PLAN AROUND CHANGES, LEARN FROM VIOLATIONS: PERSISTENT TASK STATE FOR LONG-HORIZON VLA AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Failure is not a diagnosis. In a failed long-horizon robot task, the user may have changed the goal, the world may have invalidated a grounding, the planner may be following an obsolete goal occurrence, or the policy may simply have executed poorly. Two common adaptation strategies reconstruct task state from accumulated history or train on failed trajectories. The first can forget commitments that still hold; the second can learn from events for which the agent was never responsible. We introduce COPE, a two-timescale adaptation framework built around persistent task state. It gives each commitment a stable occurrence identity and tracks whether that occurrence remains authoritative and grounded. At runtime, COPE revises affected commitments and recompiles the current planning problem. Across episodes, violations of still-valid commitments attach corrective supervision to the same occurrence; legitimate task and world changes remain state transitions rather than training targets. Across complementary controlled, simulator, and learned-policy studies, the same task-side state is exercised through both time scales. In controlled state scaling, local persistent updates remain stable as the state grows from 4 to 16 to 32 commitments, while full-state regeneration falls from 30/36 to 13/36 to 8/36 valid transitions; at 32 commitments, 27/28 full-state errors contain preservation drift. On 1,200 prospective records, occurrence-level responsibility improves the respective routing metrics by 6.67–11.50 percentage points over episode-level, terminal-skill, and flat-trace baselines. We further carry occurrence identity through actual OpenVLA adapter updates and occurrence-gated deployment, where all 18 dynamic runs switch at the designated task occurrence with exact planning and no history corruption or invalid transaction. Persistent commitments therefore tell an agent what to re-plan now and which component—if any—should learn later.

1 INTRODUCTION

A long robot task can change without starting over. A robot filling an order may receive a temporary replacement, then another interruption, before the user restores the original order. The original and temporary orders can prescribe the same object and destination. Recovering the right continuation requires identifying which occurrence of the order is authoritative and what work it still requires. Action identity alone does not resolve task occurrence identity.

The same ambiguity appears when execution fails. A revised request changes the planning problem. A displaced target can preserve the request while invalidating its grounding. A stale planner can follow an occurrence that is no longer authoritative. A visuomotor policy can fail to satisfy a requirement that still holds. These outcomes look similar at the end of an episode, yet they imply different responses. The agent must first answer a more basic question: *what is still supposed to be true, and which occurrence does the current evidence concern?*

Long horizons make that question increasingly difficult. Runtime revision can force a generative system to reproduce large amounts of task state that did not change. Learning from a final failed trajectory requires the system to decide which parts of an equally long history actually constitute a violation. Both problems depend on the same reference: the task commitment that remained

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.

authoritative at the time. An event history records what happened; the task state records which obligation remains authoritative after those events.

CoPE makes this reference persistent. We distinguish the agent's evolving belief about the physical world from an explicit task-side adaptation state,

$$z_t = \left(z_t^{\text{world}}, z_t^{\text{task}}\right).$$

The task component records what the robot still owes, by which occurrence, and under what authority. A commitment carries occurrence identity, lifecycle, lineage, grounding, restoration conditions, and append-only provenance. Its authority survives beyond the model call that created or revised it.

This state supports two adaptation regimes. At runtime, local task or world changes revise the affected commitments and a downstream planner re-optimizes from the robot's current continuation. Across episodes, a violation of a still-valid commitment reuses the same occurrence identity to assign corrective supervision to the responsible component. The same task identity can then control where a learned specialization is active at deployment. Responsibility therefore persists from task revision to failure attribution and back into execution.

The shared interface has three requirements: occurrences remain addressable, transitions preserve task authority, and attribution retains the provenance of the addressed obligation. A synchronized state and append-only log can realize this interface. The preserved occurrence supplies a common address for runtime revision, corrective supervision, and deployment scope.

Our evaluation asks three questions: does local revision preserve unaffected task state as history grows; does the retained occurrence improve prospective responsibility assignment; and does that identity remain usable through actual policy updates and scoped deployment? Controlled studies, simulator and Qwen/OpenVLA integrations, and a preregistered prospective-blind evaluation examine these links.

Our contributions are threefold. First, we formulate persistent task responsibility as an explicit adaptation state for long-horizon agents, separating evolving world belief from what the agent still owes. Second, we define occurrence-, lifecycle-, lineage-, grounding-, and provenance-aware transition semantics that preserve task authority through repeated changes. Third, we show that the same task state supports runtime recovery and prospective learning assignment, and remains usable through real OpenVLA policy updates and occurrence-gated deployment.

2 Related Work

Planning, replanning, and plan repair. Persistent goals and conditions for abandoning them are long-standing topics in agent theory (Cohen & Levesque, 1990). Situated robot replanning explicitly considers ongoing physical actions and interruption (Cashmore et al., 2019); plan repair studies how revised plans can retain useful structure (Saetti & Scala, 2022). A recent travel-agent study also compares commitment retention and edit locality under resource disruptions (Yuan et al., 2026). These studies evaluate planning and revision under a specified current task. CoPE maintains the task authority from which that planning problem is constructed after repeated suspension, override, restoration, and re-grounding.

Language-conditioned robot agents. Language-conditioned robot systems combine high-level reasoning with skills, feedback, and grounded constraints (Ichter et al., 2023; Singh et al., 2023; Huang et al., 2023; 2025). They make planning interfaces increasingly expressive. Long-horizon adaptation adds another requirement: the system must preserve which request occurrence a current plan refers to as tasks are interrupted and resumed. CoPE makes that task identity explicit and carries it into later adaptation decisions.

Memory and task state. Memory-based agents retrieve task-relevant past information or experience (Sridhar et al., 2026). Retrieval answers which parts of history should be recalled for the next decision. Persistent commitment state answers which obligation is authoritative now, which occurrence the evidence refers to, how that occurrence reached its current lifecycle state, and where a later correction should apply. Recent structured-state systems make related runtime information explicit: LocalLSTC maintains subgoals, evidence, and feedback for GUI control (Li et al., 2026), while

verification-gated mission governance maintains a task forest and blackboard with verified atomic updates (Tang et al., 2026). COPE uses the retained task occurrence as the address for subsequent learning responsibility and deployment scope.

Failure explanation and continual adaptation. Failure-explanation systems such as AHA (Duan et al., 2025) identify when manipulation breaks and why. Continual and post-training methods then decide how parameters should change. COPE supplies the task-side state that precedes that decision: failure evidence is interpreted against the occurrence that was active, applicable, and grounded when the violation arose.

3 PERSISTENT RESPONSIBILITY AND TWO-TIMESCALE ADAPTATION

3.1 TASK STATE AND EXECUTION CONTEXT

Let C_t be the committed task state after processing registered event e_t, with initial state C_0. A receipt J_i records the interpretation, proposed operation, and acceptance or rejection of event e_i. Write $J_{1:t} = (J_1, \dots, J_t)$. With a fixed executor, the state is replayable:

$$z_t^{\text{task}} = C_t = \Phi(C_0; J_{1:t}), \qquad C_t = (S_t, \mathcal{G}_t, L_t). \tag{1}$$

S_t contains commitment slots. An *occurrence* is one registered instance of a commitment; equal task predicates can have different occurrence IDs. Each slot records its stable occurrence ID, logical task ID, payload, lifecycle, grounding, priority/restoration conditions, and evidence. $\mathcal{G}_t$ records relations such as override parentage and root ancestry. L_t is append-only provenance of accepted transitions; rejected attempts are recorded in J while leaving C_t unchanged.

The execution context is $E_t = (W_t, \Pi_t, \kappa_t)$, where W_t is the current world estimate, Π_t is verified progress, and κ_t is the current physical/program continuation. Let x_t be the current observation. An object can move while an obligation survives; a user can cancel an obligation while the scene remains unchanged. Task authority and world belief evolve separately.

Let a_t be a planning/recovery decision, v_τ public evidence concerning an execution check at physical time τ, and R_τ its learning assignment. Write $C_\tau = C_{t(\tau)}$, where $t(\tau)$ is the index of the latest completed transition attempt at that time; E_τ is the recorded execution context. The runtime and learning consumers use these task references:

$$\pi_{\text{run}}(a_t \mid x_t, C_t, E_t), \qquad \pi_{\text{learn}}(R_\tau \mid v_\tau, C_\tau, E_\tau). \tag{2}$$

Attribution uses the recorded snapshot even when the route is computed later. Both consumers can therefore address the same occurrence.

3.2 APPLICABILITY, GROUNDING, AND SATISFACTION

For an occurrence o, applicability $A_t(o)$ records whether it has authority, grounding validity $\gamma_t(o)$ records whether it refers to the intended entities and world state, and satisfaction $Q_t(o)$ records whether its task predicate has been fulfilled. Cancellation changes applicability. Moving a target can invalidate grounding while preserving authority. An execution failure can leave applicability and grounding valid while satisfaction is false. These distinctions determine state revision, re-grounding, recovery, and learning.

3.3 PERSISTENT STATE TRANSITIONS

The editor proposes $P_{t+1} = f_\theta(C_t, E_t, e_{t+1})$, specifying the affected occurrence, typed operation, and local arguments. Parsing and deterministic application produce a candidate $\widehat{C}_{t+1}$, or a failure marker $\perp$. Let b_{t+1} be true exactly when a candidate exists and passes the Boolean validator $\text{Validate}(C_t, \widehat{C}_{t+1}, P_{t+1}, e_{t+1})$. Then

$$C_{t+1} = \begin{cases} \text{Commit}(\widehat{C}_{t+1}), & b_{t+1} = 1, \\ C_t, & b_{t+1} = 0. \end{cases} \tag{3}$$

Validation checks occurrence references, legal lifecycle changes, reciprocal and acyclic lineage, append-only provenance, and the live-carrier requirement for the critical workflow. These checks preserve identity outside the declared update, maintain authority, and retain the evidence behind accepted changes. Appendix E states the exact preservation conditions.

The principal operations are Suspend, Override, Expire, Revalidate, Restore, and RestoreRoot. Suspend preserves identity while removing an occurrence from the active planning set. Override creates a child and transfers active authority. Expire ends future participation while retaining the record. Revalidate refreshes grounding or restoration evidence. Restore reactivates an eligible occurrence; RestoreRoot resolves the root of its lineage before reactivation. These operations cover the authority changes exercised here.

3.4 Learning Assignment

A learning candidate must remain applicable and grounded at the checked step:

$$\Lambda_\tau(o) = A_\tau(o) \wedge \gamma_\tau(o) \wedge \neg Q_\tau(o) \wedge V_\tau(o). \tag{4}$$

$V_\tau(o)$ records evidence that a due or tested requirement of this occurrence was breached. Thus an unfinished future requirement alone does not trigger learning. Occurrence identification determines which obligation the evidence concerns; module attribution determines which component receives corrective supervision. The route records

$$R_\tau = (\mathcal{O}, \mathcal{M}, \mathcal{W}, u), \tag{5}$$

where $\mathcal{O}$ and $\mathcal{M}$ are occurrence and module target sets, $\mathcal{W}$ is the evidence window, and u is update, no-update, or abstain. The module vocabulary is editor, planner, policy, and verifier. An update requires nonempty targets and a specified window; other decisions have empty targets. The serialized route also records the first violation and target schema (Appendix F). Correction targets are obtained after the prospective route is fixed.

3.5 Two-Timescale Decisions

Runtime revision, learning assignment, and deployment share an occurrence address. Revision determines current authority; later evidence is assigned to the occurrence authoritative at the checked step; deployment consults current authority to scope the resulting specialization. Appendix E formalizes this continuity across state representations.

4 CoPE

Figure 1 diagrams this shared occurrence interface; Appendix F gives the comparison algorithms.

4.1 Specification Front End

The specification front end normalizes registered user changes, world events, and execution evidence into a public event record. It exposes the current commitment snapshot and execution context together with the observable event. Correction targets remain outside this packet so that runtime interpretation and prospective learning evaluation share the same public evidence.

4.2 Patch Reasoner

The patch reasoner maps the event record to a target occurrence and typed local transition. In the model-driven studies, the exposed vocabulary contains suspension, restoration, override, and root restoration; the simulator realization additionally exercises insertion, expiration, revalidation, and preservation.

4.3 State Transition Operator

The state transition operator applies the selected patch to the persistent responsibility state. It updates coupled lifecycle, lineage, and provenance fields and runs the validity checks in Equation (3). Its purpose is to preserve the semantics of the task state while the model handles event interpretation.

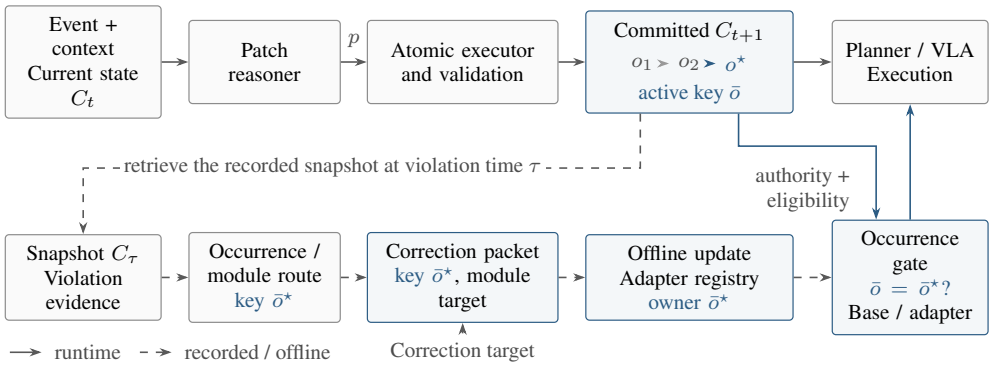

Figure 1: CoPE carries an occurrence key through task revision, violation attribution, and deployment. The learning lane depicts an accepted policy-update route; no-update and abstention decisions produce no correction packet. Validated transactions update the committed task state. Routing uses the recorded state at violation time, and a correction target enters at the packet stage. The registry associates an adapter with its occurrence owner; the gate matches current authority and eligibility to select the base or adapted policy. The inset depicts an override lineage.

4.4 Bounded Fallback

If validation rejects a proposed transition, the committed state remains unchanged. The implementation can then request a bounded retry, fall back to a conservative continuation, or defer the event. Only accepted transitions update the authoritative task state.

4.5 Planner Compiler

The planner compiler maps the committed task state and execution context to a current recovery problem. In the LIBERO/MuJoCo integration it preserves verified progress, constructs a repair goal, searches a short recovery continuation, and resumes from the interrupted physical state. The downstream planner consumes the task state; it does not reconstruct task authority itself.

4.6 VLA Compiler

The learned-policy path exposes the same committed state through a public execution interface. Qwen-based high-level modules consume task/evidence state, while OpenVLA (Kim et al., 2025) executes the resulting visuomotor continuation. The occurrence identifier is retained through the handoff.

4.7 Repair Assignment

For an update route supported by Equation (4), CoPE constructs a packet indexed by the canonical occurrence. The packet includes the responsible module, first violation time, selected observation/action window, correction target, and provenance. Legitimate task/world changes update state without entering this learning route.

4.8 Module-Specific Continual Adaptation

For the policy-update study, the visuomotor policy adapter is the trainable module. CoPE, Generic Persistent, and All-Failure start from the same OpenVLA-7B parent checkpoint and receive matched clean replay, optimizer steps, batch schedule, learning rate, and training-stream seeds. Only the correction packets selected by the routing rule differ. Child adapters are saved, reloaded, and evaluated on frozen states.

Table 1: Prospective-blind responsibility routing.

Experiment	Comparator	Δ accuracy	Paired 95% CI	Holm p
Exp2A	All-Failure Episode	+6.67 pp	[1.25, 12.17]	1.26×10^{-11}
Exp2A	Terminal Skill	+7.17 pp	[4.17, 10.25]	5.21×10^{-12}
Exp2B	All-Failure Episode	+11.50 pp	[6.92, 16.25]	1.62×10^{-8}
Exp2B	Flat Trace	+10.25 pp	[7.42, 13.25]	5.56×10^{-13}

4.9 Executable Realization

The complete implementation follows

$$\begin{aligned}&\text{event} \rightarrow \text{commitment transition} \rightarrow \text{planner/VLA continuation}\\ &\qquad \rightarrow \text{violation assignment} \rightarrow \text{offline update} \rightarrow \text{occurrence-gated deployment.}\end{aligned} \tag{6}$$

A correction assigned to one occurrence is activated only when that occurrence is authoritative; other states continue through the base policy.

5 Experiments

5.1 Experimental Setup and Protocols

The experiments examine the interfaces in Equation (6). Controlled mechanism studies test state maintenance under growing histories. Live integrations test whether the same state remains executable. Prospective-blind routing tests whether occurrence-level evidence identifies the responsible learning target before correction packets are visible. Post-training and deployment then carry that identity into real policy parameters and behavior.

We keep evidence sets separate: provider-call cases are the unit in controlled state scaling, paired task seeds are the unit in the simulator study, records are the unit in prospective routing, and training-stream seeds are the unit in the policy-update study.

5.2 Unified Mixed-Event Adaptation

The mixed-event benchmark combines legitimate task revisions, exogenous world changes, stale planning state, module-specific violations, and cases in which no parameter update is warranted. Every non-oracle route is committed from public evidence before the corresponding correction target becomes accessible. The route therefore has to resolve task authority prospectively.

The frozen evaluation contains 1,200 records and 24 predictors. Exp2A evaluates joint credit accuracy; Exp2B evaluates exact module/decision accuracy. Across four preregistered comparisons, occurrence-level responsibility improves the respective routing metric by 6.67–11.50 percentage points over coarse attribution baselines (Table 1). These comparisons support the use of occurrence-resolved task evidence for prospective responsibility assignment against the listed coarse baselines.

5.3 Repeated-Interruption Runtime Adaptation

Persistent-state scaling. We evaluate CoPE, a neutral typed sparse editor, a compact transaction editor, and full-state regeneration over 4, 16, and 32 persistent commitments. All arms receive the same common task/event information, model, seed, temperature, and completion ceiling. Table 2 reports transition validity.

The local interfaces remain stable as state grows, while FSR falls from 30/36 to 8/36. At 32 commitments, 27 of 28 FSR errors contain preservation drift outside the requested transition. Aggregate CoPE completion tokens stay near 6.9k across all three sizes, while FSR grows from 26,421 to 191,787; aggregate latency grows from 148.0 s to 996.9 s for FSR and from 56.2 s to 68.4 s for CoPE. The sparse local-update interfaces preserve unchanged records in storage, reducing the state that must be generated for a local change.

Table 2: Persistent-state scaling. Each cell reports valid post-state transitions out of 36 cases.

Commitments	CoPE	Neutral typed	Compact transaction	FSR
4	35/36	34/36	32/36	30/36
16	36/36	34/36	33/36	13/36
32	36/36	34/36	33/36	8/36

Table 3: Paired simulator representation study. NED is normalized edit distance; complete regeneration has NED$= 1$ by construction.

Stage	Method	Success	Identity	NED	Audit
B	CoPE	17/20	–	–	–
B	FSR-PC	17/20	–	–	–
B	No adaptation	3/20	–	–	–
C	CoPE	9/15	15/15	0.433	1.000
C	FSR-PC	9/15	0/15	1.000	0.344
C	FSR-PC + stable IDs	9/15	15/15	1.000	0.344
C	FSR-PC + provenance	9/15	0/15	1.000	0.933

Paired simulator study. We connect the state representation to a shared LIBERO/robosuite/MuJoCo recovery stack (Liu et al., 2023; Todorov et al., 2012). Conditions cover cancellation and redirection, temporary target unavailability, repeated redirection, restoration after a basket moves, and nested override/restoration. The final audit contains 150 formal episodes and 315 delivered events with zero controller or infrastructure failures. This total includes Stages B, C, and D; Table 3 reports the B/C comparisons.

Stage B compares CoPE, FSR-PC, and no adaptation. CoPE and FSR-PC each complete 17/20 tasks; no adaptation completes 3/20. Stage C uses the same physical execution backend while decomposing representation properties. All compared state-maintenance variants complete 9/15 tasks, while their state records differ (Table 3). CoPE preserves all pre-interruption occurrence IDs, touches only the affected subset of state, and answers the complete applicable audit set from its committed transition history. Audit is the fraction of applicable historical-authority queries answerable from retained state and provenance; Appendix A defines the metrics.

Learned-VLA integration. A Qwen3-32B/OpenVLA path (Yang et al., 2025) exercises the same state through learned visuomotor control. The reported integration totals are 32/32 trajectories, 160/160 durable event cells, 64 high-level reasoner calls, and 15,030 learned OpenVLA action calls under common public evidence, fresh observations, and prior-failure propagation. These integration counts document the execution of the state-to-policy interface; task-success comparisons are reported separately in Table 3 and Appendix B.

5.4 Actual Continual Post-Training

We instantiate the learning consumer with an OpenVLA-7B LoRA action adapter (Hu et al., 2022). Matched arms begin from the same parent checkpoint and share 105 unique terminal-butter clean-replay examples, the same replay ordering, 6 rounds of 32 optimizer steps, learning rate 10^{-6}, and the same three training-stream seeds. The routing rule determines which correction packets enter the fixed update budget.

The counts of selected corrections matching the active butter occurrence under the registered typed-match criterion are 30/30/33 for CoPE, 9/12/17 for All-Failure, and 0/0/0 for Generic Persistent. The slash-separated entries give one count per training stream, rather than fractions or total supervision counts. Each child adapter is saved, reloaded, and evaluated on the same dynamic states. Appendix B reports the three-stream policy-success outcomes. This study exercises occurrence-addressed routing through real parameter updates and reloaded policy execution.

We then carry responsibility into deployment. The base policy runs by default, and the specialized adapter is activated once the corresponding commitment occurrence becomes authoritative. In the completed 18-cell dynamic diagnostic, every run performs exactly one occurrence-conditioned switch, uses the base policy beforehand and the CoPE adapter afterward, preserves exact planning, and records zero history corruption and zero invalid transactions. In a separate clean-retention diagnostic, all 9/9 cells remain on the base policy and complete successfully. The same occurrence identity has now been used for runtime state, learning assignment, and parameter scope.

5.5 Validation under Injected Faults

The responsibility state is checked before it can drive planning or learning. Typed transitions reject illegal lifecycle and lineage changes; an independent completeness validator detects missing requirements. Across the complete targeted validation pipeline, 0/88 faults fail open: none is accepted as a valid authoritative-state update. A decomposed checker accepts 48/48 clean predicates and rejects 88/88 targeted faults with 136/136 parity against the exact checker; predicate-composition tests reject 64/64 mutations, and a structural-fuzz holdout yields 3,000/3,000 parity. The validation summary appears in Appendix D.

5.6 VLA Interface and Generalization

The tested stack combines Qwen-based high-level modules, OpenVLA-7B, LIBERO/robosuite, and the public commitment/execution interfaces described above. The same occurrence state survives the handoff from task revision to learned visuomotor execution and then back into the learning route. The tested interface connects one VLA family and one controlled robot environment.

5.7 Ablations and Classical Controls

The local-edit controls in Table 2 retain sparse generation; Neutral Typed also shares the semantic executor. Their performance supports local mutation as a mechanism for state-scaling robustness. The stable-ID and provenance variants in Table 3 separate identity preservation from historical auditability. Appendix C reports the distinct 40-case neutral transaction comparison; Appendix F specifies the named control interfaces. Routing and deployment examine the later uses of the retained occurrence state.

5.8 Efficiency and Scaling

The measured efficiency benefit comes from the local-update interface. Model output scales with the requested transition while unchanged commitments remain in the task state. The 4/16/32 scaling study therefore measures both preservation and generation cost: CoPE output remains nearly constant while full-state serialization grows with accumulated history. Format-level byte comparisons are secondary to this state-scaling effect and are reported with the alternative representation controls in the appendix.

6 Structural Analysis of CoPE

Local revision. Starting from a valid state, an atomic transition preserves global validity and the semantic state outside its write footprint when writes stay within that footprint and validation covers all affected and newly active invariants. This result assumes a fixed validation specification with sound dependency supports and states within its trusted domain (Proposition E.3). It explains the preservation mechanism examined in Table 2; Corollary E.6 relates patch and full-state generation to their required output lengths.

Responsibility assignment. Under the same public evidence, a representation supports a correct reference decision exactly when it distinguishes histories requiring different decisions (Proposition E.7). Occurrence identity and lineage retain such distinctions for restoration and violation attribution. This connects the representation to the routing comparisons in Table 1, which measure how accurately learned routers use their supplied inputs.

Scoped deployment. If the occurrence gate stays closed almost surely along base-policy trajectories, gated and base executions have the same full trajectory law, provided they share the initial joint distribution and conditional transition kernels of the environment and complete controller state (Proposition E.8). This characterizes the base branch examined by the clean gated-retention diagnostic in Appendix B. Appendix E gives the formal definitions, complete assumptions, and proofs; Appendix F specifies the CoPE and baseline algorithms.

7 Discussion and Limitations

7.1 What CoPE Is

CoPE treats task responsibility as a persistent adaptation state. The occurrence that tells the planner what remains owed is preserved through task revision, later failure attribution, and deployment of a learned correction. Planner, VLA, validator, and adapter remain replaceable consumers of that state. The representation supplies continuity among them.

7.2 Evidence Scope

The experiments measure state preservation, prospective routing, and execution of policy updates at distinct interfaces. The post-training study uses three independent training streams. It demonstrates actual updates and deployment scope; its observed success counts do not establish a consistent advantage in learning efficiency. The clean gated runs exercise the base-policy branch, so their retention follows the gate's scope mechanism.

7.3 Responsibility Across Implementations

A synchronized state and event log realizes the responsibility interface when it preserves occurrence identity, authority, lifecycle, lineage, grounding, and provenance. Lemma E.9 states the transition and consumer conditions for equivalent behavior.

7.4 Generalization

The evaluated stack uses Qwen-based reasoning, OpenVLA, and a controlled LIBERO/robosuite environment. Broader backbones and physical platforms remain to be evaluated. The state layer receives registered task/world events, and its guarantees depend on the supplied event interpretation and validation contracts. Missed perceptual events and tasks without a discrete occurrence decomposition require additional modeling.

8 Conclusion

Failure is not a diagnosis. CoPE treats persistent commitments as the shared task state behind two responses to failure. At runtime, affected occurrences are revised, unaffected commitments are retained, and the current planning problem is recompiled. Across episodes, violations of still-valid commitments route learning responsibility back to the task occurrence that generated it. The same identity can then determine where the learned specialization acts.

Beliefs describe what an agent estimates to be true. Persistent commitments tell it what it still owes the task. Long-horizon adaptation requires both.

AI Use Statement

Generative AI tools assisted with writing and editing, literature retrieval and discovery, and research ideation and execution, including implementation of code and the checking of the structural analysis and proofs (completed by the authors and checked with AI assistance). The authors take responsibility for the final manuscript, code, and results.

REFERENCES

Michael Cashmore, Andrew Coles, Bence Cserna, Erez Karpas, Daniele Magazzeni, and Wheeler Ruml. Replanning for situated robots. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 665–673, 2019. https://doi.org/10.1609/icaps.v29i1.3534.

Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2–3):213–261, 1990. https://doi.org/10.1016/0004-3702(90)90055-5.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. AHA: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *International Conference on Learning Representations*, 2025. https://proceedings.iclr.cc/paper_files/paper/2025/hash/70a06501001e1820fd1eb9ee821302d2-Abstract-Conference.html.

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. https://openreview.net/forum?id=nZeVKeeFYf9.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 1769–1782. PMLR, 2023. https://proceedings.mlr.press/v205/huang23c.html.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 4573–4602. PMLR, 2025. https://proceedings.mlr.press/v270/huang25g.html.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as I can, not as I say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 287–318. PMLR, 2023. https://proceedings.mlr.press/v205/ichter23a.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2025. https://proceedings.mlr.press/v270/kim25c.html.

Weiming Li, Helen Paik, and Yulei Sui. LocalLSTC: A long short-term control architecture for locally deployed GUI agents. *arXiv preprint arXiv:2608.25777*, 2026. https://arxiv.org/abs/2608.25777.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 44776–44791,

2023. https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie (eds.), *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. https://www-formal.stanford.edu/jmc/mcchay69.html.

Alessandro Saetti and Enrico Scala. Optimising the stability in plan repair via compilation. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pages 316–320, 2022. https://doi.org/10.1609/icaps.v32i1.19815.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation*, 2023. https://doi.org/10.1109/ICRA48891.2023.10161317.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Scaling up memory for robotic control via experience retrieval. In *International Conference on Learning Representations*, 2026. https://proceedings.iclr.cc/paper_files/paper/2026/hash/9da515b1ad19d032a7398f00f5ff9b0c-Abstract-Conference.html.

Guoqin Tang, Qingxuan Jia, Yichen Tan, Zeyuan Huang, Ning Ji, and Gang Chen. Verification-gated agentic mission-state governance for intelligent industrial multi-robot systems. *arXiv preprint arXiv:2606.31339*, 2026. https://arxiv.org/abs/2606.31339.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. https://doi.org/10.1109/IROS.2012.6386109.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. https://arxiv.org/abs/2505.09388.

Xiaofei Yuan, Yan Zhang, Shaobo Qiao, Huangleshuai He, Leyan Ni, Mingchen Ju, Lujia Yang, Sijia Xu, Yifu Tang, and Zhengyi Yang. Replan, repair, or edit? A unified empirical evaluation of travel agents for itinerary revision under resource disruptions. *arXiv preprint arXiv:2609.19654*, 2026. https://arxiv.org/abs/2609.19654.

A Detailed Runtime Protocols

A.1 Simulator Conditions

The basket task begins with milk and yogurt assigned to basket A and butter to basket B. I1 cancels yogurt and redirects butter. I2 suspends and restores butter while its target is unavailable. I3 redirects butter twice and adds a safety requirement. I4 moves basket B during suspension, requiring fresh grounding. I5 withdraws a temporary butter target and restores the original after B moves. Event schedules depend on condition and seed and are fixed independently of method.

A.2 State Metrics

Identity preservation requires every pre-interruption slot ID to remain represented. Normalized edit distance is the fraction of slots touched or re-emitted per interruption, averaged within and then across episodes; complete regeneration equals one by construction. Audit coverage counts condition-specific questions answerable from the retained operations, snapshots, or provenance, including which goal changed, what progress survived, and why suspension or restoration occurred.

B Policy-Update Details

The matched three-seed policy-update study evaluates CoPE, All-Failure, and Generic Persistent on the same six dynamic states. The observed success counts are 9/18, 8/18, and 5/18, respectively. Each arm undergoes actual training, checkpoint reload, and evaluation. The success counts describe the resulting behavior under the matched update budget. Learning-target selection and deployment scope are separate decisions; the following diagnostic evaluates the latter.

The global-adapter retention diagnostic uses three clean states and three training-stream seeds. The corresponding success counts are 5/9 for global CoPE, 8/9 for All-Failure, and 6/9 for Generic Persistent, with the round0 reference succeeding on 3/3 states. The occurrence-gated follow-up routes all nine clean cells to round0 and succeeds on 9/9; because the specialized adapter is inactive, these nine cells collapse to three distinct state-level trajectories.

C Alternative Representation Controls

A held-out 40-case sparse-state control compares CoPE with an information-matched Neutral Transaction (40-case protocol). Both produce semantically correct runtime transitions on 40/40 cases. This control compares two sparse runtime interfaces at matched semantic correctness. Under matched correctness, CoPE averages 272 proposal bytes, 117 completion tokens, and 1.24 s latency, compared with 1,711 bytes, 790 tokens, and 4.70 s for Neutral Transaction. This protocol is distinct from the Neutral Typed operation-alias control in Table 2.

Table 1 reports the preregistered comparisons against coarse attribution baselines. The Generic Persistent routing view retains occurrence identifiers, the active occurrence, and plan-suffix targets; its interface is specified in Appendix F. The listed comparisons establish their respective coarse-baseline differences.

C.1 Prospective Routing Protocol

Exp2A reports joint credit accuracy and Exp2B reports exact module/decision accuracy. The 24 frozen predictors cover six methods across four leave-one-task-out folds of the 1,200-record benchmark. Each route is fixed from public inputs before the corresponding target and held-out labels are revealed. Table 1 retains the registered paired-bootstrap confidence intervals and Holm-adjusted exact-test results.

D Responsibility-State Integrity

Targeted commission and structural faults are rejected by the typed transition layer before commit. Omission faults require the independent completeness validator because a locally well-formed transition can still omit a requirement. Across the complete shared-validation pipeline, 0/88 targeted faults fail open. A decomposed validator accepts 48/48 clean predicates and rejects 88/88 targeted faults, with 136/136 parity against the exact checker. Predicate-composition tests reject 64/64 single/double/triple mutations with attribution, and a structural fuzz holdout yields 3,000/3,000 parity.

E Structural Analysis of CoPE

The analysis follows one occurrence through revision, attribution, and deployment. Figure 1 shows the shared interfaces. The propositions below establish structural properties under explicit transition and policy contracts, linking the state-scaling, responsibility-routing, and gated-deployment experiments to their respective mechanisms.

E.1 Occurrences, Relations, and Time

Definition E.1 (Commitment occurrence)**.** An occurrence is a registered instance of a task commitment. Its record has an immutable key, a logical task identifier and payload, a lifecycle mode, a grounding status, authority and restoration conditions, and evidence references. An episode-local key is a pair $(\mathrm{trace_id}, o)$. Two occurrences may have the same payload and different keys.

The lifecycle domain of the persistent-ledger interface is `active`, `suspended`, `overridden`, and `expired`. Its grounding domain is `valid`, `degraded`, `unknown`, and `invalid`. In $C_t = (S_t, \mathcal{G}_t, L_t)$, $\mathcal{G}_t$ is the relation structure and L_t is committed provenance. The Boolean grounding predicate $\gamma_t(o)$ equals one exactly when occurrence o has grounding status `valid`.

Definition E.2 (Typed occurrence relations)**.** Let O_t be the set of occurrence keys. The relation structure is

$$\mathcal{G}_t = (O_t, \mathsf{Rel}_t^{\mathrm{override}}, \mathsf{Rel}_t^{\mathrm{dependency}}, \ldots).$$

An override edge records parentage between task occurrences; a dependency edge records a semantic dependency used by a transition or restoration condition. Each edge type has its own validity rules. The override relation is acyclic; reciprocal stored pointers must agree. A designated root or a unique-parent contract determines a restoration root. Uniqueness of the live carrier is enforced within each workflow subject to an exclusive-authority constraint.

Multiple independent commitments can be active simultaneously. Acyclic override parentage and semantic dependencies therefore play different roles: the former describes lineage, while the latter determines the fields a legal transition must read, update, or validate. The invariant supports introduced below describe these dependencies at the field level.

Committed replay. Let t index registered transition attempts and let C_t be the committed state after attempt t. Each receipt J_t records the event, interpreted patch or parsing failure, validation outcome, and state delta before the receipt is appended. Write $J_{1:t} = (J_1, \ldots, J_t)$. With the initial state C_0 and a fixed executor, the replay relation in Equation (1) is

$$C_t = \Phi(C_0; J_{1:t}) = \mathrm{Replay}(C_0, J_1, \ldots, J_t). \tag{E.1}$$

An accepted transition appends its receipt to committed provenance L. Rejected attempts leave C, including L, unchanged; their receipts enter the separate attempt log $J_{1:t}$. Replay uses the recorded interpretation and effect, so it does not rerun the proposal model. The next event e_{t+1} is processed from C_t.

Let τ denote physical evidence time and let $t(\tau)$ identify the latest completed transition attempt visible at that time. In attribution, C_τ is shorthand for $C_{t(\tau)}$, and E_τ is the recorded execution context at that physical time. Equation (4) is evaluated against this snapshot and the corresponding violation evidence:

$$\Lambda_\tau(o) = A_\tau(o) \wedge \gamma_\tau(o) \wedge \neg Q_\tau(o) \wedge V_\tau(o).$$

Here $V_\tau(o)$ records evidence of a breached condition when that condition was due or tested. Applicability and valid grounding make the occurrence eligible for responsibility analysis; the evidence and module interface then determine the route. A later task revision changes subsequent authority.

Transition contracts. For each operation, a trusted specification supplies admissible targets, preconditions, coupled field updates, and postconditions. Suspend preserves identity while changing participation; Override adds an occurrence and updates parentage and authority; Expire changes lifecycle while retaining its record; Revalidate updates grounding evidence; Restore and Restore-Root use the specified restoration guard and lineage target. The executor expands the operation into the required field updates before validation. Event correctness, written $T_e(C, C')$, states that the

transition implements the registered request. Structural validity states that the resulting records and relations satisfy their invariants; the next subsection defines its task-semantic predicate I. Completeness checks connect the event's requirements to the proposed transition, as in Appendix D.

This treatment of retained state builds on the frame problem: action descriptions specify changes while reasoning accounts for the facts that persist (McCarthy & Hayes, 1969). Here the retained objects are task occurrences, and their identities are also consumed by learning and deployment.

E.2 Dependency-Aware Transitions and Preservation

The relation graph determines which fields a change couples. Invariant supports determine which checks that change requires. This distinction lets a local transition preserve a larger persistent state, including records that the model does not regenerate.

Addresses and trusted invariant supports. Write $z = Z(C)$ for the task-semantic projection of the committed state. Formally, $z : \mathcal{A} \to \mathcal{V} \cup \{\bot\}$ is a map on a fixed countable address universe, with finitely many non-absent entries; $\bot$ denotes absence. An address identifies a record field or a relation entry. Creation and deletion are writes from or to $\bot$. Task-relevant evidence fields belong to z; transaction counters and global receipts are handled separately.

A fixed trusted specification supplies predicates I_α and selects a finite active family $\sigma(z)$ from an index set $\mathcal{I}$. Thus

$$I(z) = \bigwedge_{\alpha \in \sigma(z)} I_\alpha(z), \qquad \widetilde{I}_\alpha(z) = \big[\alpha \notin \sigma(z)\big] \vee I_\alpha(z). \tag{E.2}$$

A *sound support* $B_\alpha \subseteq \mathcal{A}$ satisfies $x|_{B_\alpha} = y|_{B_\alpha} \Rightarrow \widetilde{I}_\alpha(x) = \widetilde{I}_\alpha(y)$ throughout the trusted validation domain. Supports therefore include fields controlling membership in σ as well as fields used by the predicate. The selection rule and predicate meanings are fixed by the specification; a patch can create records, but cannot change that specification. Supports may contain the entire relation graph.

For event e and patch p, the deterministic executor produces a candidate $\widehat{z}$. Its effective write footprint W_p includes all coupled writes, including reciprocal pointers, lifecycle changes, and semantic evidence. The event and operation contract determines the allowed footprint; a proposed `affected_scope` is checked against that contract. The read footprint R_p includes every address whose value can affect execution or validation.

Proposition E.3 (Preservation under a validated transition)**.** *Suppose z and $\widehat{z}$ belong to the trusted validation domain, $I(z)$ holds, and $\widehat{z}|_{\mathcal{A}\setminus W_p} = z|_{\mathcal{A}\setminus W_p}$. Let validation establish $\widetilde{I}_\alpha(\widehat{z})$ for every $\alpha \in \sigma(z) \cup \sigma(\widehat{z})$ with $B_\alpha \cap W_p \neq \varnothing$, and for every newly active index $\alpha \in \sigma(\widehat{z}) \setminus \sigma(z)$. An atomic transition that commits $\widehat{z}$ on acceptance and retains z on rejection has*

$$I(z') \quad \textit{and} \quad z'|_{\mathcal{A}\setminus W_p} = z|_{\mathcal{A}\setminus W_p}. \tag{E.3}$$

If the commit operation extends committed provenance by appending a receipt, the previous provenance remains a prefix. Rejection leaves the entire committed state unchanged.

Domain membership is a prerequisite of acceptance; a candidate outside the trusted validation domain is rejected.

Proof. On rejection, $z' = z$, so initial validity gives $I(z') = I(z) = 1$ and the frame equality holds immediately. Consider acceptance, for which $z' = \widehat{z}$. Fix any $\alpha \in \sigma(\widehat{z})$ and distinguish three cases.

First, if $\alpha \notin \sigma(z)$, it is newly active and validation establishes $\widetilde{I}_\alpha(\widehat{z}) = 1$. Second, if $\alpha \in \sigma(z)$ and $B_\alpha \cap W_p \neq \varnothing$, the affected invariant is checked and has the same value. In both cases, activity in the candidate state implies

$$\widetilde{I}_\alpha(\widehat{z}) = \big[\alpha \notin \sigma(\widehat{z})\big] \vee I_\alpha(\widehat{z}) = I_\alpha(\widehat{z}) = 1.$$

Third, suppose $\alpha \in \sigma(z)$ and $B_\alpha \cap W_p = \varnothing$. Then $B_\alpha \subseteq \mathcal{A} \setminus W_p$, so the assumed frame equality restricts to

$$\widehat{z}|_{B_\alpha} = z|_{B_\alpha}.$$

Both states belong to the trusted validation domain. Soundness of B_α, applied to the masked predicate $\widetilde{I}_\alpha$, therefore gives

$$I_\alpha(\widehat{z}) = \widetilde{I}_\alpha(\widehat{z}) = \widetilde{I}_\alpha(z) = I_\alpha(z) = 1,$$

where the first and last equalities use activity in the respective states, and the last value follows from $I(z)$. These cases exhaust $\sigma(\widehat{z})$, hence

$$I(z') = \bigwedge_{\alpha \in \sigma(\widehat{z})} I_\alpha(\widehat{z}) = 1, \qquad z'|_{\mathcal{A}\setminus W_p} = \widehat{z}|_{\mathcal{A}\setminus W_p} = z|_{\mathcal{A}\setminus W_p}.$$

The empty active family also satisfies this conjunction.

Finally, let L and L' be the committed provenance before and after the transition. An accepted receipt J gives $L' = L \parallel (J)$, where $\parallel$ denotes sequence concatenation, so $L'_{1:|L|} = L$. Rejection gives $L' = L$ and preserves the entire committed state. □

Corollary E.4 (Preservation across an event sequence)**.** *Starting from $I(z^{(0)})$, apply any finite sequence of transitions satisfying Proposition E.3, allowing later proposals to depend on earlier outcomes. Every committed state is valid. If $U_m = \bigcup_{k=1}^{m} W_{p_k}$, then*

$$z^{(m)}|_{\mathcal{A}\setminus U_m} = z^{(0)}|_{\mathcal{A}\setminus U_m}. \tag{E.4}$$

Committed provenance retains its original prefix throughout the sequence.

Proof. Fix a realized sequence of proposals and outcomes. The base state is valid by assumption. If $I(z^{(k-1)}) = 1$, the hypotheses of Proposition E.3 at step k give, for either outcome,

$$I(z^{(k)}) = 1, \qquad z^{(k)}|_{\mathcal{A}\setminus W_{p_k}} = z^{(k-1)}|_{\mathcal{A}\setminus W_{p_k}}.$$

Induction proves validity for $k = 0, \ldots, m$.

For any address $a \notin U_m$, we have $a \notin W_{p_k}$ for every $k \le m$. Applying the stepwise frame equalities at that address yields

$$z^{(m)}(a) = z^{(m-1)}(a) = \cdots = z^{(0)}(a).$$

Since this holds for every $a \in \mathcal{A} \setminus U_m$, it proves Equation (E.4). For $m = 0$, the conclusion follows with $U_0 = \varnothing$.

Write $L^{(k)}$ for committed provenance after step k, and let $\preceq$ denote the prefix relation. Each step appends one accepted receipt or leaves the log unchanged, giving

$$L^{(0)} \preceq L^{(1)} \preceq \cdots \preceq L^{(m)}.$$

Transitivity preserves the initial prefix. The argument applies to each realized sequence, including when the next proposal depends on all preceding outcomes. □

Why the support follows dependencies. Suppose the override relation contains $a \to b \to c$. Adding $c \to a$ can satisfy local pointer and identifier checks while creating a cycle. A sound acyclicity check must account for the existing path from a to c; the affected check can therefore require a long read even when the new edge is a single write. Event correctness is a separate relation $T_e(C, C')$: expiring an eligible occurrence o_2 can preserve every structural invariant while failing an event that requested expiration of o_1. Correct event interpretation selects the transition; structural validation preserves the state properties used by subsequent consumers.

E.3 Independent Archives and Generated Output

Let $\mathrm{dom}_+(z) = \{a : z(a) \neq \bot\}$ be the stored domain. For maps with disjoint stored domains, write $z \oplus F$ for their union. The following result specifies when retaining an independent archive commutes with execution of a fixed patch. It treats p, its registered event, and its execution context as fixed inputs.

Lemma E.5 (Independent archive extension)**.** *Let $\overline{T}_p$ be the terminating deterministic semantic transition, including identity on rejection. Suppose an archive F satisfies the following noninterference conditions for evaluation of p on z and $z \oplus F$:*

1. *The stored domains of z and F are disjoint. All effective writes and fresh allocations are outside* $\mathrm{dom}_+(F)$.

2. *Execution, name resolution, guards, and allocation depend on the same explicit inputs and the same read values in both evaluations. In particular, enumeration queries and allocator state yield the same answers.*

3. *Invariant discovery and validation decompose into the same local checks and additional archive-only checks. The latter hold on F and remain unaffected by the candidate writes. All acceptance conditions obey this decomposition, including reference, authority, and schema-membership checks.*

Then

$$\overline{T}_p(z \oplus F) = \overline{T}_p(z) \oplus F. \tag{E.5}$$

Proof. Fix the patch, event, and execution context. Compare the two materialization evaluations from z and $z \oplus F$. Their shared explicit inputs and equal read answers give the same computed values and branch decisions by condition 2. This includes name resolution, enumeration, and allocation. Let z_i and $\widetilde{z}_i$ denote their semantic stores after the ith corresponding materialization instruction. We show inductively that

$$\widetilde{z}_i = z_i \oplus F, \qquad \mathrm{dom}_+(z_i) \cap \mathrm{dom}_+(F) = \varnothing.$$

The relation holds initially by condition 1. A read or control instruction leaves the stores unchanged and has the same result in both evaluations. For a write, the computed address a_i and value v_i agree. Condition 1 places a_i outside $\mathrm{dom}_+(F)$, and therefore

$$(z_i \oplus F)[a_i \leftarrow v_i] = \big(z_i[a_i \leftarrow v_i]\big) \oplus F.$$

This also covers creation and deletion, represented as writes from or to $\perp$. It preserves disjointness and proves the induction step. Equal guards also give the same early rejection, if one occurs. Termination then gives either this common rejection or candidates $\widehat{z}$ and $\widehat{z} \oplus F$.

For the candidate case, let b be the acceptance decision without the archive and $\widetilde{b}$ the decision with it. By condition 3, all local checks have the same outcomes, while the conjunction b_F of additional archive-only checks is true. Thus

$$\widetilde{b} = b \wedge b_F = b \wedge 1 = b.$$

Consequently the committed semantic states satisfy

$$\overline{T}_p(z \oplus F) = \begin{cases} \widehat{z} \oplus F, & b = 1, \\ z \oplus F, & b = 0, \end{cases} = \overline{T}_p(z) \oplus F.$$

The same equality holds after a common early rejection because each run retains its own initial state. □

The assumptions exclude, for example, an archived occurrence that creates an authority conflict or changes a global name-resolution query. Receipts extend each run's own provenance prefix; a receipt containing a whole-state digest can differ between the runs. The lemma concerns execution of the same patch; the patch reasoner can depend on its full input state.

Corollary E.6 (Serialization burden)**.** *Fix an explicit serialization grammar and measure length in bytes. A literal full-state replacement that includes n retained records, each requiring at least $b_{\min} > 0$ bytes, has*

$$L_{\mathrm{FSR}} \geq n b_{\min}. \tag{E.6}$$

A patch grammar with a fixed finite operation vocabulary can encode r operations and q identity references, each of at most $b_{\mathrm{id}} \geq 1$ bytes, using

$$L_{\mathrm{patch}} = O(1 + r + q b_{\mathrm{id}} + B_{\mathrm{args}} + B_{\mathrm{env}}), \tag{E.7}$$

where B_{args} and B_{env} count the serialized arguments and required envelope, including generated metadata and evidence. The implied constant depends only on the grammar.

Proof. Let ℓ_i be the byte length of the ith retained record in the literal replacement. Its encoding includes every such record, and any additional framing contributes a nonnegative length. Hence

$$L_{\mathrm{FSR}} \geq \sum_{i=1}^{n} \ell_i \geq \sum_{i=1}^{n} b_{\min} = n b_{\min}.$$

For the patch grammar, choose constants $c_0, c_{\mathrm{op}}, c_{\mathrm{ref}}$ that bound fixed framing, the operation name and delimiters per operation, and delimiters per identity reference, respectively. These constants depend only on the fixed grammar and its finite operation vocabulary. If ℓ_j^{id} is the encoded length of reference j, the direct encoding satisfies

$$\begin{aligned} L_{\mathrm{patch}} &\leq c_0 + c_{\mathrm{op}} r + c_{\mathrm{ref}} q + \sum_{j=1}^{q} \ell_j^{\mathrm{id}} + B_{\mathrm{args}} + B_{\mathrm{env}} \\ &\leq c_0 + c_{\mathrm{op}} r + (c_{\mathrm{ref}} + 1) q b_{\mathrm{id}} + B_{\mathrm{args}} + B_{\mathrm{env}}. \end{aligned}$$

The second inequality uses $\ell_j^{\mathrm{id}} \leq b_{\mathrm{id}}$ and $q \leq q b_{\mathrm{id}}$, the latter following from $b_{\mathrm{id}} \geq 1$. Taking $c = \max\{c_0, c_{\mathrm{op}}, c_{\mathrm{ref}} + 1, 1\}$ gives

$$L_{\mathrm{patch}} \leq c\big(1 + r + q b_{\mathrm{id}} + B_{\mathrm{args}} + B_{\mathrm{env}}\big),$$

which is Equation (E.7) with a grammar-dependent constant. □

Consecutive integer identifiers in a namespace of size N permit $b_{\mathrm{id}} = O(\log(N+1))$. A transaction that lists all protected identifiers can have B_{env} growing with the archive. Operation count and write count also differ: one `RestoreRoot` can change several linked records. These formulas isolate generated output, connecting the interfaces to the state-scaling comparison in Table 2; reads, validation, and physical execution have their own costs.

E.4 Occurrence Identity and Decision Sufficiency

Violation-time routes. Let (C_τ, E_τ) denote the committed task snapshot and recorded execution context at the physical time τ of a tested violation. Applicability, valid grounding, and the evidence in Equation (4) are evaluated against this record. The public routing interface returns a decision $d \in \{\mathrm{update}, \mathrm{no\text{-}update}, \mathrm{abstain}\}$, occurrence and module sets, first-violation time, an evidence window, a target schema, confidence, and evidence identifiers. An update names nonempty occurrence and module sets and supplies the time, window, and schema. The other decisions have empty target fields: no-update records that the considered evidence warrants no correction, whereas abstention defers that judgment. Equation (5) records the module and occurrence sets and the optional temporal fields. In prospective evaluation, the route is committed before the correction target is revealed; the completed packet then adds that target and its provenance. Thus the public evidence determines the route, and the later target supplies the training signal.

Proposition E.7 (Decision sufficiency)**.** *Let $\mathcal{H}$ and $\mathcal{V}_{\mathrm{ev}}$ be finite or countable sets of registered histories and evidence records, with any supplied execution context included in the evidence record. Let $\mathcal{D}_{\mathrm{adm}} \subseteq \mathcal{H} \times \mathcal{V}_{\mathrm{ev}}$ be the admissible pairs. For a reference decision $D : \mathcal{D}_{\mathrm{adm}} \to \mathcal{Y}$ and representation $\phi : \mathcal{H} \to \mathcal{Z}$, there exists a rule on the induced inputs such that $d(\phi(h), v) = D(h, v)$ for every $(h, v) \in \mathcal{D}_{\mathrm{adm}}$ if and only if*

$$\phi(h) = \phi(h'), \quad (h, v), (h', v) \in \mathcal{D}_{\mathrm{adm}} \quad \Longrightarrow \quad D(h, v) = D(h', v). \tag{E.8}$$

For stochastic reference decisions on a countable $\mathcal{Y}$, the same equivalence holds with equality of decision distributions.

Proof. Define the induced input set and its admissible history fibers by

$$\begin{aligned} \mathcal{U} &= \{(\phi(h), v) : (h, v) \in \mathcal{D}_{\mathrm{adm}}\}, \\ F(z, v) &= \{h \in \mathcal{H} : (h, v) \in \mathcal{D}_{\mathrm{adm}},\ \phi(h) = z\}. \end{aligned}$$

Every fiber indexed by $\mathcal{U}$ is nonempty. For necessity, suppose d realizes D. If $h, h' \in F(z, v)$, then

$$D(h, v) = d(\phi(h), v) = d(z, v) = d(\phi(h'), v) = D(h', v),$$

which is Equation (E.8).

For sufficiency, fix an enumeration of $\mathcal{H}$ and let $r(z, v)$ be the first member of $F(z, v)$ in that enumeration. Define

$$d(z, v) = D(r(z, v), v), \qquad (z, v) \in \mathcal{U}.$$

For every admissible (h, v), both h and $r(\phi(h), v)$ belong to $F(\phi(h), v)$. Equation (E.8) therefore gives $d(\phi(h), v) = D(h, v)$. It also shows that any other choice of representatives defines the same rule on $\mathcal{U}$.

For the stochastic statement, write $P_D(y \mid h, v)$ for the reference probability of output $y \in \mathcal{Y}$ and set

$$p_d(y \mid z, v) = P_D(y \mid r(z, v), v), \qquad \sum_{y \in \mathcal{Y}} p_d(y \mid z, v) = 1.$$

These nonnegative probabilities define a decision distribution at each input. Both $\mathcal{U}$ and $\mathcal{Y}$ are countable, so with their discrete σ-algebras they define a measurable probability kernel. Equality of reference distributions on each fiber yields $p_d(y \mid \phi(h), v) = P_D(y \mid h, v)$ for every y; countable additivity then gives equality on every output event. Conversely, a rule realizing the reference distributions assigns one distribution to each (z, v), so all admissible histories in its fiber have that same distribution. □

A restoration example. Consider histories h_A and h_B whose original requests are to place an object in baskets A and B, respectively. A temporary request c redirects the object to basket C in both histories. The physical scene and current predicate $\mathrm{In}(\mathrm{object}, C)$ coincide. The authority lineage is $a \to c$ in h_A and $b \to c$ in h_B, where a and b identify the original requests. Let the common evidence v withdraw the temporary request and ask to restore its root. The reference restoration decisions are (a, A) and (b, B). The reduced representation consisting only of the physical scene and current predicate maps these histories to the same value, and therefore fails Equation (E.8). Retaining the occurrence lineage distinguishes the two restoration targets. The routing results in Table 1 concern learned decisions from their supplied representations. In particular, the Generic Persistent routing interface retains public occurrence IDs, active occurrence, and plan-suffix relations; the example applies to the explicitly reduced view defined here.

E.5 Occurrence Ownership and the Deployment Gate

An episode occurrence is addressed by a key $q = (\mathrm{trace_id}, o)$. Let χ be a supplied partial map from these keys to canonical occurrence keys $\bar{o} \in \overline{\mathcal{O}}$. The correction packet and adapter registry use the same ownership interface. This abstraction permits the deployment system's registered matching rule without assuming that episode-local strings are globally unique. Let $\bar{o}^\star$ be the owner of a loaded adapter, and let $q(h)$ be the current authoritative key for its workflow in the complete pre-action history h, including past actions. Set $q(h) = \perp$ when no such key is available, with $\perp \notin \mathrm{dom}(\chi)$. The Boolean predicate $\mathrm{eligible}(q, h)$ is the deployment guard supplied by the controller. The ownership map and this guard are operational inputs to the interface. Let π_0 and π_{adapt} be the base and loaded adapter's action kernels. The index n counts policy-selection epochs; action chunks and their positions belong to the controller state. Define

$$g(h) = \begin{cases} \mathrm{eligible}(q(h), h), & q(h) \in \chi^{-1}(\{\bar{o}^\star\}), \\ 0, & \text{otherwise}, \end{cases} \tag{E.9}$$
$$\pi_g(\cdot \mid h) = g(h)\pi_{\mathrm{adapt}}(\cdot \mid h) + [1 - g(h)]\pi_0(\cdot \mid h).$$

An undefined ownership match closes the gate. Deployment eligibility is evaluated in the current committed state, whereas correction assignment uses the violation-time snapshot. The gate selects a loaded policy realization; its operation preserves the stored base parameters.

Proposition E.8 (Preservation under a closed occurrence gate)**.** *At every history h with $g(h) = 0$, $\pi_g(\cdot \mid h) = \pi_0(\cdot \mid h)$. Suppose the gated and base executions have the same initial joint law of environment and complete controller state. Suppose also that, conditional on the history and selected action, they use identical transition kernels for the environment, exogenous events, and complete controller state. Take these state and action spaces to be standard Borel spaces and all*

kernels and the gate to be measurable. Then the executions can be coupled to have identical histories through the first pre-action history at which the gate opens, and identical actions before that time. If

$$\Pr_{\pi_0}\{\exists n \geq 0 : \ g(H_n) = 1\} = 0, \tag{E.10}$$

where H_n is the random pre-action history at epoch n, their full trajectory laws coincide under that initial distribution.

Proof. For $g(h) = 0$, Equation (E.9) gives, for every measurable action set A,

$$\pi_g(A \mid h) = 0 \cdot \pi_{\mathrm{adapt}}(A \mid h) + 1 \cdot \pi_0(A \mid h) = \pi_0(A \mid h).$$

Let A_n be the action selected at epoch n. Write $K_n(\mathrm{d}h' \mid h, a)$ for the common conditional kernel of the next complete pre-action history, obtained by appending the selected action and the resulting environment state, exogenous events, and complete controller state. The assumed shared transition kernels determine this kernel. Superscripts 0 and g distinguish the base and gated executions. Define the first opening epoch along the base execution by

$$T = \inf\{n \geq 0 : g(H_n^0) = 1\}, \qquad \inf \varnothing = \infty.$$

Since H_n^0 is a pre-action history, the decision whether $T = n$ precedes the sampling of A_n^0.

Construct the coupling as follows. Sample a single initial joint state from the common initial law and use it for both executions, giving $H_0^0 = H_0^g$. At epoch $n < T$, suppose their histories are equal to h. The gate is closed at h, so draw one action from $\pi_0(\cdot \mid h)$ and assign it to both executions. Equivalently, the coupled action kernel Γ_n satisfies, for measurable action sets A, B,

$$\Gamma_n(A \times B \mid h, h) = \pi_0(A \cap B \mid h).$$

Its two marginals are the required action kernels because $\pi_g(\cdot \mid h) = \pi_0(\cdot \mid h)$. Conditional on the shared action a, draw one next history from $K_n(\cdot \mid h, a)$ and use it for both executions, giving $H_{n+1}^0 = H_{n+1}^g$. Thus induction establishes

$$H_n^0 = H_n^g \quad (n \leq T), \qquad A_n^0 = A_n^g \quad (n < T),$$

where n ranges over finite epochs. In particular, at a finite T both executions reach the same first opening history. From that epoch onward, use product couplings of their respective action and transition kernels. The diagonal kernels before opening and product kernels afterward are measurable on the assumed standard Borel spaces and have the prescribed marginals at every step. Their successive composition therefore defines a joint trajectory law with the required base and gated marginals.

Under Equation (E.10), this coupling satisfies $\Pr\{T < \infty\} = 0$, since its base marginal is the execution under π_0. For any finite N, its history-action prefixes consequently agree almost surely. For every measurable set B of such prefixes,

$$\Pr_{\pi_g}\{(H_0, A_0, \ldots, A_{N-1}, H_N) \in B\} = \Pr_{\pi_0}\{(H_0, A_0, \ldots, A_{N-1}, H_N) \in B\}.$$

Finite-prefix cylinders form a generating π-system for the trajectory product σ-algebra. Equality on these cylinders determines the same probability measure on that σ-algebra, proving equality of the full trajectory laws. □

The complete controller state in this proposition includes caches, recurrent memory, action-chunk position, and any randomness influencing future decisions. This identifies the state that must be preserved along the base branch. The clean gated-retention diagnostic in Appendix B exercises that branch. After an adapter has acted, closing its gate yields the base kernel at the resulting history; the coupling statement ends at the earlier activation.

E.6 Equivalent Persistent Carriers

Lemma E.9 (Consumer equivalence)**.** *Let X be an alternative carrier with decoder $\delta : X \to \mathcal{C}$ into the semantic commitment states, and let T_X and T_C be deterministic committed-transition maps under the specified proposal rules. Suppose its initialized state decodes to the reference initial state and, for every admitted state x, common execution context E, and event e,*

$$\delta(T_X(x, E, e)) = T_C(\delta(x), E, e). \tag{E.11}$$

Include rejected transitions in this equality. Suppose every consumer's conditional decision kernel on X equals the reference kernel evaluated at $\delta(x)$ and the same public execution context. Then the carriers give equal decoded states and equal consumer decision kernels along every common admitted sequence of contexts and events.

Proof. Fix a common admitted sequence $(E_n, e_{n+1})_{n\geq 0}$ of contexts and events, and denote the initialized carrier and reference states by x_0 and C_0. Their subsequent states are defined recursively by

$$x_{n+1} = T_X(x_n, E_n, e_{n+1}), \qquad C_{n+1} = T_C(C_n, E_n, e_{n+1}).$$

The initialization assumption gives $\delta(x_0) = C_0$. If $\delta(x_n) = C_n$, then Equation (E.11) yields

$$\begin{aligned}\delta(x_{n+1}) &= \delta(T_X(x_n, E_n, e_{n+1})) \\ &= T_C(\delta(x_n), E_n, e_{n+1}) \\ &= T_C(C_n, E_n, e_{n+1}) = C_{n+1}.\end{aligned}$$

The commutation assumption covers rejected transitions as well, so this step applies at every event in the sequence. Induction gives $\delta(x_n) = C_n$ for all n.

For any consumer j, let Q_X^j and Q_C^j denote its conditional decision kernels on the two carriers. For every measurable decision event B, the assumed consumer factorization and the decoded-state identity give

$$Q_X^j(B \mid x_n, E_n) = Q_C^j(B \mid \delta(x_n), E_n) = Q_C^j(B \mid C_n, E_n).$$

This proves equality of every consumer's decision kernel at each step of the chosen sequence. Since the sequence was arbitrary, both conclusions hold along every common admitted sequence. □

This lemma applies to an explicit graph, a synchronized state-plus-log carrier, or exact full-state regeneration whenever the stated contracts hold. It separates the retained semantic interface from the mechanism that produces it. Bijective namespace renaming preserves that interface; different learned proposals are evaluated by the experiments.

F ALGORITHMS AND BASELINE INTERFACES

The comparisons vary the state-maintenance interface or the information presented to a responsibility router. This appendix specifies the generated object, deterministic execution, and retained information for each named method. Algorithms 1–2 describe runtime maintenance; Algorithm 4 gives the learning interface.

F.1 RUNTIME STATE MAINTENANCE

COPE. The model emits a typed operation targeting an occurrence. A deterministic executor expands it into coupled lifecycle, lineage, and provenance updates. The candidate is committed atomically after validation. Unchanged records are read from persistent storage when the downstream planning problem is compiled.

Algorithm 1 COPE runtime transition

Require: Committed state C, execution context E, public event e; patch reasoner f_θ, executor Apply

$p \leftarrow \mathsf{Parse}(f_\theta(C, E, e))$; $\widehat{C} \leftarrow \bot$ ▷ Parse returns $\bot$ on failure
if $p \neq \bot$ **then**
  $\widehat{C} \leftarrow \mathsf{Apply}(C, p)$ ▷ Returns $\bot$ on invalid application
end if
if $\widehat{C} \neq \bot$ and $\mathsf{Validate}(C, \widehat{C}, p, e)$ **then**
  $C' \leftarrow \mathsf{Commit}(\widehat{C})$
else
  $C' \leftarrow C$; invoke the configured bounded fallback
end if
$\kappa' \leftarrow \mathsf{Compile}(C', E)$
return (C', κ') for downstream execution

FSR: Full-State Regeneration. FSR receives the current state, public event, execution context, and the representation's field and transition rules. Its model generates the complete replacement authoritative state, including unchanged records and required retained history. Validation precedes acceptance, and rejection preserves the previous committed state. The replacement contract determines which fields must be re-emitted. Parsing or materialization failures return $\bot$; validation is evaluated only for a well-defined candidate in all algorithms.

Algorithm 2 Full-State Regeneration (FSR)

Require: Committed state C, context E, public event e; full-state generator g_θ, representation contract b

$u \leftarrow \mathsf{Serialize}_b(C, E, e, \text{field and transition rules})$
$\widehat{C} \leftarrow \mathsf{Parse}_b(g_\theta(u))$ ▷ Generate changed and unchanged required state
if $\widehat{C} \neq \bot$ and $\mathsf{Validate}_b(C, \widehat{C}, e)$ **then**
  $C' \leftarrow \widehat{C}$
else
  $C' \leftarrow C$
end if
$\kappa' \leftarrow \mathsf{Compile}_b(C', E)$
return (C', κ')

FSR-PC. FSR-PC is the simulator-facing full-state-regeneration comparison over the persistent-commitment schema. In the paired simulator study, its repair bridge, downstream controller, interruption schedule, and physical execution stack match those of COPE. Identity behavior is specified by the evaluated representation: the model-driven development version retains stable IDs, whereas the paired simulator study reports its own identity-preservation measurement in Table 3.

F.2 LOCAL-EDIT CONTROLS

Neutral Typed. This control replaces semantic operation names with anonymous labels `N01`–`N10`. Typed signatures and the semantic executor are shared with COPE. The alias is decoded before deterministic execution. The proposal envelope checks the operation identifier and the union of allowed fields; operation-specific requirements are enforced by the executor and, in the registered benchmark wrapper, exact post-state verification. It therefore tests the operation vocabulary with the state machinery held fixed.

Compact Transaction. This control emits stable-ID JSON-style `add`, `replace`, and `remove` writes. Allowed paths address commitments, actions, progress, restorations, and facts. Its materializer applies these writes and enforces the path rules. Direct writes to protected transaction metadata are forbidden; the trusted envelope updates version, evidence, and processed-event metadata. It retains sparse generation through a generic write interface.

Generic Persistent Edit (Exp1). This control uses the same persistent ledger, protected projection, evidence, and invariant kernel as COPE, with generic occurrence-field and relation edits. The permitted field paths are `/lifecycle`, `/priority`, `/grounding_validity`, `/restore_guard`, and `/dependency_ids`. Each field write names an occurrence. The transaction envelope includes affected scope, protected IDs, a protected-state projection digest, evidence, and assertions; these fields contribute to its serialized output cost.

For these named controls, Neutral Typed and Exp1 Generic Persistent Edit retain lifecycle and lineage through their shared state machinery. This is the precise retention convention for these names in the comparisons. The separate neutral transaction comparison in Appendix C is indexed by its own carrier and evaluation protocol; it is not identified with a named control solely by the word "neutral."

Algorithm 3 Local-edit controls: shared interface, specified materializer

Require: State C, context E, event e, control b
$p_b \leftarrow \mathsf{Parse}_b(\mathsf{Generate}_b(C, E, e))$
if $p_b = \bot$ **then**
  return C ▷ Malformed proposal; retain authoritative state
end if
if $b =$ NEUTRAL TYPED **then**
  $\widehat{C} \leftarrow \mathsf{Apply}(C, \mathsf{DecodeAlias}(p_b))$
else if $b =$ COMPACT TRANSACTION **then**
  $\widehat{C} \leftarrow \mathsf{MaterializeCompact}(C, p_b)$
else if $b =$ GENERIC PERSISTENT EDIT **then**
  $\widehat{C} \leftarrow \mathsf{ApplyGeneric}(C, p_b)$ ▷ Shared ledger and invariant kernel
end if
Reject a failed materialization; otherwise validate under contract b
In the registered scaling wrapper, require exact semantic post-state agreement
Commit the valid candidate atomically; otherwise retain C

The registered Neutral Typed and Compact Transaction benchmark wrappers normalize the materialized state, check exact equality with the oracle post-state, and then commit the trusted metadata envelope. This verification uses the controlled benchmark's reference answer. The general runtime transition uses its declared executable validator; the preservation proposition states the conditions that validator must establish.

Identity, provenance, and adaptation controls. FSR-PC + STABLE IDS retains occurrence IDs while generating a complete replacement state. FSR-PC + PROVENANCE adds the retained evidence used by audit queries; its identity behavior follows its baseline variant. Both use the shared recovery backend. NO ADAPTATION retains the existing task state without applying event-driven task-state updates and uses the same physical execution stack.

F.3 Responsibility Routing and Correction Packets

Generic Persistent (Exp2). The routing control presents the public event, accepted occurrence IDs, current active occurrence, plan-suffix target occurrences, and verifier judgment:

```
view: generic_persistent
event: neutral(public.event)
entities: neutral(accepted_occurrence_ids)
active: neutral(planner.active_occurrence_id)
suffix: neutral(planner.plan_suffix_target_occurrences)
verification: verifier.judgment
```

The function `neutral` recursively removes the namespace prefixes `goal:`, `constraint:`, and `preference:` from occurrence strings. The remaining identity, active-occurrence, suffix, and event information is retained. When this relabeling is injective on the evaluated identifiers, it preserves their distinguishability. Proposition E.7 characterizes precisely when any representation supports the required route.

The common `RouteSpec` fields are `schema_version`, `trace_id`, `method`, `modules`, `occurrences`, `first_violation`, `window`, `target_schema`, `decision`, `confidence`, and `evidence_ids`. Modules are drawn from editor, planner, policy, and verifier. The validator rejects missing or extra fields, duplicate identities, non-public occurrence targets, and inconsistent decision/target combinations. An `update` route requires module and occurrence targets, violation time, window, and target schema. A `no-update` or `abstain` route has empty target fields.

Algorithm 4 CoPE responsibility routing, offline update, and deployment

Require: Public evidence v, committed snapshots, canonical-owner interface, correction source, base policy π_0

Resolve candidate occurrence keys at violation time τ
Evaluate applicability, grounding, and violation evidence in C_τ
Produce and validate a public route: decision, modules, occurrences, first violation, evidence window, and target schema
if decision is `update` **then**
    Obtain the corresponding correction target y after the route is fixed
    Construct occurrence-owned correction packet with evidence provenance
    Update the selected module offline; save the resulting adapter/checkpoint
    Register its canonical occurrence owner $\bar{o}$
else
    Emit no correction packet; preserve the route's decision and evidence
end if
At deployment, select the registered specialization when its owner matches the current authoritative occurrence and the deployment guard holds
Otherwise select π_0; execute the selected frozen policy

Task/world revisions and violation assignments are indexed by event and evidence window. A single episode may contain a legitimate task change and a later execution violation. The route distinguishes their respective responsibilities. At deployment the gate selects an already-trained policy realization.

Registered threshold decoder. The Exp2C decoder first chooses the largest positive threshold margin among the four module labels, `no_update`, and `abstain`. A winning non-update label at or above its threshold yields an empty-target route. Otherwise, module and occurrence targets are all labels at or above their respective thresholds; an empty selection falls back to the largest raw score in that group. It sets the first-violation index to `policy_step`+1 and the evidence window to $[\max(0, \tau-3), \tau]$, forms the target schema from the selected modules, and uses their maximum score as confidence. These are decoder conventions for that registered predictor, rather than definitions of violation time for every possible router.

Coarser routing controls. All-Failure Episode assigns supervision at episode level without occurrence filtering. Terminal Skill attributes the outcome to the terminal skill. Flat Trace

presents a flattened execution trace without a structured occurrence relation. These interfaces determine the attribution granularity compared in Table 1; the matched post-training protocol in Appendix B determines the training budgets of the evaluated policy-update arms.

F.4 What Each Comparison Varies

Method	Generated object / public view	State machinery or consumer
CoPE	Named typed patch	Semantic executor; persistent occurrence state
FSR / FSR-PC	Complete replacement state	Representation-specific validation; matched recovery backend where specified
Neutral Typed	Anonymous typed patch	Same signatures and semantic executor
Compact Transaction	Stable-ID generic writes	Compact materializer and protected metadata
Generic Persistent Edit	Generic field/relation transaction	Shared ledger and invariant kernel
Generic Persistent (Exp2)	Public IDs, event, active occurrence, suffix, verification	Shared route schema; namespace-normalized view
FSR-PC + Stable IDs	Complete replacement with retained IDs	Identity ablation; shared recovery backend
FSR-PC + Provenance	Complete replacement with retained evidence	Audit ablation; shared recovery backend
No Adaptation	Existing task state	No event-driven task-state revision
All-Failure Episode	Episode-level supervision	No occurrence filtering
Terminal Skill	Terminal-skill attribution	Skill-level route
Flat Trace	Flattened execution trace	No structured occurrence relation

Table 2 examines maintenance and generation under increasing retained state. Table 1 evaluates prospective responsibility assignment before correction targets are revealed. Table 3 and Appendix D examine continuation, identity, provenance, and validation, while Appendix B follows selected packets through policy updates and occurrence-gated execution.