EXECUTION AUTHORITY UNDER INTERFACE DRIFT: VERIFIED SELECTIVE MIGRATION OF TOOL CALLS

Anonymous authors
Paper under double-blind review

ABSTRACT

A tool-using agent can construct a valid rewrite of an action without evidence that the rewrite remains authorized to execute. This gap arises when an API or Model Context Protocol (MCP) contract changes after a call has been issued: a revised call may remain schema-valid even though its operation or argument roles are no longer uniquely determined. We formulate migration of an already-issued call as selective execution. CONTRACTPATCH executes a migrated call only when an independent verifier can reconstruct a unique correspondence from the old contract, new contract, and issued call; otherwise, it abstains. Unlike schema validation or checks against invented values, this certificate establishes why values retain their roles across versions. The verifier recomputes the evidence rather than trusting the proposed rewrite and requires no language-model tokens. Across 105 real-schema cases, seven language models recover many supported migrations yet still produce target-valid unsafe executions; a generic verifier reduces but does not remove these failures without sacrificing coverage. CONTRACTPATCH safely migrates all 60 adaptable calls, correctly refuses all 45 unsupported calls, and produces zero observed unsafe executions. The frozen method preserves zero observed unsafe execution across held-out REST and MCP settings, an untouched Notion release pair, 20,000 fuzz trials, and 28 end-to-end executions. On Notion, it safely automates 73.3% of adaptable calls and refuses cases whose correspondence the contracts do not establish. The results separate learned competence to propose a plausible action from the evidence required to authorize its execution.

1 INTRODUCTION

Schema validity does not establish authorization continuity. A call issued under one API or Model Context Protocol (MCP) contract (Model Context Protocol Contributors, 2025) may encounter a renamed field, a changed identifier role, or a split operation at execution time. Its rewrite can satisfy the revised JSON Schema (Wright et al., 2022) while changing meaning: an old `database_id` may fit `data_source_id` even when the contracts do not equate their identifier domains, and several replacement tools may accept the same arguments without identifying a unique branch. The runtime then possesses a well-formed action but lacks evidence that the action retains the authority of the issued call. This execution-authority gap under interface drift is not directly addressed by existing work on tool generation, policy enforcement, or API migration.

Existing research addresses how agents discover tools, construct arguments, and generate structured calls (Li et al., 2023; Qin et al., 2024; Wang et al., 2025; Qin et al., 2025; Banerjee et al., 2025; Lu et al., 2025); whether calls comply with policies over tool sequences or data flows (Xie et al., 2025; Doshi et al., 2026); and how software adapts to breaking API changes (Lamothe et al., 2021; Di Lauro et al., 2022). MCPEvol-Bench evaluates replanning and task completion as MCP toolsets evolve (Liu et al., 2026). These objectives do not establish whether the operation and live argument roles of an already-issued call remain uniquely recoverable across two contracts. A fresh plan may be useful, and an external policy may permit it, without proving this cross-version continuity.

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.

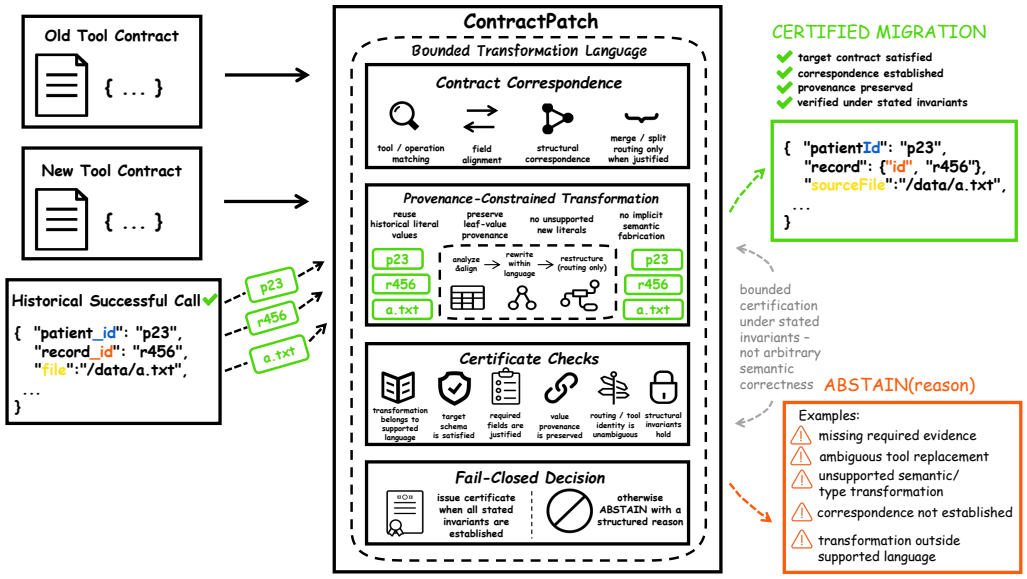


Figure 1: Authorization continuity under schema drift. The learned component proposes an action; CONTRACTPATCH searches a bounded transformation language over the old contract, new contract, and historical call. Independently verified correspondence, provenance, completeness, and validity certificates authorize execution, while unresolved ambiguity forces abstention.

To our knowledge, this is the first formulation of migration for an already-issued agent tool call across versioned tool contracts as a selective execution problem: the runtime must derive a uniquely supported replacement or abstain. CONTRACTPATCH operationalizes this criterion through bounded candidate construction and independent certificate verification. The evaluation distinguishes safe automation, unjustified execution, and conservative refusal across real REST and MCP evolution, frozen provider transfer, executable endpoints, and adversarial perturbations. This execution-level view exposes a risk–coverage boundary hidden by aggregate migration accuracy: a system can propose most supported rewrites correctly yet remain unreliable at the boundary where the contracts cease to justify one.

2 AUTHORIZATION CONTINUITY UNDER CONTRACT DRIFT

The method follows the structure of the authorization question. It first identifies what must remain continuous across contracts, then specifies which contract evidence can witness that continuity, and finally separates candidate construction from the decision to authorize execution. The algorithm is a reference realization of this evidence structure.

2.1 PROBLEM STRUCTURE

Let a tool be $t = (n, d, \Sigma)$, where n is its name, d its natural-language description, and Σ its argument schema. A contract is a finite set $\mathcal{T} = \{t_k\}_{k=1}^{K}$, and $\mathrm{Valid}_\Sigma(a) \in \{0, 1\}$ denotes complete JSON-Schema validation of argument object a. An agent produces a call $c = (n, a)$ against an old contract $\mathcal{T}^-$. At execution time, only a revised contract $\mathcal{T}^+$ may be available. Schema drift can rename, add, remove, or change fields and can also split one tool into several tools or merge several tools into one. A migration system must therefore return either a revised call or an explicit refusal:

$$F(c, \mathcal{T}^-, \mathcal{T}^+) \in \{\textsc{Adapt}(c^+), \textsc{Abstain}\}. \tag{1}$$

This setting demands selective execution. The system executes migrations that the contracts and observed call determine uniquely, and abstains whenever the available evidence admits multiple correspondences or none.

We distinguish two forms of unsafe execution. Let y be the ground-truth decision, $\hat{y}$ the system decision, and $\mathrm{Correct}(c^+)$ the semantic correctness of an executed call:

$$U_{\mathrm{M}} = \#\{y = \textsc{Adapt}, \hat{y} = \textsc{Adapt}, \neg\,\mathrm{Correct}(c^+)\}, \tag{2}$$

$$U_{\mathrm{A}} = \#\{y = \textsc{Abstain}, \hat{y} = \textsc{Adapt}\}, \tag{3}$$

$$U = U_{\mathrm{M}} + U_{\mathrm{A}}. \tag{4}$$

U_{M} is essential because a model may correctly decide that a case is migratable while constructing the wrong executable call. A metric that counts only adaptations on ground-truth abstention cases misses this failure mode.

Authorization as a proof obligation. Let $e = (c, \mathcal{T}^-, \mathcal{T}^+)$ denote the runtime evidence. A candidate call is not authorized merely because it is plausible or valid under the new schema. It must belong to the set

$$\mathcal{A}(e) = \{c^+ : \exists C\ \mathsf{Accept}(C, c) = 1\ \wedge\ C \text{ reconstructs } c^+ \text{ from } e\}. \tag{5}$$

Selective migration requires this set to contain exactly one call:

$$F(e) = \begin{cases} \textsc{Adapt}(c^+), & \mathcal{A}(e) = \{c^+\}, \\ \textsc{Abstain}, & \text{otherwise.} \end{cases} \tag{6}$$

Existence rules out unsupported rewrites; uniqueness rules out tool splits, merges, and field changes with several contract-compatible interpretations. Abstention is therefore the semantic result of an unmet proof obligation, not a confidence threshold applied to generated text.

This formulation separates search from authority. A constructor $G(e) = (\tilde{c}, C)$ may use any bounded search strategy to propose a witness, but an independently implemented verifier $V(e, \tilde{c}, C) \in \{0, 1\}$ recomputes every acceptance condition from the contracts and observed call. The execution path uses $\tilde{c}$ only when $V = 1$ and the authorized set is a singleton. Constructor scores, generation probabilities, and explanations never enter the authorization decision. This separation permits better proposal mechanisms without enlarging the set of calls that the contract evidence can authorize.

2.2 Contract evidence and certificate structure

The certificate constructor produces a bounded witness $C = (\tau, \rho, D, \delta)$. Here, τ identifies one target tool, ρ is a partial one-to-one field correspondence, D contains certified true deletions, and δ contains bounded values derived from contract evidence. Migration succeeds only when the certificate is unique and an independent verifier accepts the resulting call.

Let P^- and P^+ be the source and selected-target field sets, and let $\mathcal{L}(\mathcal{T}^-, \mathcal{T}^+)$ be the finite set of literals licensed by declared enums, quoted contract literals, and the source tool identity. The certificate acceptance predicate is

$$\begin{aligned} \mathsf{Accept}(C, c) = {} & \mathsf{UniqueTool}(\tau) \wedge \mathsf{Reciprocal}(\rho) \wedge \mathsf{Complete}(\rho, D; a) \\ & \wedge \mathsf{Bounded}(\delta; \mathcal{L}) \wedge \mathrm{Valid}_{\Sigma_\tau}(a^+) \wedge \mathsf{Conserve}(a, a^+; C). \end{aligned} \tag{7}$$

The system returns ADAPT if and only if the independently recomputed predicate in Equation 7 is true.

The constructor uses a finite, deterministic evidence extractor rather than a semantic model. It forms tool candidates from contract-visible identity, deprecation, branch, operation-family, and description cues, and retains a target only when the evidence selects it uniquely. Field candidates require compatible locations and types and a reciprocal unique lexical match. An unmatched source field is licensed as a deletion only when no plausible target replacement exists; a missing required target value may be introduced only from a uniquely selected contract literal or source-tool discriminator. These tests use normalized string and schema comparisons. Their role is deliberately limited: they instantiate the evidence relation but do not confer execution authority.

Malformed contracts, unsupported schema constructs, ambiguous matches, unlicensed value introduction, and budget violations all produce ABSTAIN. Algorithm 1 gives the complete construction order. The independently recomputed predicate in Equation 7, not the constructor or its scores, authorizes execution.

Algorithm 1 CONTRACTPATCH: proof-carrying selective migration

Require: Call $c = (n, a)$; old and new contracts $\mathcal{T}^-, \mathcal{T}^+$
Ensure: ADAPT(c^+) or ABSTAIN

if PREFLIGHT$(\mathcal{T}^-, \mathcal{T}^+)$ fails **then**
  return ABSTAIN
end if
$t^- \leftarrow$ RESOLVESOURCE$(n, \mathcal{T}^-)$
$K \leftarrow$ TOOLCANDIDATES$(t^-, a, \mathcal{T}^+)$
if $|K| \neq 1$ **then**
  return ABSTAIN
end if
$\tau \leftarrow$ the unique element of K
$\rho \leftarrow$ RECIPROCALFIELDMAP(t^-, τ)
$D \leftarrow \varnothing$; $a^+ \leftarrow \{\}$
for all live fields $i \in \mathrm{dom}(a)$ **do**
  if $i \in \mathrm{dom}(\rho)$ **then**
    $a^+_{\rho(i)} \leftarrow a_i$
  else if TRUEDELETION(i, t^-, τ) **then**
    $D \leftarrow D \cup \{i\}$
  else
    return ABSTAIN
  end if
end for
for all missing required fields j of τ **do**
  $v \leftarrow$ BOUNDEDDERIVE(j, t^-, τ)
  if $v = \perp$ **then**
    return ABSTAIN
  end if
  $\delta(j) \leftarrow v$; $a^+_j \leftarrow v$
end for
$C \leftarrow (\tau, \rho, D, \delta)$; $c^+ \leftarrow (\tau.n, a^+)$
if INDEPENDENTVERIFY$(C, c, c^+, \mathcal{T}^-, \mathcal{T}^+)$ **then**
  return ADAPT(c^+)
end if
return ABSTAIN

2.3 VERIFICATION AND GUARANTEE

The migration procedure constructs a candidate call and certificate. A separate verifier discards the constructor's correspondence claims and recomputes them from the two contracts. It requires (i) a unique licensed target, (ii) correspondence completeness for every live source argument, (iii) full target-schema validity, and (iv) multiset conservation of observed literal values after removing only licensed identity or composed-discriminator values. JSON-Schema validity establishes structural executability. Correspondence and value conservation establish that construction neither discards nor rebinds an observed value.

Let $V(a)$ denote the multiset of primitive leaf values in argument object a, V_D the values belonging to certified deletions, and V_δ the licensed derived values. The verifier's conservation predicate is

$$V(a^+) \setminus V_\delta = V(a) \setminus V_D, \tag{8}$$

where subtraction denotes multiset subtraction. The verifier also checks equality at the field locations reconstructed from ρ; multiset equality alone would permit a value swap between compatible fields.

Theorem 1 (Runtime certificate soundness)**.** *If $F(c, \mathcal{T}^-, \mathcal{T}^+) =$ ADAPT(c^+), then there is a unique certified target τ such that $\mathrm{Valid}_{\Sigma_\tau}(a^+) = 1$; every live source field is accounted for by exactly one reciprocal mapping or a certified true deletion; $\mathrm{range}(\delta) \subseteq \mathcal{L}$; and Equation 8 holds. Moreover, the procedure terminates and returns* ABSTAIN *on every failed or unsupported check within the stated finite contract budgets.*

Proof sketch. Tool selection returns only after a uniqueness check. Field construction copies values only through reciprocal correspondences, while any plausible but uncertified replacement invalidates deletion. Required target fields are filled only by bounded derivation. The independent verifier recomputes these relations, validates the complete target schema, and compares residual source and target value multisets. Acceptance therefore implies every conjunct of Equation 7, which establishes Theorem 1. The loops in Algorithm 1 range only over budgeted finite tool and field sets, and the fail-closed exception boundary establishes totality.

The theorem conditions intent preservation on the semantics expressed by the two contracts. Semantics absent from both contracts and the observed call cannot enter a contract-derived certificate. Such cases reach ABSTAIN. The verified certificate procedure consumes no language-model tokens and averages 5.46 ms per call on the main benchmark.

3 EXPERIMENTS AND RESULTS

3.1 EXPERIMENTAL DESIGN

Benchmarks. The 105-case main benchmark contains 60 provable adaptations and 45 required abstentions from Slack (61), Stripe (41), and MCP-filesystem (3) (Slack, 2021; Stripe, 2026; Model Context Protocol Contributors, 2026). It covers requiredness tightening (20), optional-field addition (33), type change (18), field removal (15), shape replacement (3), required-field addition (3), tool merge (10), and tool split (3). Transfer evaluation uses 40 held-out GitHub REST cases (27 adaptations) together with GitHub MCP and MCP Git merge and value-variation cases (GitHub, 2026b;a). An untouched pair of official Notion MCP releases, 1.9.0 and 2.5.0, yields 21 cases (15 adaptations, 6 abstentions) from nine call-bearing drift operations, including nested shape, union, and enum changes and tool replacements with or without a proven identifier correspondence (Notion, 2026).

Baselines and metrics. We evaluate two 3B-scale open models, Qwen2.5-3B-Instruct and Granite-3.0-3B-A800M-Instruct (Qwen Team, 2024; Granite Team, IBM, 2024), and five API models: DeepSeek-V3, GLM-5.3, GPT-5.6-sol, Claude Fable 5, and Grok 4.6 (DeepSeek-AI, 2024; GLM-5 Team, 2026; OpenAI, 2026; Anthropic, 2026; xAI, 2026).

Each model is paired with the same generic post-hoc verifier, which checks target-schema validity and newly invented literals but not reciprocal contract correspondence. All methods receive the same compact call-relevant projection of the old and new schemas: tool names, descriptions, nested argument structure, requiredness, types, and enums. Raw generation and verification share a fixed decision-and-call format and fixed provider-specific decoding; API or parse failures receive no safe-execution credit. The certificate procedure contributes one record per case. Appendix Table 6 supplies the unabridged-contract control. Execution precision is the fraction of executed calls that are safe. Oracle-normalized safe coverage is the number of safely executed calls divided by the number of ground-truth adaptations. Over-abstention counts provably migratable cases that are refused. Since 45 of 105 main cases should be refused, the oracle overall automation rate is

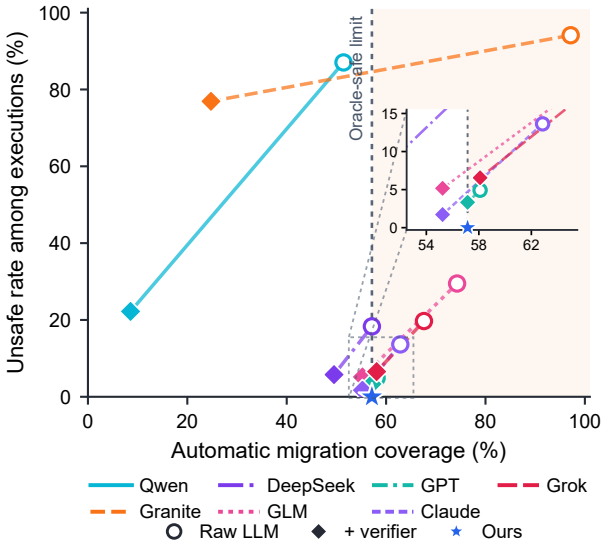


Figure 2: Risk–coverage frontier on 105 cases. Segments connect raw and verified outcomes; the dashed line marks the oracle safe-automation limit, and the inset magnifies the strongest models.

$60/105 = 57.1\%$, while its oracle-normalized coverage is 100%.

3.2 MAIN BENCHMARK RESULTS

Safe migration rises sharply with model capability. DeepSeek-V3 reaches 49 of 60 adaptable cases; GPT-5.6-sol reaches 58. Unsafe execution nevertheless persists. GPT-5.6-sol executes three unsafe calls among 61 attempted adaptations, while Claude Fable 5 and Grok 4.6 execute nine and fourteen. The remaining errors cluster at the decision boundary between a supported transformation and a merely plausible one.

Table 1: Execution-level results on the 105-case main benchmark (60 ADAPT, 45 ABSTAIN). "+V" denotes the generic verifier. Unsafe includes wrong calls on migratable cases and executions on cases that should be refused.

Model	Method	Safe	Unsafe	Over	Precision	Norm. coverage
Qwen2.5-3B	Raw	7	47	16	13.0%	11.7%
	+V	7	2	53	77.8%	11.7%
Granite-3.0-3B	Raw	6	96	0	5.9%	10.0%
	+V	6	20	34	23.1%	10.0%
DeepSeek-V3	Raw	49	11	11	81.7%	81.7%
	+V	49	3	11	94.2%	81.7%
GLM-5.3	Raw	55	23	5	70.5%	91.7%
	+V	55	3	5	94.8%	91.7%
GPT-5.6-sol	Raw	58	3	2	95.1%	96.7%
	+V	58	2	2	96.7%	96.7%
Claude Fable 5	Raw	57	9	3	86.4%	95.0%
	+V	57	1	3	98.3%	95.0%
Grok 4.6	Raw	57	14	3	80.3%	95.0%
	+V	57	4	3	93.4%	95.0%
Certificate-based method		**60**	**0**	**0**	**100.0%**	**100.0%**

The generic verifier removes many unsafe calls, but its operating point depends on the generator. For Qwen2.5-3B, unsafe execution falls from 47 to 2 as over-abstention rises from 16 to 53. The six stronger generators retain one to four unsafe executions after verification. CONTRACTPATCH executes all 60 supported migrations and refuses all 45 unsupported cases. Exact paired McNemar tests reject equality with raw Claude Fable 5 ($p = 0.000488$) and raw Grok 4.6 ($p = 0.0000153$); the GPT-5.6-sol comparison yields $p = 0.0625$. Figure 3 resolves the same progression within each model and drift category.

The repeated failures share three contract structures. Type changes can remain target-valid without providing a value conversion; tool splits can offer several plausible branches without a unique selector; and tool merges can require a discriminator that does not occur among the call arguments. These cases preserve surface plausibility while withholding the correspondence witness needed for execution. The same distinction recurs across model families in Figure 3.

3.3 PROVIDER AND DRIFT GENERALIZATION

CONTRACTPATCH produces zero observed unsafe executions throughout Figure 4; exact counts appear in Appendix Table 4. The 50-call MCP Git expansion comprises five independent drift operations instantiated with ten value configurations each. All 40 adaptable calls execute safely. The untouched Notion pair contains a stricter boundary: 11 of 15 adaptable calls execute, and all six unsupported calls are refused. Two nested `database`-to- `data_source` enum rewrites and two create-database-to-create-data-source replacements lack a licensed semantic correspondence. They account for the four over-abstentions and the 73.3% normalized coverage.

Claude Fable 5 with generic verification reaches the same Notion operating point as CONTRACTPATCH: 11 safe executions and no unsafe execution. The other five verified models retain two to six unsafe calls. Verified Claude also retains one unsafe call on the main benchmark. In contrast,

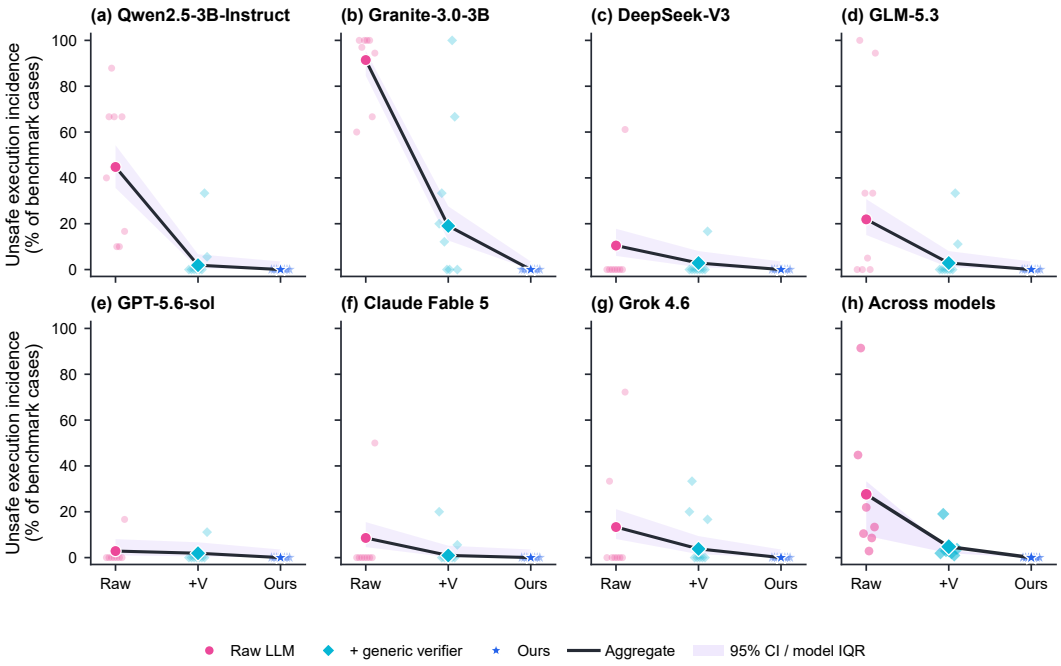


Figure 3: The same unsafe-execution pattern recurs across model families and drift categories. Points denote category-level observations, lines denote the aggregate change from raw generation through generic verification to CONTRACTPATCH, and the aggregate panel includes the across-model range. Generic verification reduces but does not uniformly eliminate unsafe execution; certificate verification reaches zero observed unsafe execution for every category.

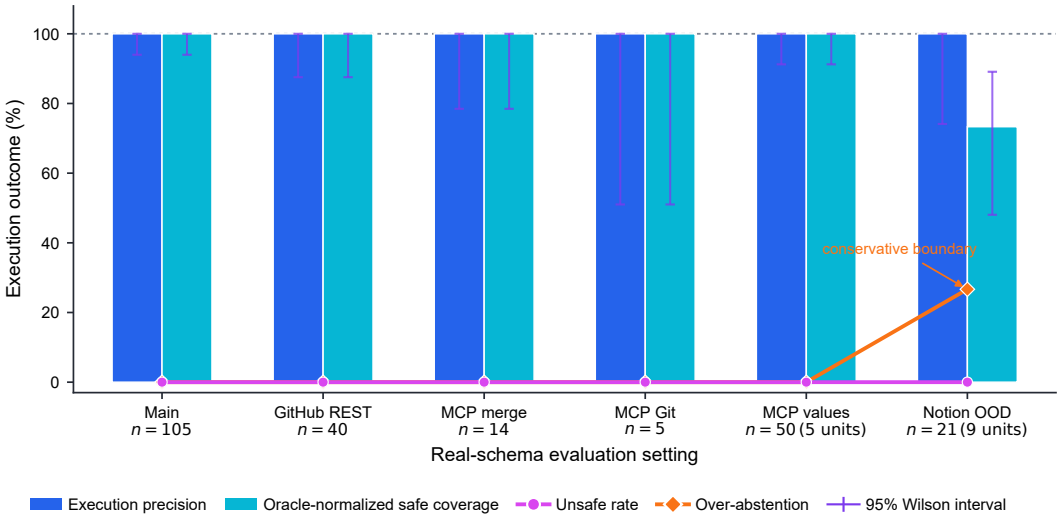


Figure 4: Performance across real-schema settings. Bars jointly report execution precision and oracle-normalized safe coverage; the overlaid lines report unsafe execution and over-abstention. The frozen method preserves execution precision across providers. Untouched Notion drift lowers safe coverage to 73.3% while execution precision remains 100%.

the certificate rule remains invariant to the upstream model. Qwen2.5-3B produces one unparsable output; all API-model runs complete without parse errors.

Table 2: Full-model comparison on the untouched Notion MCP OOD set (15 ADAPT, 6 ABSTAIN). Granite is excluded because the complete 19-to-24-tool surface exceeds its native 4,096-token context and yielded no parseable final calls.

Model	Method	Safe	Unsafe	Precision	Norm. coverage
Qwen2.5-3B	Raw	5	14	26.3%	33.3%
	+V	5	2	71.4%	33.3%
DeepSeek-V3	Raw	9	6	60.0%	60.0%
	+V	9	4	69.2%	60.0%
GLM-5.3	Raw	12	8	60.0%	80.0%
	+V	12	6	66.7%	80.0%
GPT-5.6-sol	Raw	5	3	62.5%	33.3%
	+V	5	2	71.4%	33.3%
Claude Fable 5	Raw	11	2	84.6%	73.3%
	+V	11	0	100.0%	73.3%
Grok 4.6	Raw	9	5	64.3%	60.0%
	+V	9	4	69.2%	60.0%
Certificate-based method		**11**	**0**	**100.0%**	**73.3%**

3.4 MECHANISM, ROBUSTNESS, AND END-TO-END EXECUTION

Generic verification fails when a call preserves literals but changes their roles. On Notion, models copy an observed `database_id` literal into `data_source_id`. Such a call is target-schema valid and invents no literal, but the contract does not prove the identifier domains equivalent:

$$\text{SchemaValidity} + \text{NoNewValues} \not\Rightarrow \text{SafeMigration}. \tag{9}$$

Reciprocal correspondence rejects this misbinding. The 14-case GitHub MCP merge set exhibits the dual requirement: the target needs a new operation discriminator derived from tool identity. Argument-only provenance rejects every derived discriminator; bounded identity provenance certifies all 14 migrations.

Across 20,000 fuzz trials, the method produces no exception, invalid adaptation, invented value, or silent semantic loss. The trials contain approximately 1,800 composed merges, 1,750 field renames, and 1,740 ambiguous renames. Two seeds independently exercise the same generator distribution. All 13 migrated GitHub REST calls pass live HTTP execution, response-schema checks, and semantic checks. All 15 MCP-filesystem steps pass stateful execution and postcondition checks. The execution tests verify response schemas, task semantics, and state transitions after migration. Together with the Notion refusals, they locate the empirical boundary at contract-supported semantics: accepted correspondences execute safely in the observed trials, while absent correspondences reduce coverage.

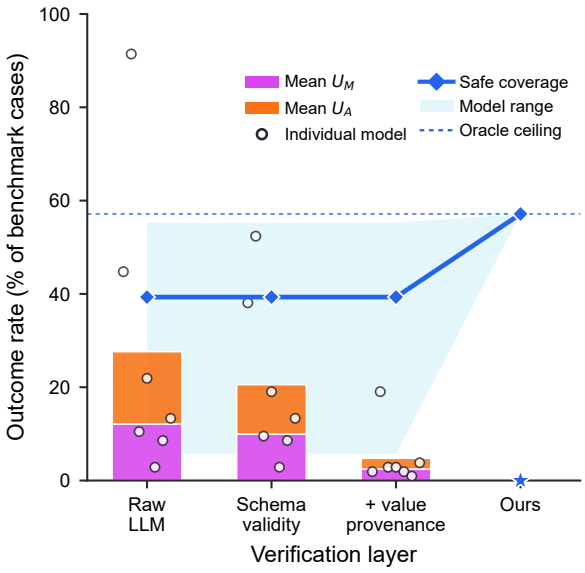


Figure 5: Failure decomposition by verification layer. Stacked bars show U_M and U_A, points denote individual models, and the line tracks safe coverage. Schema validity and value provenance do not establish contract correspondence.

4 DISCUSSION

Model scale primarily improves proposal competence. GPT-5.6-sol, Claude Fable 5, and Grok 4.6 each recover at least 57 of 60 supported migrations, yet their residual unsafe executions concentrate at the same boundary: a tool split lacks a branch witness, a type change lacks a conversion, or two identifiers share a representation without a certified domain correspondence. Syntax and plausible semantics can support a candidate at this boundary while the contracts do not uniquely authorize it. Certificate verification complements learned construction by resolving this execution decision.

Transfer results show how the available evidence controls automation. On the untouched Notion releases, the frozen method safely automates 11 of 15 adaptable calls (73.3%), refuses all six unsupported calls, and produces no observed unsafe execution. The four missed adaptations require nested enum substitution or operation replacement across identifier domains, transformations outside the frozen certificate language. Their inclusion would require new construction rules together with independently checkable witnesses. Within this scope, Table 3 records 12,995 accepted fuzz adaptations and 28 REST or stateful MCP endpoint executions, all with zero observed failures.

Table 3: Robustness and actual execution checks. Fuzzing uses an independent oracle; endpoint tests additionally check response schemas and task-level semantics or state transitions.

Protocol	Trials/steps	Adapt	Abstain	Observed failures
Independent-oracle fuzz, seed 17	10,000	6,615	3,385	0
Independent-oracle fuzz, seed 29	10,000	6,380	3,620	0
GitHub REST endpoint execution	13	13	0	0
MCP filesystem stateful execution	15	15	0	0
Unit and regression tests	23	–	–	0

The unabridged-contract control in Appendix Table 6 leaves every CONTRACTPATCH decision and output unchanged, while the two model baselines shift in opposite directions. Schema verbosity therefore offers no consistent safety gain for learned migration. ContractPatch certifies only properties visible in the two contracts and the issued call; undocumented business rules, general nested restructuring, many-to-many remapping, and semantic type conversion remain outside its present scope. Provider-supplied equivalence declarations and typed conversion witnesses could admit these transformations while preserving independent verification. Evaluations over additional temporal releases and providers can further characterize how this evidence-supported coverage changes as tool ecosystems evolve.

5 CONCLUSION

Schema drift exposes a failure mode that target-schema validation cannot detect: an executable rewrite may no longer represent the operation and argument roles authorized by the issued call. CONTRACTPATCH treats migration as selective execution and admits a revised call only when the versioned contracts uniquely justify its target operation and every live value correspondence. Across real contracts, frozen transfer settings, and executable endpoints, this criterion eliminates observed unsafe executions while refusing cases where the available evidence runs out. Reliable agentic execution therefore separates two responsibilities: learned models construct candidate actions, while independently checkable contract evidence authorizes their execution.

AI USE STATEMENT

Language models are experimental subjects in this work and were also used to assist implementation, analysis, and manuscript editing. All algorithms, labels, numerical claims, citations, and final prose were checked against the released artifacts and source documents by the authors.

REPRODUCIBILITY STATEMENT

The submission includes the frozen algorithm, formal specification, benchmark construction scripts, schema snapshots, case-level outputs, metric code, plots, and executable tests. Every reported ag-

gregate is recoverable from saved case-level records. The artifact separates development regressions from post-freeze held-out and untouched OOD evaluations, records independent drift units separately from repeated calls, and provides the prompts and provider runners required to reproduce model baselines when credentials are available.

REFERENCES

Anthropic. Claude Fable 5 and Claude Mythos 5, June 2026. URL https://www.anthropic.com/news/claude-fable-5-mythos-5.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. CRANE: Reasoning with constrained LLM generation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2836–2857. PMLR, 2025.

DeepSeek-AI. DeepSeek-V3 technical report, 2024.

Fabio Di Lauro, Souhaila Serbout, and Cesare Pautasso. To deprecate or to simply drop operations? an empirical study on the evolution of a large OpenAPI collection. In *Proceedings of the 16th European Conference on Software Architecture*, pp. 38–46. Springer, 2022. doi: 10.1007/978-3-031-16697-6_3.

Aarya Doshi, Yining Hong, Congying Xu, Eunsuk Kang, Alexandros Kapravelos, and Christian Kästner. Towards verifiably safe tool use for LLM agents. In *Proceedings of the IEEE/ACM 48th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 201–205. Association for Computing Machinery, 2026. doi: 10.1145/3786582.3786839.

GitHub. GitHub MCP Server, 2026a. URL https://github.com/github/github-mcp-server.

GitHub. GitHub REST API OpenAPI description, 2026b. URL https://github.com/github/rest-api-description.

GLM-5 Team. GLM-5: From vibe coding to agentic engineering, 2026.

Granite Team, IBM. Granite 3.0 Language Models, October 2024. URL https://github.com/ibm-granite/granite-3.0-language-models.

Maxime Lamothe, Yann-Gaël Guéhéneuc, and Weiyi Shang. A systematic review of API evolution literature. *ACM Computing Surveys*, 54(8):171:1–171:36, 2021. doi: 10.1145/3470133.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.187.

Huanxi Liu, Kun Hu, Jiaqi Liao, Qiang Wang, Pengfei Qian, Yuanzhao Zhai, Dawei Feng, Bo Ding, and Huaimin Wang. MCPEvol-Bench: Benchmarking LLM agent performance across dynamic evolutions of MCP servers, 2026.

Yaxi Lu, Haolun Li, Xin Cong, Zhong Zhang, Yesai Wu, Yankai Lin, Zhiyuan Liu, Fangming Liu, and Maosong Sun. Learning to generate structured output with schema reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4905–4918. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.243.

Model Context Protocol Contributors. Model context protocol specification, revision 2025-06-18, 2025. URL https://modelcontextprotocol.io/specification/2025-06-18.

Model Context Protocol Contributors. Filesystem MCP server, 2026. URL https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem. npm package version 2026.7.10.

Notion. Official Notion MCP Server, 2026. URL https://github.com/makenotion/notion-mcp-server.

OpenAI. GPT-5.6 system card, July 2026. URL https://deploymentsafety.openai.com/gpt-5-6.

Shengqian Qin, Yakun Zhu, Linjie Mu, Shaoting Zhang, and Xiaofan Zhang. Meta-tool: Unleash open-world function calling capabilities of general-purpose large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30653–30677. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1481.

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 *The Twelfth International Conference on Learning Representations*, 2024.

Qwen Team. Qwen2.5 technical report, 2024.

Slack. Slack API specifications, 2021. URL https://github.com/slackapi/slack-api-specs.

Stripe. Stripe's OpenAPI specification, 2026. URL https://github.com/stripe/openapi.

Jun Wang, Jiamu Zhou, Xihuai Wang, Xiaoyun Mo, Haoyu Zhang, Qiqiang Lin, Cheng Jin, Muning Wen, Weinan Zhang, Qiuying Peng, and Jun Wang. HammerBench: Fine-grained function-calling evaluation in real mobile assistant scenarios. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3350–3376. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-acl.175.

Austin Wright, Henry Andrews, Ben Hutton, and Greg Dennis. JSON Schema: A media type for describing JSON documents, draft 2020-12, June 2022. URL https://json-schema.org/draft/2020-12/json-schema-core.html.

xAI. Introducing Grok 4.6, August 2026. URL https://x.ai/news/grok-4-6.

Yuejin Xie, Youliang Yuan, Wenxuan Wang, Fan Mo, Jianmin Guo, and Pinjia He. ToolSafety: A comprehensive dataset for enhancing safety in LLM-based agent tool invocations. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 14135–14156. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.714.

A Expanded experimental protocol

Case construction and ground truth. Each case contains an old contract, a new contract, one historical call, an expected decision and, for adaptable cases, the expected target call. Cases are grounded in official versioned schema differences and executable endpoint semantics. A case is labelled ADAPT only when the target tool and every live value correspondence are determined by the recorded contracts and call; otherwise the correct decision is ABSTAIN. Ground-truth checking compares the full target tool and argument object in addition to the binary decision. Consequently, U_M can be nonzero when a model predicts ADAPT but constructs the wrong call.

Frozen OOD protocol. The transformation language and thresholds were locked before selecting and extracting the final Notion MCP release pair. The release pair was processed through the same schema-diff and case-construction pipeline without adding provider-specific aliases or rules. The four adaptable Notion calls outside the language remain over-abstentions. The Azure set supplied a development counterexample and therefore serves as a regression set.

Execution grading. For a ground-truth ADAPT case, an execution is safe only if the emitted target call exactly satisfies the expected semantic transformation and passes the target schema. A wrong executed call contributes to U_M. For a ground-truth ABSTAIN case, any executed adaptation contributes to U_A. Refusing an adaptable case is over-abstention; refusing a non-migratable case is correct abstention. Thus

$$\text{ExecutionPrecision} = \frac{N_{\text{safe}}}{N_{\text{safe}} + U_M + U_A}, \tag{10}$$

$$\text{NormalizedCoverage} = \frac{N_{\text{safe}}}{N_{\text{GT-adapt}}}. \tag{11}$$

B ADDITIONAL ANALYSES

Table 4: The frozen method across real-schema and mechanism tests. "Units" counts independent drift operations when this differs from the number of calls.

Benchmark	Cases	Units	GT adapt	Safe	Unsafe	Correct abstain	Norm. cov.
Main Slack/Stripe/MCP	105	105	60	60	0	45	100.0%
GitHub REST held-out	40	40	27	27	0	13	100.0%
GitHub MCP merge	14	14	14	14	0	0	100.0%
MCP Git original	5	5	4	4	0	1	100.0%
MCP Git value expansion	50	5	40	40	0	10	100.0%
Notion MCP untouched OOD	21	9	15	11	0	6	73.3%

Table 5: Composition of the main real-schema benchmark.

Drift category	Cases	GT adapt	GT abstain
Requiredness tightened	20	10	10
Optional field added	33	33	0
Type changed	18	0	18
Field removed	15	15	0
Field-shape replacement	3	0	3
Required field added	3	0	3
Tool merge	10	2	8
Tool split	3	0	3
Total	105	60	45

Table 6: Effect of schema verbosity on unsafe execution. Compact and full inputs use the same 105 cases, instructions, decoding configuration, and grading; Δ is full minus compact.

Method	Compact schemas	Full contracts	Δ
DeepSeek-V3	11	9	-2
GPT-5.6-sol	3	4	$+1$
CONTRACTPATCH	0	0	0

Full contracts change DeepSeek-V3 from 11 to 9 unsafe executions and GPT-5.6-sol from 3 to 4. All 13 full-contract failures are target-valid type-change adaptations on abstention cases; they therefore occur after parsing and schema validation, at the migration decision itself. The compact and full contract inputs produce identical decisions and calls for CONTRACTPATCH. The 44-case Azure development regression contains 43 safe adaptations, no unsafe execution, and one correct abstention. A repeated GLM-5.3 run agrees with the original binary decision on 104 of 105 cases; unsafe execution changes from 23 to 22 before verification and remains 3 after verification.

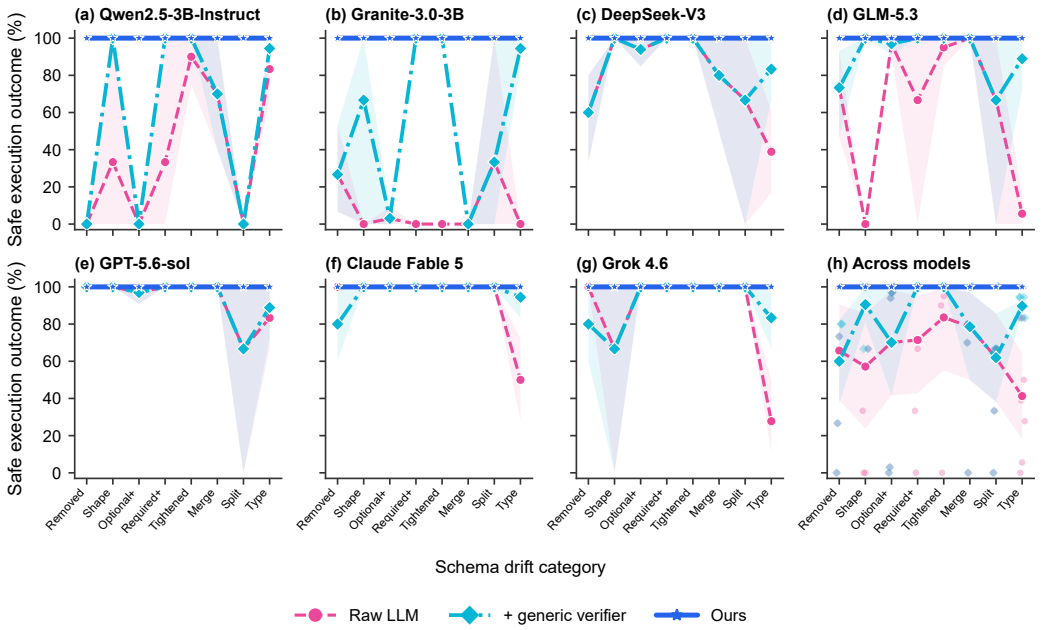


Figure 6: Category-level outcomes by model under common evaluation semantics and drift ordering. The verified certificate procedure provides the shared reference across panels.

C PROOF DETAILS

Unique target. The verifier reconstructs the tool-candidate set using the same finite, ordered rule language but does not trust the candidate serialized by the constructor. Acceptance requires this set to be the singleton $\{\tau\}$.

Source completeness and injectivity. For each live source location, the verifier requires exactly one of two disjoint witnesses: a reciprocal edge in ρ or membership in D after an independently recomputed deletion test. Reciprocal unique maxima make ρ injective. A plausible unmatched replacement invalidates the deletion witness, which prevents a live value from disappearing after its old field becomes optional or absent.

Bounded introduction and conservation. Every target value is either copied through ρ or produced by δ. The latter is accepted only if its derivation consists of uniquely selected contract literals. Location-aware multiset equality in Equation 8 then rules out both unlicensed invention and rebinding among equally typed target fields.

Validity and totality. Complete target validation is the final conjunct and is recomputed over the constructed object. All searches range over finite contract elements bounded by preflight; every partial operator has an explicit failure branch, and the outer exception boundary maps any unsupported input or runtime failure to ABSTAIN. These properties establish each conclusion of Theorem 1. Its scope excludes external-world meanings that neither contract encodes.