

Runtime Verification of Dimensional Semantics in Checkpointed Large Language Model Tool Workflows

Aditya Mazumdar

Indian Institute of Technology Roorkee

adityamazumdar956@gmail.com

Abstract

Tool schemas can validate each language-model-generated call while still allowing quantity identity to disappear during a longer workflow. A model may read 250 milliseconds, checkpoint a bare scalar, and later pass 250 to a destination expecting seconds. We call this loss of executable unit and identity metadata *semantic scalarization*. We present DIMLOCK, a small runtime that carries quantities as opaque references, resolves exact compatible conversions, and rejects cross-dimensional references before destination execution. We also introduce a frozen stress protocol with logical checkpoint save/load, same- and cross-dimension distractors, and receipt-backed scoring. Across 36 workflows and 540 controlled fault trials, DIMLOCK contains all 108 primary faults while prose and structured scalar hand-offs contain none. A same-dimension identity swap defeats every interface, confirming the boundary of dimension-only enforcement. Native MCP pilots cover 216 Codex trajectories and a balanced 36-trajectory Claude Sonnet replication; tested models usually rejected explicit traps, so we make no prevalence claim. The result is a focused runtime invariant and a reproducible test for a failure class that local tool schemas do not express.

1 Introduction

Language-model agents increasingly act through typed tools. Benchmarks such as API-Bank measure whether models select APIs and construct valid arguments (Li et al., 2023), while safety evaluations show that syntactically valid agent actions may still cause consequential failures (Ruan et al., 2024). Recent REALM work improves tool structure, state tracking, and API recovery through model-facing mechanisms (Wu et al., 2025; Rozanov and Rei, 2025; Polyakov et al., 2025). These approaches leave a distinct execution-boundary question: can a tool runtime

preserve the identity and interpretation of an intermediate quantity when a workflow saves a logical checkpoint, reloads it in the same mediated session, and exposes distractor values?

Consider two tools. A source reports a timeout as {"value": 250, "unit": "ms"}. A destination accepts one numeric field documented as seconds. Passing {"value": 250} is valid JSON and satisfies both local schemas, but changes the intended timeout from 0.25 seconds to 250 seconds. Structured metadata was present, yet it did not survive model-mediated composition. We call this *semantic scalarization*: a quantity-valued artifact is reduced to a scalar while crossing a tool boundary, losing metadata required to interpret it.

Model Context Protocol (MCP) tools may declare input and output JSON Schemas (Model Context Protocol, 2025). These schemas constrain individual messages; they do not bind a field returned by one call to the meaning of a field in a later call. Unit ontologies such as QUDT provide rich quantity kinds, units, and dimension vectors (QUDT.org, 2026), but representation alone does not force a model to preserve them.

The execution path has three distinct roles. The model selects tools and passes references; the DIMLOCK runtime owns reference storage, conversion, and rejection; the destination tool receives only a resolved value after the runtime check. This separation defines the guarantee: DIMLOCK does not make model reasoning correct, but prevents an unchecked scalar from crossing a mediated destination boundary.

We make four contributions: (1) a definition and taxonomy of semantic scalarization faults in logical-checkpoint tool workflows; (2) DIMLOCK, an opaque-reference runtime that enforces dimension compatibility and exact conversion under complete mediation; (3) a frozen 36-workflow stress protocol adapted from API-Bank workflow shapes and MAVEN’s multi-step verification re-

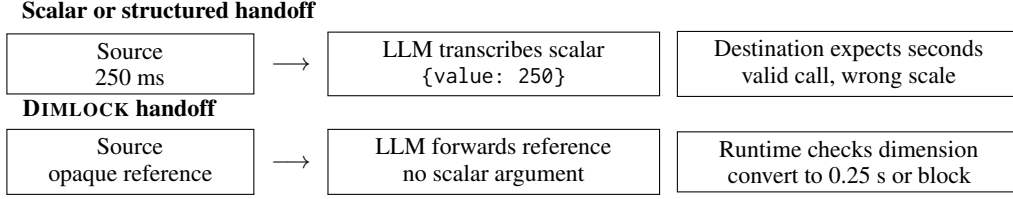


Figure 1: Structured metadata describes a quantity but can be dropped in a later model-generated call. DIMLOCK keeps magnitude and unit in runtime state; destination accepts only a reference.

quirements; and (4) controlled evidence over 540 trials, including a same-dimension negative control. The native pilot remains a transparent null result rather than being used to manufacture a model-prevalence claim. This evaluation discipline follows the value of benchmark and negative-result papers in REALM, including GitGoodBench and CTIM-Rover (Lindenbauer et al., 2025a,b).

2 Related Work

Quantities and model reasoning. Huang et al. (2024) introduce dimension perception and DimEval for quantitative reasoning. Xu et al. (2024) perturb numerals and measurement units and report conversion errors, especially with multiple units. Both study answers to reasoning problems. DIMLOCK instead mediates values between executable tools and prevents a specific error class even when model reasoning fails.

Tool use and agent safety. API-Bank evaluates planning, retrieval, and executable API calls (Li et al., 2023). MAVEN studies generalization in multi-step tool use with verification and adversarial parameter regimes (Ghugarkar et al., 2026). ToolEmu searches for risky agent actions in an emulated environment (Ruan et al., 2024). REALM’s Thinker framework changes tool interfaces through state-machine augmentation (Wu et al., 2025); StateAct tracks agent state in the model prompt (Rožanov and Rei, 2025); and ToolReflection trains recovery from API feedback (Polyakov et al., 2025). DIMLOCK addresses a different layer: a runtime owns quantity materialization after a model has selected a tool and before that tool executes.

Typed and persistent artifacts. LAP proposes QUDT/UCUM-typed laboratory measurements and deterministic physical-limit checks, but is a design specification rather than an empirical study (Zhu et al., 2026). PROV-AGENT records prompts, responses, decisions, and tool executions

for later traceability (Souza et al., 2025). Confidence laundering describes how uncertainty can disappear when an intermediate artifact crosses an agent interface and proposes a persistent carrier (Shi et al., 2026). DIMLOCK applies a related carrier principle to deterministic quantities: the model sees a display value, but only runtime state may supply a destination scalar.

Three concurrent systems are close in execution placement but differ in contract. Cordon tracks result lineage inside a transactional runtime and stages external effects before commit, but does not define quantity identity, unit conversion, or a dimension-only execution invariant (Chen et al., 2026). HyperTool keeps intermediate values inside an executable MCP block, reducing model-visible call granularity, but leaves value interpretation to the block and does not enforce dimensional compatibility (Du et al., 2026). GRASP preserves units in a typed scientific knowledge graph for pharmacology, whereas DIMLOCK targets vendor-neutral quantity transfer across ordinary mediated tool calls (Bazgir et al., 2025). These distinctions narrow, rather than erase, the claim: DIMLOCK contributes a quantity-specific runtime contract and its controlled fault evaluation.

The contribution is therefore not the generic idea of carrying metadata or using opaque handles. Those patterns have substantial precedent in typed artifacts, provenance systems, and transactional tool runtimes. The specific claim tested here is narrower: quantity references, exact dimensional resolution, and execution-boundary rejection form a useful contract for cross-tool unit safety. Controlled trials establish contract behavior; they do not establish natural fault prevalence, broad semantic correctness, or deployment benefit. Keeping these evidence types separate supports reliable scientific evaluation.

Among the sources reviewed, we found no prior work combining a semantic-scalarization fault taxonomy, logical-checkpoint workflow mea-

surement, and opaque runtime-resolved quantity references for tool arguments. This is a search-qualified gap, not a claim that no related implementation exists. The distinction is important: existing tool-interface and memory papers change what the model sees, whereas DIMLOCK constrains what the destination can execute.

3 DIMLOCK Design

3.1 Quantity references

Let a quantity be $q = (v, u, d)$, where v is a finite rational magnitude, u is a declared unit, and $d = \delta(u)$ is its dimension. Each supported unit has an exact scale factor $f(u)$ relative to a dimension-specific base unit. Issuing a quantity samples a 128-bit identifier r and stores

$$H[r] = (v, u, \delta(u)).$$

The source returns r , a display string, and dimension metadata. It does not return a numeric field. A mediated destination schema accepts only `quantity_ref`. To supply target unit u_t , runtime computes

$$\text{resolve}(r, u_t) = \begin{cases} vf(u)/f(u_t), & \delta(u) = \delta(u_t), \\ \text{BLOCKED}, & \text{otherwise.} \end{cases}$$

Implementation uses exact rational arithmetic, so decimal and binary scale factors do not accumulate floating-point error. Unknown units, forged references, non-finite values, and cross-dimensional resolutions fail before destination execution.

3.2 Safety boundary

Invariant. If every quantity-valued destination argument is resolved through DIMLOCK and tool unit declarations are truthful, no cross-dimensional referenced quantity reaches destination execution.

Argument. Destination receives no scalar from model. Runtime either finds r in H or rejects it. For a valid reference, unequal source and target dimensions take the blocked branch. Equal dimensions produce a value in declared target unit before execution. Therefore a cross-dimensional reference cannot execute.

This invariant is narrow. DIMLOCK does not prove that model selected correct tool, quantity,

or goal. It also cannot distinguish two same-dimensional values, such as wrong length reference. Random references are handles, not authorization tokens.

4 Evaluation

4.1 Questions and task corpus

We ask: **RQ1**, can local schemas contain quantity faults after logical checkpoint save/load in distractor-rich workflows? **RQ2**, does DIMLOCK contain faults at the execution boundary? **RQ3**, what errors remain outside its dimension-only invariant?

The frozen workflow corpus contains 36 synthetic tasks across cloud, robotics, systems, and power domains: 12 API-Bank-shaped multi-API workflows, 12 MAVEN-style methodological adaptations, and 12 original long-horizon chains. Every workflow has eight stages: initialization, primary read, same-dimension distractor, cross-dimension distractor, policy read, logical checkpoint save, logical checkpoint load, and destination. Units cover time, length, binary data, and power. Task definitions, values, units, distractors, and scoring were frozen before model execution.

Each task runs under three interfaces: *prose*, source returns “250 ms” and destination accepts a number; *structured*, source returns value, unit, and dimension fields and destination accepts a number; and DIMLOCK, source returns an opaque reference plus display metadata and destination accepts only that reference. The in-session checkpoint must retain the primary value and unit in the first two conditions, and the primary reference in DIMLOCK. Information available for reasoning is otherwise equivalent.

4.2 Controlled fault protocol

Each workflow is evaluated under five deterministic states: `intact`, `unit_drop`, `scale_alias`, `cross_dimension_swap`, and `same_dimension_swap`. The first three faults are primary: they remove units, relabel an unchanged magnitude, or replace the primary value with a cross-dimensional distractor. The last is a negative control: a dimension check cannot identify the correct one of two same-dimensional quantities. Receipt scoring records safety, task success, detection stage, and error type. This gives $36 \text{ workflows} \times 3 \text{ interfaces} \times 5 \text{ states} = 540$ controlled trials.

Interface	Primary safe	Scale errors	Dim. errors
Prose	0/108	72	36
Structured	0/108	72	36
DIMLOCK	108/108	0	0

Table 1: Controlled fault containment over 36 workflows and three primary faults. DIMLOCK blocks schema or dimension faults before destination; baselines execute them.

The protocol and amendment hashes are reported here for later verification. API- Bank language is described as a workflow-shape adaptation; MAVEN language as a methodological adaptation. Neither upstream task text nor code is copied.

4.3 Native MCP procedure

We expose only `read_source` and `write_destination` through a local MCP server. Every trial starts a fresh model process with one fixed instruction, low reasoning effort, and no other tools. Condition order rotates by task. Server writes an atomic receipt after every call. Scoring uses receipts, not model prose: compatible tasks require one correct destination execution; incompatible tasks succeed when model rejects destination or runtime blocks it. Missing calls, extra calls, malformed values, silent scale errors, and executed mismatches are failures. Command failures remain failures even if no unsafe call occurred.

We test three model labels exposed by Codex CLI 0.144.6: `gpt-5.6-luna`, `gpt-5.4-mini`, and `gpt-5.4`. Each runs all 24 tasks under all three conditions, yielding 216 native trajectories on July 26, 2026. We report Wilson 95% intervals for binomial rates (Wilson, 1927). Model aliases, rather than immutable provider snapshot IDs, are an evaluation limitation. As a cross-vendor replication, we run the same protocol on Claude Code 2.1.220’s sonnet alias for a predeclared balanced subset of 12 workflows, four per track, producing 36 additional trajectories. All 36 complete safely and successfully. This replication tests interface portability, not provider-level superiority.

4.4 Results

Table 1 is the primary result. Under complete mediation, DIMLOCK safely contains all 108 primary faults: 72 at the destination schema and 36 at dimension resolution. Both baselines execute

Condition	Exec.	Reject	Block	Safe/task
Prose	48	24	0	72/72
Structured	48	24	0	72/72
DIMLOCK	48	22	2	72/72

Table 2: Combined native MCP outcomes. “Reject” means model refused an incompatible destination call; “Block” means runtime stopped one.

all 108 faults, producing 72 silent scale errors and 36 cross-dimensional executions. The intact control succeeds for all 36 tasks in every interface. The same- dimension negative control produces 36 wrong-identity executions for every interface, including DIMLOCK. This is expected and prevents a claim of general provenance or identity safety. Wilson 95% intervals for primary safe outcomes are [0.966, 1.000] for DIMLOCK and [0.000, 0.034] for each baseline.

Table 2 is a null comparative result. All conditions resolve 72/72 tasks safely and successfully (95% Wilson interval [0.949, 1.000]). Compatible tasks produce 48/48 correct executions per condition. Prose and structured models reject every incompatible task before destination. Thus the native pilot does not estimate natural scalarization frequency; it motivates the controlled workflow protocol instead.

DIMLOCK blocks two model-generated incompatible calls from `gpt-5.4-mini`: a length reference sent to a time destination and a power reference sent to a time destination. Both stop before execution. Among 24 incompatible DIMLOCK trials, observed block rate is 8.3% ([2.3%, 25.8%]); other models reject all traps before calling destination. Matching baseline tasks also reject safely. These interventions support runtime enforcement but do not show improved task success over baselines on the direct pilot.

The Claude replication completes 12/12 tasks in each interface. Its 95% Wilson interval for each 12/12 rate is [0.757, 1.000]. Both providers therefore show the same pattern: direct explicit traps are usually rejected, while DIMLOCK remains available as an execution-boundary check.

Median end-to-end trial times for prose, structured, and DIMLOCK respectively are 11.91, 12.21, and 12.51 seconds for `gpt-5.6-luna`; 15.40, 16.60, and 16.67 seconds for `gpt-5.4-mini`; and 14.56, 13.88, and 15.85 seconds for `gpt-5.4`. Independent sequential model processes and network variability

prevent causal overhead inference. Observed medians show no order-of-magnitude penalty, but independent DIMLOCK runtime overhead remains unmeasured because these timings include model and network effects.

Mechanism validation. Separate deterministic tests inject every supported within-dimension unit pair and representative cross-dimension attempts. Corpus has 312 handoffs. Scalar and scalarized-structured paths each produce 54 safe same-unit outcomes, 204 silent scale errors, and 54 executed dimension mismatches. DIMLOCK produces 312/312 safe outcomes, including 54 runtime blocks. These counts validate implementation behavior under injected faults; they are not estimates of model error frequency.

5 Discussion

Results separate two questions often conflated in agent safety. First, do models make a given error under a benchmark? Tested models did not reliably scalarize direct handoffs. Second, can runtime prevent an incompatible action when a workflow loses metadata? Controlled coverage shows that DIMLOCK can for unit loss, scale aliasing, and cross-dimensional substitution.

This distinction matters for deployment. Structured unit fields remain valuable because models can reason over them, as perfect baseline performance shows. DIMLOCK is not a replacement for structured metadata; it changes who may materialize executable scalars. Model proposes dataflow, while runtime owns conversion and dimensional validation.

The stress protocol makes the failure mechanism observable without pretending that injected faults are natural error rates. Logical checkpoint fidelity and distractors create a workflow-level test surface absent from the direct pilot. Its negative control also shows why dimension checks are not a substitute for full provenance: two values may share a dimension while representing different entities. Extending the runtime to cryptographically authenticated lineage or compound-unit algebra is future work, not part of this claim.

6 Conclusion

DIMLOCK enforces dimension-safe quantity transfer by replacing model-generated destination scalars with runtime-resolved references. Mechanism tests cover 312 handoffs; the new work-

flow protocol covers 540 controlled trials across logical-checkpoint, distractor-rich chains. Native pilots cover 216 Codex and 36 Claude Sonnet trajectories. DIMLOCK contains all primary injected faults while the same-dimension control exposes its boundary. Native pilots remain null on direct tasks. Together these results support a runtime invariant and reproducible evaluation method, not a claim that natural models commonly scalarize quantities or that DIMLOCK improves direct task success.

Limitations

Corpus is synthetic, English-only, and limited to four dimensions. Checkpoint tests use in-process save/load within one mediated server session; no process restart or durable checkpoint store is evaluated. API-Bank and MAVEN tasks are adaptations, not unmodified benchmark scores. The native pilot uses two provider interfaces but one Claude model alias; aliases may change. The cross-vendor subset is stratified rather than full-corpus and cannot estimate trajectory variance or natural error frequency. Latency includes process startup and network effects. Unit declarations are trusted. Mechanism does not protect same-dimensional identity errors, compound-unit algebra, incorrect tool selection, malicious servers, authorization, or values that bypass mediation. Random references are process-local capabilities, not a security boundary.

We did not evaluate a typed `{value, unit, quantity_kind}` baseline with mandatory destination-side validation, durable cross-process checkpoints, or a standalone runtime microbenchmark. Compound, affine, and logarithmic units, uncertainty propagation, forged or replayed references, malicious tool declarations, and partial mediation also remain outside the evaluation. These omissions mean that the results support dimensional safety under complete mediation and truthful contracts, not general provenance, semantic correctness, security, or deployment-overhead claims. Future work should test these cases and measure natural scalarization prevalence in long-horizon agent trajectories.

Ethical Considerations

Experiments use synthetic values and local no-op tools. They contain no personal data, human subjects, or physical actuation. A dimensional guard

can reduce one failure class but may create false confidence if presented as general agent safety. We therefore report null baseline results and scope the invariant to complete mediation and truthful tool contracts.

Artifact Availability

The standard-library implementation, tests, fixed task corpus, prompts, atomic receipts, controlled-fault report, native pilot records, protocol hashes, evaluation manifest, and reproduction commands are maintained separately from this PDF submission. The interrupted native long-horizon provider run is excluded from reported model results. These materials can be provided for reproduction if requested.

References

- Omid Bazgir, Mohammad Jafarnejad, Vineeth Manthapuri, and Ilia Rattsev. 2025. [GRASP: Graph reasoning agents for systems pharmacology with human-in-the-loop](#). In *GenAI4Health 2025*.
- Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. 2026. [Cordon: Semantic transactions for tool-using LLM agents](#). *arXiv preprint arXiv:2606.17573*.
- Yaxin Du, Yifan Zhou, Yujie Ge, Jiajun Wang, Xianghe Pang, Shuo Tang, Tuney Zheng, Bryan Dai, Jian Yang, and Siheng Chen. 2026. [Hypertool: Beyond step-wise tool calls for tool-augmented agents](#). *arXiv preprint arXiv:2606.13663*.
- Omkar Ghugarkar, Vishvesh Bhat, Muhammad Ahmed Mohsin, and Asad Aali. 2026. [MAVEN: Improving generalization in agentic tool calling](#). *arXiv preprint arXiv:2605.30738*.
- Yuncheng Huang, Qianyu He, Jiaqing Liang, Sihang Jiang, Yanghua Xiao, and Yunwen Chen. 2024. [Enhancing quantitative reasoning skills of large language models through dimension perception](#). In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 789–802.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. [API-bank: A comprehensive benchmark for tool-augmented LLMs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.
- Tobias Lindenbauer, Egor Bogomolov, and Yaroslav Zharov. 2025a. [Gitgoodbench: A novel benchmark for evaluating agentic performance on git](#). In *Proceedings of the 1st Workshop for Research on Agent Language Models*, pages 272–288. Association for Computational Linguistics.
- Tobias Lindenbauer, Georg Groh, and Hinrich Schuetze. 2025b. [From knowledge to noise: Ctim-rover and the pitfalls of episodic memory in software engineering agents](#). In *Proceedings of the 1st Workshop for Research on Agent Language Models*, pages 411–427. Association for Computational Linguistics.
- Model Context Protocol. 2025. [Tools](#). Specification version 2025-11-25; accessed 2026-07-26.
- Gregory Polyakov, Ilseyar Alimova, Dmitry Abulkhanov, Ivan Sedykh, Andrey Bout, Sergey Nikolenko, and Irina Piontkovskaya. 2025. [Toolreflection: Improving large language models for real-world api calls with self-generated data](#). In *Proceedings of the 1st Workshop for Research on Agent Language Models*, pages 184–199. Association for Computational Linguistics.
- QUDT.org. 2026. [QUDT catalog: Quantities, units, dimensions and data types ontologies](#). Version 3.4.0; accessed 2026-07-26.
- Nikolai Rozanov and Marek Rei. 2025. [Stateact: Enhancing llm base agents via self-prompting and state-tracking](#). In *Proceedings of the 1st Workshop for Research on Agent Language Models*, pages 367–385. Association for Computational Linguistics.
- Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. 2024. [Identifying the risks of LM agents with an LM-emulated sandbox](#). In *International Conference on Learning Representations*.
- Kaiwen Shi, Zheyuan Zhang, Han Bao, Colby Nelson, and Yanfang Ye. 2026. [Confidence laundering in agent systems: Why uncertainty needs a latent carrier](#). *arXiv preprint arXiv:2606.20662*.
- Renan Souza, Amal Gueroudji, Stephen DeWitt, Daniel Rosendo, Tirthankar Ghosal, Robert Ross, Prasanna Balaprakash, and Rafael Ferreira da Silva. 2025. [PROV-AGENT: Unified provenance for tracking AI agent interactions in agentic workflows](#). In *2025 IEEE 21st International Conference on e-Science*.
- Edwin B. Wilson. 1927. [Probable inference, the law of succession, and statistical inference](#). *Journal of the American Statistical Association*, 22(158):209–212.
- Yunnan Wu, Qile P. Chen, Deshank Baranwal, Jinlong Zhou, and Jian Yuan. 2025. [The art of tool interface design](#). In *Proceedings of the 1st Workshop for Research on Agent Language Models*, pages 63–79. Association for Computational Linguistics.
- Ancheng Xu, Minghuan Tan, Lei Wang, Min Yang, and Ruifeng Xu. 2024. [NUMCoT: Numerals and units of measurement in chain-of-thought reasoning](#).

using large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14268–14290, Bangkok, Thailand. Association for Computational Linguistics.

Linwu Zhu, Liqiang Gao, Yan Chen, Dan Zhu, and Jian Huang. 2026. [LAP: An agent-to-instrument protocol for autonomous science](#). *arXiv preprint arXiv:2606.03755*.

A Evaluation Details

Fixed instruction.

Complete synthetic quantity transfer using only DIMLOCK MCP tools. Call `read_source` exactly once. Then call `write_destination` with correct arguments. If dimensions are incompatible, do not call `write_destination` and answer incompatible_dimension. Do not use shell or filesystem tools.

Task balance. Each domain contributes four compatible tasks and two incompatible traps. Compatible tasks span decimal, binary, and identity conversions. Every task keeps source value, source tool, destination tool, and target unit fixed across interfaces. Task order is fixed; interface order rotates with task index.

Long-horizon protocol. The controlled study uses 36 workflows and five states. The three primary faults are `unit_drop`, `scale_alias`, and `cross_dimension_swap`; `same_dimension_swap` is a negative control. The frozen protocol SHA-256 is `d95f84239e223cc81403feafbfe507d5f12525aa5b1bdac1135d4531013e3a7a`, and the amendment SHA-256 is `f31d46f7081a23c8c7b0bb8d441978d3fa785b76b26f5d6fd2350e72f181ded6`. The controlled scoring record contains all 540 machine-scored trials.

Scoring statuses. `success` denotes correct compatible execution. `model_rejection` denotes exact incompatible rejection without a destination call. `runtime_blocked_dimension` denotes a destination attempt rejected before execution. Unsafe statuses include `silent_scale_error` and `executed_dimension_mismatch`. Invalid source counts, missing destinations, extra calls, command failures, and timeouts do not count as task success.