TOOLTRIALS: AGENTS LEARN OPAQUE TOOL CONTRACTS BY EXPERIMENT

Anonymous authors
Paper under double-blind review

ABSTRACT

An agent can read a tool's schema and still choose the wrong action: labels such as `standard`, `compat`, and `strict` often name distinct operational contracts whose semantics are visible only through execution. Existing approaches rewrite documentation or accumulate task trajectories, but the examples they observe need not distinguish competing behaviors. We introduce TOOLTRIALS, which treats a new tool as an active system-identification problem. Given a finite executable candidate library containing the target and a legal query pool, it maintains candidate contracts, selects the input–policy trial that maximally partitions them, executes the trial, and compiles the identified mapping into a reusable behavioral card. Across 48 opaque implementations, eight behavior families, 288 held-out tasks, and 10,272 planned agent decisions, actively selected receipts reach 84.0–93.1% success versus 53.1–54.5% for passive observations. Compiling those receipts into cards reaches 98.3–99.0%, the oracle range. A matched control applies the identical compiler to model-designed probes and reaches 97.6–100.0%, localizing the downstream gain to verified contract compilation while TOOLTRIALS supplies deterministic, auditable probe selection. Three independent passive draws preserve 44.3–50.7-point card gains. A prospectively frozen non-randomized study identifies 18/18 installed Python, Node, Ruby, and GNU provider/mode contracts and predicts 90/90 held-out outputs; two random trials identify 52.4%. On a separately blinded natural-mode study, opaque case identifiers eliminate answer cues: cards solve 120/120 decisions across three model families, versus 105/120 from schema alone and 107/120 from fixed receipts. Deterministic transfer additionally verifies 480 GNU-wrapper actions and 756 two-tool plans. Two stored trials detect all 480 tested policy-mapping changes, while five-fold repetition preserves 98.5–98.6% identification under 10% output noise. TOOLTRIALS turns tool exploration from incidental trial-and-error into a small, planned, transferable experiment whose evidence and conclusion are executable.

1 INTRODUCTION

Consider a data agent calling `merge(current, patch, policy)`. The schema lists three legal values— `standard`, `compat`, and `strict`—but does not say whether null deletes a field, overwrites it, or leaves the current value untouched. All three calls type-check. Only one produces the requested state. A language model may attach a plausible prior to the labels, yet that prior is unrelated to the implementation currently installed.

This form of *operational opacity* appears whenever an interface names modes but leaves their exact behavior underspecified: compatibility policies, pagination conventions, tie-breaking rules, merge semantics, or version-dependent defaults. It differs from not knowing which API implements a function. The agent knows the tool, arguments, and return type; execution identifies which contract lies behind each legal mode. Our experiments show that schemas alone yield only 31.6–31.9% exact action success across randomized mappings of familiar labels to behaviors.

Recent systems improve tool use by compressing documentation (Yuan et al., 2025), gathering diverse trial trajectories (Qu et al., 2025; Fang et al., 2025), learning from task feedback (Hallinan et al., 2026), or training over large tool collections (Qin et al., 2024; Wang et al., 2026). These methods establish that interaction and experience are valuable. They leave open a more elementary question:

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.

which interaction should an agent perform when the purpose is to distinguish executable contracts? A convenient or naturally occurring example may produce the same output under every hypothesis and convey no information.

TOOLTRIALS answers this question with contrastive experiments. It represents the plausible contracts for a tool as hypotheses, predicts how each would respond to legal input–policy pairs, and executes the pair with maximum expected information gain. The observation removes inconsistent hypotheses. Once a single contract remains, the two compact input–output receipts and the resulting mode mapping form a versioned behavioral card. Any compatible agent can reuse the card without repeating the exploration or invoking a separate editor model.

We construct a controlled evaluation in which schemas, names, inputs, and output types are identical while operational behavior changes. Eight families span merge, filter, schedule, deduplication, pagination, quantization, batch, and join semantics. For each family, all six bijections between three policy labels and three behaviors produce 48 independent executable implementations and 288 held-out tasks. DeepSeek V4 Flash and GLM 5.3 Flash solve each task from a schema, two passive observations, a two-trial TOOLTRIALS card, or an oracle contract under matched prompts and action budgets.

We then move beyond the controlled executors. Eight additional families invoke the installed GNU implementations of `sort`, `head`/`tail`, `cut`, `grep`, `tr`, `paste`, `wc`, and `uniq`. This suite tests 480 held-out actions and 756 unique-output two-call plans emitted as free-form JSON. Unlike these deliberately permuted wrappers, a prospectively frozen natural-contract study builds candidates directly from installed Python, Node, Ruby, and GNU provider defaults or compatibility modes. Six families cover tie rounding, query parameters, path globbing, Base64 decoding, regular expressions, and version sorting. A separate audited agent study turns these natural contracts into 40 unique-answer tasks and adds OpenAI Luna as a third model family. Finally, controlled output corruption measures noisy identification, and replaying card trials after every alternative policy mapping measures detection of tested policy-mapping changes.

Our contributions are:

- **Active tool-contract identification.** Given a finite executable candidate library, TOOLTRIALS selects legal trials by their partition of operational contracts and compiles the result into a reusable, versioned card. Under the additional bijection and separating-input assumptions, two trials are sufficient and necessary for the three-mode class studied here.
- **Fresh, executable evidence.** The evaluation contains 48 controlled behavior twins, 48 GNU CLI-backed implementations, 18 non-randomized installed provider/mode contracts, 288 held-out controlled tasks, 480 GNU-wrapper tasks, 756 compositional plans, two main model families plus an audited third-model transfer, and complete call-level artifacts. The 10,272 planned LLM decisions measure action selection; GNU and composition results are labeled separately as deterministic executor verification, not agent planning.
- **A mechanism-resolving empirical result.** Two planned receipts improve LLM exact execution by 29.5–39.9 points over equally sized passive examples and reach 84.0–93.1% under the same call budget as validated self-designed probes. The matched compiler control shows that verified mappings—whether supported by model-designed or actively selected evidence—reach 97.6–100.0%; TOOLTRIALS uniquely fixes probe selection and identifies every controlled mapping within the tight two-call bound. Across three passive draws, cards retain 44.3–50.7-point gains. On authentic provider/mode contracts, active selection improves exact identification from 52.4% to 100%, predicts every held-out output, and yields 120/120 blinded natural-mode decisions across three models.

2 TOOL USE AS ACTIVE SYSTEM IDENTIFICATION

2.1 OPERATIONAL CONTRACTS

A tool exposes a schema S, a policy label $p \in \mathcal{P}$, an input $x \in \mathcal{X}$, and an executable response $y = f_h(p, x)$ governed by an unknown contract $h \in \mathcal{H}$. The hypotheses share S and differ only in operational behavior. In our main class, $|\mathcal{P}| = 3$ and each contract is one of the six bijections from

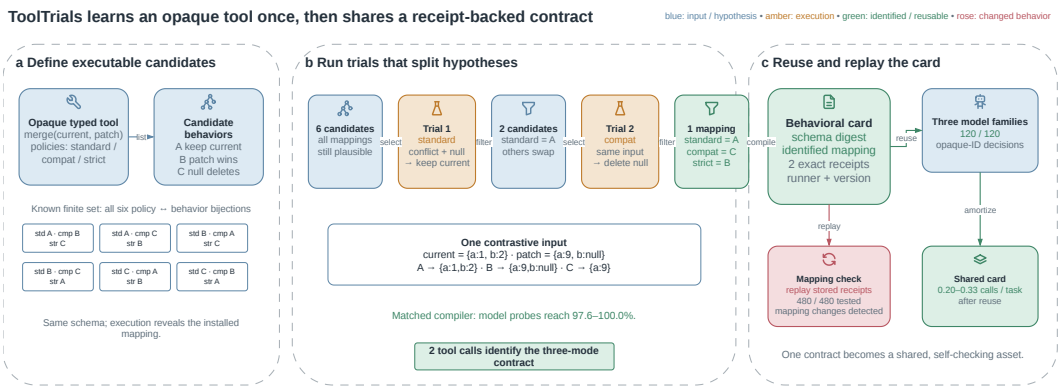

Figure 1: TOOLTRIALS turns an opaque interface into a reusable, receipt-backed contract. It enumerates executable hypotheses, chooses trials that split their predicted outputs, reduces the candidate set from $6 \to 2 \to 1$, and stores both the identified mapping and exact receipts. Two model families reuse the card at 98.3–99.0% held-out success; a matched compiler control exposes why, while the same selector identifies all 18 authentic installed provider/mode contracts and replay detects all 480 tested policy-mapping changes.

Component	General selector needs	Tight two-call result additionally needs
Candidates	finite executable library containing the target	three behaviors and a bijective label mapping
Queries	legal calls with comparable outputs	one input separating all behaviors
Observations	output comparator and update rule	deterministic, equal-cost calls

Table 1: What TOOLTRIALS assumes and what it discovers. Candidate construction is an input to the selector; the installed contract and contrastive trials are discovered.

policy labels to three concrete behaviors. Candidate behaviors can be assembled from documented alternatives, version histories, adapter implementations, or maintainer-provided compatibility modes. TOOLTRIALS identifies the running contract from this constrained, executable candidate set.

A trial is $q = (p, x)$. Given surviving hypotheses H_t, its predicted outputs induce a partition

$$\Pi(q, H_t) = \big\{\{h \in H_t : f_h(p, x) = y\} : y \in \mathcal{Y}\big\}. \tag{1}$$

With a uniform prior, TOOLTRIALS selects the trial maximizing partition entropy,

$$q_t^* = \arg\max_q - \sum_{G \in \Pi(q, H_t)} \frac{|G|}{|H_t|} \log_2 \frac{|G|}{|H_t|}, \quad H_{t+1} = \{h \in H_t : f_h(q_t^*) = y_t\}. \tag{2}$$

The query pool contains legal, locally executable calls. Costs or side effects can be included as penalties; our benchmark calls are deterministic and equal-cost, so entropy alone determines the choice.

Why two trials? Suppose one separating input produces a distinct output for each of B behaviors. A trial on one policy reveals its assigned behavior. Because the mapping is bijective, $B-1$ trials reveal the final assignment by elimination. Fewer than $B-1$ trials leave at least two unqueried policies exchangeable. Thus the three-behavior class has a tight two-trial identification bound. TOOLTRIALS discovers these separating trials from predicted partitions rather than receiving them as annotations.

2.2 FROM OBSERVATIONS TO A REUSABLE CARD

The card stores the tool and implementation identifier, schema digest, each selected input–policy pair and exact output, the surviving contract, and runner metadata. The human-readable view names the operational modes, for example `standard=keep-current-on-conflict`. The machine

Suite	Families	Impl.	Tasks/plans	Executions	Unit / role
Behavior twins	8	48	288	4,608 LLM	8-family inference
Independent repeats	8	48	288	3,456 LLM	8-family stability
Passive resamples	8	48	288	1,728 LLM	8-family stability
Probe selection	8	48	–	38,400 local	exhaustive local
Public GNU CLI	8	48	480	385k local	exact local
Provider/mode defaults	6	18	90	36k selections	6-family census
Blinded natural agents	6	18	40	480 LLM	40 paired tasks
Two-tool plans	7 pairs	252	756	2,268 local	exact local
Noise study	16	96	–	230,400 trials	simulation
Version changes	16	96	–	480 changes	exhaustive replay

Table 2: Evaluation scale. "Impl." counts independent policy mappings; two-tool implementations are ordered mapping pairs. LLM counts are planned denominators, with missing answers scored as failures. Families are the primary independent units; local rows are deterministic verification and do not enter inferential claims.

view retains the receipts that support this statement. Downstream agents receive the card as ordinary context, while the system can replay the receipts when its tested policy mapping changes. This separation is useful: identification is deterministic, and language models focus on selecting an action for the user's requested output.

For non-deterministic tools, the same query can be repeated and the modal observation used for filtering. Section 4.5 measures this extension under controlled output corruption.

3 Evaluation

3.1 Research questions and evaluation suites

RQ1: Do planned trials improve agent action selection? The behavior-twin suite contains eight tool families: merge, filter, schedule, dedupe, paginate, quantize, batch, and join. Each has three plausible behaviors and all six policy-to-behavior permutations, yielding 48 implementations. We reserve six inputs per implementation for evaluation. Each task specifies an exact target output and offers the three legal policy calls plus no-call in shuffled order. Exactly one action executes to the target. Trial inputs never appear as held-out task inputs.

We compare eight matched conditions. **Schema** gives the interface; **Passive** gives two random observations; **Active receipts** gives two TOOLTRIALS-selected outputs; **ToolTrials** adds their deterministic mapping; and **Oracle** gives the mapping without receipts. In **Self-explore**, each model designs two probes per family; schema validation exposes only missing argument keys. **Self-explore+compiler** applies the exact TOOLTRIALS filter and renderer to those frozen probes, changing only probe selection. **Explore+rewrite** instead asks a second inference pass to map the same observations, a matched DRAFT/Play2Prompt-style loop under the same two-call budget. DeepSeek V4 Flash and GLM 5.3 Flash receive identical tasks and are scored by execution, without a model judge. The eight-condition matrix has 4,608 planned decisions (4,582 parsed); two repeat runs add 3,456 (3,432 parsed). Every missing answer is a failure. Raw-event audit excludes five tool-using GLM schema batches and one DeepSeek self-explore batch; clean repairs replace four, while two nonreturning repairs count as 24 failures. All prompts, events, outputs, timing, tokens, and cost remain archived.

Three frozen passive draws per model add 1,728 planned decisions under a predeclared $2 stop; two further prompt sets remain preserved but unrun.

RQ2: Does active selection use probes efficiently? For every controlled implementation, we compare TOOLTRIALS at budgets 0–4 with 200 uniformly shuffled random-probe sequences per budget. Identification requires a singleton hypothesis set. RQ1 inference treats the eight behavior families as the independent units: 10,000 family-bootstrap samples form 95% intervals, and all 2^8 paired sign assignments give exact two-sided tests. Implementation-clustered estimates remain a secondary artifact diagnostic, never the headline test.

Model	Schema	Passive	Probe receipts	Probe+compiler	Rewrite	Active receipts	TOOLTRIALS	Oracle
DeepSeek V4 Flash	31.9	53.1	85.1	**100.0**	84.7	93.1	98.3	98.3
GLM 5.3 Flash	31.6	54.5	91.3	97.6	96.9	84.0	**99.0**	96.9

Table 3: Exact held-out action success (%) over the planned 288 decisions per cell. Missing outputs count as failures. Applying the same deterministic compiler to either selected or model-designed identifying probes reaches the oracle range.

RQ3: Do contracts transfer to public tools and composition? Eight GNU wrappers cover ordering, line selection, fields, matching, case mapping, joining, counting, and uniqueness. All six hidden mappings and ten held-out inputs per wrapper yield 480 exact-output tasks comparing cards, conventional labels, oracle, and random trials.

Seven ordered tool pairs add 756 unique-output compositions. A model-free planner renders each card-implied policy pair as free-form JSON and success requires exact execution to the target.

A prospectively frozen study removes randomized aliases. Its 18 targets are installed Python, Node, Ruby, and GNU defaults or compatibility modes across tie rounding, query parsing, path globbing, Base64 decoding, regex interpretation, and version sorting. Six legal calls per family support no-trial, two-fixed, 2,000 random-sequence, and TOOLTRIALS comparisons. Five disjoint calls per family test the identified candidate's held-out outputs directly.

Forty unique-answer natural-mode tasks add DeepSeek, GLM, and OpenAI Luna under schema, fixed-receipt, TOOLTRIALS, and oracle conditions (480 decisions). Each semantic case name is replaced by a salted opaque digest before prompts are assembled, and task order is independently frozen. Tool-disabled sessions run from an empty directory; raw-event scans gate every call, and missing batches remain failures.

RQ4: Are cards robust and useful after updates? We independently corrupt each probe output with probability $\epsilon \in \{.05, .10, .20\}$ by substituting another currently plausible output. Across 200 trials for every implementation, we compare one, three, or five repeated executions per selected query with majority aggregation. For drift, each learned card is replayed against every other policy mapping in its family: 96 originals times five alternatives gives 480 tested policy-mapping changes.

4 RESULTS

4.1 TWO PLANNED TRIALS RECOVER ORACLE-LEVEL AGENT PERFORMANCE

Table 3 gives the central result. Without receiving an inferred mapping, active receipts reach 93.1% for DeepSeek and 84.0% for GLM, gains of 39.9 and 29.5 points over two passive examples. Both conditions execute two legal calls and show only their exact outputs; trial selection is the sole difference. Family-clustered 95% intervals remain positive: [33.0, 49.0] and [16.7, 41.0], with exact eight-family $p = .0078$ and .0156. Compiling the same receipts into a named mapping contributes a further 5.6 and 14.9 points, separating the benefit of information acquisition from card rendering.

Schema validation makes self-exploration a strong baseline. Both models initially omit the required `values` object for quantization; those outputs are archived, and a retry that reveals only the required key yields legal probes for 8/8 families and identifies 48/48 mappings. Held-out success from the raw probe receipts is 85.1% and 91.3%; TOOLTRIALS cards improve on those receipts by 13.2 and 7.6 points, respectively.

The matched compiler control resolves the mechanism. Applying the same candidate filter and card renderer to the frozen model-designed probes yields 100.0% for DeepSeek and 97.6% for GLM. Relative to this control, TOOLTRIALS is -1.7 points for DeepSeek (family CI $[-3.5, -0.3]$, exact $p = .25$) and $+1.4$ for GLM (CI $[-0.7, 4.5]$, $p = .75$). Thus near-oracle action selection follows when identifying evidence is compiled into a verified mapping. TOOLTRIALS's added contribution is to choose that evidence deterministically: it identifies every mapping in the tight two-call budget, with fixed execution and audit semantics rather than a model-generated probe set.

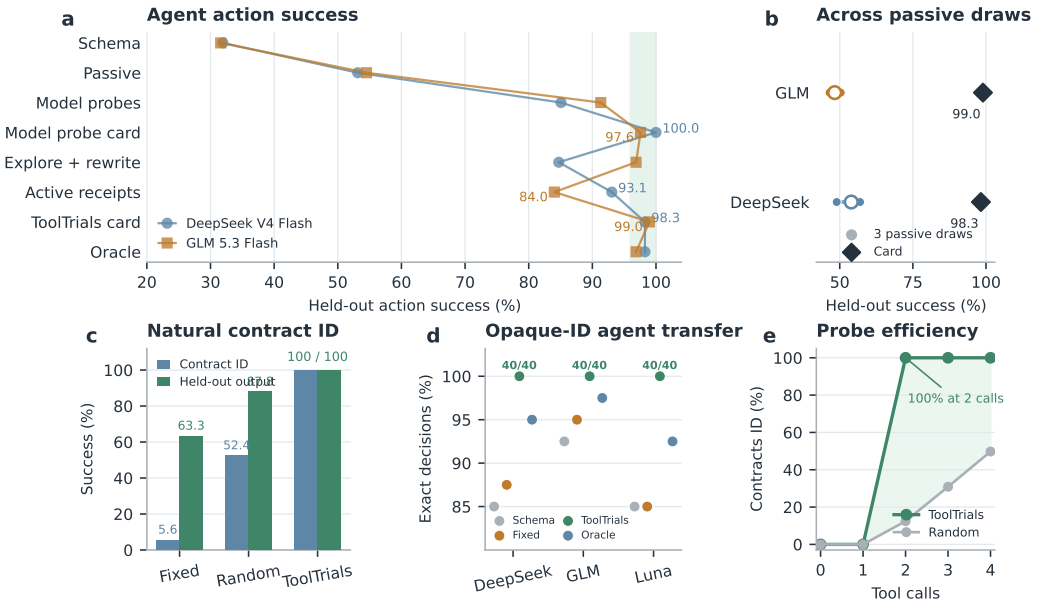

Figure 2: Planned experiments convert opaque modes into stable, near-oracle behavior. (a) The complete evidence ladder shows that the identical compiler turns either model-designed or actively selected identifying probes into near-oracle cards. (b) TOOLTRIALS exceeds every passive draw by 41.3–52.5 points. (c) On 18 installed provider/mode contracts, active selection identifies and predicts 100%. (d) With answer-bearing case names replaced by opaque IDs, cards solve 40/40 decisions for each of three models and outperform schema and fixed receipts. (e) Controlled mappings are all identified in two calls.

Method / budget	0	1	2	3	4
Random: singleton identified	0.0	0.0	12.4	30.9	49.8
TOOLTRIALS: singleton identified	0.0	0.0	**100.0**	100.0	100.0
Random CLI: task success	33.8	50.3	66.1	82.0	93.4
TOOLTRIALS CLI: task success	33.8	–	**100.0**	–	–

Table 4: Probe efficiency (%). The first two rows cover 48 controlled implementations and 9,600 random sequences per budget; the lower rows execute 480 GNU CLI tasks.

Explore+rewrite reaches 84.7% for DeepSeek and 96.9% for GLM. TOOLTRIALS improves by 13.5 points for DeepSeek (family CI [3.5, 24.7], $p = .0625$) and is within 2.1 points for GLM (CI $[-1.0, 5.6]$, $p = .375$). It deterministically recovers all 96 mappings; the rewrite recovers 86. TOOLTRIALS also matches the DeepSeek oracle and is 2.1 points above the GLM oracle (CI $[-2.1, 8.3]$, $p = .875$), while exceeding 94.4% in every model–family cell.

Across three complete passes, TOOLTRIALS spans 98.3–99.7% for DeepSeek and 99.0–99.3% for GLM (population SD 0.6 and 0.2 points); active receipts span 92.7–94.8% and 82.6–86.8%. The primary 283/288 and 285/288 card scores include every missing output as failure, separating parser failures from executable but wrong actions.

Across three frozen passive draws, DeepSeek averages 53.9% (49.0–56.9%) and GLM 48.3% (46.5–50.3%). Cards improve by 44.3 and 50.7 mean points and beat each best draw by 41.3 and 48.6 points (Figure 2b).

4.2 CONTRASTIVE TRIALS IDENTIFY WHAT RANDOM TRIALS LEAVE AMBIGUOUS

One contrastive query reduces the six candidates to two; a second identifies the contract. Random selection frequently spends its budget on an input for which all behaviors agree. After four calls

Method	CLI actions	Two-tool plans	Calls per plan
Schema-default mapping	33.8	10.2	2
TOOLTRIALS card	**100.0**	**100.0**	2
Oracle contract	100.0	100.0	2

Table 5: Exact execution success (%) on 480 held-out GNU-wrapper actions and 756 free-form JSON plans. ToolTrials cards match the oracle in every family and pair.

Method	Contract ID	Held-out output	Mean calls
No trial	0.0	57.8	0.00
Two fixed examples	5.6	63.3	1.94
Two random trials	52.4	87.8	1.71
TOOLTRIALS	**100.0**	**100.0**	**1.33**

Table 6: Exact identification and output prediction (%) on 18 non-randomized, installed provider/mode contracts. Random selection averages 36,000 frozen target–sequence runs; every method has a two-call maximum.

it identifies only 49.8% of controlled contracts. The GNU-wrapper suite shows the downstream consequence: two random probes yield 66.1% task success, while two TOOLTRIALS probes yield 100%. Even four random probes remain 6.6 points below the two-call active card.

4.3 CARDS TRANSFER TO GNU-BACKED WRAPPERS AND TWO-TOOL PLANS

Table 5 extends the finding from in-process behavior twins to GNU-backed executables behind randomized labels. The same procedure identifies all 48 mappings and executes all 480 held-out tasks. Performance is 100% in each of the eight wrapper families, including cases where the modes correspond to materially different programs, such as adjacent `uniq`, global first-occurrence filtering with `awk`, and sorted global uniqueness with `sort -u`.

Deterministic composition amplifies mapping uncertainty: a conventional mode prior must be correct twice, and succeeds on 10.2% of unique-output plans. The two identified cards produce 756/756 correct action sequences over seven ordered pairs. Each output is obtained by executing the emitted policies in order, rather than matching a multiple-choice label. This result verifies downstream planning sufficiency: the learned mappings support exact composition without special composition demonstrations.

4.4 AUTHENTIC PROVIDER AND MODE DIFFERENCES REMOVE RANDOMIZED ALIASES

Table 6 directly tests the algorithm outside policy permutations. The candidates are executable behaviors already present in Python 3.11, Node 22, Ruby 3.1, GNU grep 3.8, and GNU sort 9.1. TOOLTRIALS identifies all 18 targets and predicts all 90 disjoint outputs exactly. It needs one call when a single query creates three output groups and two calls for binary-output families such as path globbing, hence 1.33 calls on average. Random two-call selection identifies 52.4%; two ordinary examples identify only 5.6%. The 47.6-point identification gain shows that candidate construction from real implementations does not remove the need for contrastive query selection.

The blinded natural-mode study removes semantic answer cues from every case identifier. Across DeepSeek, GLM, and Luna, TOOLTRIALS cards solve 120/120 planned decisions, versus 105/120 from schema alone, 107/120 from fixed ordinary receipts, and 114/120 from the oracle mapping. Every model returns all 40 decisions. The card improves over schema by 15.0, 7.5, and 15.0 points with 15 paired wins and no losses across models; DeepSeek and Luna each have a paired 95% bootstrap interval of [5.0, 27.5] points, while GLM's is [0.0, 17.5]. This opaque-ID result turns the transfer study into a direct card-benefit test rather than a cue-following task (Figure 2d).

Domain	Output noise	1 execution	3 executions	5 executions
Behavior twins	5%	90.4	98.7	99.9
Behavior twins	10%	80.9	95.1	**98.6**
Behavior twins	20%	63.6	82.0	90.3
GNU wrappers	5%	89.9	98.8	99.8
GNU wrappers	10%	80.3	95.2	**98.5**
GNU wrappers	20%	63.9	82.5	90.2

Table 7: Contract identification (%) under independent output corruption. Repeating the same selected trial preserves its contrastive value while majority aggregation restores the observation. Each cell contains 9,600 runs.

4.5 Repeated trials tolerate noise; stored trials detect drift

Five-fold repetition raises identification at 10% output noise from 80.9% to 98.6% for behavior twins and from 80.3% to 98.5% for GNU-backed wrappers (Table 7). This is a direct cost–reliability knob: deterministic tools use two calls, while a noisy tool can use six or ten calls without changing the query selection rule.

The same receipts make cards self-checking. Replaying their two selected inputs detects 240/240 alternative mappings in the controlled suite and 240/240 in the GNU-wrapper suite. Because two observed policy assignments determine a three-way permutation, every changed mapping alters at least one receipt. A tool registry can therefore invalidate the old card with two local calls and rerun identification only when needed.

We also test candidate-set misspecification. We replace one policy with either a novel output or a duplicate, non-bijective behavior in every controlled and CLI mapping. A full separating-input confirmation across the three policies detects all 576/576 cases, so the system fails explicitly instead of silently publishing the nearest card. A four-mode extension identifies all 24 bijections in the tight three-query budget; random probing identifies 78.8% even after six calls.

4.6 Cost and amortization

The confirmatory two-model matrix, active-receipt and self-exploration ablations, explore-and-rewrite baseline, and two full repeat runs cost $2.836 in model inference; the preceding positive pilot costs $0.312. Round 6 adds the initial natural-agent study, three passive draws per model, and validated baseline updates for $1.507, below its predeclared $2 stop. The blinded transfer, audited control repairs, and matched compiler add 102 calls for $0.449, below their preregistered $1.20 stop. Contract identification, CLI transfer, composition, noise, and drift checks are local executions. A two-call card is reused across six held-out tasks per controlled implementation and ten per GNU-wrapper implementation, amortizing to 0.33 and 0.20 exploratory calls per task respectively. More reuse drives this cost toward zero. The complete experimental ledger is $14.516 because it also preserves a $9.412 invalidated Gemini run; unavailable and invalid calls remain visible rather than being silently removed from accounting.

The card prompt is 1.51× longer than the oracle prompt across 144 frozen batches, so near-oracle accuracy is not a shorter-prompt effect; Appendix D reports the distribution.

5 Analysis

A contrastive input matters more than a plausible input. For merge, a fresh key is realistic but non-identifying. ToolTrials combines a conflict, a null, and a fresh key, making keep-current, patch-wins, and null-deletes yield distinct outputs. Two policy calls determine the third by elimination. The same principle separates pagination, deduplication, and join semantics.

Candidate libraries can come from executable implementation choices. The workflow collects executable alternatives, normalizes I/O, enumerates legal calls, and defines exact comparison. The natural study uses 18 bindings, 36 probe fixtures, and 30 held-out fixtures without a target label;

generic code handles selection, filtering, confirmation, and storage. Candidate coverage remains upstream work: absent targets are rejected and equivalent candidates remain non-singletons.

The card changes the model's job. Without it, a model must infer both label meaning and the action reaching the goal. TOOLTRIALS resolves the first problem before any task, leaving transparent action selection. Filtering and evaluation remain model-independent, so one card transfers across providers and its provenance is inspectable as two receipts rather than unsupported prose.

6 RELATED WORK

Tool-using language agents. ReAct and Toolformer interleave action or learn invocation (Yao et al., 2023; Schick et al., 2023); ToolLLM and MCP-Flow scale tool coverage (Qin et al., 2024; Wang et al., 2026); ToolEyes evaluates broad tool competence (Ye et al., 2025). TOOLTRIALS instead identifies which executable contract a specific installed tool implements.

Learning tool documentation through experience. EasyTool condenses documentation (Yuan et al., 2025); DRAFT, Play2Prompt, and ToolObserver learn descriptions from experience (Qu et al., 2025; Fang et al., 2025; Hallinan et al., 2026). TOOLTRIALS adds the missing experimental-design layer: exact candidate partitions yield a verifiable two-call contract and reusable change-detection receipts. Against a matched trial–feedback–rewrite loop, it improves DeepSeek by 13.5 points and matches GLM within 2.1 points.

Active learning and system identification. TOOLTRIALS instantiates active learning and system identification (Settles, 2009; Ljung, 1999): executable contracts are hypotheses and legal calls are interventions, yielding auditable selection without training.

7 DEMONSTRATED SCOPE AND EXTENSIONS

The study establishes deterministic, low-cost identification for finite policy libraries: TOOLTRIALS supplies probe selection, a tight bound for the studied class, confirmation for two misspecification classes, repetition under controlled noise, and auditable receipts. The matched control localizes near-oracle accuracy to compilation once probes identify the contract. Headline gains cover two models with blinded Luna transfer, while GNU and composition tests establish executable mapping and downstream planning sufficiency. Candidate generation, stateful trials, continuous parameters, cost-aware design, and open-ended task completion define the direct extensions beyond this finite-library scope. Raw-event scans exclude tool-using runs and incomplete clean repairs remain scored as failures, preserving the stated evaluation boundary.

8 CONCLUSION

TOOLTRIALS chooses executions that separate plausible behaviors. Two receipts raise action success to 84.0–93.1%; verified cards reach 98.3–99.0% and beat every passive draw by at least 41.3 points. A matched compiler turns independently designed identifying probes into equally strong cards, pinpointing the reusable mechanism while TOOLTRIALS fixes the experiment itself. The selector identifies 18/18 installed contracts, predicts 90/90 held-out outputs, and supports 120/120 blinded natural-mode decisions across three models. Its contracts are executable conclusions: reusable, inspectable, and cheap to revalidate.

AI USE STATEMENT

Generative AI tools assisted with research ideation and methodology feedback, literature discovery, implementation and editing of research code, drafting portions of the manuscript, and language polishing. The authors selected the final methodology, inspected and tested AI-assisted code, reran and checked the experiments, verified numerical results and citations against source artifacts, and reviewed all AI-assisted text. The authors take responsibility for the paper's text, claims, results, and released artifacts.

REFERENCES

Wei Fang, Yang Zhang, Kaizhi Qian, James Glass, and Yada Zhu. PLAY2PROMPT: Zero-shot tool instruction optimization for LLM agents via tool play. In *Findings of the Association for Computational Linguistics: ACL*, 2025.

Skyler Hallinan, Thejas Venkatesh, Xiang Ren, Sai Praneeth Karimireddy, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. OpaqueToolsBench: Learning nuances of tool behavior through interaction. *arXiv preprint arXiv:2602.15197*, 2026.

Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, 2 edition, 1999.

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

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. From exploration to mastery: Enabling LLMs to master tools via self-driven interactions. In *International Conference on Learning Representations*, 2025.

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.

Burr Settles. Active learning literature survey. *Computer Sciences Technical Report 1648, University of Wisconsin–Madison*, 2009.

Wenhao Wang, Peizhi Niu, Zhao Xu, Zhaoyu Chen, Jian Du, Yaxin Du, Xianghe Pang, Keduan Huang, Yanfeng Wang, Qiang Yan, and Siheng Chen. MCP-Flow: Facilitating LLM agents to master real-world, diverse and scaling MCP tools. In *Proceedings of the Association for Computational Linguistics*, 2026.

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.

Junjie Ye, Guanyu Li, Songyang Gao, Caishuang Huang, Yilong Wu, Sixian Li, Xiaoran Fan, Shihan Dou, Tao Ji, Qi Zhang, Tao Gui, and Xuanjing Huang. ToolEyes: Fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. In *International Conference on Computational Linguistics*, 2025.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. EASYTOOL: Enhancing LLM-based agents with concise tool instruction. In *Proceedings of NAACL-HLT*, pp. 951–972, 2025. doi: 10.18653/v1/2025.naacl-long.44.

A Behavior Families and Contracts

Family	Three candidate behaviors
Merge	keep current on conflict; patch wins including null; patch wins and null deletes
Filter	include listed fields; exclude listed fields; mask listed fields
Schedule	higher priority first; lower priority first; preserve input order
Dedupe	keep first per key; keep last per key; keep highest score per key
Paginate	zero-based offset; one-based offset; one-based page number
Quantize	round down; round up; round nearest
Batch	all eligible; first eligible; last eligible
Join	inner join; left outer join; left anti join
GNU sort	lexicographic; stable numeric; version-aware
GNU selection	first n lines; last n lines; all but last n lines
GNU fields	first field; second field; all except first field
GNU match	case-sensitive match; inverse match; case-insensitive match
GNU case	uppercase; lowercase; swap case
GNU paste	comma; colon; space delimiter
GNU count	lines; words; bytes
GNU unique	adjacent runs; global first; sorted global

Table 8: All evaluated behavior families. Every family instantiates all six bijections between three public policy labels and these three concrete behaviors.

B Authentic Candidate Libraries

Family	Executable candidates	Active calls
Tie rounding	Python `round`; Node `Math.round`; Ruby `round`	1.00
Query parameters	Python `parse_qs`; WHATWG object conversion; Ruby `CGI.parse`	1.00
Path globbing	Python `fnmatch`; Ruby default; Ruby pathname mode	1.67
Base64 decoding	Python lenient; Python strict; Node `Buffer`	1.67
Regex interpretation	GNU grep fixed; basic; extended	1.67
Version sorting	GNU sort lexical; numeric; version	1.00

Table 9: The non-randomized study's installed candidate libraries. Active calls are averaged across the three targets; every target is identified within the two-call cap.

The frozen query pool has six calls per family. Ordinary examples occupy indices 0–1; indices 2–5 contain documented edge cases such as half ties, repeated/blank parameters, dotfiles and path separators, missing/invalid Base64 padding, regular-expression metacharacters, and version-like numbers. Candidate outputs are generated by the installed runtime rather than encoded as answer labels. The complete selected receipts, runtime versions, random sequences, and 90 held-out outputs are included in the artifact.

C Prompt and Evaluation Controls

Every condition receives the same schema, current input, exact required output, and four shuffled actions. The passive, self-explore, self-explore+compiler, explore-and-rewrite, active-receipts, and ToolTrials conditions receive exactly two executed observations. Active receipts omit the inferred mapping; the self-explore observations come from two probes designed once per model and family, with one schema-only retry when the argument object is invalid; self-explore+compiler applies the same deterministic candidate filter and card grammar used by ToolTrials; explore-and-rewrite adds an LLM-generated mapping from those observations; ToolTrials renders the deterministically filtered mapping; the oracle renders that mapping without receipts. Batches contain 12 independent tasks. The parser accepts only a JSON object covering every case in a batch. Malformed batches are preserved; the original main run attempted one single-task repair, and the validated baseline permits

one format-only retry for two malformed quantization batches. The scorer resolves the selected label to its concrete policy, executes the target tool, and compares the exact output to the requested output.

D Prompt-Length Control

Across 144 frozen batches per condition, the receipt-backed card prompt averages 9,851 characters (range 7,332–11,332), versus 6,505 (5,400–6,866) for the oracle mapping. The card is therefore 1.51× longer; its near-oracle result cannot be attributed to receiving a shorter prompt. Exploratory-call amortization is distinct from token compression.

E Additional Main Results

Model	Condition	Batch	Dedupe	Filter	Join	Merge	Paginate	Quantize	Schedule
DeepSeek	Schema	38.9	16.7	33.3	47.2	16.7	36.1	33.3	33.3
DeepSeek	Passive	66.7	50.0	66.7	55.6	33.3	33.3	52.8	66.7
DeepSeek	ToolTrials	94.4	100	100	100	100	100	97.2	97.1
GLM	Schema	55.6	72.2	30.6	33.3	30.6	33.3	27.8	63.9
GLM	Passive	61.1	63.9	69.4	44.4	36.1	36.1	58.3	66.7
GLM	ToolTrials	100	97.2	100	100	100	100	100	94.4

Table 10: Exact task success (%) by model and behavior family.

F Invalidated Gemini Smoke Test

The preregistered expansion gate ran Gemini 3.5 Flash on four families and 48 tasks per condition. Its apparent scores were 100% for schema, ToolTrials, and oracle, and 91.7% for passive examples. Trace audit found that all four schema sessions and two passive sessions invoked OpenCode filesystem tools; schema sessions directly read the raw benchmark containing target mappings and answers. We therefore invalidate the entire smoke test and draw no model comparison from it. In contrast, an exhaustive event scan additionally found tool events in five retained GLM schema batches and one DeepSeek self-explore batch. Those runs are excluded and replaced only by clean, tool-disabled reruns; two incomplete GLM reruns count as failures. The new blinded natural transfer and matched-compiler runs have zero tool events. The invalid Gemini run's $9.412 cost and all superseded traces remain in the artifact ledger.

G Proof of the Probe Bound

Proposition. Let B named policies be bijectively mapped to B behaviors, and let an input x^* produce pairwise-distinct outputs under those behaviors. Identifying the mapping needs and suffices to use $B - 1$ single-policy trials in the worst case.

Proof. For sufficiency, execute x^* under any $B - 1$ policies. Each distinct output reveals one policy–behavior assignment, and bijectivity determines the unqueried assignment. For necessity, after fewer than $B - 1$ single-policy trials at least two policies remain unqueried. Swapping the behaviors assigned to those policies produces a different bijection that is observationally identical on every completed trial. Thus no method can distinguish all mappings. □

H Artifact Map and Reproduction

The review artifact at `https://anonymous.4open.science/r/tooltrials` contains the raw benchmark, prompts, events, scores, executables, drift trials, scripts, tests, and accounting. Principal commands are:

```
python3 scripts/analyze_main.py
```

```
python3 scripts/analyze_repeats.py
python3 scripts/analyze_identification.py
python3 scripts/run_cli_transfer.py
python3 scripts/run_authentic_contracts.py
python3 scripts/analyze_robustness.py
python3 -m pytest -q tests
```