EffectSet: A Batch Tool Call Is More Than One Outcome

Anonymous authors
Paper under double-blind review

Abstract

A batch tool call can apply some requested effects, reject others, and leave still others uncertain. Treating that call as one success or failure forces a language agent to reconstruct item-level state from overlapping responses, which can abandon unfinished work or repeat completed effects. We introduce EffectSet, a deterministic runtime representation that folds tool events by target item into settled, verify, retry, and escalate sets, then exposes the residual obligations to the agent. We also introduce PartialCallBench, an executable benchmark with 288 fresh tasks, twelve response grammars, 6,912 target items, and 1,496 model–task decisions from Claude Haiku and GPT-5.6 Luna. On the 120-task main split, exact residual-action accuracy rises from 29.2% with raw transcripts and 72.5% with raw-plus-tagged transcripts to 86.7% with item-grouped canonical histories, then 100% when each item receives a derived state. A fully crossed, tags-only control sharpens the mechanism: grouping changes Haiku and Luna by -8.3 and $+4.2$ points, whereas derived state reaches 100% for both models, gaining 20.8 and 8.3 points over call-ordered tags. An independently generated 48-task Luna split and four unseen synthetic grammars also reach 100% with EffectSet. More decisively, a 32-task replay grounded in four independently documented public batch protocols raises exact accuracy from 71.9% raw and 75.0% tagged to 100% with the state ledger, a paired gain of 25.0 points over tags. The representation is linear-time, processes 120,000 events in 78.2 ms, and reduces prompt characters by 25.2%. EffectSet turns partial batch execution into explicit state that agents can act on exactly and efficiently.

1 Introduction

Suppose an agent asks a tool to publish 48 documents. The response reports 31 created, six rejected before execution, four invalid, and seven with no terminal receipt. A later verification call confirms three of the uncertain documents and proves four absent. The correct next action is neither “retry the batch” nor “stop”: the agent must retry exactly ten items, verify none, escalate four, and never repeat the 34 settled effects.

Current tool-agent interfaces make this decision unnecessarily difficult. Agent benchmarks evaluate executable trajectories and final state (Yao et al., 2024; Lu et al., 2025; Trivedi et al., 2024), while function-calling benchmarks score individual calls (Yan et al., 2024; Wu et al., 2026). In each case, the history normally remains call-shaped. When several calls partially overlap, the model must join item identities, interpret heterogeneous status codes, preserve earlier confirmations, and derive the unfulfilled subset inside its context window. Adding status labels helps interpretation but leaves this reconciliation problem intact.

Our central insight is that the recovery unit should be an effect, not a call. EffectSet compiles heterogeneous responses into canonical item events and folds those events into a small monotone ledger. A confirmed effect remains settled; an uncertain effect requires verification; a confirmed absence or retryable rejection can be retried; and a permanent rejection is escalated. The model receives either the ledger alone or the ledger plus its residual action sets. This is a deterministic context transformation, not a learned policy or a model judge.

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.

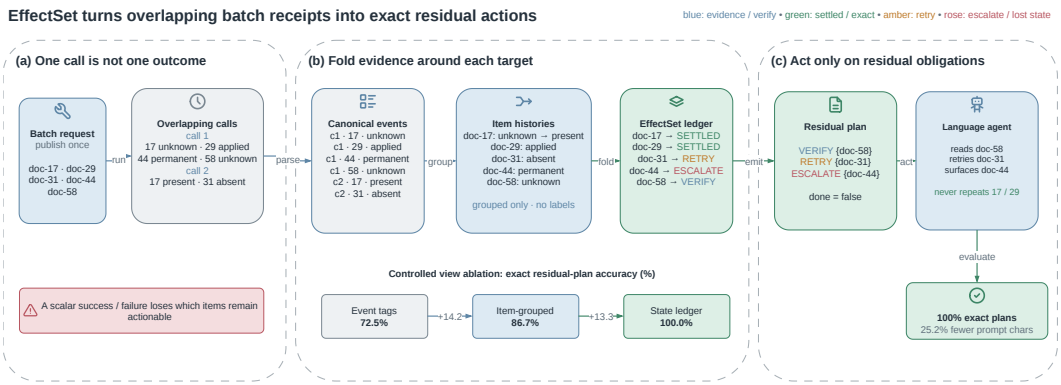

Figure 1: EffectSet changes the state unit from a whole call to the individual effects that survive across calls. Matched view ablations separate call ordering, item grouping, and explicit state derivation; the resulting residual plan returns only actionable obligations to the agent.

We make the abstraction measurable with PartialCallBench. Fresh tasks contain partially overlapping calls, mixed outcomes, later confirmations, genuine all-done cases, and exact executable labels. The main and confirmation splits vary batch size from 12 to 64 items across HTTP multi-status, CLI, SQL batch, and workflow responses. A transfer split replaces all four response grammars and application domains. PublicReplay-32 then reconstructs traces from the independently specified response contracts of Elasticsearch Bulk, Microsoft Graph batch, Google Cloud Storage batch, and JSON-RPC (Elastic, 2026; Microsoft, 2026; Google Cloud, 2026; JSON-RPC Working Group, 2013). Every condition uses identical goals and answer instructions; only the evidence view changes.

The results isolate a large, repeatable representation effect. Haiku improves from 29.2% on raw transcripts and 72.5% on raw-plus-tagged transcripts to 86.7% with item-grouped canonical histories and 100% with a derived state per item. Luna independently improves from 56.2% raw and 70.8% raw-plus-tags to 100%. A fully crossed control then removes the raw-text difference: on the same 24 tasks and fixed four-task packing, call-ordered tags alone score 79.2% for Haiku and 91.7% for Luna, grouping scores 70.8% and 95.8%, and derived state reaches 100% for both. The clean control assigns the stable gain to state derivation rather than grouping alone.

The external replay sharpens that mechanism. Across the four public protocol families, raw and tagged traces obtain 71.9% and 75.0% exact plans, item grouping alone obtains 62.5%, and the state ledger obtains 32/32. On an identical 24-task matrix, the ledger also reaches 24/24 for both Haiku and Luna, improving over call-ordered tags-only views by 20.8 and 8.3 points. State derivation is the stable operation across response surfaces and model families; grouping alone is model-dependent.

Our contributions are:

- An item-level execution abstraction. EffectSet reconciles partial responses across calls into auditable per-item evidence and exact residual sets in linear time.
- A controlled and externally grounded benchmark. PartialCallBench contains 288 model-facing tasks, twelve response grammars, 6,912 target items, 11,796 canonical events, two model families, a same-task cross-model matrix, and an independently checked oracle.
- A mechanism-level empirical result. A matched tags-only control shows that grouping alone changes accuracy by -8.3 and +4.2 points across the two models, while derived state reaches 100% for both and all four public protocol families. An explicit residual summary adds zero.

2 Effect Sets for Partial Batch Execution

2.1 Problem setup

Let I be the set of target items for a goal. A tool call c_t returns zero or more events $e = (t, i, o, r)$, where $i \in I$, o is a protocol-specific outcome, and r is a receipt reference. An adapter maps o to one of seven canonical semantics:

$$\{\text{applied}, \text{present}, \text{duplicate}, \text{absent}, \text{retryable}, \text{unknown}, \text{permanent}\}.$$

The desired output is a partition of unresolved obligations into V (verify), R (retry), and E (escalate), plus a Boolean done. A prediction is exact only when all three sets and done match the executable oracle.

This formulation separates two questions often conflated by a scalar call status. First, did the requested effect occur for item i? Second, what operation is now permitted? unknown does not imply failure and therefore requires a read before retry, whereas absent proves that retry is permitted. Likewise, a duplicate response can be evidence that the desired effect is already present.

2.2 Monotone item reconciliation

For every target i, EffectSet stores state $s_i \in \{\text{settled}, \text{verify}, \text{retry}, \text{escalate}\}$ and an append-only evidence list. New items start in retry. Events are processed in chronological order with

$$s_i \leftarrow \begin{cases} \text{settled}, & o \in \{\text{applied}, \text{present}, \text{duplicate}\}, \\ \text{verify}, & o = \text{unknown}, \\ \text{retry}, & o \in \{\text{absent}, \text{retryable}\}, \\ \text{escalate}, & o = \text{permanent}, \end{cases} \qquad \text{unless } s_i = \text{settled}. \tag{1}$$

The final guard encodes monotonicity: once the effect is confirmed, a later duplicate, stale rejection, or repeated report cannot reopen it. The residual plan is $V = \{i : s_i = \text{verify}\}$, $R = \{i : s_i = \text{retry}\}$, and $E = \{i : s_i = \text{escalate}\}$; done holds exactly when all three are empty. Runtime is $O(|I| + |e|)$ and the ledger retains the receipt references supporting every state.

A controlled representation staircase. We expose a matched canonical staircase to separate the contribution of each operation. Tags-only lists canonical events in call order without the raw transcript. Folded-unlabeled groups that event history by target item but includes neither a derived state token nor residual action lists. Ledger-only adds the state s_i and its receipt-backed evidence but still omits V, R, E, and done; the model must derive the requested partition. Full EffectSet adds the explicit residual summary. The 24-task, two-model crossed control evaluates the first three views with identical packing; the larger main suite evaluates raw, raw-plus-tags, grouped, ledger-only, and full views.

3 PartialCallBench

3.1 Task construction

Each task samples 12–64 target items and assigns each a terminal category: settled, retry, verify, or escalate. We then sample a short evidence history consistent with that category. Examples include unknown→present, retryable→applied, and unknown→absent→permanent. Events are interleaved across up to four overlapping calls, so reading only the last call loses state. Every tenth controlled task is genuinely complete. Partition sizes vary, so a fixed class-count heuristic cannot solve the benchmark.

The main suite renders these events as HTTP 207 records, aligned CLI rows, SQL batch diagnostics, or workflow prose across records, jobs, documents, and packages. The transfer split uses disjoint domains and nested NDJSON, XML multistatus, YAML result lists, and

Split	Tasks	Items	Events	Views	Model
Pilot	40	480	818	3×2	Haiku, Luna
Main	120	3,360	5,709	5	Haiku
Confirmation	48	1,792	3,096	4	Luna
Grammar transfer	48	896	1,529	3	Haiku
Public protocol replay	32	384	644	4	Haiku
Total	288	6,912	11,796	1,496 model–task decisions	

Table 1: Model-facing evaluation scale. "Views" counts evidence conditions; the pilot applies three views to both models. The decision total also includes a frozen four-view, two-model replay of 24 main tasks.

pipe-delimited diagnostics. The semantic contract is stated in every condition; the test is reconciliation, not memorization of status codes.

PublicReplay-32 adds eight schema-faithful traces for each of four independently specified batch interfaces. The adapters use the public identity and status contracts, map 152 requested-but-unreturned identities to unknown, and reject duplicate or unrecognized response identities instead of guessing. Target names and outcome combinations are newly generated; this is a documentation-grounded replay rather than production traffic. Separately, four immutable fixtures project official example response bodies into 15 hand-annotated subresponses; all four match the adapters exactly. A bare 409 conflict or 412 failed precondition remains unknown until a read verifies the postcondition—it is not treated as proof of success.

3.2 Independent oracle validation

The reference labels are computed by a table-driven oracle implemented separately from the EffectSet class. It agrees on all 528 generated tasks, including a 240-task mechanism suite, and on all 2,800 one-item histories of length one through four over the seven canonical event types. The scorer compares structured sets directly; no LLM judge is used. The source-fixture audit is independent of the replay generator and additionally checks reordered identities, permanent response errors, and an unrecognized null id.

4 Evaluation

4.1 Conditions and controls

Raw transcript preserves call boundaries and native response grammar. Event tags appends canonical item/status rows to the raw transcript but does not reconcile them. Tags-only retains the same call-ordered canonical rows without raw text and appears in the fully crossed common-task control. Folded-unlabeled groups those canonical histories by item without assigning a derived item state. Ledger-only folds all events by item and exposes the supporting receipts while omitting the residual answer. EffectSet adds a deterministic residual summary. All conditions receive the same target set, semantic legend, question, and exact JSON output schema.

Claude Haiku and GPT-5.6 Luna run at low effort through their native read-only agent harnesses. Prompts batch independent tasks only for cost efficiency; the public replay and common-task matrix fix four tasks per call across views. The parser requires one complete JSON object per task; all 1,496 evaluated decisions parse. We report task-paired bootstrap 95% intervals with 10,000 samples and repeat the calculation by resampling inference batches. Tasks, prompts, raw outputs, parsed answers, timings, token/cost metadata, and seeds are retained.

Split	Model	Raw	Raw+tags	Grouped	State ledger	EffectSet
Main	Haiku	29.2	72.5	86.7	100.0	100.0
Confirmation	GPT-5.6 Luna	56.2	70.8	–	100.0	100.0
Grammar transfer	Haiku	87.5	81.2	–	–	100.0

Table 2: Exact residual-action accuracy (%). The main view staircase rises from call-shaped evidence to grouped canonical histories and then derived item state. Complete EffectSet reaches 100% for the main model, an independently generated Luna split, and four unseen response grammars.

4.2 Research questions

RQ1 asks whether item-level reconciliation improves exact residual actions over call-shaped evidence. RQ2 separates call ordering, item grouping, derived state labels, and the explicit residual summary. RQ3 tests a disjoint model/task split, unseen grammars, public protocol contracts, and an identical-task cross-model matrix. RQ4 measures the operational consequence of predicted partitions and the local runtime cost.

5 Results

5.1 Reconciliation makes residual actions exact

Table 2 gives the central result. On the 120-task main split, EffectSet improves exact accuracy by 70.8 points over raw transcripts and 27.5 points over canonical event tags. The independently generated Luna split confirms gains of 43.8 and 29.2 points. Both task- and inference-batch-resampled intervals remain positive. The result is exact at the task level: every item must be assigned to the correct action set, and done must also match. Wilson 95% intervals for the 100% task rates are [96.9, 100.0] on the 120-task main split and [92.6, 100.0] on the 48-task confirmation split.

The raw-plus-tag baseline shows why semantic normalization is not enough. Its item-level accuracy is 94.6% on the main split and 99.0% on confirmation, yet one wrong item makes the whole residual plan incorrect or incomplete. Reconciliation turns those event semantics into persistent item state and removes the remaining transition burden.

5.2 Derived item state is the load-bearing operation

The larger main staircase rises from 72.5% with raw-plus-tags to 86.7% with grouped canonical histories and 120/120 after adding state. The first transition describes the complete view change because it also removes the raw transcript. The matched common-task control isolates grouping: relative to call-ordered tags-only, grouping changes Haiku by -8.3 points and Luna by +4.2 points. Derived state is stable where grouping is not: it raises tags-only accuracy by 20.8 points [4.2, 37.5] for Haiku and 8.3 points [0.0, 20.8] for Luna, reaching 24/24 for both. Adding the residual lists changes no prediction: ledger-only and full EffectSet both reach 100% on the main Haiku and confirmation Luna splits. Explicit state removes the transition burden; the aggregate summary is operational convenience rather than the source of accuracy.

The experiments also rule out a token-volume explanation. Folded-unlabeled uses 44.3% fewer input characters per main task than raw history, ledger-only uses 36.7% fewer, and full EffectSet uses 25.2% fewer. In the fixed-packing control, tags-only averages 2,518 characters per task, grouped histories 2,150, and the state ledger 2,420. The ledger is therefore shorter than tags-only while improving both models, so neither more context nor canonical vocabulary explains the result.

Figure 2 exposes the scale interaction. Event tags fall from 82.5% at 12 items to 55.0% at 48, while raw transcripts reach 22.5% at 48. Grouped-unlabeled histories retain 82.5% at

View	Evidence for pkg-2447-08	Prediction
Raw	UNKNOWN in call 1; ABSENT in call 2	verify ×
Raw + tags	Same tagged events remain separated by call	verify ×
Grouped	call-1 UNKNOWN; call-2 ABSENT	retry ✓
State ledger	RETRY <- unknown, absent	retry ✓

Table 3: A 12-item SQL-batch case. Grouping the two reports for one item makes the latest confirmed absence visible without supplying its derived state.

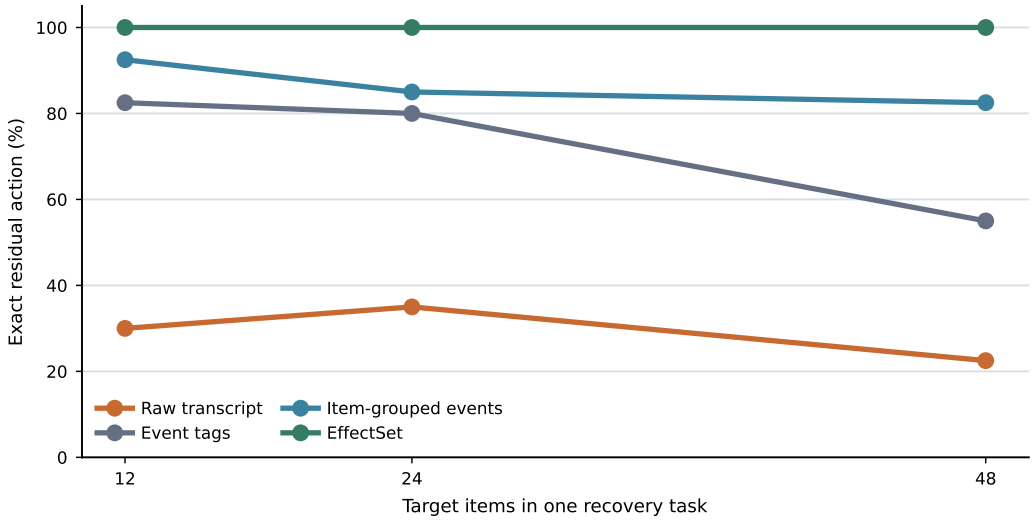

Figure 2: Main-split exact accuracy by target-set size. Item grouping recovers much of the gap without derived state labels; complete EffectSet remains 100% from 12 to 48 items.

48 items, and EffectSet remains 100% at every size and in each of the four main response protocols and domains.

5.3 State derivation transfers across response surfaces

The transfer split replaces every controlled grammar and application domain. EffectSet again reaches 48/48 exact decisions. Raw transcripts reach 87.5% and event tags 81.2%; the paired EffectSet gains are 12.5 points [4.2, 22.9] and 18.8 points [8.3, 31.2]. The ordering of the two baselines changes, but the reconciled representation remains exact. This supports the intended adapter boundary: protocol-specific parsing ends at canonical events, while the ledger and residual planner are shared.

Table 4 closes the two principal external-validity gaps. On public protocol replays, the state ledger improves over raw responses by 28.1 points [12.5, 43.8] and over event tags by 25.0 points [9.4, 40.6]. All four protocol families reach 8/8, and the 32/32 Wilson interval is [89.3, 100.0]. Grouping alone is not sufficient here: it reaches 62.5%, localizing the robust external gain to deterministic state derivation.

The common-task matrix removes both the model/split confound and the raw-text difference. Ledger-only reaches 24/24 for both models, improving over tags-only by 20.8 points [4.2, 37.5] for Haiku and 8.3 points [0.0, 20.8] for Luna. The direction is preserved despite different baseline ceilings. Each prompt contains exactly four tasks; the mean input is 2,518 characters for tags-only, 2,150 for grouped histories, and 2,420 for the ledger, so the state-ledger gain does not come from extra task packing or a longer prompt. On existing main transition slices, the ledger is exact on 100 tasks containing unknown→present, 98 containing unknown→absent, and 104 with permanent escalation.

Setting	Model	N	Raw	Raw+tags	Tags only	Grouped	State	State gain†
Public protocols: all	Haiku	32	71.9	75.0	–	62.5	100.0	+25.0
Elasticsearch Bulk	Haiku	8	62.5	75.0	–	62.5	100.0	+25.0
Microsoft Graph batch	Haiku	8	75.0	75.0	–	62.5	100.0	+25.0
Google Storage batch	Haiku	8	75.0	75.0	–	62.5	100.0	+25.0
JSON-RPC batch	Haiku	8	75.0	75.0	–	62.5	100.0	+25.0
Common main tasks	Haiku	24	–	83.3	79.2	70.8	100.0	+20.8
Common main tasks	GPT-5.6 Luna	24	–	91.7	91.7	95.8	100.0	+8.3

Table 4: Exact residual-action accuracy (%) beyond the original split assignment. The public replay imports response shapes and identity rules from four independent specifications. The common matrix gives both models the same 24 tasks and four-task packing. Derived state reaches 100% in every row. †Public gains use raw+tags; common-task gains use the matched tags-only view.

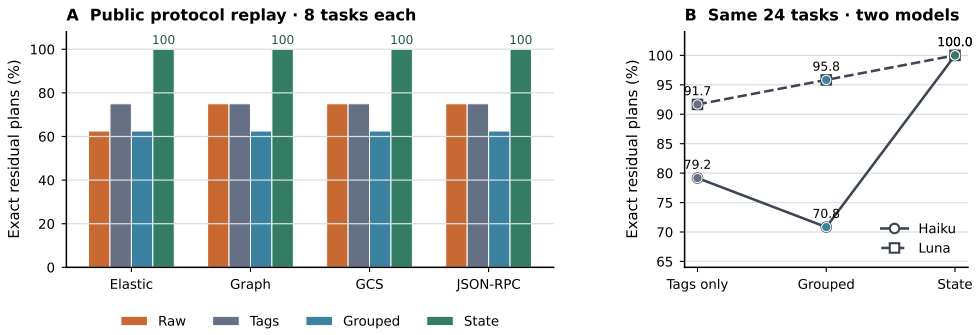

Figure 3: The state ledger is the only view that remains exact across every public protocol (left) and both models on the same 24 tasks (right). Bars show eight tasks per protocol; points compare call-ordered tags without raw text, item grouping, and derived state on 24 tasks per model. The common matrix fixes four tasks per prompt.

5.4 Local mechanism checks and efficiency

A 240-task deterministic suite varies later contradictory reports and removes one property at a time. Reading only the last call obtains 0% exact plans; overwriting settled state with the latest report obtains 1.2%; collapsing unknown into retry obtains 0.4%. The full monotone fold obtains 100%. These are implementation-level checks, not substitute model results: they show that history, uncertainty, and settlement monotonicity are each necessary to compute the defined residual plan.

The implementation processes 120,000 events over 40,000 items in 78.2 ms, or 1.53 million events/s, with 15.1 MiB peak traced memory. All model-facing experiments required $10.72 of directly metered Haiku inference and 1,701,898 Luna tokens. The public replay and same-task matrix added $2.51 of Haiku inference and 559,691 Luna tokens. All benchmark generation, exact scoring, bootstrap analysis, oracle validation, and scale sweeps run locally.

6 Related Work

Tool-using language agents. ReAct interleaves reasoning and actions (Yao et al., 2023), Toolformer learns when to call tools (Schick et al., 2023), ToolLLM scales instruction tuning over large API collections (Qin et al., 2024), and Gorilla connects models to broad API sets (Patil et al., 2024). AgentBench, τ-bench, ToolSandbox, AppWorld, BFCL, and MCPMark expand executable evaluation across interactive worlds and tool surfaces (Liu et al., 2024; Yao et al., 2024; Lu et al., 2025; Trivedi et al., 2024; Yan et al., 2024; Wu et al., 2026).

PartialCallBench isolates a different decision: what remains to be done after one batch call has several item-level outcomes.

State and context for agents. MemGPT treats context management as an operating-system problem (Packer et al., 2023), and Agentic Context Engineering evolves reusable context (Zhang et al., 2026). Speculative Actions overlaps independent work while preserving outcomes (Ye et al., 2026). EffectSet contributes an execution-specific context unit: a receipt-backed item state that survives call boundaries and directly determines residual obligations.

Distributed execution semantics. Transactions, sagas, and long-running workflows distinguish commit, compensation, and partial progress (Gray & Reuter, 1993; Garcia-Molina & Salem, 1987; Helland, 2007). EffectSet does not introduce a new transaction protocol. It imports their central state discipline into the agent interface: preserve confirmed effects, distinguish unknown from absent, and retry only the residual set.

7 Scope and Limitations

The study targets monotone, per-item batch effects with deterministic adapters and explicit item identities. This scope supports exact labels and a clean causal comparison of evidence views. Ordered dependencies between items, compensating actions, ambiguous identity resolution, and probabilistic postconditions require richer state machines than Equation 1. The controlled response grammars and domains are synthetic by design. PublicReplay-32 adds independently documented response contracts and adapter behavior, but remains a schema-faithful replay rather than sampled production traffic. The two model families cover distinct runners and context behavior; broader live-workflow studies can extend this envelope.

8 Conclusion

A partially successful batch call is a set of persistent item effects, not one scalar outcome. EffectSet makes that structure explicit. Across 1,496 model–task decisions, derived item state reaches 100% on the controlled main split, two models on identical tasks, four unseen grammars, and four public protocol families. The public replay adds 25.0 points over event tags, while the matched tags-only control identifies state derivation as the stable mechanism. EffectSet is a compact, auditable, and fast execution abstraction for exact residual planning after partial batch execution.

AI Use Statement

Generative AI assisted with implementation, experiment orchestration, analysis code, and language editing. The authors defined the study, validated every executable label, and verified all reported numbers. The anonymous artifact contains the implementation and supporting evidence.

References

Elastic. Bulk API. https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html, 2026. Accessed 2026-09-24.

Hector Garcia-Molina and Kenneth Salem. Sagas. In Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, pp. 249–259, 1987. doi: 10.1145/38713.38742.

Google Cloud. JSON API batch requests. https://cloud.google.com/storage/docs/batch, 2026. Accessed 2026-09-24.

Jim Gray and Andreas Reuter. Transaction Processing: Concepts and Techniques. Morgan Kaufmann, 1993.

Pat Helland. Life beyond distributed transactions: An apostate's opinion. In Conference on Innovative Data Systems Research, 2007.

JSON-RPC Working Group. JSON-RPC 2.0 specification: Batch. https://www.jsonrpc.org/specification#batch, 2013. Accessed 2026-09-24.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, et al. AgentBench: Evaluating LLMs as agents. In International Conference on Learning Representations, 2024.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Felix Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In Findings of the Association for Computational Linguistics: NAACL 2025, 2025. doi: 10.18653/v1/2025.findings-naacl.65.

Microsoft. Combine multiple HTTP requests using JSON batching. https://learn.microsoft.com/en-us/graph/json-batching, 2026. Accessed 2026-09-24.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In Advances in Neural Information Processing Systems, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, et al. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In International Conference on Learning Representations, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In Advances in Neural Information Processing Systems, 2023.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024.

Zijian Wu, Xiangyan Liu, Xinyuan Zhang, Lingjun Chen, Fanqing Meng, et al. MCPMark: A benchmark for stress-testing realistic and comprehensive MCP use. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=uobROwBsJm.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Berkeley function calling leaderboard. https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html, 2024.

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, 2023.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. arXiv preprint arXiv:2406.12045, 2024. URL https://arxiv.org/abs/2406.12045.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster AI agents. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=P0GOk5wslg.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, et al. Agentic context engineering: Evolving contexts for self-improving language models. In International Conference on Learning Representations, 2026. URL https://openreview.net/forum?id=eC4ygDs02R.

A Canonical Event Semantics

Canonical event	Item state	Permitted next action
applied / present / duplicate	settled	none
unknown	verify	read postcondition first
absent / retryable	retry	resend this item
permanent	escalate	surface irrecoverable item

Table 5: The adapter vocabulary used by every response grammar. Once settled, an item remains settled under all later events.

B Pilot and Additional Controls

Split	Model	Raw	Event tags	EffectSet	Gain over raw
Pilot	Haiku	80.0	100.0	100.0	+20.0
Pilot	GPT-5.6 Luna	85.0	87.5	100.0	+15.0

Table 6: The frozen 40-task pilot independently passed its 15-point expansion gate on both model families.

The main task-paired gain over event tags is 27.5 points with 95% interval [20.0, 35.8]. Confirmation yields 29.2 points [16.7, 41.7]. Resampling the inference batches instead of individual tasks yields intervals [15.0, 42.5] and [14.6, 45.8], respectively. All parse rates are 100%.

C Reproduction Map

```
python3 scripts/analyze_pilot.py --split main \
  --runs data/processed/main-v1/runs-haiku \
  --out data/processed/main-v1/analysis-haiku
python3 scripts/build_local_evidence.py
python3 scripts/validate_oracle.py
python3 scripts/validate_public_fixtures.py
python3 scripts/render_paper_assets.py
python3 -m pytest -q
```

The package preserves generated task JSON, prepared prompts, every raw runner response, normalized task predictions, exact task-level scores, paired and batch-blocked intervals, runtime/cost metadata, source code, tests, and the independent oracle audit.