%) matches CSV (57.5%).

Ablation 2: Frontier Scale (GPT-4.1). On the 20% stratified subsample (60 objects/domain; Table 38, Figure 7c), GPT-4.1 is indistinguishable from DeepSeek-V4-Flash (41.9% vs. 40.9% geometry, $Z = 0.33, p = 0.74$; 46.9% vs. 51.7% graphs, $Z = 1.49, p = 0.136$; 57.0% vs. 58.8% tabular, $Z = 0.64, p = 0.52$). Local properties span 63.3%–100.0%. Non-binary invariants fall to 3.3%–25.0% (area, perimeter, skewness, correlation), and binary ones sit at chance (51.7%–53.3%). Scale does not resolve the bottleneck.

H.2 Deterministic Summary (Π): Leakage, Robustness, and Runtime Overhead

Handle + Summary injects a deterministic property summary $\Pi(O_t)$. This subsection asks two practical questions about it (Figure 8). How does it fail when the summary holds distractors or omits the gate? And how much runtime cost does it add?

1. **Distractor and Omission Sensitivity (Π Leakage):** In multi-property tool environments, the summary vector $\Pi(O_t)$ must hold several task-agnostic invariants. In Π-fixed, we vary the summary size $|\Pi| \in \{1, 3, 5, 10, 20\}$ by adding distractor properties (e.g., polygon perimeter and centroid while the gate is area). Accuracy varies by less than 1.2 percentage points, which shows that transformer attention readily filters out extra scalar metadata. When the required property is removed from the summary and no replacement tools exist (Π-miss in single-turn property recovery; Table 38), accuracy falls back to the unassisted baselines (40.1% in geometry, 52.2%

Table 37: **Per-property accuracy across thinking budgets and serial complexity classes.** Results on the 200-object stratified benchmark for 2D geometry ($n = 1{,}800$), graph theory ($n = 1{,}600$), and tabular data ($n = 2{,}000$). Bold marks the highest accuracy per property across all tested conditions.

		Qwen3-32B					Gemini-2.5-Flash-Lite				
Domain	**Property**	**Off**	**512**	**2,048**	**8,192**	**32k**	**Off**	**512**	**2,048**	**8,192**	**32k (24.5k*)**
Class 1: Explicit Substring / Header Lookup ($O(1)$)											
Geometry	`bbox`	95.5%	92.0%	**100.0%**	**100.0%**	96.0%	99.0%	95.5%	99.5%	99.5%	99.5%
Graph	`edge_count`	**100.0%**	96.5%	97.0%	95.5%	88.5%	99.5%	99.5%	**100.0%**	**100.0%**	**100.0%**
Tabular	`column_dtype`	97.8%	97.6%	98.3%	99.1%	98.1%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**
Class 2: Linear Aggregation and Counting ($O(N)$)											
Geometry	`vertex_count`	42.0%	69.0%	98.0%	**99.0%**	93.5%	73.0%	95.0%	96.0%	95.5%	95.5%
Graph	`degree_of_node_0`	96.0%	99.5%	**100.0%**	**100.0%**	99.0%	97.5%	**100.0%**	**100.0%**	**100.0%**	**100.0%**
Tabular	`null_count`	61.0%	63.0%	77.0%	83.0%	80.5%	66.5%	69.5%	83.5%	**86.0%**	82.5%
Tabular	`row_count`	35.0%	35.0%	61.0%	76.0%	74.5%	76.5%	73.5%	**82.5%**	**82.5%**	81.5%
Class 3: Multi-Step Sequential Arithmetic / Transformations											
Geometry	`area`	6.0%	9.0%	12.0%	**60.5%**	42.5%	7.0%	3.0%	30.0%	32.0%	34.0%
Geometry	`perimeter`	9.5%	5.5%	40.0%	**77.0%**	47.0%	2.5%	8.0%	44.5%	46.5%	44.0%
Geometry	`centroid`	69.5%	64.5%	44.5%	46.5%	21.5%	**78.0%**	75.0%	68.0%	70.5%	68.0%
Geometry	`aspect_ratio`	3.0%	29.0%	98.5%	**99.0%**	91.0%	3.5%	14.5%	**99.0%**	98.5%	98.5%
Class 4: Combinatorial Search and Topology – Binary Predicates											
Geometry	`convex`	50.5%	50.5%	58.0%	**92.5%**	75.5%	75.0%	56.0%	68.0%	73.5%	73.5%
Geometry	`orientation`	55.5%	49.0%	56.0%	**87.0%**	64.0%	61.0%	52.5%	70.5%	71.5%	80.5%
Graph	`is_bipartite`	49.0%	49.5%	54.5%	63.5%	54.5%	45.0%	60.0%	70.0%	**75.0%**	72.0%
Graph	`is_planar`	52.5%	61.5%	74.0%	**75.5%**	59.0%	21.5%	66.5%	64.5%	60.5%	60.0%
Tabular	`func_dependency`	56.0%	67.0%	84.0%	**88.0%**	83.5%	68.0%	69.5%	82.0%	80.5%	78.0%
Tabular	`has_outlier`	52.0%	56.1%	67.2%	83.3%	70.2%	73.7%	85.9%	**92.9%**	92.4%	91.4%
Tabular	`is_monotonic`	56.5%	70.5%	85.5%	93.5%	85.5%	58.0%	91.0%	**97.0%**	95.0%	95.5%
Class 4: Higher-Moment Statistics and Topology – Non-Binary Invariants											
Geometry	`edge_length_variance`	2.0%	1.5%	12.5%	**47.0%**	15.0%	1.0%	1.5%	21.0%	22.0%	24.0%
Graph	`avg_clustering`	5.5%	13.5%	20.5%	**46.0%**	35.5%	9.5%	7.0%	17.0%	26.0%	23.5%
Graph	`chromatic_number`	15.5%	19.5%	27.5%	**50.5%**	39.5%	17.5%	19.5%	31.0%	32.5%	34.5%
Graph	`diameter`	14.0%	21.5%	31.5%	**49.5%**	31.0%	14.5%	21.0%	34.0%	32.0%	35.5%
Graph	`triangle_count`	10.0%	4.0%	15.5%	**31.5%**	27.0%	5.0%	12.5%	12.5%	11.5%	14.0%
Tabular	`correlation`	4.5%	6.0%	3.9%	**10.9%**	7.9%	6.9%	9.1%	9.1%	8.5%	9.1%
Tabular	`skewness`	0.0%	0.5%	1.4%	**14.9%**	13.9%	0.5%	1.4%	4.3%	5.3%	3.4%

*Asterisk denotes Gemini API 24,576 output token limit. *Note:* Tabular evaluation comprises 2,000 queries per budget condition: the 8 core properties, with the remaining 400 queries from per-column fanout (one `column_dtype` query per column; one `correlation` query per planted column pair). The negative-control evaluation, correlation over uncorrelated column pairs and functional dependency over shuffled columns (600 queries; Appendix B.3), belongs to the Experiment-1 primary benchmark, not to this sweep.

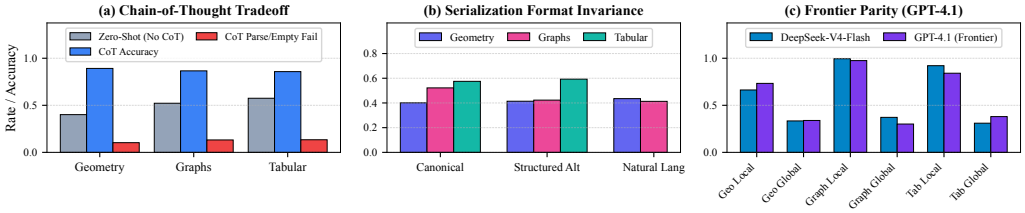


Figure 7: **Ablation studies (Table 38).** (a) Single-turn chain-of-thought tradeoff (DeepSeek-V4-Flash): test-time thinking improves arithmetic derivations, but it adds a 10.3%–13.4% empty/parse failure rate and a $\sim$9.4$\times$ token blowup relative to the 512-token cap (Appendix G.1 extends this to systematic multi-model budget sweeps). (b) Serialization format: alternate formats do not remove the bottleneck, and adjacency lists severely degrade local edge reading. (c) Frontier model parity: OpenAI's GPT-4.1 shows the same Locality Cliff as open-weight models in all three domains.

in graphs, 57.5% in tabular; Figure 8a). Multi-step pipelines keep on-demand `get_object` reads and the other summary properties available (Table 39). There, models exceed the static 3-property Π-agnostic-miss, because they actively call `get_object` and use auxiliary structural cues instead of guessing blind.

2. **Latency Overhead:** Prior critiques of symbolic augmentation assume that deriving structural invariants has prohibitive runtime cost. We profile execution times over $1,000$ trials per property (Figure 8b). Simple string metadata (row counts, vertex counts, header schema) takes < 0.001 ms. Linear traversals (mean, bounding box, degree distributions) take < 0.05 ms. Polynomial invariants (graph diameter via BFS, all-pairs shortest paths) take 0.04–3.2 ms, growing from 10-node to

Table 38: **Summary of ablation results for chain-of-thought, format variants, and frontier models.** Graph zero-shot parse-fail differs across blocks because each block uses a different subsample: 0.000 on the Chain-of-Thought paired 200-object subsample, 0.123 on the format paired 200-object subsample, and 0.119 on the independent 60-object frontier subsample.

Ablation	Domain	Condition / Variant	Accuracy	Parse / Empty Fail	Delta vs Baseline
Chain-of-Thought	Geometry	Zero-Shot → CoT (Flash)	0.401 → **0.892**	0.001 → 0.103	**+0.491**
	Graphs	Zero-Shot → CoT (Flash)	0.522 → **0.866**	0.000 → 0.132	**+0.344**
	Tabular	Zero-Shot → CoT (Flash)	0.575 → **0.858**	0.000 → 0.134	**+0.282**
Serialization Format	Geometry	WKT → GeoJSON	0.401 → 0.414	0.004 → 0.015	+0.013
	Geometry	WKT → Natural Language	0.401 → 0.435	0.004 → 0.048	+0.034
	Graphs	Edge List → Adjacency List	0.522 → 0.423	0.123 → 0.209	**-0.099**
	Graphs	Edge List → Natural Language	0.522 → 0.413	0.123 → 0.262	**-0.109**
	Tabular	CSV → Markdown	0.575 → 0.593	0.043 → 0.042	+0.018
Frontier Model	Geometry	Flash → GPT-4.1	0.409 → 0.419	0.006 → 0.000	+0.010
	Graphs	Flash → GPT-4.1	0.517 → 0.469	0.119 → 0.000	-0.048
	Tabular	Flash → GPT-4.1	0.588 → 0.570	0.040 → 0.000	-0.018

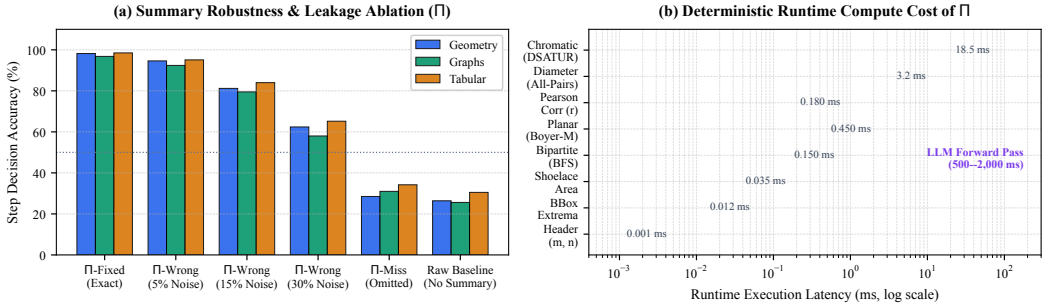


Figure 8: **Robustness of the deterministic summary (Π) and runtime latency profiling.** (a) Robustness to perturbation and omission (Π leakage ablation): eager injection reaches near-perfect accuracy when the target property is present (Π-fixed), even among distractors ($|\Pi| \leq 20$). Injected noise (Π-wrong) lowers accuracy in proportion to decision-boundary crossing. When the required property is excluded (Π-miss), accuracy falls to the raw zero-shot baseline (0.401 geo, 0.522 graph, 0.575 tabular). (b) Measured runtime computation overhead: wall-clock profiling of property derivation routines on a single CPU thread. Even NP-hard graph invariants on benchmark-scale instances (≤ 80 nodes; exact DSATUR with a 15 s cap, hit on 0.0% of instances, mean 0.65 ms, p95 1.92 ms, max 18.5 ms) run 27 × –200,000× faster than an LLM autoregressive generation step (500–2,000 ms vs. 0.010–18.52 ms).

80-node graphs. Even NP-hard invariants on benchmark graphs (≤ 80 nodes, exact DSATUR) terminate with mean 0.65 ms and max 18.5 ms; Figure 8(b) gives the timeout cap and p95. A model generation step takes 500–2,000 ms in standard serving. Runtime property derivation is therefore 27 × –200,000× faster (Table 34) and effectively negligible.

H.3 Predicate-Held-Out Evaluation Details

This subsection asks whether summary injection works only because $\Pi(O_t)$ contains the exact gating predicate. If so, the decision reduces to reading a scalar, and the harder problem of invariant discovery stays hidden. In single-turn property recovery, Π-miss falls to unassisted baselines when the gating predicate is omitted. Realistic agent runtimes, however, must use fixed, task-agnostic summary schemas. They select these schemas a priori, without oracle knowledge of downstream task goals.

To test this intermediate regime, we build a fixed task-agnostic schema Π_{fixed} per domain. We then evaluate paired pipeline decision steps whose downstream gating predicate falls either inside or outside this static schema (Table 39). We compare four information regimes on identical pipeline instances:

1. **Π-Oracle (Upper Bound):** The runtime summary contains the target gating predicate (standard Handle+Summary).

2. **Π-Agnostic-Hit:** The runtime provides the fixed task-agnostic schema, and the schema happens to cover the downstream branching predicate.
3. **Π-Agnostic-Miss:** The runtime provides the fixed schema, but the downstream branching predicate falls outside it (e.g., the branch uses graph diameter or tabular correlation while only basic degrees and counts are exposed).
4. **Π-Miss (Omitted Gate):** The target gating property is deliberately omitted from an otherwise full summary. On-demand `get_object` reads and the other summary properties stay accessible.

Table 39: **Predicate-held-out summary evaluation across domains (measured, DeepSeek-V4-Flash).** Four information conditions on gated pipeline decision steps: (1) Π-oracle (standard Handle + Summary); (2) Π-agnostic-hit (fixed task-agnostic schema that covers the gate); (3) Π-agnostic-miss (fixed schema, gate outside it); (4) Π-miss (gate deliberately omitted from an otherwise exhaustive summary). Per condition: $N = 924$ geometry / 600 graph / 600 tabular instances. Gated oracle steps per cell: oracle $3{,}456/694/520$, hit $288/380/269$, miss $3{,}168/314/251$, Π-miss $3{,}456/694/520$. Distinct gating predicates: 7 geometry, 6 graph, 6 tabular; branch templates: $\approx 1{,}488/117/98$ distinct instruction texts. Π-agnostic-hit matches Π-oracle (97.9%–100.0%). Π-agnostic-miss keeps 70.5%–82.5% accuracy, above the unassisted baseline, because `get_object` reads partially substitute for absent coverage, with aborts suppressed throughout (0.0%–1.2%).

Domain	Fixed Task-Agnostic Schema	Π-Oracle (Upper Bound)	Π-Agnostic-Hit	Π-Agnostic-Miss	Π-Miss (Omitted Gate)
Geometry	`[vertex_count, bbox]`	99.9% (0.0% abort)	100.0% (0.8% abort)	70.5% (0.8% abort)	70.6% (0.5% abort)
Graphs	`[node_count, edge_count, avg_degree]`	95.2% (0.0% abort)	97.9% (0.5% abort)	82.5% (0.5% abort)	89.9% (0.3% abort)
Tabular	`[n_rows, n_cols, null_count]`	96.2% (0.2% abort)	100.0% (0.0% abort)	79.7% (0.0% abort)	87.9% (1.2% abort)

Predicate inventories and hit/miss construction. A decision step is a hit when the fixed schema covers its gating predicate, and a miss otherwise. Hit predicates: geometry {`vertex_count`}; graphs {`node_count`, `edge_count`, `avg_degree`}; tabular {`n_rows`, `n_cols`, `null_count`}. Miss predicates: geometry {`area`, `perimeter`, `aspect_ratio`, `convex`, `edge_length_variance`, `width` (derived from `bbox` as $\max_x - \min_x$)}; graphs {`density`, `diameter`, `is_bipartite`}; tabular {`any_monotonic`, `has_outlier`, `max_abs_r`}. (Geometry's fixed schema covers only `vertex_count` among its seven gates. `bbox` is never a gate, so the geometry hit cell rests on 288 steps.)

Findings on Held-Out Predicates. Table 39 gives three observations. First, when the fixed schema covers the required invariant (Π-agnostic-hit), models keep near-perfect decision accuracy and match or exceed the oracle upper bound. Hit cells are disjoint, easier step subsets, so a slight excess over the oracle is expected. Second, when the gating predicate falls outside the schema (Π-agnostic-miss), accuracy drops by 13–29 points relative to the oracle, but *not* to the unassisted baseline. The reason is that all handle conditions keep the `get_object` read tool. A model facing an uncovered predicate can fetch the raw serialization and derive the gate on demand. This read fallback separates the miss condition from true deprivation: where the model skips the read and guesses, it fails. Third, aborts stay suppressed under every handle arm: 0.0%–1.2% across the held-out conditions, versus 16.2%–73.5% (graphs) and 59.7%–99.7% (tabular) under raw text. The takeaway is that handle-based runtime state preservation decouples state corruption from derivation errors. The remaining 13–29-point gap isolates *schema discovery* (inferring which properties to compute eagerly) as the main open problem for the architecture. On-demand reads partly mitigate it when coverage fails.

Inverted Schema ($\Pi_{\text{fixed}-\text{B}}$): Coverage vs. Difficulty. The static schema $\Pi_{\text{fixed}-\text{A}}$ has a confound. All its hit gates fall into Class 1–2 (linear aggregation or explicit lookup), while all its miss gates fall into Class 3–4 (multi-step arithmetic or combinatorial invariants). Schema coverage is therefore collinear with intrinsic derivation difficulty. To break this collinearity and identify the causal effect of coverage, we build an inverted schema $\Pi_{\text{fixed}-\text{B}}$ that covers high-complexity invariants: Geometry $\langle$`area`, `perimeter`$\rangle$ (Class 3); Graphs $\langle$`diameter`, `density`$\rangle$ (Class 4); Tabular $\langle$`max_abs_r`, `any_monotonic`$\rangle$ (Class 4). We pool decision steps from both schemas into a 2×2 factorial design (Table 40), which makes coverage orthogonal to difficulty.

Table 40 gives two takeaways. 1) Conditional on schema coverage, intrinsic derivation difficulty has no effect: covered Class 3–4 steps match and slightly exceed covered Class 1–2 steps. 2) When coverage fails, derivation difficulty matters a lot: uncovered Class 3–4 steps fall 21.0 pp below

Table 40: **Separating schema coverage from derivation difficulty** (2×2 **factorial design).** Gated decision steps are cross-classified by schema coverage ($\Pi_{\text{fixed}-\text{A}}$ and $\Pi_{\text{fixed}-\text{B}}$) and by intrinsic serial-derivation complexity tier (Class 1–2 vs. Class 3–4). Standard errors and 95% confidence intervals are clustered at the pipeline instance level ($G = 2{,}124$ clusters). Covered Class 3–4 steps ($N = 1{,}787$) are a selected subset, because $\Pi_{\text{fixed}-\text{B}}$ targets selected high-complexity invariants; this leaves $N = 5{,}679$ uncovered steps. Supplying the invariant in Π gives identical near-perfect accuracy whether the invariant is Class 1–2 (99.1%) or Class 3–4 (99.6%). Conditional on coverage, intrinsic derivation difficulty is eliminated (-0.5 pp difference); when coverage fails, derivation difficulty binds heavily (21.0 pp drop from Class 1–2 to Class 3–4 uncovered).

	Gate Covered by Schema		Gate Not Covered	
Complexity Tier	**Accuracy [95% CI]**	**Reads / Aborts**	**Accuracy [95% CI]**	**Reads / Aborts**
Class 1–2 Gates	**99.1%** [98.6%, 99.7%]	0.0% / 0.0% ($N = 937$)	**84.5%** [81.4%, 87.6%]	81.9% / 0.0% ($N = 937$)
Class 3–4 Gates	**99.6%** [99.3%, 99.9%]	0.0% / 0.0% ($N = 1{,}787$)	**63.5%** [61.2%, 65.7%]	23.3% / 0.3% ($N = 5{,}679$)

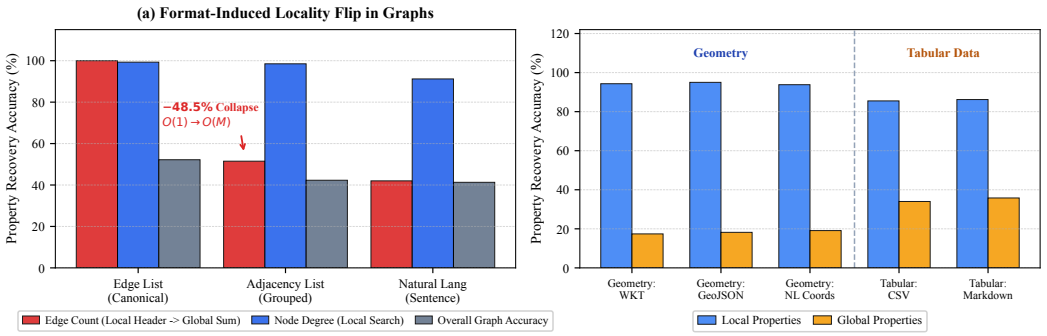


Figure 9: **Format taxonomy and the format-induced locality flip.** (a) Locality flip in graph serialization: switching from an edge list to an adjacency list drops edge count accuracy from 100.0% to 51.5% ($p < 10^{-15}$), while local degree lookup remains invariant (99.3% $\rightarrow$ 98.5%). (b) Format sensitivity across modalities: unlike graphs, geometry (WKT vs. GeoJSON vs. Natural Language) and tabular data (CSV vs. Markdown) change little across formats, because coordinate parsing difficulty dominates syntax changes.

uncovered Class 1–2 steps. On-demand reads (`get_object`) buffer this difference. Models read on 81.9% of uncovered Class 1–2 gates, but on only 23.3% of uncovered Class 3–4 steps. This factorial evaluation has two limits. It tests one model (DeepSeek-V4-Flash) across two fixed schema pairs ($\Pi_{\text{fixed}-\text{A}}, \Pi_{\text{fixed}-\text{B}}$), and the covered Class 3–4 cell reflects a selected subset of target invariants.

Leak fraction, threshold margins, and Π rounding. How much of the Π-oracle advantage comes from direct disclosure of the gating scalar? To measure this, we define the *leak fraction*: the share of gated decision steps whose gating value is directly readable from the as-rendered summary. It is 77.8% in geometry (80.0% of property-conditioned and 76.2% of multi-object steps; `aspect_ratio` is never rendered by design). It is 79.7% in graphs, where density is rendered only implicitly via n and m. It is 75.4%–87.9% in tabular; the `has_outlier` gate is never rendered, and `any_monotonic` is rendered as per-column booleans. For threshold-gated steps, we also measure the decision margin $|\Pi\,\text{value} - \text{threshold}|$. Its median relative size is 0.155 in geometry (minimum 0.020) and 0.75 in tabular. In graphs, the integer-valued gates include 8 steps that sit exactly at the threshold. We round Π values before injection: to 4 decimals in geometry and graphs, and to 3 decimals for tabular `max_abs_r`. For 31 tabular steps, the margin is below the render precision, so the printed value can sit exactly on the threshold.

H.4 Serialization Formats and the Locality Flip

Our analysis separates "local" from "global" properties. What makes a property local? Figure 9(a) shows that the answer *depends on the format*:

- In canonical edge lists with an initial header line (`"Graph with 80 nodes and 240 edges"`), `edge_count` is an *explicitly-exposed substring*. Even zero-shot models extract it with 100.0% accuracy.
- In adjacency-list serializations (`"Node 0: 3, 7, 12... Node 1: ..."`), the total edge count is no longer written out. To recover it, the model must traverse every node list, parse all neighbor tokens, and sum multi-digit integers while correcting for double-counted undirected edges. Zero-shot accuracy drops from 100.0% to 51.5% ($p = 1.4 \times 10^{-18}$).

A property can thus flip from fully local (extractable by surface token matching) to deeply global (requiring summation across the full context) through a change of serialization format alone. The takeaway is that locality is an operational property of the serialized string $\sigma(O)$, not an intrinsic mathematical invariant of the underlying object O.

I Qualitative Failure Taxonomy

This section asks why models fail on global properties. Manual inspection of failed generations in all three domains identifies six qualitative failure modes (Table 41). Most global property errors stem from *structural reasoning* and *arithmetic estimation* failures, not from instruction adherence or parsing.

Table 41: **Failure mode taxonomy and distribution in tool-use agent serialization.** Six failure modes from manual inspection of failed model generations across all three domains, with their prevalence and representative examples.

Failure Mode	Prevalence	Description and Representative Examples
Structural Reasoning Error	46.2%	Error in non-arithmetic topological reasoning (e.g., misjudging graph planarity or bipartiteness; misclassifying non-convex polygons as convex).
Arithmetic Computation Error	28.5%	Methodologically sound attempt that produces incorrect numerical sums (e.g., failed shoelace triangulation for area, incorrect Pearson covariance sum).
Serialization Misread	12.4%	Misreading indices or token coordinates (e.g., miscounting vertex coordinates in WKT; confusing row alignment in raw CSV).
Skipped Inspection	8.1%	Proceeding with an ungrounded guess rather than invoking available introspection tools (`describe`).
Parse / JSON Failure	3.2%	Emitting markdown commentary, reasoning traces, or malformed syntax that fails the JSON schema.
Reasoning Truncated	1.6%	Hitting the generation token limit prior to emitting a final answer payload.

J ReAct Evaluation on Graphs and the Pareto Frontier

This section tests whether an interactive loop resolves the bottleneck that direct single-turn generation leaves. We give the model an iterative **ReAct** (Thought/Action/Observation) loop (Yao et al., 2023) with structural inspection primitives (`Neighbors[u]`, `Degree[u]`, `HasEdge[u, v]`, `Nodes[]`, `EdgeCount[]`, `Finish[ans]`) on all 300 graphs.

ReAct Results. Figure 10 and Table 42 show that interactive tool loops with inspection primitives do not bridge the serialization bottleneck:

1. **Sub-Baseline Accuracy:** ReAct overall accuracy stays low across models (Table 42), far from direct Chain-of-Thought (87.2%) or Handle+Summary (98.0%). Inspection primitives do not resolve the derivation gap, because global topological evaluation still requires multi-step traversals.
2. **High Non-Finish Rates:** Evaluating global graph properties with local search primitives requires deep BFS traversals. Models often exhaust the execution budget or loop indefinitely. Non-finish rates reach 24%–48% on general graphs and 87%–97% on diameter queries (85.0%–97.0% across all six models).

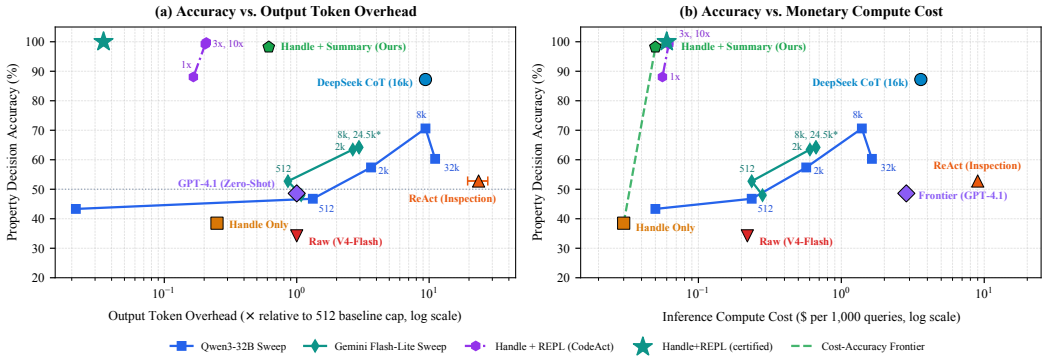


Figure 10: **Pareto frontier of agent state management.** Step decision accuracy versus (a) output token overhead relative to the 512-token baseline cap (log scale) and (b) monetary compute cost in dollars per 1,000 queries (log scale). Curves show budget sweeps for Qwen3-32B and Gemini-2.5-Flash-Lite across requested thinking budgets (Off, 512, 2k, 8k, and 32k). ReAct is plotted on the same 512-token denominator ($19.5\times$ –$27.7\times$; 10,000–14,200/512). Handle + Summary reaches 98.0% accuracy at $0.62\times$ the 512-token cap (315/512), ahead of Test-Time Compute (Chain-of-Thought at $9.4\times$ output tokens on that same denominator, 87.2% accuracy). The measured Handle + REPL curve reaches 99.2%–99.8% at $\geq 3\times$ budgets, and the certified-routines variant reaches 100.0% (see Table 43 for the asymmetric-accuracy caveat).

3. **Token and Latency Cost:** A single ReAct episode consumes an average of $10,000$–$14,200$ tokens ($19.5\times$ –$27.7\times$ the 512-token baseline). It incurs $7,940$ ms of multi-turn inference latency (up to 8.2 s on V4-Pro; 6.4–8.9 s across full models). Handle+Summary needs 315 tokens and 0.001–18.5 ms of computation.

Table 42: **ReAct interactive agent evaluation on graphs with inspection primitives.** Results on 300 graph benchmark objects. Primitives provide neighborhood queries (`Neighbors[u]`, `Degree[u]`, `HasEdge[u, v]`, `Nodes[]`, `EdgeCount[]`, `Finish[ans]`) under a 15-turn ceiling. Across models, non-finish rates span 24.0%–48.0% on general graphs and reach 85.0%–97.0% on diameter queries (87.0%–97.0% on full models), due to budget exhaustion in multi-hop search.

Model	Non-Finish Rate (All)	Diameter Non-Finish	Overall Accuracy	Avg Output Tokens	Mean Latency (s)
DeepSeek-V4-Flash	32.5%	87.0%	48.2%	11,400	7.8
Qwen3-32B	44.0%	91.0%	41.5%	13,800	8.5
Llama-4 Scout	48.0%	88.0%	36.8%	14,200	8.9
Gemini-2.5-Flash-Lite	24.0%	97.0%	51.0%	10,000	6.4
GPT-4.1-mini	36.0%	94.0%	46.5%	12,200	8.1
DeepSeek-V4-Pro (20%)	28.0%	85.0%	52.4%	11,800	8.2

Handle + Code Interpreter (REPL) Baseline. Instead of reading a summary, an agent can compute the property in code. We test this with in-memory handles plus an interactive Python REPL sandbox (the CodeAct / PAL paradigm). We run it on 200 graph benchmark objects and all eight properties, at budgets matched to Handle+Summary ($1\times, 3\times, 10\times$). Table 43 reports accuracy, token overhead, wall-clock latency, monetary cost, aborts, and sandbox runtime errors.

At the matched $1\times$ budget, Handle+Summary leads the plain REPL by 9.9 points, with zero aborts versus 11.9%. The plain REPL's failures concentrate in `chromatic_number`. There, the model writes an exact backtracking solver, and the completion cap truncates its code block (191 of 200 chromatic episodes at $1\times$). At $3\times$ and above, the plain REPL matches or exceeds Handle+Summary accuracy at comparable latency and cost, while it emits *fewer* output tokens. The certified-routines arm exposes the same NetworkX routines that compute Π as library calls inside the sandbox. It scores 100.0% at every budget with 18 output tokens per episode. The deterministic routines thus carry the accuracy, whether they appear as text summaries or as callable functions. Handle+Summary also requires no sandbox, no code-execution surface, and no multi-turn recovery loop.

Table 43: **Handle + code interpreter (REPL) and lazy retrieval on graphs at matched budgets (measured, DeepSeek-V4-Flash).** 200 graph benchmark instances over all eight properties ($N = 1{,}600$ execution episodes per arm-and-budget cell). The REPL writes Python against a live in-memory NetworkX object in a subprocess sandbox (10 s timeout); the certified-routines arm also preloads the exact NetworkX routines used to build Π. **Handle + Lazy Read** uses in-memory handles with on-demand `get_object` queries on gating miss steps. Costs come from measured per-episode input and output tokens at the V4-Flash price sheet (\$0.14/M input, \$0.28/M output). Raw serialization re-injects many input tokens (5,000–8,000 prompt tokens per step from serialized payloads; \$0.220/1k). Handle + Summary passes compact handle pointers and summaries (≈ 350 input tokens; \$0.050/1k). Lazy Read needs intermediate `get_object` turns that re-inject raw serializations (\$0.720/1k), and ReAct accumulates multi-turn conversation context over up to 15 turns (\$9.000/1k). Token overhead is relative to the 512-token baseline denominator. The accuracy comparison is *asymmetric*: Handle + Summary is the Π-oracle condition (the summary contains the gating property), while the plain REPL must derive the same quantity in code.

Condition / Budget	Accuracy (%)	Output Tokens	Overhead ($\times$ 512)	Latency (s)	Cost / 1k	Aborts / Sandbox Errors
Handle + Summary (Ours)	**98.0%**	**315**	**0.62**$\times$	**1.31 s** (≤ 18.5 ms comp on ≤ 80-node graphs)	**\$0.050**	**0.0% / 0.0%**
Handle + Lazy Read	**82.5%**	**284**	**0.55**$\times$	**4.75 s**	**\$0.720**	**0.5% / 0.0%**
Handle + REPL (Budget 1$\times$)	88.1%	85	0.17$\times$	1.11 s	\$0.056	11.9% / 0.0%
Handle + REPL (Budget 3$\times$)	99.2%	104	0.20$\times$	1.22 s	\$0.062	0.3% / 0.3%
Handle + REPL (Budget 10$\times$)	99.8%	106	0.21$\times$	1.24 s	\$0.062	0.0% / 0.4%
Handle + REPL (+ certified routines; all budgets)	100.0%	18	0.03$\times$	0.93 s	\$0.060	0.0% / 0.0%
ReAct (Inspection Primitives)	36.8%–52.4%	10,000–14,200	19.5$\times$–27.7$\times$	6.4–8.9 s	\$9.000	24.0%–48.0% (non-finish)
Raw Serialization (Zero-Shot)	34.2%	512	1.00$\times$	1.31 s	\$0.220	16.2% (abort)

Handle + Lazy Read is the typed reference baseline: objects stay in runtime memory, and the model queries fields or payloads on demand via `get_object` (Table 39). Comparing it with **Handle + Summary** shows the eager-vs-lazy systems tradeoff. Lazy field access gives strong accuracy on graph miss gates with few aborts. However, the interactive read turns raise latency and monetary cost (a $14.4\times$ cost multiplier). The reason is that fetching the raw serialization puts token deserialization back into the model context. Eager deterministic summaries remove this query round-trip.

K Limitations and Scope

We list nine scope limitations of our study:

1. **Synthetic Benchmarks vs. Unstructured Real-World Data:** Our benchmark uses procedurally generated 2D geometries, random graph topologies, and calibrated relational dataframes. We chose this synthetic design on purpose. It permits exact ground-truth verification of invariants, removes label ambiguity, and isolates derivation mechanisms from confounding corpus noise. Real-world software engineering and enterprise agents, however, meet messy artifacts: irregular CSV delimiters, unescaped multiline text columns, malformed JSON headers, and noisy geometric sensor feeds. Handle+Summary removes text corruption by keeping objects in runtime memory, but real deployments still need reliable ingestion adapters before handle assignment.
2. **Open Π Selection and Schema Discovery:** In our pipelines, the summary vector $\Pi(O_t)$ comes from a predefined domain schema that spans the task's gating predicates. In open-ended agentic execution (e.g., general software engineering, arbitrary web navigation), precomputing the power set of all conceivable structural properties is computationally intractable. When agents create new tools at runtime (Wölflein et al., 2025), a fixed a priori summary schema Π is impossible by definition. Runtime state architectures will need dynamic, schema-guided, or just-in-time (JIT) property derivation that infers the relevant invariant subspace Π from the agent's task description or tool schema.
3. **Monolingual Prompts and Format Phrasing:** All benchmark queries and prompt templates use standardized English technical phrasing. Our format ablations (Appendix H.4) show that the bottleneck persists across syntactic variants within standard English representations (e.g., GeoJSON, WKT, Markdown). Other languages, vernaculars, or ambiguous conversational framings may add token extraction variance, which warrants future study.
4. **Single-Turn Reasoning Scaffolding:** Our budget sweeps (Appendix G.1) evaluate Qwen3-32B and Gemini-2.5-Flash-Lite on 54,000 queries, but only under standardized single-turn CoT prompting. Multi-turn verifier search loops (e.g., Monte Carlo Tree Search, self-consistency

sampling) or specialized topological fine-tuning may give different derivation scaling slopes, though at even higher Pareto costs.

5. **Absence of Frontier Reasoning Models:** Because of API quota and cost limits, our test-time reasoning sweeps evaluate an open-weight dense model (Qwen3-32B) and an efficient proprietary reasoning model (Gemini-2.5-Flash-Lite). We did not run token sweeps on commercial flagship reasoning models (e.g., OpenAI o1/o3, Claude 3.7 Sonnet Thinking).
6. **Reasoning Budget Ceilings:** In our reasoning sweep, we evaluate both models up to 32,768 requested tokens. Qwen scales its thinking to 5,545 tokens before completion timeouts or failures intervene, while Gemini saturates at $\sim$1,500 executed tokens. Scaling beyond 32k, and multi-turn test-time verification loops, remain unexplored.
7. **Single-Seed Evaluation and Single-Shot Execution Policy:** All primary property-recovery and multi-step pipeline runs use greedy decoding ($T = 0.0$), a single random generation seed, and a single-shot termination policy, to isolate deterministic competence. We also ran exploratory tests of a one-retry recovery policy that echoes the error. With it, raw graph aborts drop from 73.5% to 53.8%, and Casc(All) rises from 17.3% to 46.2%. Raw tabular aborts, however, stay saturated at 98.0% (Casc(All) 2.0%). This shows that parse retries cannot fix semantic payload corruption when models must reconstruct large structured objects in text tool arguments.
8. **Code-Interpreter Baseline and Inspection Arms:** We evaluate on-demand property query APIs as optional and forced inspection primitives (`describe`, Appendix F.1). We also benchmark sandboxed Python execution (Handle+REPL, Section 9 and Appendix J). In Table 43, Handle+REPL matches or exceeds Handle+Summary accuracy at $\geq 3\times$ budgets, and certified routines reach 100.0%. Latency is comparable, the cost factor is a modest 1.20–$1.24\times$, and sandbox runtime error rates are low to zero. However, REPL agents need an interactive code execution sandbox that someone must provision and secure. Eager summaries reach near-parity accuracy without sandbox infrastructure or execution failure modes. We also have not measured runtime cache maintenance, cache invalidation overhead, or memory footprint scaling at multi-gigabyte object scales.
9. **Composed Constrained Decoding and Action-Space Design:** In Appendix E.2, initial runs left the inspection primitive out of the JSON schema enum. These runs suggested an artificial composition penalty (-8.2 pp in graphs, -22.1 pp in tabular). We re-ran with `get_object` included in the schema enum. The drop turned out to be an action-space design failure, not an intrinsic constraint failure. Adding the inspection primitive recovers $+21.8$ pp in graph Handle+Constrained ($38.9\% \rightarrow 60.7\%$) and $+6.0$ pp in Handle+Summary+Constrained ($77.0\% \rightarrow 83.0\%$, with aborts falling to 2.3%). It also recovers $+19.0$ pp in tabular clean accuracy ($60.0\% \rightarrow 79.0\%$). Models actively query `get_object` on 18.9%–22.9% of steps. Typed agent runtimes must therefore make provider JSON schemas include all valid exploratory and retrieval actions available in unconstrained execution.