A TYPED LATENT-STATE ARCHITECTURE FOR EXPERIENCE-CONDITIONED STRUCTURAL REVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a typed structural latent-state architecture in which experience revises the executable structures governing an agent's subsequent prediction, action, and learning. Its learnable state is a discrete structural world model: concepts are typed Metadata clusters whose generation structures, scopes, and relations participate directly in identity and computation. We implement finite slices in software agents in simulated worlds. In early studies, matched version-space and complete-context cache controls reproduce the tested prediction and reuse behaviors. In later studies, experienced outcomes change method selection and active interfaces, while failure attribution produces structures that persist across processes and guide later candidate generation. Across twelve new experience streams, an interface-revision policy improves predictive accuracy over the frozen predecessor in ten and ties in two, but yields lower cost-adjusted returns in all twelve under the tested tariffs. Revision does not by itself ensure model adequacy: a connected attribution–revision–reuse loop can still generalize incorrectly when the true relation lies outside its available hypothesis family. Matched interventions on what later computation consumes establish executable experience-conditioned revision of an agent's computational state.

1 INTRODUCTION

A learning agent's current explanation of its world need not be a final explanation. It must nevertheless support decisions, retain what those decisions depended on, and change when experience supplies a reason to change it. This suggests an architectural question: what happens when the agent's learnable state consists of executable, typed structures, including structures used to interpret its own previous decisions?

We study a **typed structural latent-state architecture**, implemented here with discrete executable structures. A concept is a cluster of typed fields with generation structures, relations and scope conditions. A stored value is one readout of this state. Revising a generating expression or an active observation interface can change the agent's next prediction, method choice or investigation. The structures therefore occupy the learning and decision path itself. Their role is not exhausted by retrieving text for a separate, fixed predictor.

The term *latent* refers to an inferred internal model of relations and conditions that are not given directly by the current observation. The implementations use discrete executable structures, rather than a learned continuous embedding. We do not assume that a stored structure is true merely because it is well typed, consistent with its observations, or successfully used once. The architecture makes these different conditions representable; the experiments examine which conditions the implemented learners actually establish.

We instantiate finite slices in simulated worlds. The studies progress from ordinary candidate elimination and reuse controls to paid experience, interface revision, failure attribution and later reuse. They are development studies with explicit continuation boundaries, not a single unchanged learner evaluated on an independent benchmark. This distinction matters: an author repairing a mechanism between studies is different from a learner updating its state inside a study.

Three findings organize the paper. First, several early behaviors are reproduced by ordinary version-space learning and complete-context caching. Second, later outcomes change executable state.

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.

Across 12 new experience streams, interface revision improves subsequent accuracy over its frozen predecessor in ten and ties in two, but yields lower cost-adjusted return in all twelve. Attribution results are also acquired, persisted and consumed by later processes. Third, the resulting feedback loop can remain wrong when its available hypothesis family omits the world relation. Preserving a decision's generation path helps locate this failure, but does not by itself repair hypothesis coverage.

Our contribution is an executable architectural formulation, a set of matched interventions on its state consumers, and a concrete account of where experience-conditioned revision succeeds or fails in finite implementations. We distinguish demonstrated feedback coupling and repeated local predictive improvement from competitive cost, broad robustness or autonomous hypothesis-language growth. Cost-adjusted outcomes are unfavorable in the tested revision setting; broader robustness and language growth remain unestablished. The results motivate an agent that continually improves its provisional model, while showing why persistence, self-description and successful local validation alone do not establish such improvement.

2 Typed structural state

2.1 Concepts, generation structures and identity

Represent a Metadata item as $m = (\tau_m, G_m)$, where τ_m is a structural type and G_m is an executable generation structure or an anchor to an observed input. A cached value gives the extended form $m = (\tau_m, G_m, v_m)$; the value does not determine the whole item. A concept is

$$C = (M_C, R_C, \Sigma_C), \qquad M_C = \{k_i \mapsto m_i : \tau_i\}_{i \in I}.$$

M_C is a typed field map, R_C records typed relations and dependencies, and Σ_C contains the declared interface and applicable structural constraints. A name addresses the concept. Two items with the same current output can retain different generation identities. For example, a zero obtained from a conjunction and a zero obtained from a parity expression need not have the same behavior under a changed input.

Within an executable expression, composition follows typed input/output interfaces and can share subexpressions in a DAG. A reference, an observed association and a generation dependency are different edge kinds. We do not infer a causal dependency merely from a reference edge, or require that every semantic relation across concepts be an edge in one global acyclic graph. An atomic operator is atomic relative to the implementation's current language and resolution; this definition does not prohibit later refinement.

In the finite implementations, these objects appear as executable programs, candidate transformations, record-reading expressions, method-selection rules and scoped references to observations. Their content, rather than only their descriptive label, participates in execution. The experiments do not instantiate every possible Metadata type or the full proposed world-model architecture.

2.2 Current state and retained experience

For exposition, write the persistent state as

$$Z_t = (\mathcal{C}_t, \mathcal{H}_t, \mathcal{I}_t, D_t, P_t, B_t).$$

Here $\mathcal{C}_t$ contains available structural objects; $\mathcal{H}_t$ contains currently considered hypotheses; $\mathcal{I}_t$ identifies active interfaces and method-selection conditions; D_t contains actual observations, decisions and outcomes; P_t contains unresolved cases; and B_t records available resources and incurred costs. This is a common description of the experimental components, not a claim that all studies use one identical serialized class.

The current hypothesis set is conditioned on a language Γ_t and applicable observations:

$$\mathcal{H}_t = \{h \in \Gamma_t : h \text{ is compatible with the observations applicable at } t\}.$$

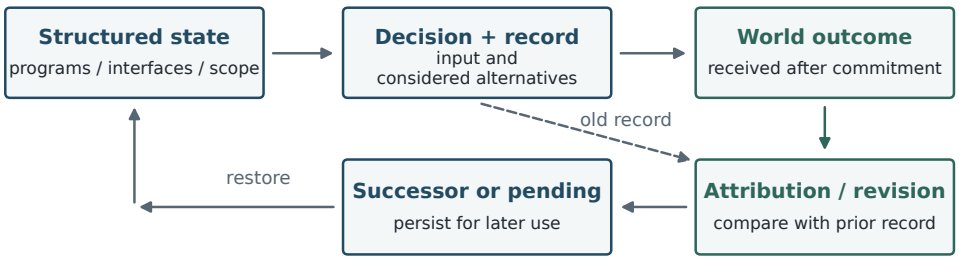


Figure 1: The revision loop and its state consumers. The historical record remains fixed while current executable structures may change. Each study implements a stated slice; the diagram does not assert one uninterrupted experiment.

In the deterministic tasks, compatibility is equality of predicted and observed outputs. Selecting a local answer from this set does not establish that the world belongs to Γ_t. Similarly, an empty set is a state of unresolved model/evidence incompatibility, not a successful explanation. It can support a pending record and later investigation without forcing a new answer.

An immutable decision record stores the input references, evaluated alternatives, selected action and known gaps at the time of decision. It is primarily resumable context for the agent itself. A later record may reject the old interpretation while retaining what was actually available then. Immutability applies to the historical record, not to the current hypothesis. In G32, the record is the explicit program's actual computation DAG; it is not a reconstruction of an unobserved neural chain of thought. Hypothetical branches are kept distinct from outcomes the agent actually experiences.

2.3 Experience-conditioned revision

The architectural loop separates proposing a model, deciding what to observe, experiencing a result, and modifying the state that will be used next:

$$(\widehat{y}_t, a_t, r_t) = \mathrm{Decide}(Z_t, o_t), \qquad e_t = \mathrm{World}(a_t), \qquad Z_{t+1} = \mathrm{Revise}(Z_t, r_t, e_t).$$

r_t is committed before e_t. Revision may eliminate hypotheses, change a field reader or selection condition, store an attributed structural difference, or retain a case for later input. A newly installed structure is then an input to subsequent computation. These possibilities describe the architecture; each experimental row below states which possibilities actually ran.

The implemented transition pattern is:

1. Restore the current state and obtain the observations available for the task.
2. Read applicable structures and generate candidate explanations or methods under the current language.
3. Compute an action or a proposed observation, recording the actual alternatives considered and reasons for unexpanded alternatives when available.
4. Commit the decision record; obtain and charge the actual world response.
5. Use the resulting discrepancy or missing comparison to update hypotheses, investigate, reconstruct or remain pending.
6. Persist a successor state. A later process uses that successor without rewriting the earlier record.

The prototype supplies the primitives, candidate constructors, prices and selection procedures. Some *outputs* of these procedures are learned from experience, such as the selected record path and revised branch condition. Providing a learning algorithm is compatible with measuring learning; replacing that algorithm between experiments is not counted as an autonomous learner update. In particular, current evidence does not show autonomous invention of an unrestricted new hypothesis language.

2.4 CONDITIONS FOR OPERATIONAL SELF-EXPLANATION

Self-explanation here means that the agent can recover and inspect the actual dependencies of its own computation. It requires the following architectural conditions, beyond storing readable descriptions.

1. **Mediated internal changes.** Changes to executable concepts, active interfaces, the updater itself and accepted shared knowledge must pass the agent's versioned admission-and-update procedure. An external contribution can propose a change or supply evidence; its author cannot directly assign an accepted internal state or rewrite a lived record. The procedure may initially be supplied by the designer.
2. **Execution-grounded history.** Before an outcome arrives, the record binds the actual input and state versions, operators invoked, alternatives evaluated, stopping conditions and unresolved dependencies. Later interpretation is a new record. Uncomputed alternatives remain uncomputed; a plausible retrospective narrative cannot replace the executed path.
3. **Qualified shared knowledge.** Shared objects retain their generation structure, source, evidence or proof assumptions, applicable scope and version dependencies. Admission checks these relations. A mathematical proof under stated assumptions, applicability to this occurrence and the consumer's own experience are distinct statuses; neither a contributor's assertion nor a stored success automatically supplies all three.
4. **Self-readable dependencies.** The agent can resolve the retained references and use them in later diagnosis or revision. Every dependency needed for the claimed reconstruction must remain available; an omitted dependency is an explicit explanatory gap.

Under complete records and reproducible operator semantics, these conditions permit reconstruction of the recorded decision from its actual prior state and inputs (Appendix G). The guarantee concerns why that computation occurred, including a mistaken one. G32 demonstrates pre-outcome recording and later self-readback in a finite diagnostic; the present experiments do not jointly establish all four conditions for a complete persistent organism. Testing the full claim must distinguish an external alteration that bypasses the specified update path from an unsupported update admitted by that path. The latter is a failure of the proposed mechanism itself.

3 EXPERIMENTAL DESIGN AND EVIDENCE SCOPE

We use deterministic modular-relation and Boolean-program worlds, plus a finite probabilistic decision diagnostic. An experiment controller has the world function; a learner receives only its declared inputs and requested responses. Thus a world may be exhaustively checkable by the investigator without being fully observed by the learner. Outcomes used for author-side audits are not added to learner experience unless the experimental protocol actually delivers them.

The central continuation studies use known Boolean world functions with new input occurrences. G19 and G20 each evaluate 32 occurrences from 16 known functions, with 16 service outputs per occurrence. G29 and G30 use the corresponding finite work setting. These are not hundreds of independent world samples. G31 uses 36 specified worlds and reports a common unseen-output subset. G32 instead supplies the target relation and varies missing facts, available observations and prices. Its complete enumeration is a finite diagnostic, not a population estimate.

Experience-stream repeats (R1). We hold the implementations, method library, tariffs and world collection fixed while generating 12 new streams before interface learning. Each appends one or three paid observations to each of 56 training episodes, changes arrival order and service inputs, and reruns method acquisition/comparison before G19 and G20. Two pilots and the original run are excluded. The 12 seeds were fixed before inspecting pilot performance tables. We retain streams without a revision target. Paired intervals resample whole streams, conditional on the shared ancestry and world collection; Appendix F gives the protocol and individual results.

Matched interventions. We compare actual use with storing the same object without using it; a revised interface with its frozen predecessor; historical observations with a current-only input; and structural implementations with ordinary programs given the same relevant information. Strong controls can use complete candidate search or previously obtained complete programs. We do not claim an architectural gain when a weaker control merely repeats work that a normal cache avoids.

Separate cost scopes. For a declared horizon T, the relevant account has the form

$$C(T) = C_{\text{acquire}} + C_{\text{install}} + \sum_{t \le T} (C_{\text{observe},t} + C_{\text{plan},t} + C_{\text{use},t} + C_{\text{revise},t} + C_{\text{persist},t}).$$

This is an accounting decomposition, not a claim that every study measures every physical cost. We label observed operation counts, internal wall time and simulated charges separately. W denotes the specified work-currency of a study, not GPU FLOPs or purchased tokens. Acquisition is not charged again merely because a later process restores its result. Public subsidy changes who pays a charge; it does not remove the charge. The G20 and G29 tables cover their stated new work, not the entire prehistory of the parent agent. No financial total is pooled across different studies.

Stored results retain source references and implementation checks. Development revisions and post-hoc inspections are labeled as such. Separate checking code written in the same research process is implementation cross-checking, not independent authorship or external replication. The artifact index identifies the original local study IDs, so the reader can distinguish a genuine continuation from a new diagnostic with a related question.

4 RESULTS

4.1 EARLY CONTROLS: TYPED STATE DOES NOT MAKE ORDINARY REUSE UNIQUE

An ordinary version-space reconstruction matches the tested plural predictor both online and on its later targets (P0). In the finite reuse study P1, a typed store and a complete-context cache also agree on correctness, target checks and expression execution. Revalidating each request separately costs more checks, but that difference is already obtained by the ordinary cache.

Table 1. P1 finite reuse. Counts aggregate the same 36 configurations within each method. Execution nodes exclude qualification checks.

Method	Correct / requests	Target checks	Execution nodes
Typed state	5,355/5,355	1,380	8,415
Complete-context cache	5,355/5,355	1,380	8,415
Reverify	5,355/5,355	41,055	8,415

P2 supplies the same solved programs to the native fragment mechanism and the released Stitch library learner. On the later program pool, total application-node representation, including the library, is 6,603 for the native fragment and 6,681 for Stitch, compared with 6,831 without a library. All three expand to 38,880 executed primitive nodes. The native candidate shape is supplied in advance; Stitch is an actual library-learning control. This study separates a representation difference from an execution difference and does not establish general superiority of either library.

These controls establish where the proposed architecture overlaps ordinary learning and programming mechanisms. The later question is whether experience changes the particular structural objects consumed by the agent, and what those changes accomplish.

4.2 EXPERIENCED OUTCOMES REVISE ACTIVE INTERFACES

In the original G19 continuation, a bounded constructor generates 296 new record-reading expressions from paid method-comparison records and selects one for use. It reads the ninth input component in the tenth observation record. The new interface delivers 485/512 correct outputs, versus 512/512 for both the inherited interface and complete method. G20 then purchases comparisons on all 12 occurrences in the implicated readout cell, including successful occurrences, and activates a previously generated reader of the ninth component of the eighth observation. It removes the frozen predecessor's 15 errors on later inputs. This is an update to an existing candidate interface, not a new primitive or proof that interface construction improved the original learner. The original costs and controls remain in Appendix C.

Repeatability across new experiences. R1 reruns this sequence from before interface learning. Its 12 newly executed loss tables yield 11 distinct active-expression sets. G19's new interface beats the inherited interface in four streams, ties in three and loses in five; the mean difference is -5.25 correct outputs per 512 (paired 95% interval [-22.25, 11.50]). New fields do not ensure a better decision rule.

G20 installs revisions in 11 streams; one has no observed failure and keeps its interface. Compared with freezing the same predecessor, revision improves subsequent accuracy in ten streams and ties in two, with a mean gain of 18.17 outputs per 512 ([10.08, 26.83]). Against the complete method, it ties in seven and loses in five. Table 2 includes all streams, including the no-target outcome.

Table 2. R1: G20 means across 12 experience streams. All policies continue the same G19 new-interface state within each stream and face 32 matched occurrences, with 512 delivered outputs. Acquisition includes preparation, paid comparisons and revision; W means are rounded to the nearest unit. Correct counts are means, not a pooled independent sample.

Method	Correct / 512	Use W	Acquisition W	Net W
REVISE	508.92	1,068,420	1,028,252	2,992,495
FROZEN	490.75	1,030,562	15,732	3,861,207
RETIRE	512.00	1,070,050	15,732	4,034,218
FIXED_C	512.00	1,060,439	15,732	4,043,829

REVISE updates the interface, FROZEN retains it, RETIRE removes active fields, and FIXED_C always selects the complete method. In this batch RETIRE also selects complete search. The complete method attains 512/512 in every stream. Net return including new acquisition is lower for REVISE than both FROZEN and FIXED_C in all 12 streams: mean differences are -868,711.83 W and -1,051,334.42 W, respectively. Even in all seven accuracy-matched comparisons with FIXED_C, REVISE's subsequent-use cost is higher. Thus the repeated local predictive benefit has not produced a cost-adjusted advantage over either control in the observed horizon.

A post-hoc common-unseen audit retains the ten-improvement/two-tie pattern against FROZEN (Appendix F). The no-target stream is also informative: it delivers 512/512 in G19, triggers no revision, then delivers 503/512 in G20. Absence of observed failure in one window is insufficient to certify the next. Neither these failures nor the unfavorable costs establish an architecture-wide lower bound or returns under a different future reuse distribution.

4.3 Attribution becomes executable state, but scope fails

G28 constructs an attribution workflow for a saved failure: it compares the actually used generation expression with candidate revisions under the old observations and later outcome. G29 invokes this procedure from a saved continuing agent, using its own historical records. The derived difference changes an inner AND to OR and changes an input pair in a subexpression. The procedure is author supplied; the difference is obtained during the learner's paid call rather than installed from the author's earlier answer.

G29's USE and STORE controls both acquire and persist the same K. They differ in whether K participates in candidate generation for later work. USE actually invokes the new route on four occurrences. Its recorded loading and candidate construction precede world queries. This intervention establishes consumption, rather than mere presence, of the learned object.

Table 3. G29 attribution and subsequent use. Charges include the current continuation's restoration, research, work and persistence. Earlier parent costs are not charged again.

Method	Correct / outputs	New provider charges W
USE	505/512	1,403,832
STORE	512/512	1,426,913
FROZEN	512/512	1,420,761
FULL	512/512	1,401,114

FROZEN preserves the old method selector without buying attribution; FULL directly uses complete search. USE and STORE both pay for the acquired K.

The new local candidate family contains the true function for one world and omits it for another. In the latter, occurrence 004 nevertheless delivers 16/16 correct outputs; occurrence 020 later delivers 9/16. Both retained programs were consistent with their limited checks. An author-only full-domain inspection finds 64 and 128 errors, respectively, over the 512-point Boolean input domain. Those additional outputs are diagnostic evidence, not experience supplied to the learner.

The causal failure is specific: an attributed difference generates a restricted candidate family, and the supplied stop rule accepts a locally surviving candidate without establishing coverage of the new world. G29 also does not yet feed every earlier same-world outcome back into the current task's observations. A successful local use therefore cannot establish the general applicability of the attributed transformation.

G30 addresses the latter omission by indexing the continuing agent's own old outcomes and using them in subsequent filtering. On the common 493 service points unseen before delivery, HISTORY, complete search with history and ordinary complete-program reuse all obtain 493 correct; a current-only version obtains 492. Ordinary program reuse has the smallest reported new provider charges, 1,199,834 W, compared with HISTORY's 1,254,483 W. History can constrain later computation, while a capable ordinary program library remains a necessary comparator. This repair does not establish that the restricted constructor now covers arbitrary new mechanisms.

4.4 Reopening a case is distinct from resolving it

G31 continues from G30's complete-program-reuse state. New work leaves 15 pending cases. A paid attribution supplies one additional K; the indexed rule reopens nine pending cases, while an all-pending rule reopens all 15. In both, the newly generated candidates are rejected by the old observations, so the new-K route issues no new probe and resolves none of these cases. Other predefined search routes can still obtain usable programs. Their success is not attributed to the failed new-K route.

This result distinguishes three events often conflated in a learning loop: remembering that a question exists, reconstructing candidate explanations, and obtaining discriminating new evidence. It also exposes a trigger limitation: cases that passed earlier checks are absent from the pending index. The implementation has not learned a general rule for when new knowledge should reopen previously accepted interpretations. Pending itself is an allowed outcome; the unresolved question is whether the right new evidence or structure later changes what the learner can do.

4.5 A decision record can preserve why an action was reasonable

G32 isolates interpretation of a previous pause or action. Its world consists of six bits, with either parity or conjunction as the supplied target. Unobserved bits have a supplied independent fair prior. Correct/incorrect answers earn $+R/-R$, and HOLD earns zero. A finite dynamic program compares stopping with priced observations; a second ordinary implementation exploits the given parity/conjunction identities to lower planning cost. Decisions are recorded before outcomes.

Across the 2,064 paired cases with zero computation price, full planning and the specialized method agree on their initial decision, final status/action, observation spending and wallet. With computation priced, the cheaper method changes 208 initial HOLD decisions and 180 initial ACT decisions to QUERY. The relation was supplied by the author, so these are consequences of an available cheaper computation, not autonomous discovery of a planning algorithm.

A saved minority-outcome loss retains the uncertainty and favorable expected value that supported the original action; later evidence narrows its possible worlds without rewriting that rationale. Conversely, a correctly computed answer can cost more than its reward. Appendix D retains both examples. Readback was author-invoked; learning when to purchase it is untested. G32 is a separate diagnostic with G30 account lineage, not the whole G30 learner body.

5 WHAT THE FINITE SLICES ESTABLISH

Structural state is operational. The strongest positive evidence is not a diagram or a prose explanation. It is an intervention on what a subsequent computation consumes: a paid update changes an active interface in G20; USE differs from STORE in G29; old outcomes enter hypothesis filtering in G30. R1 adds repeated local predictive improvement after revision, alongside consistently worse cost-adjusted returns in the tested horizon. These findings support the experience-conditioned revision claim within the implemented languages and curricula while separating it from competitive efficiency.

Feedback coupling is weaker than reliable expansion. G29's connected acquisition-attribution-use chain generalizes incorrectly. G31 reopens cases without creating a surviving explanation or a new probe on that route. These observations identify a missing competence: improving the set of hypotheses and its applicability conditions, including cases in which the current family agrees for the wrong reason. Adding labels for uncertainty does not itself supply that competence.

Structure can change computation, conditionally. A separate observation-planning study, G9, gives a capable cached planner an additional bound derived from a supplied Boolean generation expression. On a parity case where only half the necessary observations can be purchased, exact risk evaluations fall from 59 to 1 with the same selected action. Internal median time falls from 1.260 to 0.359 ms over three process repetitions; process startup and output can dominate this difference. Other cases do not benefit, and all arms still prepare the finite joint distribution. This establishes a local computational use of a relation, not an architecture-wide complexity improvement. Appendix B states the bound and its assumptions.

The current K remains provisional. A learner can use a structure under its current conditions while retaining uncertainty about broader scope. Its history permits later comparison with new evidence. We do not require an assertion of final truth to use a K; we do require the empirical paper to distinguish what changed, which evidence caused the change and whether future behavior improved. The decision record serves the learner's future reasoning rather than assigning every external judgment automatic influence over its state.

6 RELATED WORK

Candidate elimination has a direct antecedent in Mitchell (1977). Li, Littman and Walsh (2008) formalize learning that can decline a prediction when it does not know. Our P0 equality result confirms this overlap on the tested stream; our finite learners do not obtain a general KWIK guarantee under hypothesis misspecification.

DreamCoder learns program abstractions together with a search-guiding neural model. Stitch synthesizes reusable library functions from a program corpus. The experiments here focus on persistent outcomes, applicability and later state revision; P2 actually runs Stitch, but we have not implemented a matched full DreamCoder agent. We do not treat library learning, compositional abstraction or an experience-dependent program state as unprecedented in isolation.

Adaptive Functional Programming provides a relevant account of dependency recording and change propagation. In our setting some structures describing a world or a method's applicability must themselves be inferred and may be wrong. This motivates keeping acquisition, executable dependency and qualified use separate; it does not make memoization a new mechanism.

World Models studies learned internal environment representations for control. Our discrete typed latent state is a different implementation choice, and these experiments are not a comparison against modern neural world models. Hay et al. (2012) formalize the value of selecting computations. G32 uses this established metareasoning perspective to distinguish missing information from computation that was not worth buying. The issue studied here is how such distinctions can remain available across the agent's later revisions.

7 LIMITATIONS AND NEXT DECISIVE TEST

The raw modalities, operator languages, constructors, budgets, simulated prices and update procedures are supplied. Some studies continue the same saved learner; others introduce a new diagnostic or an

author-modified mechanism. The architecture has not autonomously redesigned every component. The evaluated worlds are small and mostly reuse known function families. Observed local checks do not establish arbitrary model coverage, and persistent storage alone does not measure retention of useful behavior across a changing task distribution.

Counts over finite grids are descriptive. R1 varies new experiences under a fixed ancestor and function collection, not independent world families; its paired bootstrap intervals resample streams, not service outputs. Twelve streams do not establish deployment robustness. Cost fields omit some physical overhead; logical work, memory representation and wall time cannot be exchanged without a measured conversion. This draft does not include a new, adequately tuned neural architectural comparison or a matched end-to-end competing continual learner. The measured costs cover finite work horizons. They do not value every future structural capability or measure controlled changes in the hypothesis language.

A decisive continuation would hold the update mechanism fixed, let genuinely later outcomes change the learner's own reopening or applicability decisions, and compare against a capable ordinary learner with the same histories and resources. It should distinguish affordable missing evidence from a currently inadequate hypothesis family, retain reasonable pending decisions, and measure quality and cumulative cost on later unseen occurrences. Such a run is a proposed test, not part of this paper's results. The current evidence already supports the narrower executable revision claim, while leaving its reliability and economic scope explicit.

8 Conclusion

We formulate learnable state as typed Metadata clusters with executable generation structures and scope conditions. Finite implementations connect experienced outcomes to interface revision, attribution and later reuse. Across 12 new experience streams, revision improves accuracy over its frozen predecessor in ten and ties in two, yet lowers cost-adjusted return in all twelve; complete computation remains stronger. Ordinary learning and caching reproduce several behaviors, and a connected feedback loop can still reuse a family that omits the true relation. The remaining challenge is reliable adaptation of hypotheses and validation scope, evaluated with acquisition and continued-use costs.

AI USE STATEMENT

Generative AI systems were used in conceptual formalization, hypothesis and experiment design, implementation, result interpretation, source checking, mathematical derivations, drafting and figure preparation. The research originated from human-proposed structural concepts and interaction with AI collaborators. The present evidence extraction reads saved experimental outputs, and the reported implementation checks were also developed with AI assistance. We distinguish these checks from independent external replication and do not claim an unperformed manual human audit.

REPRODUCIBILITY STATEMENT

The anonymous supplement contains this paper and its textual appendices, all 56 saved JSON sources from which the reported fields and tables are re-extracted, a field-level claim map, the relevant implementation sources and frozen configurations, and a standard-library verifier that runs after unpacking in an empty directory. It also contains bounded runnable checks for P1 and G32. These materials support saved-result, table and implementation auditing; they do not constitute an end-to-end replay of every ancestral agent state. The supplement README and dependency map state the exact boundary and commands.

REFERENCES

Acar, U. A., Blelloch, G. E., and Harper, R. Adaptive Functional Programming. *ACM Transactions on Programming Languages and Systems*, 28(6), 2006. Author manuscript.

Bowers, M., Olausson, T. X., Wong, L., Grand, G., Tenenbaum, J. B., Ellis, K., and Solar-Lezama, A. Top-Down Synthesis for Library Learning. *Proceedings of the ACM on Programming Languages*, 7(POPL), 2023. Paper.

Ellis, K., Wong, C., Nye, M., Sable-Meyer, M., Cary, L., Morales, L., Hewitt, L., Solar-Lezama, A., and Tenenbaum, J. B. DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning. 2020 preprint. Paper.

Ha, D. and Schmidhuber, J. World Models. 2018. Paper.

Hay, N., Russell, S., Tolpin, D., and Shimony, S. E. Selecting Computations: Theory and Applications. *UAI*, 2012. Paper.

Li, L., Littman, M. L., and Walsh, T. J. Knows What It Knows: A Framework for Self-Aware Learning. *ICML*, 2008. Paper.

Mitchell, T. M. Version Spaces: A Candidate Elimination Approach to Rule Learning. *IJCAI*, 1977. Paper.

O'Donnell, R. Analysis of Boolean Functions. Cambridge University Press, 2014; corrected author version, 2021. Author version.

A STUDY LINEAGE AND REPRODUCTION STATUS

The experiment identifiers below locate preserved artifacts; they are not independent replications or a chronology of one unmodified runtime.

Study	Actual object or continuation	Evidence role
P0	Reconstruction from an existing finite stream	Ordinary version-space equality
P1	Finite relation reuse with complete conditions	Cache, qualification and restoration control
P2	Shared solved-program source	Native and Stitch fragment comparison
G9	Supplied programs and priors	Conditional planning-work difference
G19	Continuing learner and prior paid records	Generated/selected interface
G20	G19 continuing state	Paid comparison changes active interface
R1	12 new G17-G20 experience streams; fixed ancestors	Repeated local benefit and unfavorable total return
G28	Saved G26 decision and outcome	Author-side attribution workflow
G29	G27 state, invoking that workflow	Acquisition, storage and later consumption
G30	G29 USE state	Historical outcomes enter current filtering
G31	G30 PROGRAM_REUSE state	Pending reconstruction after K arrival
G32	Separate supplied-model diagnostic	Pre-action computation and later readback

G19 constructs candidates from record paths, aggregations and comparisons using a bounded provided grammar. The concrete record path is selected empirically, not known to have a universally valid semantic interpretation. G20 revises a particular questioned readout cell using additional paid comparisons. G29 receives a reusable difference inferred from an earlier failure but uses it under a separately supplied local candidate/stop rule. G30 incorporates prior observations using their stable world provenance; its result does not establish how to infer such identities in an unknown deployment.

The reported results are preserved source-bound runs. The manuscript's table extractor rereads their JSON and records each used field and file hash. The anonymous supplement contains derived, path-sanitized exports of all 56 cited JSON sources, a new manifest for those exported bytes, the extractor's field-level claim map and expected tables, selected source/configuration snapshots, and a verifier that runs from the unpacked package root. This makes numerical quotation reproducible without rerunning learning. Bounded P1 and G32 checks are runnable from the same package. The dependency map separately identifies the larger parent-state closure required for complete end-to-end replay. Existing experiment checks recompute outputs and relevant account transitions, but remain within the same AI-assisted research process rather than independent external replication.

The continuous prototypes often use known world functions with newly acquired inputs. G30 originally reports 512 outputs; its common unseen subset has 493. G31 reports 576 outputs, with 564 common unseen points. Only the latter denominators support the respective unseen-input statements. This is stricter than equating a new process with a new generalization sample.

B A SUPPLIED GENERATION RELATION CAN BOUND OBSERVATION SEARCH

G9 provides a positive computational contrast under restrictive assumptions. Let $Y = F(X)$ for independent fair bits, and let $f(z) = 1 - 2F(X)$ with $z_i = 1 - 2X_i$. Using standard Boolean Fourier identities from O'Donnell (2014), write

$$f(z) = \sum_A c_A \chi_A(z), \qquad \mathbb{E}[(\mathbb{E}[f \mid z_T])^2] = \sum_{A \subseteq T} c_A^2.$$

At a batch-search node, P is already selected, R remains eligible and b is the remaining cardinality budget. The quantity

$$U(P, R, b) = \sum_{A:\ A \subseteq P \cup R,\ |A \setminus P| \le b} c_A^2$$

upper bounds the second moment of the posterior mean for every feasible completion. Individual terms in this sum need not be jointly attainable; it is an optimistic relaxation. With abstention loss $0 < h \le 1/2$ and $m_T = \mathbb{E}[f \mid z_T]$,

$$\mathbb{E}[\min\{h, (1 - |m_T|)/2\}] \ge h(1 - \sqrt{U(P, R, b)}).$$

This follows from $\min\{h, (1-x)/2\} \ge h(1-x)$ for $x \in [0, 1]$ and Cauchy–Schwarz. Adding mandatory query costs gives a lower bound for branch pruning. The derivation uses standard identities and is not asserted to be a new general theorem. Unlike a singleton-information rule, it retains complementary terms when their support fits the remaining budget.

The implementation compiles coefficients from the given AST and falls back when its bounded compilation attempt is unsuitable. Both controls retain marginal caches and receive the same program. In case g9-008, both stop, but the specialized bound avoids most exact risk evaluations:

Method	Exact risk evaluations	Internal time (ms)
superset_bnb	59	1.260
spectral_bnb	1	0.359

Times are medians of three fresh-process repetitions and include model preparation, planning and forecast construction; they exclude complete subprocess startup and output. The internal saving therefore does not imply a full-process speedup. Both arms still prepare 4,096 joint assignments. Other program structures can make the bound loose or compilation unhelpful. This experiment does not learn the bound or remove exponential work from general inference.

C Additional continuation results

G19. Each policy receives the same grown starting state. The NEW_ALL control deliberately evaluates every candidate reader; it decomposes reader cost and is not a strong alternative learning architecture. Shared new acquisition is 602,086 W and is not included in the later-work column.

Method	Correct / outputs	Later-work cost W
NEW_ACTIVE	485/512	1,034,148
NEW_ALL	485/512	1,326,152
OLD_ACTIVE	512/512	1,092,791
FIXED_B	469/512	987,108
FIXED_C	512/512	1,084,666

G20 original continuation. These are the original single-stream values, not a thirteenth R1 replicate. Preparation is included with revision. REVISE has 843,430 W extra acquisition over FIXED_C and 11,573 W greater subsequent-use cost on this batch.

Method	Correct	Work W	Preparation + revision W	Net W
REVISE	512/512	1,122,704	859,080	3,138,216
FROZEN	497/512	1,097,464	15,650	3,856,886
RETIRE	512/512	1,120,747	15,650	3,983,603
FIXED_C	512/512	1,111,131	15,650	3,993,219

G30. The common-history methods and ordinary complete-program reuse all reach 493/493 on common unseen points; LOCAL reaches 492/493. The full delivery and new-charge totals below include already-seen outputs and are retained as work-account totals, not mislabeled generalization.

Method	Correct / outputs	New provider charges W
HISTORY	512/512	1,254,483
LOCAL	511/512	1,448,254
FULL_HISTORY	512/512	1,257,905
PROGRAM_REUSE	512/512	1,199,834

G31. The FULL control expands only cases already registered as pending and retains the earlier accepted pages. It is not an omniscient full-search control for all 36 worlds. Accordingly, its shared 513/564 unseen-output result does not mean complete search failed to identify these finite functions when actually invoked.

Method	Revisited	Pending now usable	Unseen correct
INDEXED	9	0	513/564
ALL_PENDING	15	0	513/564
STORE_ONLY	0	0	513/564
PROGRAM_REUSE	15	15	513/564
FULL	15	15	513/564

D COMPUTATION RECORDS AND HINDSIGHT

In G32, the complete-enumeration diagnostic covers every unknown-bit assignment in 288 conditions, giving 6,192 cases per method. An earlier four-draw sample missed rare conjunction outcomes; it remains preserved and is not used as a zero-error claim. Counts with different gap sizes have different numbers of assignments. Descriptive totals must not be interpreted as an unknown real-world workload's error probabilities.

In the saved conjunction example, two unknown fair bits give $P(Y = 0) = 3/4$. With $R = 16$ and query cost 8, answering zero has expected value 8, querying has value 4, and HOLD has value 0. The chosen answer loses 16 in the remaining quarter of worlds. Later readback narrows four possible worlds to one while preserving the original rationale.

The initial decision, its actual computed branches and any unexpanded alternatives are committed before the world responds. A later, author-invoked readback first loads the old record and old conditions, saves that reading, and only then receives the saved outcome. It does not construct motives from success or failure. A matched win/loss pair has the same pre-outcome reading; the minority loss supplies a new constraint, not a retroactive change in the original probability.

All policies use the current opportunity window's supplied utility. Later windows offer a price reduction and then a free missing fact; waiting itself is unpriced. The original planner does not optimize over that future schedule. The specialized method uses author-provided parity/conjunction identities, and the purchase gate uses an optimistic upper bound on potential gain. In one retained example, planning costs 14.875 W and six observations cost 6 W for a correct answer rewarded 16 W. With the common intake and prefix charges, the result is a net loss of 11.890625 W. This is a concrete gap in the supplied purchase procedure, not a judgment that exploratory learning is inherently unprofitable.

E WHAT REMAINS OUTSIDE THE PRESENT EMPIRICAL CLAIM

The main text does not report the next continuous experiment as completed. In particular, it has not demonstrated an agent learning its own general reopening rule from the saved decision trees, indefinite reliable hypothesis-space expansion, competitive scaling against a tuned neural learner, guaranteed absence of forgetting, or a complete organism's self-explanation. A future study may test these questions without requiring a final-truth declaration by the agent.

The conceptual division remains useful: retain the actually generated context, let reality constrain current hypotheses, and judge a revision by what it changes in later use. Reliability and the conditions of useful reuse remain distinct from how long the agent can persist.

F REPEATED EXPERIENCE STREAMS (R1)

F.1 FROZEN DESIGN AND ACQUISITION LINEAGE

R1 supplies new experience before interface construction rather than rerunning a fixed learned state. Each stream starts from the same G7 archive and 100,000,000 W grant, inherited G16 episodes, partial public library, 56 learning functions and 16 later functions. For each learning episode a deterministic seed-dependent hash chooses one or three unseen observation points with equal probability; it then selects 16 distinct service inputs and a new arrival order. Actual world receipts are appended without rewriting the old history. Each additional observation costs 1,000 W plus the existing prediction-work and payload charges, which are recorded before G17.

The unchanged G17 selector acquires the available methods; G18 buys its missing comparisons and fits its inherited interface. The new paid loss table and actual record views then enter G19. G19 presents one- and three-observation prefixes; G20 presents two- and four-observation prefixes. Later inputs exclude the preceding stages' receipts. G20 future inputs are committed before G20 research, not selected after seeing its outcomes. Subsequent method queries can nevertheless expose a service point; the separate common-unseen audit below accounts for that.

G19 compares NEW_ACTIVE, OLD_ACTIVE and FIXED_C. G20 starts all four policies from the same NEW_ACTIVE G19 terminal state, preserving within-stream ancestry and matching occurrences. Its comparison is therefore not a never-research-from-birth counterfactual. Upstream G17/G18 charges are recorded separately. Original core algorithms and tariffs remain fixed; a path adapter directs each stream to its own records. An explicit no-target branch keeps the old interface and charges the state/evidence check rather than fitting an empty dataset. Incomplete research cohorts retain the existing keep-old rule.

Two pilots (91001, 91002) are excluded. The formal seeds 92001-92012 were fixed from runtime and storage considerations before pilot performance tables were inspected; formal execution followed source freezing. No outcome-dependent sample extension was made. A preliminary checkpoint-file write failure and a verifier correction for G18's additional packet-size accounting field occurred; neither changed a formal outcome or its algorithm. The original G19 construction and nonempty G20 revision were exactly reproduced before running the new streams.

F.2 PER-STREAM OUTCOMES

Each correct count below is out of 512. The last column is G20 REVISE minus FIXED_C net return, including new acquisition. Seed 92003 had no observed G19 failure and no revision target. All other streams installed a revision; 92006 installed one without improving later accuracy against FROZEN. No such stream was excluded.

Seed	G19 new / old	G20 revise / frozen	Complete	Net difference W
92001	471 / 502	512 / 509	512	-953,034
92002	482 / 482	512 / 488	512	-1,010,177
92003	512 / 462	503 / 503	512	-83,645
92004	466 / 462	512 / 462	512	-1,611,212
92005	501 / 467	506 / 494	512	-672,869
92006	506 / 512	509 / 509	512	-530,354

Seed	G19 new / old	G20 revise / frozen	Complete	Net difference W
92007	446 / 512	507 / 475	512	-1,706,077
92008	480 / 504	512 / 484	512	-933,266
92009	483 / 475	512 / 501	512	-1,253,519
92010	463 / 463	512 / 485	512	-1,369,653
92011	472 / 504	512 / 486	512	-913,177
92012	470 / 470	498 / 493	512	-1,579,030

The REVISE-versus-RETIRE comparison separates later-use cost from accuracy for every stream. Across all 12 streams, the mean REVISE-minus-RETIRE use-cost difference is -1,630.25 W, with a paired bootstrap 95% interval of [-6,561.59, 2,744.95] W. In all seven streams where the two policies have equal accuracy, REVISE has greater use cost. The other five streams have lower REVISE use cost and lower accuracy. None of these twelve streams shows lower use cost at equal or greater accuracy than RETIRE.

Seed	Accuracy difference	Use-cost difference W
92001	0	3,441
92002	0	6,307
92003	-9	-18,892
92004	0	4,299
92005	-6	-5,400
92006	-3	-6,245
92007	-5	-3,055
92008	0	2,084
92009	0	6,162
92010	0	4,839
92011	0	3,312
92012	-14	-16,415

The active reader expressions differ across experiences; this is measured state variation, not itself a failure or a success criterion. The meaningful paired comparisons are the subsequent outputs and costs.

F.3 Paired intervals and common unseen outputs

Intervals use 20,000 percentile bootstrap resamples of the 12 paired whole streams, with fixed resampling seed 19317. The target is variation in this new-experience protocol conditional on the common ancestry, function collection and implementation. Shared task structure limits generalization beyond that setting. These are reported estimates, not a proof of universal consistency or an architecture-wide cost floor.

Comparison	Metric	Mean difference	95% interval
G19 new - old	Correct outputs	-5.25	[-22.25, 11.50]
G19 new - old	Net W	-42,771.92	[-168,831.49, 85,716.26]
G20 revise - frozen	Correct outputs	18.17	[10.08, 26.83]
G20 revise - frozen	Net W	-868,711.83	[-1,087,388.60, -634,285.14]
G20 revise - complete	Correct outputs	-3.08	[-5.83, -0.83]
G20 revise - complete	Net W	-1,051,334.42	[-1,304,914.45, -783,560.83]

The mean additional acquisition cost for REVISE over FROZEN or FIXED_C is 1,012,520.42 W; subsequent-use cost is greater by 37,858.08 W over FROZEN and 7,980.67 W over FIXED_C. Mean additional income over FROZEN is 181,666.67 W. In all seven accuracy-matched streams, use cost exceeds FIXED_C; the only two use-cost reductions against FIXED_C occur with nine and fourteen additional errors.

The post-hoc common-unseen audit excludes inherited observations, earlier-stage and research receipts, earlier service answers, and pre-delivery queries made by any matched arm on the current

occurrence. Original delivery and cost accounts are retained. The following pooled counts describe delivered outputs, not independent worlds. The per-stream accuracy pattern against FROZEN remains ten improvements and two ties.

Method	Common unseen deliveries
REVISE	5,996/6,033
FROZEN	5,779/6,033
RETIRE	6,033/6,033
FIXED_C	6,033/6,033

F.4 Verification and interpretation boundary

All 12 formal streams passed saved-state, receipt and scalar-output checks, exact interface reconstruction and full subsequent-use replay within the same AI-assisted collaboration. Execution transferred between collaborators mid-batch without changing code, configuration or seeds; this is not independent-team replication. The supplement supplies the source-freeze manifest, per-stream verification and cost summaries, and common-unseen audit; it does not contain the complete per-occurrence object stores and event histories.

The repeated result is a local improvement over a learned predecessor accompanied by unfavorable finite-horizon return. It does not isolate learned representation from every possible alternative selector, estimate a deployment distribution, establish autonomous growth of an unrestricted hypothesis language, or measure long-run return under changed reuse conditions. The original single stream remains evidence of its particular continuation, not an extra independent sample.

G Conditional reconstruction and evaluation responsibility

Write a recorded computation as $s_{j+1} = f_j(s_j, u_j)$. Suppose the initial state, each actual operator version, all consumed inputs and all nondeterministic choices are available through the decision record. Induction on the recorded steps reconstructs the terminal state and selected action. A retained dependency DAG can share computations without changing this argument. This is a conditional reconstruction property, not a claim that the modeled world is correct or that every uncomputed alternative was understood.

Mediated updates make the relevant state lineage identifiable: an accepted successor refers to the prior state, the actual update procedure, its proposed change and the evidence inspected. Historical records remain fixed; the agent can replace a hypothesis and record why. Admission includes shared knowledge that will affect computation, while retaining its borrowed origin and assumptions. A digest alone cannot establish that the recorded object was actually executed or that its contents are justified.

An evaluation should therefore report both the modification path and the resulting behavior. Directly replacing internal state outside the declared procedure is a distinct intervention; it does not alone refute the conditional reconstruction property. A normal external contribution that the procedure wrongly accepts is an admission failure. Missing execution dependencies, an unrecoverable record, a mismatch on replay or failure of the subject's required readback refutes the corresponding explanatory condition. These distinctions assign a failure to a concrete component without making the architecture immune to counterexamples.