COMMIT BEFORE BINDING: TYPED EFFECT PROGRAMS FOR HELD-OUT TOOL FAMILIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool interfaces certify call shape, not state-change intent: a schema-valid agent can still mutate the wrong resource, cross an authorization boundary, or commit an irreversible transition. RAVEL transfers the missing invariant as a provider-independent typed effect program over resource identity, validation, permission, mutation scope, and recovery; provider operations and arguments become available only after the program composes, while uncertain critical bindings trigger read-only probes or pre-mutation abstention. We evaluate this commitment-first architecture on 612 resource-mutating tasks in six leave-one-family-out folds, exposing target documentation and schemas under a common 15-call budget while withholding target-family trajectories and supervision. Against same-backbone Adapter Transfer, RAVEL raises fold-mean success from 40.3% to 48.3% (paired gain 8.0 points, 95% CI [6.1, 9.8]) and lowers task-weighted catastrophic error from 10.1% to 4.7% (RAVEL-minus-Adapter CI [-6.9, -3.8]). A latent-plan control matched for source data, annotation and verification effort, capacity, corruption training, hard negatives, probing, and abstention attributes a further 3.2-point success gain and 2.1-point catastrophic-error reduction to the joint typed structure. All six family-wise catastrophic-error intervals favor RAVEL; under misleading descriptions it retains 77% of clean completion versus 64% for Adapter Transfer, and an independently assembled 184-task AppWorld-derived suite reproduces the ordering (44.2% versus 40.7% for the matched control and 35.9% for Adapter Transfer). Familywise replay identifies scheduler, dependency, trigger, and retry state as the residual representation bottleneck. Making commitments executable before an unfamiliar call can alter state therefore improves completion and mutation integrity while revealing where resource-centric effects require temporal structure.

1 INTRODUCTION

A version-control agent asked to delete a merged branch can select the documented endpoint, supply well-typed arguments, and still act on the wrong repository or cross a protected-branch boundary. Object storage, databases, and workflow engines expose the same mismatch as a wrong bucket, transaction scope, or trigger phase. The interface describes how to issue a call; it does not carry the operational commitment that should survive provider change: which resource may change, which observation licenses the change, which permission applies, when the effect becomes external, and which recovery obligation remains.

Existing tool-generalization benchmarks emphasize tool discovery, ranking, invocation, or outcome completion. They do not jointly withhold an entire operational family, execute state-changing trajectories, and distinguish successful completion from unauthorized, wrong-resource, or irreversible mutations (Qin et al., 2024; Li et al., 2023; He et al., 2025; Wu et al., 2026; Trivedi et al., 2024; He et al., 2026; Lu et al., 2026; Liu et al., 2026a). This leaves the transfer object underspecified: an agent may learn provider syntax without learning the invariant that licenses a mutation.

RAVEL makes that invariant executable. A source-trained mapper first constructs a provider-independent effect program over retrieval, filtering, validation, mutation, authorization, and recovery. Provider operations and arguments are absent from this stage. Once the complete program passes type and composition checks, a residual decoder binds its slots to the target schema. State-changing calls remain masked until resource, validation, permission, and recovery obligations are satisfied; uncertain

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.

critical slots invoke compatible read-only probes, and a calibrated gate can abstain immediately before mutation. This ordering exposes the exact point at which a generic intent becomes an authorized provider action.

The evaluation isolates both the complete system and the representation. Six folds hold out REST APIs, CLI tools, databases, version control, workflow engines, or cloud storage, for 612 resource-mutating tasks in total. Each target family contributes documentation and schemas at inference, while trajectories, labels, demonstrations, updates, and calibration examples come only from the other five families. Relative to same-backbone Adapter Transfer, RAVEL improves success from 40.3% to 48.3% and reduces catastrophic error from 10.1% to 4.7%. A matched latent-plan control equalizes source data, annotation and verification effort, capacity, corruption training, hard negatives, probes, abstention, and optimization; replacing its unconstrained codes with typed slots, composition, reversibility, and operation masks adds 3.2 success points and removes 2.1 catastrophic-error points.

The family map turns the aggregate gain into a representation diagnosis. All six catastrophic-error intervals favor RAVEL. Cloud Storage and Databases gain 11.8 and 9.5 success points, while Workflow Engines retain a smaller 3.5-point completion gain and concentrate failures in scheduler phase, dependency release, trigger timing, and retry state. A failure-derived temporal vocabulary raises Workflow success by 8.2 points on that diagnostic suite while the other-five-family estimate changes by -0.2 points. RAVEL also retains 77% of clean success under misleading descriptions, preserves its ordering after all 87 authored variants are removed, and reproduces that ordering on an independently assembled AppWorld-derived domain holdout.

Contribution hierarchy. The primary contribution is commitment-first binding: a typed, provider-independent effect program becomes a hard precondition for provider-specific mutation. Two linked study contributions make this mechanism measurable: an executable whole-family holdout protocol that reports completion, mutation integrity, and selective utility together; and a family-resolved failure analysis that connects inspectable resource state to large gains and delayed scheduler state to a targeted temporal extension. The matched latent control, documentation perturbations, authored-free split, and separate AppWorld-derived suite test the same mechanism from complementary directions.

2 Related Work

Generalization beyond familiar interfaces. Toolformer, ToolLLM, Gorilla, API-Bank, RESTGPT, ToolAlpaca, and ToolkenGPT establish selection, grounding, and invocation across broad tool inventories (Schick et al., 2023; Qin et al., 2024; Patil et al., 2024; Li et al., 2023; Song et al., 2023; Tang et al., 2023; Hao et al., 2023). GenTool evaluates ranking under zero-to-one and weak-to-strong simulated transfer (He et al., 2025); ToolLIFT lifts tool-specific trajectories into function-level plans (You et al., 2026); FlowAgent learns continuous planning trajectories (Huang et al., 2026); and OpenAgent measures controlled query, action, observation, and domain shifts (Wu et al., 2026). Their transfer objects are rankings, trajectory abstractions, latent plans, or shift robustness. RAVEL instead learns an executable pre-mutation invariant while every interface, trace, label, and update from one operational family is withheld. The empirical isolation uses same-backbone Adapter Transfer for the complete-system contrast and an equal-capacity latent plan for the typed-structure contrast.

Executable and trajectory-level evaluation. ReAct, Reflexion, CodeAct, SWE-agent, and AutoGen organize reasoning, feedback, code execution, and multi-agent action (Yao et al., 2023; Shinn et al., 2023; Wang et al., 2024a; Yang et al., 2024; Wu et al., 2024). AgentBench and MINT measure interactive competence; AppWorld and SWE-bench verify environment outcomes; TRAJECT-Bench decomposes selection, parameterization, and ordering (Liu et al., 2024; Wang et al., 2024b; Trivedi et al., 2024; Jimenez et al., 2024; He et al., 2026). SIMMER adds world-model execution to expose latent and irreversible planning failures (Lu et al., 2026). Our evaluation crosses this trajectory view with a transfer intervention: executable terminal success, protected-resource integrity, permission, recovery, and latent-transition labels are measured after withholding a complete software family. Paired document perturbations and familywise replay then identify whether the transferred commitment survives both interface and state-machine shift.

Effect structure and selective mutation. ToolGate gates known tools with Hoare-style contracts, Contract2Tool infers preconditions and effects for causal tool filtering, Cordon stages cross-step

effects in semantic transactions, and ETAS gives agent programs effect and trace semantics (Liu et al., 2026b; Babu & Iyer, 2026; Chen et al., 2026; Tan et al., 2026). These systems formalize or mediate execution for a supplied tool set. RAVEL addresses the complementary transfer problem: it learns a provider-independent commitment program from source families, completes that program before any held-out-family operation is bound, and measures completion and mutation integrity under whole-family withholding. This design connects classical planning, compositional effect systems, and constrained generation (Fikes & Nilsson, 1971; Lucassen & Gifford, 1988; Shin et al., 2021; Cheng et al., 2023) with calibrated deferral (Guo et al., 2017; Naeini et al., 2015; Geifman & El-Yaniv, 2017; El-Yaniv & Wiener, 2010; Hendrycks & Gimpel, 2017; Lakshminarayanan et al., 2017; Gal & Ghahramani, 2016; Liu et al., 2026a). The matched latent plan isolates the joint representation-and-ordering effect; the temporal extension probes a failure-derived vocabulary within Workflow.

3 Commitment Transfer Across Held-Out Families

3.1 Evaluation Setting

Let $\mathcal{F} = \{F_1, \ldots, F_M\}$ contain operational families whose members expose different provider interfaces. We use *interface* for one sandboxed product wrapper and schema version, *operation* for one callable endpoint or command in that schema, and *task* for one initialized sandbox episode with an executable goal. An interface $t = (d_t, s_t, \Xi_t)$ has documentation d_t, schema s_t, and optional traces Ξ_t. Fold m trains on $\mathcal{F} \setminus F_m$ and tests on F_m. At inference, d_t and s_t for $t \in F_m$ are visible; target-family traces, labels, demonstrations, and parameter updates are withheld. Source-family graphs may therefore contain trace nodes, whereas target-family graphs contain only documentation and schema nodes. This is benchmark-family holdout: it tests transfer without benchmark supervision from F_m, independently of what a pretrained backbone may have encountered.

A family is a predeclared stratum defined by its dominant state transition, authorization boundary, and recovery unit. REST covers remote record-resource services; CLI covers process- and filesystem-scoped command workflows outside the specialized resource strata; Databases center transaction and query state; Version Control centers revision graphs and repository transactions; Workflow Engines center scheduled dependency state; and Cloud Storage centers object namespaces and ACL-scoped mutations. Assignment follows the task wrapper and executable checker, so a product name never determines a fold by itself. Two annotators independently apply this rubric before split construction and adjudicate disagreements, producing mutually exclusive task strata with distinct mutation state machines. The record schema, family rubric, executable predicates, protected-state assertions, corruption operators, and catastrophic-error taxonomy are fixed before fold training; a fold is then formed by removing its family's benchmark-side interfaces and supervision.

The central factorization assigns cross-family commitments to an effect program and provider realizations to residual slots. The matched latent plan preserves annotation effort, training, and capacity while removing typed commitment structure, so its contrast with RAVEL estimates the joint effect of typed slots, composition, reversibility, and operation masks. A separate exploratory extension studies whether adding temporal primitives addresses errors concentrated in the Workflow family.

3.2 Commitment Effect Programs

Definition 1 (Effect Sketch)**.** An effect sketch σ is a typed program over six primitive effects:

$$\begin{aligned} e ::= &\textsc{Retrieve}(\tau_{\text{in}}, \tau_{\text{out}}) \mid \textsc{Filter}(\tau_{\text{in}}, \tau_{\text{pred}}, \tau_{\text{out}}) \mid \textsc{Mutate}(\tau_{\text{in}}, \tau_{\text{scope}}, \tau_{\text{rev}}) \\ &\mid \textsc{Validate}(\tau_{\text{in}}, \tau_{\text{cond}}) \mid \textsc{Commit}(\tau_{\text{scope}}, \tau_{\text{perm}}) \mid \textsc{Rollback}(\tau_{\text{ref}}). \end{aligned} \tag{1}$$

Data slots lie in $\mathcal{T}_{\text{data}}$, mutation scopes in $\mathcal{T}_{\text{mut}}$, permissions in $\mathcal{T}_{\text{perm}}$, predicates in $\mathcal{T}_{\text{pred}}$, and $\tau_{\text{rev}} \in \{\top, \bot\}$.

Each primitive is a typed state transition $\Gamma_i \vdash e_i : \Gamma_i \Rightarrow \Gamma_{i+1}$, where Γ_i maps resource references to data, mutation scope, permission, predicate, and reversibility types. Data slots hold resource identifiers and observed values; mutation slots pair a target resource with its effect scope; permission slots identify principal, action, and scope; predicate slots encode existence, identity, precondition,

or postcondition checks; and recovery slots hold an inverse operation or saved-state reference. The grammar lists admissible primitives, not a mandated order. RETRIEVE introduces an observed resource, FILTER narrows its identity, and VALIDATE records a satisfied predicate. MUTATE declares an intended state delta and recovery type but does not emit an external call. Every MUTATE must be covered by a COMMIT carrying the applicable permission; ROLLBACK consumes the saved pre-state or inverse reference. Table 7 gives the complete operational contract. A program $\sigma = e_1 \circ \cdots \circ e_L$ is well formed only if types compose, every mutation has resource and scope identity, its required validations precede its covering commit, and its reversibility obligation is resolved. Provider binding begins only after the complete program passes these checks.

Definition 2 (Residual Slots)**.** Given σ and tool t, $\mathcal{R}(\sigma, t)$ contains interface-specific resource identifiers, pagination policies, credentials, transaction boundaries, rate limits, and rollback mechanisms that remain to be bound.

For fold m, all learned parameters are fit on the five source families:

$$\begin{aligned} \Theta_m &= (\eta_m, \gamma_m, \phi_m, \psi_m, \omega_m) = \mathrm{Train}(\mathcal{D}_{\mathcal{F}\setminus F_m}), \\ \mathrm{Score} &= \frac{1}{M}\sum_{m=1}^{M} \left[\mathrm{SR}_m - \lambda_c \mathrm{Cat}_m\right], \qquad \mathrm{Calls}(\pi_{\Theta_m}) \le K. \end{aligned} \tag{2}$$

Here $\mathrm{SR}_m = \mathrm{SR}(\pi_{\Theta_m}, F_m)$ and likewise for Cat; target-family observations enter only the reported score. We use $K = 15$ and select $\lambda_c = 3.0$ on source-family validation.

4 RAVEL: DELAYED COMMITMENT BINDING

Backbone-to-program dataflow. RAVEL implements commitment factorization as a hard execution order: the effect program must be complete before a provider operation or argument can be selected. The frozen DeepSeek-Coder-V2 backbone with trainable LoRA parameters η encodes the request, documentation sentences, schema names and descriptions, and accumulated trajectory tokens. Its contextual states initialize request and interface-node representations. RAVEL builds $G_t = (\mathcal{V}_t, \mathcal{E}_t)$ from those schema-field and sentence nodes; source graphs additionally include example, error, and trace nodes, while held-out-family graphs contain only exposed documentation and schema. Compatibility, co-occurrence, dependency, exclusion, and document–schema alignment edges are supervised from source schemas and traces. A six-layer, 256-dimensional, eight-head graph attention network with parameters γ (Veličković et al., 2018) yields $\mathbf{g}_t$. A four-layer transformer mapper M_ϕ cross-attends to the backbone request states and $\mathbf{g}_t$ and decodes a task-level program

$$p(\hat{\sigma} \mid G_t, q) = \prod_{\ell=1}^{L} p(\hat{e}_\ell \mid \hat{e}_{<\ell}, \mathbf{g}_t, q), \qquad \hat{\sigma} = \arg\max_{\sigma \in \mathcal{G}} p(\sigma \mid G_t, q), \tag{3}$$

using beam search with type masks. The output is a discrete, fully type-checked $\hat{\sigma}$; no provider operation token is available to the mapper.

Commitment-first residual binding. After $\hat{\sigma}$ is fixed, $R_\psi(\hat{\sigma}, G_t, q)$ receives the backbone trajectory states, graph states, and typed program and emits a serialized provider operation plus arguments. The four-layer, 256-dimensional causal decoder has separate operation and argument heads. Source traces align an operation with the maximal contiguous effect block whose postconditions its replay verifies; a multi-label compatibility scorer then infers $C_t(o) \subseteq \{1, \ldots, L\}$ from operation documentation, input/output types, and graph state. Target-family compatibility annotations never enter this scorer. Thus a transaction API may stage and commit in separate calls, whereas an immediately effective delete call may cover adjacent MUTATE and COMMIT effects atomically. At decision k, the operation head can select only an o compatible with the next unfulfilled effect block; the argument head grounds its typed slots in graph nodes, task entities, and prior observations. Each binding has confidence c_s. A slot is safety-critical when an incorrect resource, scope, permission, validation, or recovery binding can instantiate an unauthorized, wrong-resource, or irreversible mutation. Below $\tau_r = 0.6$, RAVEL selects the compatible read-only operation with the largest expected reduction in the argument head's candidate-binding entropy, where possible outcomes are induced by the operation schema and scored from source-family traces.

Guarded execution. RAVEL maintains an effect pointer and a typed environment of bound resources, observations, permissions, and recovery references. Read-only calls may execute as soon as their input types are bound. A state-changing operation o is emitted only under the invariant

$$\mathrm{Emit}(o) \wedge \mathrm{ChangesState}(o) \Rightarrow \mathrm{Bound}(\mathrm{resource}, \mathrm{scope}, \mathrm{recovery}) \wedge \mathrm{Validated} \wedge \mathrm{Authorized}. \quad (4)$$

Here *Validated* means every predicate required by the covered mutation block is present as a successful observation in the environment, and *Authorized* means the permission slot of its covering COMMIT is bound and satisfied before dispatch; permission cannot be supplied by the postcondition of the same call. The guard applies to every state-changing call, including rollback, scheduling, triggering, retry, and backfill calls. For a call that jointly realizes MUTATE and COMMIT, all preconditions of the covered block must therefore hold before dispatch. Execution repeats four steps: bind the next compatible effect block, probe any low-confidence critical slot, evaluate Equation 4 and the risk gate, then dispatch or abstain. A returned observation advances only the maximal contiguous prefix of effects in $C_t(o)$ whose postconditions are verified; one effect may require several calls, in which case the pointer remains fixed. A tool error does not advance the pointer, and rebinding a potentially state-changing operation requires a read-only observation that its postcondition remains unmet. Probe calls consume the same 15-call budget as task actions. The rollout succeeds when all program postconditions are observed; the executable benchmark checker is used only for evaluation and is never visible to the agent.

Selective mutation. Before candidate mutation a_k, the residual adapter exposes a 256-dimensional decision state $\mathbf{h}_k$ that jointly summarizes task q, trajectory prefix $y_{<k}$, proposed residual bindings, binding confidences, and a_k. The sketch encoder is a six-dimensional normalized primitive-count vector, making the risk-head input fixed at 518 dimensions:

$$p_{\mathrm{cat},k} = \alpha_\omega([\mathbf{g}_t; \mathrm{enc}(\hat{\sigma}); \mathbf{h}_k]), \qquad \text{abstain before } a_k \text{ if } p_{\mathrm{cat},k} > \tau_a. \quad (5)$$

The detector uses focal loss (Lin et al., 2017). The final source-family validation split selects λ_c from $\{1, 2, 3, 4\}$, τ_r from $\{0.4, 0.5, 0.6, 0.7, 0.8\}$, and τ_a from the validation precision–coverage curve; the selected values are 3.0, 0.6, and 0.35.

Optimization. Training has two stages and a calibration pass. A master pool contains 120 manually labeled seed operations and human-verified bootstrapped effects; before each fold, every operation label, trace, and interface from F_m is removed, so Stage 1 learns sketch tokens exclusively from the five source families. Stage 2 minimizes

$$\mathcal{L} = \mathcal{L}_{\mathrm{task}} + 0.5\mathcal{L}_{\mathrm{effect}} + 2.0\mathcal{L}_{\mathrm{abstain}} + 0.3\mathcal{L}_{\mathrm{consist}}. \quad (6)$$

$\mathcal{L}_{\mathrm{task}}$ combines teacher-forced negative log likelihood for the next operation and grounded arguments with binary cross-entropy from a final-trajectory outcome head to the executable success label. Gradients pass through backbone LoRA states, graph, mapper, operation, argument, and trajectory states; discrete environment transitions supply labels and are not differentiated. Effect tokens use cross-entropy, catastrophic labels use focal loss, and adjacent call predictions use KL consistency. Type and operation masks used at inference are also applied during teacher forcing. Stage 2 jointly tunes the LoRA matrices η, GAT γ, mapper M_ϕ, residual decoder R_ψ, and risk head α_ω using only the fold's source-family trajectories; all base DeepSeek weights remain frozen. It corrupts 30% of source documentation examples and draws 15% of batches from near-signature hard negatives. The matched latent control retains the same trainable tensor shapes and parameter count. The calibration pass temperature-scales α_ω and selects τ_a using source-family validation only.

The backbone is DeepSeek-Coder-V2-236B (DeepSeek-AI et al., 2024) with LoRA (Hu et al., 2022) ($r = 16$, $\alpha = 32$ on attention projections). Training uses 8×A100 80GB GPUs, CUDA 12.4, PyTorch 2.3.1, DeepSpeed ZeRO-3, bfloat16, learning rate 2×10^{-5}, cosine decay, and eight epochs.

5 EXPERIMENTS

5.1 PROTOCOL

The benchmark contains 612 tasks across REST APIs (118), CLI Tools (94), Databases (108), Version Control (86), Workflow Engines (92), and Cloud Storage (114). We import 525 multi-step resource-mutation tasks from ToolBench, API-Bank, and GAIA and add 87 authored variants that preserve

the goal while changing naming, permission, or rollback conventions (Qin et al., 2024; Li et al., 2023; Mialon et al., 2024). Normalization converts each candidate into one record containing an initial sandbox snapshot, natural-language goal, allowed interface and operations, reference trace, executable terminal-state predicate, and protected-resource assertions. We replay the normalized trace, exclude non-mutating or non-replayable candidates, and deduplicate by normalized goal, operation signature, and end-state predicate before family assignment. Appendix A reports the 28 provider interfaces and task counts.

Family withholding covers every benchmark-side operation label, trace, demonstration, adapter update, and calibration example. Controlled transfer comparisons share the frozen DeepSeek backbone, while prompt and retrieval references retain the backbones named in Table 1. Reference solutions contain 3–8 calls, while the 15-call cap accommodates read-only probes and recovery attempts. The primary evaluation runs on clean held-out-family documentation.

Corruption is a separate paired stress evaluation. For each task, five fixed seeds select identical corruptions for every method; metrics first average the five outcomes within a task, and the paired bootstrap resamples tasks together with all five outcomes. *Dropped Fields* removes 40% of non-required schema-description fields while retaining names, types, and executable validity. *Renamed Parameters* applies a one-to-one, type-preserving opaque rename consistently across schema references. *Misleading Descriptions* replaces one operation description with a fluent description from a near-signature operation having different side effects; two raters verify grammaticality, schema validity, and semantic conflict before adjudication.

We compare Prompt + Docs, Schema Retrieval, Code Generation, Adapter Transfer, and a Tool-Specific FT reference using 15 labeled target-family examples per interface. Adapter Transfer is the complete-system comparator: it shares the DeepSeek backbone, source documentation, schemas, traces, 15-call budget, and LoRA capacity. GenTool evaluates simulated tool ranking, ToolLIFT requires tool-specific trajectories for plan lifting, FlowAgent learns continuous trajectories, and OpenAgent studies a different controlled-shift protocol; their published outputs do not instantiate this suite's executable mutation contract. The matched latent-plan control directly targets the closest common abstraction—a learned plan between request and action—while matching RAVEL's seed-labeling and verification budget, corruption schedule, hard negatives, probes, abstention head, mapper and decoder capacity. It replaces the typed program, slots, composition rules, and operation masks with learned 256-dimensional unconstrained codes.

For that control, each annotated effect step supervises one learned 256-dimensional code at the same sequence position. A codebook with cardinality equal to RAVEL's primitive vocabulary maps the aligned source annotation ID to a learned vector; the mapper predicts a normalized vector, and its code loss is cross-entropy over cosine similarities to all codebook vectors. The highest-similarity vector conditions the residual decoder at inference. The control retains RAVEL's plan length, four-layer mapper and residual decoder, parameter budget, call and outcome losses, and identical abstention and consistency objectives. Its codes have no declared input/output types, resource or permission slots, reversibility field, composition rule, or operation mask. This construction preserves annotation budget, aligned primitive IDs, and optimization capacity while ablating the joint typed commitment structure.

SR requires the executable end state and counts an abstention as unsuccessful. ICR pools parse and type failures over calls actually issued, including probe calls. Calls includes all actions before completion or abstention. Cat is the task-weighted fraction with an unauthorized, wrong-resource, or irreversible mutation; refusal itself is not catastrophic, while any harmful call before refusal remains Cat. Each fold uses three training seeds, and each checkpoint is evaluated with 20 fixed decoder seeds per task. Table 1 pools checkpoint–decoder–task indicators for aggregate Cat. Familywise Cat averages the twenty decoder seeds for the checkpoint selected by the median source-validation objective; count-based audits additionally fix a predeclared canonical decoder seed. Both selections occur before target evaluation. Two independent raters inspect the goal, checker state, trace, and sandbox replay, then adjudicate disagreements (pre-adjudication $\kappa = 0.79$). Aggregate paired intervals use a nested bootstrap that resamples families, shared training seeds, decoder seeds, and paired tasks within family, retaining family task counts as Cat weights. Familywise Cat intervals resample decoder seeds and paired tasks for the preselected checkpoint. All intervals use 10,000 resamples.

Table 1: Six-fold held-out-family transfer. SR is the fold mean; ICR is pooled; Cat is task-weighted over all checkpoint–decoder–task trials; Calls includes probes and pre-abstention calls. All methods have 15 calls; † uses 15 labeled target examples per interface. The Llama row tests implementation portability; matched mechanism contrasts use DeepSeek.

Method	Backbone	SR ↑	ICR ↓	Cat ↓	Calls ↓
Prompt + Docs	GPT-4o	34.7	26.3	13.1	12.4
Prompt + Docs	Claude-3.5-Sonnet	33.2	27.8	13.9	12.7
Prompt + Docs	Llama-3.1-70B	28.6	31.2	16.4	13.1
Schema Retrieval	GPT-4o	39.1	21.8	11.2	11.6
Code Generation	GPT-4o	36.8	23.4	12.0	13.2
Adapter Transfer	DeepSeek-Coder-V2	40.3	19.7	10.1	10.9
Tool-Specific FT†	DeepSeek-Coder-V2	42.1	18.9	9.1	10.2
Matched latent plan	DeepSeek-Coder-V2	45.1	14.9	6.8	10.1
RAVEL	DeepSeek-Coder-V2	48.3	12.6	4.7	9.8
RAVEL	Llama-3.1-70B	43.7	15.8	6.9	10.5

Table 2: Family-resolved transfer on clean documentation. SR point estimates average all checkpoint–decoder trials. Cat estimates and paired intervals use twenty decoder seeds for each fold's median-validation checkpoint. The Average row is an unweighted family macro.

Held-out family	n	Adapter		RAVEL		ΔSR	ΔCat [95% CI]
		SR	Cat	SR	Cat		
REST APIs	118	42.8	8.7	50.4	4.1	7.6	−4.6 [-7.3, -2.0]
CLI Tools	94	39.6	11.2	48.2	5.0	8.6	−6.2 [-9.4, -3.0]
Databases	108	43.1	8.3	52.6	3.4	9.5	−4.9 [-7.5, -2.2]
Version Control	86	37.8	12.4	44.6	5.8	6.8	−6.6 [-10.2, -3.3]
Workflow Engines	92	35.2	13.1	38.7	7.2	3.5	−5.9 [-9.5, -2.1]
Cloud Storage	114	43.3	7.9	55.1	2.7	11.8	−5.2 [-7.6, -2.8]
Average	612	40.3	10.3	48.3	4.7	8.0	−5.6

5.2 RQ1: Does Commitment Structure Improve Completion and Mutation Integrity?

Table 1 supplies two nested tests. Relative to same-backbone Adapter Transfer, RAVEL gains 8.0 SR points (95% paired hierarchical-bootstrap CI [6.1, 9.8]), lowers ICR by 7.1 points, reduces Cat by 5.4 points (RAVEL minus Adapter CI [-6.9, -3.8]), and uses 1.1 fewer calls. Matching annotation and verification budgets, corruption training, hard negatives, probes, abstention, and plan capacity yields 45.1 SR and 6.8 Cat; the joint typed structure then adds 3.2 paired SR points (95% CI [1.8, 4.7]) and reduces Cat by 2.1 points (CI [-3.2, -1.0]). Component removals localize complementary roles: removing effect sketches costs 9.2 SR points, while removing the abstention detector raises Cat from 4.7% to 11.3% (Appendix E). The full-stack, matched-plan, and component comparisons align on a single conclusion: explicit commitment structure improves completion and harmful-mutation risk together.

5.3 RQ2: Which State Transitions Benefit, and Which Need Temporal Effects?

The family view separates a broad safety gain from a heterogeneous completion gain. All six Cat intervals lie below zero. Cloud Storage and Databases improve SR by 11.8 and 9.5 points; in these state machines, object identity, permission scope, validation, and recovery are directly inspectable before mutation. Workflow Engines improve SR by only 3.5 points while reducing Cat by 5.9 points, and replayed failures cluster around scheduler phase, dependency release, trigger timing, and backfill state. The aggregate in Table 1 pools checkpoint–decoder–task trials, whereas Table 2 fixes the median-validation checkpoint and averages its twenty decoder seeds; this predeclared distinction

yields 4.7% RAVEL Cat in both views while allowing task-weighted and family-macro comparator values to differ.

Temporal mechanism probe. The replay categories suggest four missing transitions: SCHEDULE, DEPEND, TRIGGER, and RETRY/BACKFILL. Their labels and mapping rules are derived from source-family queue, trigger, and dependency analogues; backbone, source data, optimization, residual decoder, probes, thresholds, and the 15-call budget remain fixed. Both variants use a ten-coordinate primitive-count encoder and a 522-dimensional risk-head input, with the base model's four added coordinates set to zero. On the 92 Workflow tasks, the temporal variant raises SR from 38.7% to 46.9%, a paired 8.2-point increase (95% CI [3.8, 12.8]), and lowers Cat from 7.2% to 4.8% (CI [-4.7, -0.4]). The other-five-family SR changes from 50.2% to 50.0% (CI [-1.1, 0.8]). Appendix H reports the complete metric table and category counts.

5.4 RQ3: DO COMPLETION GAINS PERSIST UNDER DOCUMENTATION AND TASK SHIFT?

RAVEL retains 77% under misleading descriptions versus 64% for Adapter and 52% for Schema Retrieval. Relative to Adapter, RAVEL's paired margins are 10.3 points for dropped fields (95% CI [6.4, 14.0]), 12.3 for renaming ([8.2, 16.1]), and 11.5 for misleading descriptions ([7.4, 15.5]). RAVEL also exceeds the corruption-trained matched latent plan by 5.1 points under dropped fields (CI [2.4, 7.9]), 6.5 under renaming ([3.5, 9.3]), and 7.5 under misleading descriptions ([4.1, 10.8]). Because this stress test records SR, these estimates support completion robustness under document shift; the widening margin over the matched control shows that typed structure retains its completion advantage after matching corruption training and hard negatives.

Independent executable slice. We separately construct 184 resource-mutation tasks from the public AppWorld release associated with its ACL 2024 paper (Trivedi et al., 2024). We use the released task collection as a source pool rather than preserve its leaderboard partitions: tasks are repartitioned into held-out Commerce (46), Communication (42), Productivity (49), and Travel/Media (47) domain folds. Inclusion requires a persistent app-state mutation and an executable final-state predicate; read-only, subjective-generation, and non-replayable tasks are excluded. We deduplicate normalized goals and operation signatures within the pool and remove any task whose normalized goal–operation pair overlaps the six-family suite.

For each domain fold, the mapper, residual decoder, and risk head are fit on the other three AppWorld domains; no target-domain trace, label, example, update, or calibration item is used. Architecture, optimization, thresholds, Cat taxonomy, checkpoint selection, and the 15-call cap are copied from the primary protocol. The checker reads the post-execution AppWorld state and applies the task's deterministic goal predicate plus the same unauthorized, wrong-resource, and irreversible-mutation labels. Thus this suite changes the apps, domain partition, and task construction while preserving the estimands.

The independent pool reproduces the full ordering. RAVEL exceeds the matched control by 3.5 SR points (95% paired CI [1.1, 5.8]) and reduces Cat by 2.5 points (CI [-5.0, -0.4]); its advantage over Adapter is 8.3 SR points and 6.2 Cat points. Appendix D supplies a complementary shift: after all 87 authored tasks are removed and the remaining 525 are grouped by goal template and product, RAVEL again leads both comparators overall and on Workflow Engines.

5.5 RQ4: HOW MUCH UTILITY DOES PRE-MUTATION SELECTION PRESERVE?

The gate operates on decision instances immediately before a candidate mutation. The test population contains 214 harmful-event positives and 1986 negatives; training and source-validation use disjoint 512/4688 and 126/1074 positive/negative pools. At $\tau_a = 0.35$, the gate intercepts 155 of 214 harmful events, giving 72.4% raw event coverage at 95% precision from 163 flagged decisions. Event interception, task-level Cat, and whole-task abstention remain separate estimands; Appendix Figure 2 and Table 15 report calibration and task-level utility.

In the canonical rollout, 168 abstentions leave 444 accepted trajectories; their 66.7% success corresponds to 296 successful tasks, or 48.4% overall, consistent with the 48.3% multi-seed mean. For Workflow Engines, 39 abstentions leave 53 accepted trajectories; 36 successes give the reported 67.9% accepted SR and 39.1% canonical-rollout SR, close to the 38.7% multi-seed mean. Removing

Table 3: Paired documentation stress tests of completion. Values are SR with retention from each method's clean score in parentheses. All methods receive the same task-level corruption manifests.

Method	Clean	Drop 40%	Rename	Mislead
Prompt + Docs	34.7	24.3 (70%)	20.1 (58%)	16.8 (48%)
Schema Retrieval	39.1	29.6 (76%)	25.7 (66%)	20.4 (52%)
Adapter Transfer	40.3	33.8 (84%)	29.4 (73%)	25.7 (64%)
Matched latent plan	45.1	39.0 (86%)	35.2 (78%)	29.7 (66%)
Code Generation	36.8	25.9 (70%)	21.6 (59%)	17.3 (47%)
RAVEL	48.3	44.1 (91%)	41.7 (86%)	37.2 (77%)

Table 4: Independent AppWorld-derived validation. SR is domain-fold mean $\pm$SE; Cat brackets are 95% task-bootstrap intervals.

Method	SR	ICR	Cat [95% CI]	Calls
Adapter Transfer	35.9$\pm$1.8	22.8	12.5 [9.1, 16.6]	11.6
Matched latent plan	40.7$\pm$1.7	17.4	8.8 [6.0, 12.4]	10.8
RAVEL	44.2$\pm$1.6	14.1	6.3 [4.0, 9.1]	10.2

the detector leaves SR near 47.1% but raises Cat to 11.3% (Appendix E). Cloud combines high accepted utility with 82.8% event coverage over 29 pre-gate harmful candidates. Workflow coverage is 63.6% over 22 such candidates. These decision-level candidates include intercepted events and may occur multiple times within a task; the 7.2% Workflow Cat rate averages post-gate task indicators over twenty decoder seeds.

6 Discussion and Conclusion

Three comparisons identify the source of RAVEL's gain. The complete stack improves same-backbone Adapter Transfer by 8.0 success points and 5.4 catastrophic-error points. Holding source information, annotation effort, capacity, probes, and abstention fixed, the typed commitment structure contributes another 3.2 success points and 2.1 catastrophic-error points over a latent plan. Component removals then separate execution roles: effect sketches drive completion, while the pre-mutation detector supplies the largest safety contribution. Documentation attacks, the authored-free split, and the AppWorld-derived suite preserve this ordering under three distinct shifts.

The family-resolved evidence explains when the abstraction transfers. Resource identity, permission scope, validation, and recovery can be observed before mutation in Cloud Storage and Databases, where success gains are largest; delayed scheduler transitions concentrate the Workflow errors. Adding temporal effects changes Workflow success by 8.2 points while leaving the other-five-family estimate effectively unchanged, connecting the failure taxonomy to a targeted representation change rather than a uniform capacity increase.

The evaluated regime is benchmark-family holdout over software-resource tasks with visible target documentation and schema, 3–8-call reference trajectories, sandbox-observable effects, and source-side effect supervision. Within this regime, the pre-mutation gate controls rather than eliminates harmful actions, and its benefit depends on read-only observations that expose identity, permission, validation, and recovery state. The temporal vocabulary was selected after inspecting the same Workflow suite, so its result is mechanism-directed evidence; the confirmatory claims rest on the predeclared base RAVEL comparisons. This source-side investment—120 labeled seed operations, 68 annotation hours, and 41 verification hours reused across six folds—is amortized when providers share an operational core.

RAVEL turns the invariant behind a tool call into an executable transfer object. Requiring resource, evidence, permission, and recovery commitments before provider binding raises completion, cuts harmful mutations in every held-out family, and exposes temporal state as a concrete next boundary for commitment representations.

AI USE STATEMENT

Generative AI tools were used to assist with literature search, manuscript drafting and editing, structural feedback, and reference and figure formatting. The authors reviewed and verified all AI-assisted text, citations, equations, figures, analyses, and reported values, and take responsibility for the final content of this work.

REFERENCES

Rahul Suresh Babu and Laxmipriya Ganesh Iyer. Contract2Tool: Learning preconditions and effects for reliable tool-augmented LLM agents. *arXiv preprint arXiv:2606.07904*, 2026. doi: 10.48550/arXiv.2606.07904. URL https://arxiv.org/abs/2606.07904.

Zheng Chen, Hanqing Liu, Duling Xu, Dong Dong, Jialin Li, Bangzheng Pu, and Jidong Zhai. Cordon: Semantic transactions for tool-using LLM agents. *arXiv preprint arXiv:2606.17573*, 2026. doi: 10.48550/arXiv.2606.17573. URL https://arxiv.org/abs/2606.17573.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lH1PV42cbF.

DeepSeek-AI, Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y. Wu, Yukun Li, Huazuo Gao, Shirong Ma, Wangding Zeng, Xiao Bi, Zihui Gu, Hanwei Xu, Damai Dai, Kai Dong, Liyue Zhang, Yishi Piao, Zhibin Gou, Zhenda Xie, Zhewen Hao, Bingxuan Wang, Junxiao Song, Deli Chen, Xin Xie, Kang Guan, Yuxiang You, Aixin Liu, Qiushi Du, Wenjun Gao, Xuan Lu, Qinyu Chen, Yaohui Wang, Chengqi Deng, Jiashi Li, Chenggang Zhao, Chong Ruan, Fuli Luo, and Wenfeng Liang. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. *arXiv preprint arXiv:2406.11931*, 2024. doi: 10.48550/arXiv.2406.11931. URL https://arxiv.org/abs/2406.11931.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11(53):1605–1641, 2010. URL https://www.jmlr.org/papers/v11/el-yaniv10a.html.

Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971. doi: 10.1016/0004-3702(71)90010-5. URL https://doi.org/10.1016/0004-3702(71)90010-5.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059. PMLR, 2016. URL https://proceedings.mlr.press/v48/gal16.html.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, pp. 4878–4887, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. ToolkenGPT: Augmenting frozen language models with massive tools via tool embeddings. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45870–45894, 2023. doi: 10.52202/075280-1988. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8fd1a81c882cd45f64958da6284f4a3f-Abstract-Conference.html.

Jie He, Jennifer Neville, Mengting Wan, Longqi Yang, Hui Liu, Xiaofeng Xu, Xia Song, Jeff Z. Pan, and Pei Zhou. GenTool: Enhancing tool generalization in language models through zero-to-one and weak-to-strong simulation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 1097–1122, 2025. doi: 10.18653/v1/2025.findings-acl.61. URL https://aclanthology.org/2025.findings-acl.61/.

Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, Yue Xing, Jiliang Tang, and Benoit Dumoulin. TRAJECT-Bench: A trajectory-aware benchmark for evaluating agentic tool use. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TZWnWvsQ0X.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Hkg4TI9xl.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Tairan Huang, Siyu Shang, Qiang Chen, Xiu Su, and Yi Chen. Tools as continuous flow for evolving agentic reasoning. *arXiv preprint arXiv:2605.07339*, 2026. doi: 10.48550/arXiv.2605.07339. URL https://arxiv.org/abs/2605.07339.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6402–6413, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

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. URL https://aclanthology.org/2023.emnlp-main.187/.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, 2017. doi: 10.1109/ICCV.2017.324. URL https://doi.org/10.1109/ICCV.2017.324.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Xun Liu, Yi Evie Zhang, Vira Kasprova, Parisa Rabbani, Pardis Sadat Zahraei, Tianyu Zhang, Ali Ebrahimpour-Boroojeny, and Varun Chandrasekaran. AgentAbstain: Do LLM agents know when not to act? *arXiv preprint arXiv:2607.10059*, 2026a. doi: 10.48550/arXiv.2607.10059. URL https://arxiv.org/abs/2607.10059.

Yanming Liu, Xinyue Peng, Jiannan Cao, Xinyi Wang, Songhang Deng, Jintao Chen, Jianwei Yin, and Xuhong Zhang. ToolGate: Contract-grounded and verified tool execution for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9653–9684, 2026b. doi: 10.18653/v1/2026.findings-acl.470. URL https://aclanthology.org/2026.findings-acl.470/.

Xiaoxin Lu, Ranran Haoran Zhang, and Rui Zhang. SIMMER: Benchmarking latent failures in LLM executable planning with a world model. *arXiv preprint arXiv:2606.14574*, 2026. doi: 10.48550/arXiv.2606.14574. URL https://arxiv.org/abs/2606.14574.

John M. Lucassen and David K. Gifford. Polymorphic effect systems. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 47–57. ACM, 1988. doi: 10.1145/73560.73564. URL https://doi.org/10.1145/73560.73564.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: A benchmark for general AI assistants. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, volume 29, pp. 2901–2907, 2015. doi: 10.1609/aaai.v29i1.9602. URL https://ojs.aaai.org/index.php/AAAI/article/view/9602.

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*, volume 37, pp. 126544–126565, 2024. doi: 10.52202/079017-4020. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e4c61f578ff07830f5c37378dd3ecb0d-Abstract-Conference.html.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/28e50ee5b72e90b50e7196fde8ea260e-Abstract-Conference.html.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551, 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.

Richard Shin, Christopher H Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7699–7715. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.608. URL https://aclanthology.org/2021.emnlp-main.608/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. RestGPT: Connecting large language models with real-world RESTful APIs, 2023. URL https://arxiv.org/abs/2306.06624.

Huiri Tan, Yikun Wang, Puyang Zhang, Shangyu Li, and Jiasi Shen. ETAS: An effect-typed language for agent systems. *arXiv preprint arXiv:2607.17780*, 2026. doi: 10.48550/arXiv.2607.17780. URL https://arxiv.org/abs/2607.17780.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023. doi: 10.48550/arXiv.2306.05301. URL https://arxiv.org/abs/2306.05301.

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 (Volume 1: Long Papers)*, pp. 16022–16076. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 50208–50232. PMLR, 2024a. URL https://proceedings.mlr.press/v235/wang24h.html.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=jp3gWrMuIZ.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Proceedings of the First Conference on Language Modeling*, 2024. URL https://colmweb.org/2024/AcceptedPapers.html.

Weiming Wu, Song-Lin Lv, Rui Zhu, Zi-Jian Cheng, and Lan-Zhe Guo. Can agents generalize to the open world? unveiling the fragility of static training in tool use. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. URL https://icml.cc/virtual/2026/poster/65179.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652, 2024. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

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. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xiuhui You, Jiayi Luo, Zichao Shen, Qingyun Sun, and Ziwei Zhang. ToolLIFT: Lifting tool-specific trajectories into function-level graphs for generalizable tool planning. *arXiv preprint arXiv:2608.03468*, 2026. doi: 10.48550/arXiv.2608.03468. URL https://arxiv.org/abs/2608.03468.

A Commitment-Audit Construction and Split Controls

The main suite contains REST APIs (Stripe, GitHub, Slack, Twilio, SendGrid; 118 tasks), CLI Tools (docker, kubectl, terraform, ansible, Helm; 94), Databases (PostgreSQL, MongoDB, Redis, Elasticsearch, ClickHouse; 108), Version Control (git, SVN, Mercurial, Perforce, Fossil; 86), Workflow Engines (Airflow, Prefect, Temporal, Argo; 92), and Cloud Storage (S3, GCS, Azure Blob, MinIO; 114). Each task includes initial sandbox state, an allowed interface and operation set, a natural-language goal, and executable end-state and protected-state checks.

Candidate ToolBench, API-Bank, and GAIA tasks are normalized into this record, filtered to multi-step resource operations, and deduplicated by normalized goal, parameter signature, and end-state

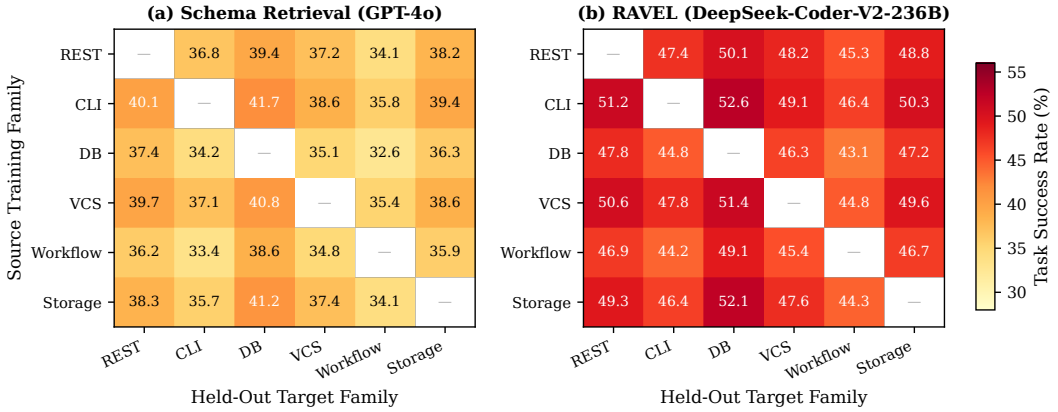


Figure 1: Pairwise single-source to target-family success matrix for Schema Retrieval and RAVEL. Columns are single-source averages and therefore differ from the five-source leave-one-family-out gains in Table 2.

predicate. The 87 authored variants hold goal semantics constant while changing naming, permission, and rollback conventions. Before each fold, a screen removes held-out product aliases, copied examples, canonical signatures, trace fragments, and prompt templates from the five source families. Product- and goal-grouped sensitivity in Appendix D removes the authored portion entirely.

Table 5: Benchmark inventory. An interface is one sandboxed provider wrapper; the checker unit identifies the persistent state used for terminal success and protected-state assertions.

Family	Interfaces	Tasks	Dominant transition	Checker unit
REST APIs	5	118	Record mutation	Remote resource
CLI Tools	5	94	Process/filesystem action	Filesystem/process state
Databases	5	108	Query/transaction	Database snapshot
Version Control	5	86	Revision transaction	Repository graph
Workflow Engines	4	92	Scheduled dependency	DAG/run state
Cloud Storage	4	114	Object/ACL mutation	Namespace and ACL
Total	28	612		

The inventory makes the split unit explicit: all operations and tasks attached to every interface in one row are target-side in that fold. Imported and authored records pass the same replay, checker, deduplication, and family-assignment pipeline.

Table 6: Complete per-family point estimates underlying the family-resolved analysis. The FT reference uses 15 labeled target-family examples per interface. The Average row is the unweighted family macro.

	Schema Ret.		Adapter		FT Ref.		RAVEL	
Family	SR	Cat	SR	Cat	SR	Cat	SR	Cat
REST APIs	41.2	9.8	42.8	8.7	44.3	7.8	50.4	4.1
CLI Tools	38.4	12.1	39.6	11.2	41.7	9.6	48.2	5.0
Databases	42.7	9.4	43.1	8.3	45.2	7.4	52.6	3.4
Version Control	36.1	13.3	37.8	12.4	39.4	10.8	44.6	5.8
Workflow Engines	33.8	14.6	35.2	13.1	37.6	11.4	38.7	7.2
Cloud Storage	42.4	8.0	43.3	7.9	44.4	7.6	55.1	2.7
Average	39.1	11.2	40.3	10.3	42.1	9.1	48.3	4.7

B Baseline and Training Details

Prompt + Docs inserts the complete target documentation into the model context. Schema Retrieval uses GPT-4o to retrieve source interfaces by schema embedding and adapt their call patterns. Code Generation asks GPT-4o to generate a Python wrapper executed in the sandbox. Adapter Transfer applies the source-trained DeepSeek LoRA to the held-out family. Tool-Specific FT starts from the same source model and adds 15 labeled examples per target interface. RAVEL and Adapter receive the same source documentation, schemas, traces, backbone, LoRA capacity, optimizer, and inference budget.

The master pool labels 120 seed operations across the six-family inventory. For fold m, preprocessing removes every F_m interface, operation label, and trace before bootstrapping or optimization. Bootstrapped sketches are checked against documentation, schema, and one replayable source trace; raters verify primitive order, scope, reversibility, permission, and validation slots. Stage 2 uses batches of 64 trajectories with gradient accumulation to an effective batch of 256, gradient clipping at 1.0, and validation every 500 updates. Early stopping monitors Equation 6 on a source-family validation split. The two stages and final calibration pass use three training seeds; each checkpoint is evaluated from the same sandbox state under twenty fixed decoder seeds.

Table 7: Operational semantics of the provider-independent effect program. "External call" describes what the primitive licenses during residual binding; declaring an effect never bypasses the pre-dispatch guard.

Primitive	Required environment and transition	External-call contract
RETRIEVE	Bound query or parent resource; adds an observed resource reference	Licenses a read-only lookup
FILTER	Observed collection and predicate; narrows resource identity	Licenses read-only selection or pagination
VALIDATE	Bound resource and condition; records a successful precondition observation	Licenses only the corresponding checked predicate
MUTATE	Bound resource, scope, and recovery type; stages an intended delta	Emits no call by itself
COMMIT	Covered mutation, satisfied validations, permission, and recovery obligation	Licenses the compatible state-changing call
ROLLBACK	Saved pre-state or inverse reference and rollback permission	Licenses a guarded compensating call

End-to-end execution example. For "delete the merged branch `release-x` from repository `r`," backbone states initialize repository, branch, merge-status, permission, delete, and restore nodes. The mapper emits a provider-independent sequence that retrieves `r`, filters `release-x`, validates the merged predicate, declares a reversible branch mutation, attaches the repository-write permission, and records the recovery reference. Only then does the residual decoder consider provider operations. It first binds a read-only branch query; the returned branch identifier, merge observation, and permission evidence update Γ. If the provider's delete is immediately effective, its compatibility set covers the adjacent MUTATE–COMMIT block, but Equation 4 prevents dispatch until the validation and authorization observations are present. The result advances both effects on a successful deletion response and retains the saved reference for rollback. The matched latent control receives the same graph, observations, capacity, and losses, but its code sequence cannot enforce this pre-dispatch type and permission check.

C Full Matched-Control Results

Table 8: Matched latent-plan structure ablation by family. It matches RAVEL in backbone, source data, annotation and verification budgets, corruption schedule, hard negatives, probing, abstention, mapper/decoder capacity, and plan length; unconstrained codes replace typed slots, composition rules, and operation masks. Overall Cat pools checkpoint–decoder–task trials; family Cat averages twenty decoder seeds for the median-validation checkpoint.

Family	Adapter		Matched latent plan		RAVEL	
	SR	Cat	SR	Cat	SR	Cat
REST APIs	42.8	8.7	47.4	6.3	50.4	4.1
CLI Tools	39.6	11.2	44.9	7.2	48.2	5.0
Databases	43.1	8.3	48.6	5.5	52.6	3.4
Version Control	37.8	12.4	41.9	8.2	44.6	5.8
Workflow Engines	35.2	13.1	37.1	8.9	38.7	7.2
Cloud Storage	43.3	7.9	50.6	4.6	55.1	2.7
Overall	40.3	10.1	45.1	6.8	48.3	4.7

RAVEL exceeds the matched control in every family. The largest margins occur in Cloud Storage and Databases, where constrained resource and permission slots align with inspectable state. Table 16 separately reports the exploratory temporal extension for Workflow Engines.

D Authored-Free Grouped Sensitivity

This regime removes all 87 authored variants, leaving 525 imported tasks and 0 authored tasks. We group by normalized goal template and product before family withholding, preventing near-goal variants from crossing partitions.

Table 9: Authored-free grouped transfer. Panel A reports all 525 imported tasks with SR as held-out-family fold mean $\pm$SE; Panel B isolates Workflow Engines. Cat intervals are task bootstrap.

Method	SR $\pm$SE	Cat [95% CI]
Panel A: all imported tasks		
Adapter Transfer	37.9$\pm$1.8	11.6 [10.1, 13.2]
Matched latent plan	42.8$\pm$1.6	8.9 [7.4, 10.3]
RAVEL	45.6$\pm$1.4	6.2 [5.1, 7.4]
Panel B: Workflow Engines		
Adapter Transfer	33.6	12.9
Matched latent plan	35.8	10.1
RAVEL	37.4	7.4

The authored-free split preserves RAVEL $>$ matched control $>$ Adapter overall and within Workflow Engines. Together with the independent AppWorld slice, this sensitivity demonstrates that the ordering persists across imported-only tasks, grouped goals, and a separately assembled executable suite.

E Commitment Components and Supervision Analysis

Table 10: RAVEL component ablations. The signature-similarity replacement is internal to the RAVEL stack and differs from Schema Retrieval despite the same displayed 39.1 SR.

Variant	SR	Cat	ΔSR
RAVEL	48.3	4.7	—
w/o effect sketches	39.1	9.8	-9.2
w/o residual adapter	43.6	7.4	-4.7
w/o abstention detector	47.1	11.3	-1.2
w/o low-risk probing	45.8	7.2	-2.5
w/o corruption training	44.2	6.1	-4.1
w/o hard negatives	46.4	5.9	-1.9
Random sketch assignment	33.8	16.4	-14.5

The component removals produce complementary observed drops. Removing effect programs yields the largest SR decrease; removing the residual adapter, corruption training, or probes also lowers SR and raises Cat; and removing abstention produces the largest Cat increase. The random-assignment result is consistent with a dependence on correct semantic alignment.

Table 11: Source-side information available to each method. All zero-target-label methods receive the same target documentation and 15-call budget.

Method	Src. doc.	Schema	Traces	Grammar	Eff. labels	Verify	Corr. train	Hard neg.
Prompt + Docs	N	N	N	N	N	N	N	N
Schema Retrieval	N	Y	N	N	N	N	N	N
Code Generation	N	N	N	N	N	N	N	N
Adapter Transfer	Y	Y	Y	N	N	N	N	N
Tool-Specific FT	Y	Y	Y	N	N	N	N	N
Matched latent	Y	Y	Y	N	Y	Y	Y	Y
RAVEL	Y	Y	Y	Y	Y	Y	Y	Y

Table 12: Source-side supervision cost. The grammar, annotation protocol, and verified master pool support all six folds; each training run filters the held-out family's labels and traces.

Quantity	Value
Manually labeled seed operations	120
Callable source operations	414
Seed coverage of operations	29.0%
Manual sketch labeling	68 person-hours
Bootstrapped operations reviewed	294
Human verification	41 person-hours
Evaluation-fold reuse	6 folds
Observed gain over Adapter	$+8.0$ SR, -5.4 Cat

This reusable source-side investment covers 29.0% of callable operations manually and amortizes one verified annotation protocol across six held-out-family evaluations, with target-family items excluded separately in every fold.

F Target-Supervision Scaling

RAVEL's zero-target-label SR is competitive with the 50-example reference while retaining lower Cat. Sufficient target supervision eventually matches the transfer system on completion, quantifying the target-label budget that the source-side resource representation replaces.

Table 13: Target-family supervision scaling under the same backbone, tasks, and 15-call budget. RAVEL uses zero target labels.

Method	Labels/interface	SR	Cat [95% CI]
Adapter Transfer	0	40.3	10.1 [8.6, 11.8]
Tool-Specific FT	5	41.0	9.8 [8.3, 11.4]
Tool-Specific FT	15	42.1	9.1 [7.7, 10.7]
Tool-Specific FT	50	49.5	6.5 [5.2, 8.1]
RAVEL	0	48.3	4.7 [3.6, 5.9]

G Abstention Audit and Security Evidence

Potentially harmful calls are independently labeled by two raters who inspect the task, gold end state, complete trace, documentation, and sandbox replay. The fixed taxonomy contains permission violations, wrong-resource modifications, and irreversible mutations without a safe recovery path. Disagreements are resolved by joint replay and adjudication. Parse errors, failed read-only probes, and refusals do not enter Cat unless an earlier call has already produced a harmful state transition.

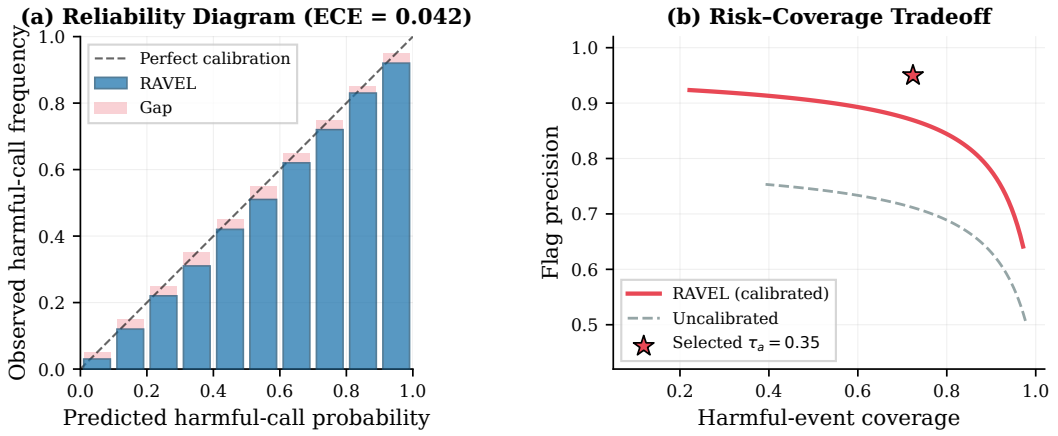


Figure 2: Decision-level calibration and precision–coverage. AUROC is 0.847 (95% CI [0.806, 0.883]), AUPRC is 0.793 ([0.742, 0.838]), and ECE is 0.042 ([0.031, 0.055]). The selected source-validation threshold reaches 95% precision and 72.4% event coverage ([66.2, 78.1]).

Table 14: Calibration and audit denominators. Coverage is event-level; abstention and Cat are task-level.

Quantity	Value
Train harmful / negative decisions	512 / 4688
Validation harmful / negative decisions	126 / 1074
Test harmful / negative decisions	214 / 1986
Flagged true harmful events at 95% precision	155
All predicted-positive decisions	163
Task-level abstentions	168
Audit raters / pre-adjudication κ	2 / 0.79
RAVEL task Cat 95% CI	[3.6, 5.9]
Adapter task Cat 95% CI	[8.6, 11.8]

Table 15: Task-level abstention utility at $\tau_a = 0.35$ for the median-validation checkpoint and predeclared canonical decoder seed. Accepted SR and Cat condition on tasks that proceed; aggregate metrics count abstentions as unsuccessful.

Slice	Abstain	Rate	Accepted SR	Accepted Cat	Calls
All families	168/612	27.5%	66.7%	6.5%	9.8
Cloud Storage	25/114	21.9%	70.8%	3.4%	9.2
Workflow Engines	39/92	42.4%	67.9%	13.2%	10.7

H Workflow Failure Anatomy

Table 16: Temporal mechanism probe on Workflow Engines and the other-five-family aggregate. SR, ICR, and Calls average checkpoint–decoder trials; Cat averages twenty decoder seeds for the median-validation checkpoint.

Method	Workflow Engines				Other five	
	SR	Cat	ICR	Calls	SR	Cat
Adapter Transfer	35.2	13.1	21.4	11.5	41.3	9.7
Matched latent plan	37.1	8.9	17.6	10.8	46.7	6.4
RAVEL	38.7	7.2	16.9	10.6	50.2	4.2
RAVEL + temporal	46.9	4.8	12.8	10.3	50.0	4.1

The complete metric view preserves the same selective pattern as the main-text intervals: temporal effects improve Workflow completion and mutation integrity while the other-five-family aggregate remains stable. The category ledger below shows where the remaining failures and pre-gate harmful candidates concentrate.

At the canonical decoder seed, base-to-temporal failed-task counts change from 17 to 14 for scheduler state, 13 to 11 for dependency release, 9 to 8 for trigger timing, 7 to 6 for retry/backfill, and remain 5 and 5 for rollback and resource/permission failures. The corresponding pre-gate harmful-candidate counts change from 6/5/4/3/2/2 to 4/3/2/2/2/1 in the same order. Overall, the extension removes seven failed tasks and eight pre-gate harmful candidates, concentrated in delayed-state categories rather than a uniform tool-selection deficit.