PACT-AGENT: PROSPECTIVE ACTION-CONTRACT TESTING FOR SELF-EVOLVING LANGUAGE AGENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Language-model agents can adapt through external memory, but most systems learn retrospectively from trajectories generated without considering what those trajectories can identify. An extracted rule may therefore fit an observed outcome without having predicted it. We introduce **PACT-Agent** (**P**rospective **A**ction-**C**ontract **T**esting), a self-evolving agent grounded in Active Inference. PACT-Agent represents uncertain environmental dynamics as competing mechanisms in unresolved rules and connects LLM reasoning to explicit numerical inference through predictive contracts. During online interaction, a Worker agent predicts mechanism-conditioned traces and minimizes a contracted *Expected Free Energy (EFE)* objective to select actions that balance information gain against utility. After execution, a Builder agent minimizes local *Variational Free Energy (VFE)*, accepting memory revisions only when improved trajectory interpretability outweighs added description complexity. Across multiple LLM-agent benchmarks, PACT-Agent consistently improves over strong memory-based agents. These results show that prospective predictions turn Active Inference into an explicit and auditable procedure for agent self-evolution.

1 INTRODUCTION

Large language model (LLM) agents can navigate websites (Kim et al., 2023; Sodhi et al., 2024), manipulate software interfaces (Hong et al., 2024; Qian et al., 2023), and act in embodied environments (Ahn et al., 2022; Wake et al., 2023). They remain brittle when an environment contains conventions absent from pre-training: `Save` may stage rather than commit an edit, a dialog may expose a hidden confirmation state, or the same action may have different effects across contexts. Fine-tuning can absorb such knowledge but requires task-specific data and repeated optimization (Zeng et al., 2024; Putta et al., 2024). External memory instead lets an agent accumulate reusable experience while keeping the backbone fixed.

Most memory-based agents distill demonstrations, skills, rules, or reasoning patterns after completing a trajectory (Zhu et al., 2023; Wang et al., 2024; 2025b; Zhao et al., 2024; Fu et al., 2024; Chen et al., 2024; Ouyang et al., 2025). This retrospective workflow learns only from the evidence that the original policy happened to generate. If an agent edits a record, clicks `Save`, and later observes the new value, the trajectory may still support several mechanisms: `Save` persisted the edit, a later `Confirm` committed it, or another transition applied it. A Builder can fit a plausible rule afterward, but cannot recover a distinction the trajectory never tested. The problem is therefore not only what to remember, but what evidence to collect before writing memory.

Active Inference (Friston et al., 2017), grounded in the Free Energy Principle (Friston, 2010), couples action and learning under an internal generative model. Before observing an outcome, expected free energy (EFE) favors policies that pursue preferred outcomes while reducing uncertainty about latent causes. Afterward, variational free energy (VFE) evaluates an explanation by predictive error and complexity. This suggests a coherent self-evolution loop: use EFE to choose task-relevant tests, update mechanism beliefs from prediction errors, and use VFE to decide whether the evidence warrants changing long-term memory. Directly applying this idea to LLM agents is difficult, however, because hypotheses and predictions live in free-form text, outcome descriptions may be incompatible, and predictions can be rewritten after the result is known. Merely prompting an LLM to "minimize free energy" therefore leaves the numerical theory detached from the implementation.

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.

We introduce **PACT-Agent** (**P**rospective **A**ction-**C**ontract **T**esting), which makes a finite local generative model explicit before interaction. An unresolved Rule stores competing explanations of environment mechanisms and their beliefs. To test these explanations, we introduce *predictive contracts*, which specify, before execution, what each explanation predicts under candidate policies and how the resulting observations will be checked. This makes exploration a question of which actions can distinguish the mechanisms that matter for the task.

The system operates through two complementary components. A Worker agent proposes short policies, and a numerical module selects among them by explicitly minimizing *Expected Free Energy* (EFE), balancing task progress and interaction cost against expected information gain. Crucially, the same predictions used to select an action subsequently determine the Bayesian update of mechanism beliefs from its observed outcome. After each task, a Builder agent proposes local memory revisions. A numerical gate accepts a revision only if it reduces *Variational Free Energy* (VFE) on the completed trajectory, requiring any added description complexity to be justified by improved explanatory power.

Our main contributions are as follows:

- We introduce predictive contracts as a shared interface for task-directed mechanism testing, numerical action selection, and evidence-traceable belief revision.
- We instantiate Active Inference with finite-model EFE and local VFE objectives, and separate retrospective memory admission from prospective validation.
- We evaluate PACT-Agent on web automation, mobile control, and embodied planning, and the significant performance demonstrates the effectiveness of our agent self-evolution method.

2 Related Works

2.1 Self-Evolving LLM Agent

Memory-based Adaptation. A central approach to agent self-improvement is to retain experience in external memory while keeping the language model fixed. A prevalent strategy, GITM Zhu et al. (2023) and Voyager Wang et al. (2024), stores successful task trajectories in a *skill library* for in-context retrieval, while AWM (Wang et al., 2025b) extracts workflows for later tasks. ExpeL Zhao et al. (2024), AutoManual Chen et al. (2024) and ReasoningBank Ouyang et al. (2025) distill more general guidance from experience. These methods have made memory construction and retrieval central components of agent learning. PACT-Agent extends this view from retrospective consolidation to prospective evidence collection: an unresolved Rule remains a set of competing mechanisms until an interaction produces an observation that separates them.

Explicit runtime optimization. Recent methods attach numerical state to external memory rather than relying only on textual reflection. MemRL Zhang et al. (2026) learns utility estimates for episodic memories. JitRL Li et al. (2026) estimates advantages from retrieved experience and implements a KL-regularized policy update through logit modulation. Bayesian-Agent Wu et al. (2026) maintains context-conditioned skill reliability and uses posterior statistics to guide skill revision operations. PACT-Agent instead centers adaptation on *competing explanations of environmental dynamics*. Rather than only estimating whether a skill is useful, it predicts how alternative mechanisms produce different observable consequences.

2.2 Active Inference and Information Gathering

Active Inference (Friston et al., 2017; Parr et al., 2022) couples belief updating with action selection through variational and expected free energy. Discrete-state and deep implementations (Friston et al., 2017; Fountas et al., 2020) provide computational realizations of these objectives. For language model agents, posterior-sampling exploration (Arumugam & Griffiths, 2025) and BED-LLM (Choudhury et al., 2025) show how Bayesian beliefs and expected information gain can guide information gathering. PACT-Agent connects local information gathering to persistent memory revision. The same saved predictions govern policy evaluation and online belief updates, while a separate local model-comparison test governs which retrospective revisions are retained. Our derivations apply to this finite, explicitly represented model, rather than asserting exact inference over the full environment.

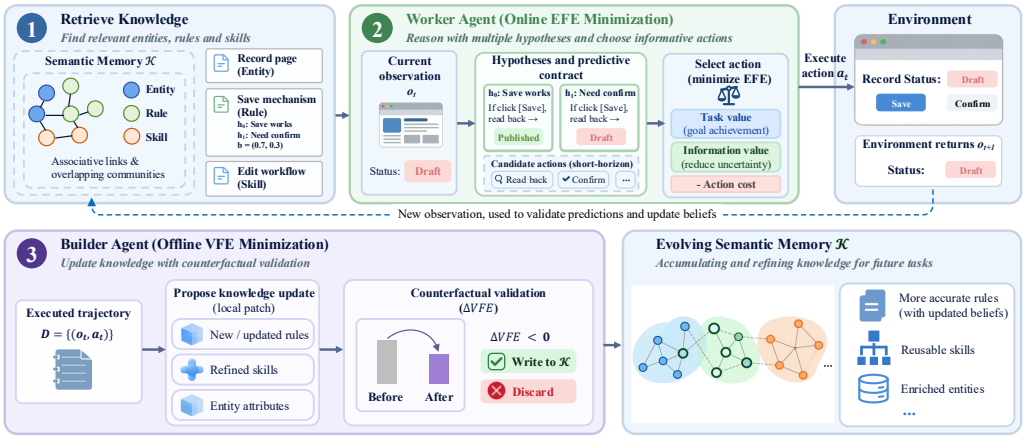

Figure 1: **Overview:** PACT-Agent instantiates the Active Inference loop into two LLM agents with a shared semantic memory $\mathcal{K}$. During online interaction, the Worker retrieves relevant knowledge, reasons with multiple hypotheses via predictive contracts and selects actions by minimizing expected free energy (EFE). After task completion, the Builder analyzes the executed trajectory and updates the memory by minimizing variational free energy (VFE) with counterfactual validation.

3 METHODS

The framework instantiates the two Active-Inference processes, as summarized in Figure 1. *Online*, before the next result is known, the Worker constructs a local predictive model and the numerical module minimizes EFE to choose a policy that is useful for the task and informative about the relevant mechanism. *Offline*, after the trajectory is observed, the Builder proposes a revised local model and a numerical gate compares its VFE with that of the committed memory. Predictive contracts connect the two stages by ensuring that the epistemic value considered before execution and the evidence used after execution refer to the same precommitted predictions.

3.1 PRELIMINARIES: ACTIVE INFERENCE

We consider an agent acting under partial observability. At step t, it receives an observation o_t, chooses an action a_t, and seeks to satisfy a user goal g. The environment's state and dynamics are not directly available. Active Inference (Friston et al., 2017) describes the agent through an internal generative model $P(\tilde{o}, \tilde{z}, \pi, \mathcal{K})$, where $\tilde{o}$ is an observation sequence, $\tilde{z}$ is a sequence of latent states, π is a policy, and $\mathcal{K}$ denotes the learnable memory. A policy may depend on observations rather than prescribing a fixed action sequence. Given observed data, the **variational free energy** (VFE) is an upper bound on negative log evidence (Friston, 2010; Friston et al., 2017):

$$\mathcal{F} = \underbrace{-\mathbb{E}_Q[\ln P(\tilde{o}|\tilde{z}, \pi, \mathcal{K})]}_{\text{Prediction Error}} + \underbrace{D_{\mathrm{KL}}\left[Q(\tilde{z}, \pi, \mathcal{K}|\tilde{o}) \,\|\, P(\tilde{z}, \pi, \mathcal{K})\right]}_{\text{Complexity}}, \tag{1}$$

The prediction-error term measures how well the inferred latent variables and model explain observed data, whereas the complexity term penalizes unsupported deviation from the prior. Optimizing beliefs and model structure therefore trades explanatory fit against complexity.

VFE evaluates explanations of observations that have already occurred. Planning instead evaluates future observations under a candidate policy. Let ω be a possible future action–result trace, with $\tilde{z}_{>t}$ as its latent causes. A goal specifies preferences over traces through $P_{\mathrm{pref}}(\omega \mid g) = \exp(U_g(\omega))/Z_g$. Then the pragmatic–epistemic form of **Expected Free Energy (EFE)** is:

$$G(\pi) \doteq \underbrace{-\mathbb{E}_{Q(\omega|\pi)}[U_g(\omega)]}_{\text{Pragmatic term}} \underbrace{-\,\mathbb{E}_{Q(\omega|\pi)}[D_{\mathrm{KL}}(Q(\tilde{z}_{>t} \mid \omega, \pi) \,\|\, Q(\tilde{z}_{>t} \mid \pi))]}_{\text{Information gain}} \tag{2}$$

Minimizing EFE favors traces that satisfy the goal while revealing information about the represented latent cause. Thus, VFE evaluates an explanation after evidence is available, whereas EFE selects

which evidence-producing policy to execute beforehand. Sections 3.3 and 3.4 restrict these standard objectives to finite mechanism sets and fixed replay records, yielding quantities that can be computed directly from an LLM-produced contract rather than inferred from unobserved verbal reasoning.

3.2 Mechanism Memory and Predictive Contracts

Knowledge graph as memory. We write the adaptable memory as $\mathcal{K} = \mathcal{V}_E \cup \mathcal{V}_R \cup \mathcal{V}_S$. Entity records describe observable objects, locations, and state fields; Rule records describe local environment dynamics; and Skill records describe reusable procedures and their Rule dependencies. The backend follows a conventional hybrid graph-retrieval pipeline inspired by GraphRAG and LightRAG (Han et al., 2024; Guo et al., 2024): exact-name and dense retrieval are fused and reranked, after which typed dependencies are expanded.

An unresolved *Rule* does not assert a single transition when the environment is still ambiguous. Instead, it stores a finite latent mechanism variable H_R with candidate values $\mathcal{H}_R = \{h_0, \ldots, h_{J-1}\}$. Each value h specifies one concrete explanation of how actions produce observable effects within the shared scope of R. The belief of a mechanism at the current step is:

$$b_t(h) := Q(H_R = h \mid x_t, \mathcal{K}), \qquad b_t(h) \geq 0, \qquad \sum_{h \in \mathcal{H}_R} b_t(h) = 1, \tag{3}$$

where $x_t = (g, \tilde{o}_{\leq t}, \tilde{a}_{<t})$ is the information available before the next environment action. The belief is stored by the system and updated from real outcomes. When a mechanism set is first proposed, we let the Worker or Builder agents produce the subjective prior: the LLM will output positive scores $s(h)$, and normalize them as $b_{\text{init}}(h) = s(h)/\sum_{h' \in \mathcal{H}} s(h')$.

Predictive contracts. The core interface between semantic reasoning and numerical inference is a predictive contract C_t, constructed by the Worker agent at each environment step. Concretely, a contract contains (i) the referenced Rule version and its complete mechanism set, (ii) explicit result labels together with criteria for recognizing them, (iii) a small set of short policies that may branch only on visible outcomes, (iv) one common observable subgoal, and (v) the trace predicted for each policy under each mechanism. After the current observation has been incorporated, the numerical module also stores with the contract the belief snapshot used for scoring.

A simple example clarifies the distinction across these concepts. A record-persistence Rule may ask "whether editing a record requires an additional confirmation step". Its contains two mechanism h_0: "`Save` immediately persists the new value," and h_1: "`Save` only stages the edit and `Confirm` is required." After editing a particular record, the Rule is reused, but the Worker must create a new contract C_t that binds this record and its current state. It defines labels such as `new`, `old`, and `ack`; compares policies such as `ReadBack`, `Confirm`→`ReadBack`, and "read first, then confirm only if needed"; and records that a pre-confirmation read should return `new` under h_0 but `old` under h_1. Thus, a mechanism is a reusable candidate explanation and a contract is a prospective test of those explanations in one concrete decision.

3.3 Worker Agent: Minimizing Expected Free Energy

The Worker follows a ReAct-style interaction loop (Yao et al., 2023), but predictive contracts impose a fixed temporal interface. At every environment step t, the Worker receives the goal, the current raw observation o_t, retrieved semantic memory, and the pending contract C_{t-1}. Worker emits: (i) a review that applies the pre-written observation criteria in C_{t-1}, (ii) a task-relevant semantic state z_t, (iii) a new contract C_t for the next decision.

Grounding the previous return. Every contract defines, for each action that may be executed, a finite result alphabet and an observation criterion for each label. After a_{t-1} is executed, the Worker uses the criteria saved in C_{t-1} to map the raw return o_t to a grounded result label y_t. For example, a `read` action may produce $y_t \in \{\texttt{new}, \texttt{old}, \texttt{other}\}$. The label `other` denotes a definite but unanticipated value and therefore remains evidence against mechanisms that excluded it. The review cites the concrete observation that supports the label, preventing later belief revision from relying on an ungrounded verbal judgment.

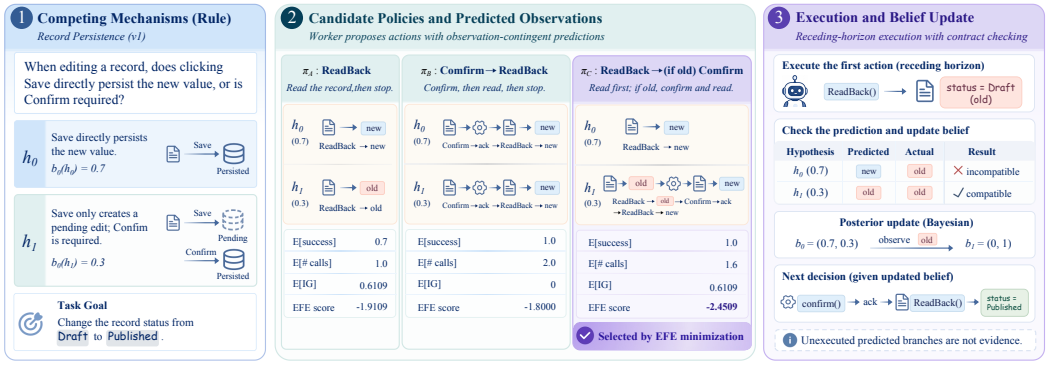

Figure 2: An illustration of one predictive-contract decision in FEP-Agent. Left: two competing mechanisms for the record persistence rule with initial beliefs. Middle: three candidate policies with observation-contingent predictions and their evaluation scores. Right: receding-horizon execution, contract checking, belief update, and the subsequent decision.

Constructing the next contract. Using the updated observable context x_t, the Worker identifies one local mechanism uncertainty that can affect the next decision and proposes a small policy set Π_t, where a policy may condition later actions on observed results. All policies are evaluated against the same observable subgoal. For each $\pi \in \Pi_t$ and $h \in \mathcal{H}_R$, the Worker predicts its trace $\hat{\omega}^{\pi}_{t,h} = ((a^1, y^1), \ldots, (a^m, y^m))$, where each y is a contract-defined result label. Combining these predictions with the current mechanism belief $b_t(h)$ yields the finite trace distribution

$$p_t(\omega \mid \pi) = \sum_{h \in \mathcal{H}_R} b_t(h)\, \mathbf{1}\{\omega = \hat{\omega}^{\pi}_{t,h}\}. \tag{4}$$

In our methods, we adopt deterministic contracts over a short planning horizon: for a fixed mechanism and policy, the contract specifies one observable action–result trace. This is a local modeling choice, not a claim that the full environment is deterministic. The stochastic extension, in which a mechanism specifies a distribution over traces, is given in the Appendix.

We define the preference utility $U_g(\omega) = \beta r_g(\omega) - \alpha n(\omega)$, with $\beta > 0$ and $\alpha \geq 0$, where $r_g(\omega) \in \{0, 1\}$ indicates whether the common subgoal is visibly satisfied, and let $n(\omega)$ be the number of environment calls in the trace.

Proposition 3.1 (Contracted EFE objective)**.** *Conditioned on x_t, suppose that H_R is the unresolved latent cause represented within the contract horizon and that each pair (h, π) produces the saved trace $\hat{\omega}^{\pi}_{t,h}$. Then the EFE in Equation* (2)*, restricted to this finite local model, is equivalent to*

$$\overline{G}_t(\pi) = -\beta\, \mathbb{E}_{p_t(\omega|\pi)}[r_g(\omega)] + \alpha\, \mathbb{E}_{p_t(\omega|\pi)}[n(\omega)] - I(H_R; \omega \mid \pi, x_t). \tag{5}$$

For a deterministic contract, $I(H_R; \omega \mid \pi, x_t) = \mathrm{H}[p_t(\omega \mid \pi)]$.

The proof is given in the Appendix. Its first two terms measure predicted task progress and interaction cost; the last term measures how much the complete observable trace is expected to distinguish the competing mechanisms. The Worker proposes semantic hypotheses and predictions, and then the numerical module computes the EFE score for each $\pi \in \Pi_t$ based on Eq. 5 and selects the minimum $\pi^*_t = \arg\min_{\pi \in \Pi_t} \overline{G}_t(\pi)$.

Only the first action a_t of π^*_t is executed. The remaining branches are counterfactual predictions used for action selection. When the next return is available, the saved contract supplies a first-action likelihood $p_{t,h}(y_{t+1} \mid a_t)$ for each mechanism. The posterior is

$$b_{t+1}(h) = \frac{b_t(h)\, p_{t,h}(y_{t+1} \mid a_t)}{\sum_{h'} b_t(h')\, p_{t,h'}(y_{t+1} \mid a_t)}. \tag{6}$$

For deterministic contracts, the likelihood is one when the saved prediction matches the grounded result label and zero otherwise. In the Appendix, we prove Eq. 6 is also the closed-form minimizer of a local VFE. Thus, the same prospective contract controls both the epistemic value used before action and the belief revision performed after action. The Rule version is then suspended from numerical planning and sent to the Builder for revision; a mechanism proposed after seeing that result cannot be credited with having predicted it.

3.4 BUILDER AGENT: VFE-GATED SEMANTIC MEMORY REVISION

After a task terminates, the Builder will propose local revisions to $\mathcal{K}$. It first localizes mechanism questions, retrieves the corresponding committed memory, and proposes a discrete patch. For example, it may add a candidate, replace an invalid mechanism, or compress a redundant Rule. A separate numerical gate determines whether the proposal becomes part of $\mathcal{K}$ by computing VFE.

Let $D = \{(x_i, a_i, y_i)\}_{i=1}^{N}$ be the records from the completed trajectory that are relevant to the proposed patch. Each x_i contains only information available before the executed action a_i, and y_i is the grounded result obtained from the immutable observation criterion. The committed memory $\mathcal{K}_{\text{old}}$ and every proposed memory $\mathcal{K}_{\text{new}}$ are scored on D based on the following proposition.

Proposition 3.2 (Local Builder VFE)**.** *For a fixed local memory proposal $\mathcal{K}$ and replay set D, let $q(h)$ be any distribution over the candidate mechanisms of $\mathcal{K}$, and let $L_{\text{tok}}(\mathcal{K})$ denote the token length of the local predictive closure. Define $L_D(h, \mathcal{K}) = \prod_{i=1}^{N} p_{\mathcal{K}}(y_i \mid x_i, a_i, h)$ and $Z_D(\mathcal{K}) = \sum_h b_{0,\mathcal{K}}(h) L_D(h, \mathcal{K})$. The local variational objective*

$$\begin{aligned} \mathcal{F}_B(q, \mathcal{K}; D) &= \underbrace{-\mathbb{E}_{q(h)}[\log L_D(h, \mathcal{K})]}_{\textit{Prediction Error}} + \underbrace{D_{\text{KL}}(q \| b_{0,\mathcal{K}}) + \lambda L_{\text{tok}}(\mathcal{K})}_{\textit{Complexity}}, \\ \mathcal{F}_B^*(\mathcal{K}; D) &:= \min_q \mathcal{F}_B(q, \mathcal{K}; D) = -\log Z_D(\mathcal{K}) + \lambda L_{\text{tok}}(\mathcal{K}) \end{aligned} \tag{7}$$

is minimized by the posterior proportional to $b_{0,\mathcal{K}}(h) L_D(h, \mathcal{K})$. Consequently, a proposed patch reduces the local VFE exactly when

$$\Delta \mathcal{F}_B := \mathcal{F}_B^*(\mathcal{K}_{\text{new}}; D) - \mathcal{F}_B^*(\mathcal{K}_{\text{old}}; D) < 0. \tag{8}$$

The proof is given in the Appendix. The first term of Eq. 8 is the improvement in log evidence, and the second is the additional storage and retrieval cost. Proposition 3.2 turns memory admission into a model-comparison test. This implies that a modification proposed by the builder will be rejected if it introduces mechanisms that are overly complex or redundant. Because $L_D(h, \mathcal{K})$ uses one mechanism h to explain the complete replay, a proposal cannot choose a different favorable mechanism for each record. The gate therefore accepts either a more coherent explanation whose evidence gain justifies its added text, or a shorter representation that preserves evidence; verbose restatements with unchanged predictions are rejected. If a proposal fails, the module returns the evidence change, token-cost change, and contradicted replay records so that the Builder can revise the same local patch against the same committed baseline.

Passing the gate establishes only that the proposed model is a better complexity-regularized fit to the current trajectory. A mechanism first formulated after observing D is therefore stored with provenance and remains `unresolved`. It becomes `resolved` only after a later contract prospectively distinguishes its candidates, the posterior exceeds a fixed validation threshold, and no scope conflict remains. Accepted Rule revisions may trigger corresponding Skill edits; a Skill depending on an unresolved Rule must retain an observation check or conditional branch. The complete execution order is given in the Algorithm 1 in the Appendix.

4 EXPERIMENTS

4.1 EXPERIMENT SETUP

Benchmarks. We evaluate PACT-Agent in three environments with different observation and action interfaces. **WebArena** (Zhou et al., 2024) contains realistic self-hosted websites; following ReasoningBank (Ouyang et al., 2025), we use 684 tasks from Shopping (187), Admin (182), GitLab (180), Reddit (106), and Multi-site (29), with accessibility trees as observations. **AndroidWorld** (Rawles et al., 2025) contains 116 tasks across 20 Android applications and exposes screenshots plus accessibility trees, with tap, long-press, swipe, and text actions. **ALFWorld** (Shridhar et al., 2021) contains 135 held-out text-based embodied tasks from six task families.

Baselines. Baselines. We include a non-evolving ReAct agent (Yao et al., 2023); trajectory and insight memories (ExpeL (Zhao et al., 2024), AdaPlanner (Sun et al., 2023), AWM (Wang et al., 2025b), AutoManual (Chen et al., 2024), ReasoningBank (Ouyang et al., 2025), G-Memory (Zhang

Table 1: Success rate (%) on the 684-task WebArena subset.

Methods	Reddit	Shopping	Admin	Gitlab	Multi	Overall
Gemini-2.5-flash						
ReAct Yao et al. (2023)	55.7	39.0	44.5	33.9	10.3	40.5
ExpeL Zhao et al. (2024)	61.3	43.9	46.2	36.1	10.3	43.7
AutoManual Chen et al. (2024)	65.8	47.1	48.9	38.9	13.8	46.9
ReasoningBank Ouyang et al. (2025)	67.0	49.7	51.1	40.6	13.8	48.8
MemRL Zhang et al. (2026)	64.2	46.0	48.4	38.3	10.3	45.9
JitRL Li et al. (2026)	66.0	48.1	50.0	38.9	10.3	47.4
PACT-Agent	**73.6**$_{\pm 1.1}$	**55.6**$_{\pm 0.8}$	**56.0**$_{\pm 0.6}$	**48.3**$_{\pm 0.9}$	**20.7**$_{\pm 1.9}$	**55.1**$_{\pm 0.6}$
Gemini-2.5-pro						
ReAct Yao et al. (2023)	71.7	45.5	51.1	35.0	6.9	46.7
ExpeL Zhao et al. (2024)	73.6	48.1	52.2	38.9	10.3	49.1
AutoManual Chen et al. (2024)	77.4	52.4	54.9	50.0	13.8	54.7
ReasoningBank Ouyang et al. (2025)	80.2	51.9	56.6	44.4	13.8	53.9
MemRL Zhang et al. (2026)	75.5	50.3	53.3	43.3	10.3	51.5
JitRL Li et al. (2026)	77.4	51.9	54.9	45.0	10.3	53.1
PACT-Agent	**83.0**$_{\pm 1.0}$	**61.0**$_{\pm 0.9}$	**61.5**$_{\pm 0.8}$	**67.2**$_{\pm 1.1}$	**27.6**$_{\pm 2.0}$	**64.7**$_{\pm 1.0}$

et al., 2025)); a value-augmented episodic memory (MemRL (Zhang et al., 2026)); and mobile systems reported under their original backbones (M3A (Rawles et al., 2025), MobileUse (Li et al., 2025), Gemini 2.5 Computer Use (DeepMind, 2025), Mobile-Agent-v3 (Ye et al., 2025), UI-TARS-2 (Wang et al., 2025a), UI-Venus-1.5 (Venus Team, Ant Group, 2026), AgentProg (Tian et al., 2025), and Agent S3 (Gonzalez-Pumariega et al., 2025)).

Implementation Details. The main backbone is Gemini-2.5 (Comanici et al., 2025) ALFWorld uses GPT-4o-mini for comparison with prior memory agents and Qwen3-32B (Yang et al., 2025) as an open-weight backbone. Unless stated otherwise, all benchmarks share $\beta = 2$, $\alpha = 0.1$, and $\lambda = 0.01$. We use a building–testing (B–T) protocol: memory is constructed on the designated building stream, frozen, and then evaluated. Success rates are the mean$\pm$standard deviation over three independent runs, each rebuilding $\mathcal{K}$ from scratch. Appendix E provides more details and prompts.

4.2 Main Results

WebArena. As shown in Table 1 shows that PACT-Agent reaches 55.1% with Gemini-2.5-flash and 64.7% with Gemini-2.5-pro. These scores exceed the strongest listed baseline by 6.3 and 10.0 points, respectively, and PACT-Agent leads on every domain. Notably, MemRL and JitRL also attach numerical values to memory, yet gain only 4.8–6.9% over ReAct: their values estimate *how useful* a cached experience has been, but cannot tell *which explanation* of an ambiguous page holds. The largest margin appears in GitLab, where a submission may take effect immediately, create a pending request, or silently fail without permission. The contract turns such ambiguity into a cheap discriminative check, selected precisely because its predicted trace differs across mechanisms.

AndroidWorld. Table 3 reports 81.0% success for PACT-Agent, exceeds AutoManual Reasoning-Bank under the same Gemini-2.5-pro backbone, and better than specialized GUI systems. Mobile tasks frequently expose delayed rendering, transient dialogs, and application-specific commit semantics; the Rule belief and suspension mechanism let the agent preserve these alternatives instead of collapsing immediately to one textual instruction.

ALFWorld. With GPT-4o-mini, PACT-Agent reaches 93.1%, compared with 88.8% for G-Memory and 85.9% for AutoManual (Table 2). With open-weight Qwen3-32B as the Worker, PACT-Agent reaches 96.3%, 3.0 points above G-Memory on the same model. The numerical module is particularly helpful for weaker backbones, since stating what each mechanism predicts is far easier for the LLM than weighing hypotheses by itself.

Appendix E further reports building and testing token costs and robustness tests.

4.3 Ablation Studies

Table 4 isolates the decisions introduced by PACT-Agent on WebArena with Gemini-2.5-flash.

Table 2: Success rate (%) of LLM agent methods on 135 test tasks of ALFWorld.

Methods	Put	Clean	Heat	Cool	Examine	Put two	ALL
GPT-4o-mini							
ReAct Yao et al. (2023)	79.2	25.8	39.1	40.9	55.6	35.3	45.2
ExpeL Zhao et al. (2024)	66.7	64.5	39.1	63.6	61.1	35.3	56.3
AdaPlanner Sun et al. (2023)	87.5	48.4	69.6	72.7	66.7	41.2	64.4
AutoManual Chen et al. (2024)	95.8	80.6	87.0	77.3	**100.0**	76.5	85.9
G-Memory Zhang et al. (2025)	95.8	90.3	86.9	86.4	83.3	88.2	88.8
PACT-Agent	**100.0**$_{\pm 0.0}$	**93.5**$_{\pm 1.9}$	**91.3**$_{\pm 2.5}$	**90.5**$_{\pm 2.7}$	88.9$_{\pm 3.2}$	**92.6**$_{\pm 3.2}$	**93.1**$_{\pm 1.1}$
Qwen3-32B							
ReAct Yao et al. (2023)	91.7	80.6	78.3	81.0	83.3	66.7	80.7
ExpeL Zhao et al. (2024)	91.7	83.9	82.6	85.7	88.9	72.2	84.4
AdaPlanner Sun et al. (2023)	87.5	83.9	82.6	85.7	77.8	77.8	83.0
AutoManual Chen et al. (2024)	95.8	93.5	91.3	90.5	**94.4**	83.3	91.9
G-Memory Zhang et al. (2025)	95.8	93.5	91.3	**95.2**	**94.4**	88.9	93.3
PACT-Agent	**100.0**$_{\pm 0.0}$	**96.8**$_{\pm 1.9}$	**95.7**$_{\pm 2.5}$	**95.2**$_{\pm 2.7}$	**94.4**$_{\pm 3.2}$	**94.4**$_{\pm 3.2}$	**96.3**$_{\pm 0.7}$

Table 3: Success rate (%) of different methods on AndroidWorld.

Methods	SR (%)
Domain-specific models	
MobileUse Li et al. (2025)	62.9
Agent S3 Gonzalez-Pumariega et al. (2025)	68.1
Mobile-Agent-v3 Ye et al. (2025)	73.3
UI-TARS-2 Wang et al. (2025a)	73.3
UI-Venus-1.5 Venus Team, Ant Group (2026)	77.6
AgentProg Tian et al. (2025)	78.0
Gemini-2.5-pro	
M3A Rawles et al. (2025)	59.7
ReAct Yao et al. (2023)	65.9
AutoManual Chen et al. (2024)	72.4
ReasoningBank Ouyang et al. (2025)	73.7
PACT-Agent	**81.0**$_{\pm 0.9}$

Table 4: Ablation study on WebArena (Gemini-2.5-flash). We systematically remove components to validate their contribution.

Methods	SR (%)	Δ
PACT-Agent (Full)	**55.1**$_{\pm 0.6}$	–
ReAct (no self-evolving)	40.5$_{\pm 0.7}$	-14.6
Prompt-level EFE & VFE	52.6$_{\pm 0.7}$	-2.5
Worker: Contracted EFE		
LLM-selected policy	53.8$_{\pm 0.6}$	-1.3
w/o Information Gain	51.4$_{\pm 0.5}$	-3.7
w/o Belief Update	52.2$_{\pm 0.6}$	-2.9
Builder: VFE-gated Revision		
LLM-judged admission	54.3$_{\pm 0.6}$	-0.8
w/o Complexity ($\lambda = 0$)	53.6$_{\pm 0.5}$	-1.5
w/o Prospective Validation	53.4$_{\pm 0.7}$	-1.7

Worker decisions. The most direct test keeps the whole architecture but replaces every numerical computation with its prompt-level description (verbal confidence for action selection, self-judged counterfactual validation for admission). This FEP-inspired variant drops by 2.5%, indicating that the gain comes from computing the objectives rather than describing them to the LLM. Within the Worker, removing the information gain causes the largest drop (-3.7%): contracts are still written, but the most goal-directed policy is always taken, so competing mechanisms remain undiscriminated. An LLM choosing among the same contracted policies (-1.3%) favors those that directly complete the subgoal over equally useful ones that also resolve the mechanism. Without belief update (-2.9%), an answered test keeps its full information gain, so it is repeated while contradicted mechanisms still steer later steps.

Builder decisions. Setting $\lambda = 0$ admits every patch that fits the current trajectory, inflating the memory from 71.8k to 109k tokens with over-specific Rules that mislead retrieval (-1.5%). LLM-judged admission (-0.8%) retains part of this control, but accepts 82% of the proposals versus 62% under the gate. Promoting post-hoc mechanisms directly to `resolved` (-1.7%) lets Skills drop their observation checks prematurely, so a wrongly fitted explanation is executed unconditionally. More ablations are given in the Appendix.

4.4 Analysis

We analyze the WebArena building stream with Gemini-2.5-flash. Web pages keep introducing local conventions, so the interesting question is not whether exploration vanishes, but how a bad retrospective fit is caught and repaired.

Non-monotonic EFE minimization. Fig. 3(a) evaluates a fixed 120-task probe, stratified across the five sites, every 19 building tasks. The selected contracted EFE falls from -0.91 to -1.58 nat, and

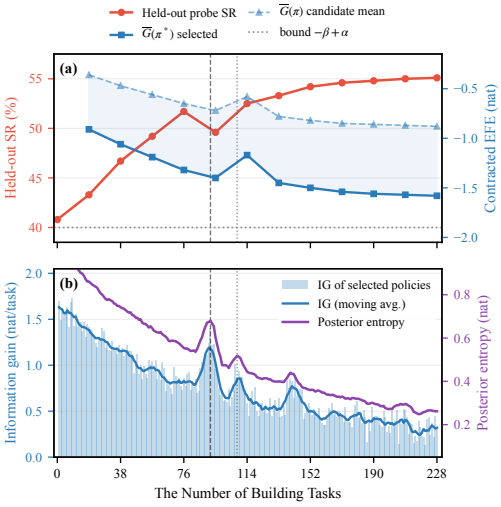

Figure 3: Worker-side dynamics on WebArena (Gemini-2.5-flash). (a) Probe success rate and the contracted EFE of the selected policy versus the mean of the other candidates. (b) Information gain of the selected policies and the posterior entropy of retrieved Rules. Dashed and dotted lines mark a mid-stream misfit and its later repair.

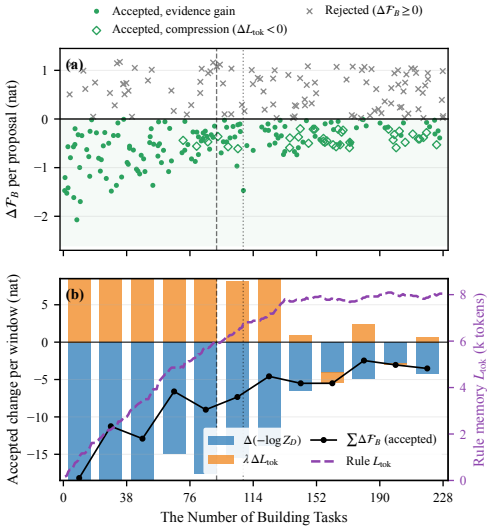

Figure 4: Builder-side dynamics on WebArena. (a) $\Delta\mathcal{F}_B$ of every proposal. (b) Accepted changes in 19-task windows, split into $\Delta(-\log Z_D)$ and $\lambda\Delta L_{\text{tok}}$, with the Rule memory size.

its margin over the other candidates widens from 0.55 to 0.70 nat. It does not approach the bound $-\beta+\alpha$: a web subgoal is rarely completed by one sure call. The probe success rate rises from 40.8% to 51.7% by the 76th task, then drops to 49.6% around the 95th task, when an Admin publishing Rule fitted on a single trajectory is retrieved on later CMS tasks. A later prospective contract suspends that Rule (dotted line); the probe recovers to 52.5% and ends at 55.1%. The dip is therefore a failure of retrospective fit, not of the EFE score itself. Fig. 3(b) makes the same point in the sufficient statistics. Early selected policies yield about 1.51 nat of information gain per task. Posterior entropy of retrieved Rules starts near 0.85 nat, because many Rules still carry two or three live candidates, and ends near 0.26 nat rather than zero: unseen Multi-site pages continue to reopen local uncertainty. The only sharp later spikes sit at the Admin misfit, its repair, and a cross-site batch around the 141st task.

VFE-gated revision under residual uncertainty. Fig. 4 shows how the gate treats that stream. Of 285 proposals, 104 (36.5%) are rejected: almost all add 50–120 tokens while barely changing $-\log Z_D$, typically a narrower restatement of a site-specific Rule. Accepted evidence gains total 114.2 nat before the misfit and 38.9 nat after the repair. Compressions become common only in the second half, once several Rules describe the same submit-and-read pattern. The Rule memory therefore grows quickly at first, then flattens around 8.0k tokens (71.8k for the whole memory). The accepted patch at the repair is large because the suspended Admin Rule could no longer explain the replay set; a costly revision is admitted only when the evidence term pays for it.

5 Conclusion

We presented PACT-Agent, which turns agent self-evolution into prospective mechanism testing. Predictive action contracts expose a finite local generative model: contracted EFE selects informative actions and updates the same mechanism beliefs, while a VFE gate admits only revisions whose replay evidence pays for their description cost. Results on three benchmarks support this coupled evidence pipeline. Extending contracts to interacting mechanisms and longer stochastic horizons remains future work.

6 AI USE STATEMENT

Generative AI tools were used to assist with literature search, language editing, code development, and discussion of paper writing. All AI-assisted outputs, including technical claims, implementations, and manuscript text, were independently reviewed and verified by the authors. The authors take full responsibility for the content of this work.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, and et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

Dilip Arumugam and Thomas L. Griffiths. Toward efficient exploration by large language model agents. *ArXiv*, abs/2504.20997, 2025.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

Deepro Choudhury, Sinead Williamson, Adam Goli'nski, Ning Miao, Freddie Bickford-Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. Bed-llm: Intelligent information gathering with llms and bayesian experimental design. *ArXiv*, abs/2508.21184, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, and et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *ArXiv*, abs/2507.06261, 2025.

Google DeepMind. Introducing the gemini 2.5 computer use model, Oct 07 2025.

Zafeirios Fountas, Noor Sajid, Pedro A.M. Mediano, and Karl Friston. Deep active inference agents using monte-carlo methods. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Karl J. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11: 127–138, 2010.

Karl J. Friston, Thomas H. B. FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. Active inference: A process theory. *Neural Computation*, 29:1–49, 2017.

Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

Gonzalo Gonzalez-Pumariega, Vincent Tu, Chih-Lun Lee, Jiachen Yang, Ang Li, and Xin Eric Wang. The unreasonable effectiveness of scaling agents for computer use. *ArXiv*, abs/2510.02250, 2025.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. *ArXiv*, abs/2410.05779, 2024.

Haoyu Han, Yu Wang, Harry Shomer, Kai Guo, Jiayuan Ding, Yongjia Lei, Mahantesh M. Halappanavar, Ryan A. Rossi, Subhabrata Mukherjee, Xianfeng Tang, Qianru He, Zhigang Hua, Bo Long, Tong Zhao, Neil Shah, Amin Javari, Yinglong Xia, and Jiliang Tang. Retrieval-augmented generation with graphs (graphrag). *ArXiv*, abs/2501.00309, 2024.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Geunwoo Kim, Pierre Baldi, and Stephen Marcus McAleer. Language models can solve computer tasks. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Ning Li, Xiangmou Qu, Jiamu Zhou, Jun Wang, Muning Wen, Kounianhua Du, Xingyu Lou, Qiuying Peng, Jun Wang, and Weinan Zhang. Mobileuse: A hierarchical reflection-driven GUI agent for autonomous mobile operation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

Yi-Bo Li, Zijie Lin, Ailin Deng, Xuan Zhang, Yu-Fei He, Shuo Ji, Tri Cao, and Bryan Hooi. Just-in-time reinforcement learning: Continual learning in llm agents without gradient updates. *ArXiv*, abs/2601.18510, 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. *ArXiv*, abs/2509.25140, 2025.

Thomas Parr, Giovanni Pezzulo, and Karl J. Friston. *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press, 03 2022. ISBN 9780262369978. doi: 10.7551/mitpress/12441.001.0001.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Ramesh Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *ArXiv*, abs/2408.07199, 2024.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, Maosong Sun, and Wei Liu. Communicative agents for software development. *ArXiv*, abs/2307.07924, 2023.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Paloma Sodhi, S.R.K Branavan, Yoav Artzi, and Ryan McDonald. Step: Stacked LLM policies for web actions. In *First Conference on Language Modeling*, 2024.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In *Neural Information Processing Systems (NeurIPS)*, 2023.

Shizuo Tian, Hao Wen, Yuxuan Chen, Jiacheng Liu, Shanhui Zhao, Guohong Liu, Junpeng Ren, Yunxin Liu, and Yuanchun Li. Agentprog: Empowering long-horizon gui agents with program-guided context management. *Proceedings of the 24th Annual International Conference on Mobile Systems, Applications and Services*, 2025.

Venus Team, Ant Group. UI-Venus-1.5 technical report. *ArXiv*, abs/2602.09082, 2026.

Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, 11: 95060–95078, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research (TMLR)*, 2024.

Haoming Wang, Haoyang Zou, Huatong Song, Jiazhan Feng, Junjie Fang, Junting Lu, Longxiang Liu, Qinyu Luo, Shihao Liang, Shijue Huang, Wanjun Zhong, Yining Ye, Yujia Qin, Yuwen Xiong, Yuxin Song, Zhiyong Wu, Bo Li, Chen Dun, Chong Liu, Fuxing Leng, Han rui Wang, Hao Yu, Haobin Chen, and et al. Ui-tars-2 technical report: Advancing gui agent with multi-turn reinforcement learning. *ArXiv*, abs/2509.02544, 2025a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning (ICML)*, 2025b.

Xiao-Jun Wu, Ce-Hao Yang, Honghao Liu, Xueyuan Lin, Wenjie Zhang, Zhichao Shi, Xuhui Jiang, Chengjin Xu, Jia Li, and Jian Guo. Bayesian-agent: Posterior-guided skill evolution for llm agent harnesses. *ArXiv*, abs/2606.08348, 2026.

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, Shijie Yang, Shuai Bai, Shusheng Yang, Sibo Xu, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, 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*, abs/2505.09388, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Jiabo Ye, Xi Zhang, Haiyang Xu, Haowei Liu, Junyang Wang, Zhaoqing Zhu, Ziwei Zheng, Feiyu Gao, Junjie Cao, Zhengxi Lu, Jitong Liao, Qi Zheng, Fei Huang, Jingren Zhou, and Ming Yan. Mobile-agent-v3: Fundamental agents for gui automation. *ArXiv*, abs/2508.15144, 2025.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms. In *ACL (Findings)*, 2024.

Gui-Min Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *ArXiv*, abs/2506.07398, 2025. URL https://api.semanticscholar.org/CorpusID:279250852.

Shengtao Zhang, Jia-Qian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yu-Tao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *ArXiv*, abs/2601.03192, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Y. Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, Y. Qiao, Zhaoxiang Zhang, and Jifeng Dai. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *ArXiv*, abs/2305.17144, 2023.

A Notation and Complete Procedure

A.1 Core Notation

Symbol	Meaning
g	User task or current verified subgoal.
η	Parameters and structural variables of the internal generative model.
$\mathcal{K}$	Current structured semantic memory, the editable part of η.
z_t	Textual semantic state inferred by the Worker.
x_t	Observable decision context before the next environment action.
$H; h \in \mathcal{H}$	Local latent mechanism variable and one candidate mechanism value.
$b_t(h)$	Posterior mass $Q_\eta(H = h \mid x_t)$ before action a_t.
C_t	Predictive contract persisted before action a_t.
$\pi \in \Pi_t$	Observation-contingent short policy and the current finite policy set.
$\omega \in \Omega_t$	The visible action-result trace realization and its finite support.
$p_{t,h}(\omega \mid \pi)$	Contract prediction for trace ω conditional on mechanism h.
$p_t(\omega \mid \pi)$	Trace prediction after marginalizing H with b_t.
D	The current trajectory records used for one Builder comparison.

A.2 End-to-End Procedure

Algorithm 1 makes explicit the temporal order of the implementation. The Worker reviews the pending contract and proposes the next one in a single call. The numerical module then updates the reviewed Rule belief, scores the new contract with that updated belief, and executes only the selected first action. Structural memory revision is deferred until the trajectory is complete.

Algorithm 1 PACT-Agent

Require: Goal g, environment, committed memory $\mathcal{K}$, action budget B
Observe o_1; set the pending contract to $\varnothing$
for $t = 1, \ldots, B$ **do**
  Retrieve relevant memory; append the pending contract and its raw return
  $(\textit{review}_t, z_t, C_t) \leftarrow \textsc{Worker}(g, o_{\le t}, \mathcal{K}, C_{t-1})$
  if C_{t-1} yields an observable, unprocessed result **then**
    Update its Rule belief by Eq. 6; suspend the Rule if no mechanism supports the result
  end if
  if the task has terminated **then**
    break
  end if
  Validate C_t and attach the updated belief snapshot
  Score valid policies by Eq. 5; otherwise use an admissible ordinary action
  Persist C_t and execute the selected policy's first action
end for
Construct fixed replay records D from the completed trajectory
for each local patch proposed by the Builder **do**
  Compute Eq. 7 and 8; commit iff $\Delta\mathcal{F}_B < 0$
  If rejected, return the score decomposition and re-propose on the same D
end for
return the single committed memory $\mathcal{K}$

B Derivations of Operational Free-Energy Objectives

B.1 Preliminary: Derivations of VFE and EFE

Derivations of VFE. Start from the evidence:

$$-\ln P(\tilde{o}) = -\ln \sum_{\tilde{z},\pi,\eta} P(\tilde{o}, \tilde{z}, \pi, \eta) \tag{9}$$

$$= -\ln \mathbb{E}_Q \left[\frac{P(\tilde{o}, \tilde{z}, \pi, \eta)}{Q(\tilde{z}, \pi, \eta \mid \tilde{o})} \right], \tag{10}$$

By Jensen's inequality $-\ln \mathbb{E}[X] \le \mathbb{E}[-\ln X]$, we obtain:

$$-\ln P(\tilde{o}) \le \mathbb{E}_Q[\ln Q(\tilde{z}, \pi, \eta \mid \tilde{o}) - \ln P(\tilde{o}, \tilde{z}, \pi, \eta)] := F[\tilde{o}, \tilde{z}, \pi, \eta] \tag{11}$$

Defining the right-hand side as the Variational Free Energy (VFE), this is the standard derivation that VFE is the upper bound of surprise/negative log evidence. It can also be written as:

$$F[\tilde{o}, \tilde{z}, \pi, \eta] = -\mathbb{E}_Q[\ln P(\tilde{o}|\tilde{z}, \pi, \eta)] + \mathbb{E}_Q[\ln Q(\tilde{z}, \pi, \eta \mid \tilde{o}) - \ln P(\tilde{z}, \pi, \eta)] \tag{12}$$

$$= \underbrace{-\mathbb{E}_Q[\ln P(\tilde{o}|\tilde{z}, \pi, \eta)]}_{\text{Prediction Error}} + \underbrace{D_{\mathrm{KL}}\left[Q(\tilde{z}, \pi, \eta|\tilde{o}) \,\|\, P(\tilde{z}, \pi, \eta)\right]}_{\text{Complexity}} \tag{13}$$

Derivations of EFE. Let $\tilde{z}_{>t}$ denote the latent causes of the future trace ω, and let $Q(\omega, \tilde{z}_{>t} \mid \pi) = Q(\tilde{z}_{>t} \mid \pi)P(\omega \mid \tilde{z}_{>t}, \pi)$ be the predictive joint under policy π (the model parameter η and the history z_t are omitted from conditions for simplicity). The expected free energy associated with the future trace is conventionally written as:

$$G_t(\pi) := \mathbb{E}_{Q(\omega, \tilde{z}_{>t}|\pi)}[\log Q(\tilde{z}_{>t} \mid \pi) - \log P(\omega, \tilde{z}_{>t} \mid g, \pi)]\,. \tag{14}$$

Factor the preferred generative model as $P(\omega, \tilde{z}_{>t} \mid g, \pi) = P_{\mathrm{pref}}(\omega \mid g)P(\tilde{z}_{>t} \mid \omega, g, \pi)$. The standard posterior-matching approximation identifies the intractable conditional $P(\tilde{z}_{>t} \mid \omega, g, \pi)$ with the variational posterior $Q(\tilde{z}_{>t} \mid \omega, \pi)$. Substituting this factorization into Eq. 14 gives

$$G_t(\pi) \approx \mathbb{E}_Q[\log Q(\tilde{z}_{>t} \mid \pi) - \log Q(\tilde{z}_{>t} \mid \omega, \pi) - \log P_{\mathrm{pref}}(\omega \mid g)] \tag{15}$$

$$= -\mathbb{E}_{Q(\omega|\pi)}[D_{\mathrm{KL}}(Q(\tilde{z}_{>t} \mid \omega, \pi) \| Q(\tilde{z}_{>t} \mid \pi))] - \mathbb{E}_{Q(\omega|\pi)}[\log P_{\mathrm{pref}}(\omega \mid g)]\,. \tag{16}$$

The expected KL in the first term is the mutual information $I_Q(\tilde{z}_{>t}; \omega \mid \pi)$. For the Boltzmann preference model $P_{\mathrm{pref}}(\omega \mid g) = \exp(U_g(\omega))/Z_g$,

$$-\mathbb{E}_Q[\log P_{\mathrm{pref}}(\omega \mid g)] = -\mathbb{E}_Q[U_g(\omega)] + \log Z_g. \tag{17}$$

Because $\log Z_g$ is shared by all policies evaluated under the same preference model, it can be omitted for policy selection. Equation 16 therefore yields Eq. 2.

B.2 Derivation of the Contracted EFE Objective

For a contract C_t with $h \in \mathcal{H}_R$, define

$$q_t(h, \omega \mid \pi) = b_t(h)p_{t,h}(\omega \mid \pi), \qquad p_t(\omega \mid \pi) = \sum_h q_t(h, \omega \mid \pi). \tag{18}$$

Whenever $p_t(\omega \mid \pi) > 0$, the hypothetical posterior after observing the complete trace is

$$q_t(h \mid \omega, \pi) = \frac{b_t(h)p_{t,h}(\omega \mid \pi)}{p_t(\omega \mid \pi)}. \tag{19}$$

Restricting the latent variable $\tilde{z}_{>t}$ in Eq.2 to the explicitly represented mechanism H_R gives the epistemic term

$$\mathbb{E}_{p_t(\omega|\pi)}[D_{\mathrm{KL}}(q_t(H_R \mid \omega, \pi) \| b_t(H_R))] = \sum_{h,\omega} q_t(h, \omega \mid \pi) \log \frac{q_t(h \mid \omega, \pi)}{b_t(h)} \tag{20}$$

$$= I_{q_t}(H_R; \omega \mid \pi, x_t). \tag{21}$$

Substituting $U_g(\omega) = \beta r_g(\omega) - \alpha n(\omega)$ into Eq.2, and using Eq.21, yields

$$G_t(\pi) \doteq -\beta \mathbb{E}_{p_t}[r_g(\omega)] + \alpha \mathbb{E}_{p_t}[n(\omega)] - I_{q_t}(H_R; \omega \mid \pi, x_t), \tag{22}$$

which proves Proposition 3.1.

The same information term also has the entropy decomposition

$$I_{q_t}(H_R; \omega \mid \pi, x_t) = \mathrm{H}[p_t(\omega \mid \pi)] - \sum_h b_t(h)\, \mathrm{H}[p_{t,h}(\omega \mid \pi)]. \tag{23}$$

This form is used directly for stochastic contracts in Sec.C.

Information Gain for Deterministic Contracts Under Eq.4, every conditional $p_{t,h}(\cdot \mid \pi)$ is a point mass, so the second term of Eq.23 is zero. Hence

$$I_{q_t}(H_R; \omega \mid \pi, x_t) = \mathrm{H}[p_t(\cdot \mid \pi)]. \tag{24}$$

The same result can be seen by grouping mechanisms that predict the same trace. Let $\mathcal{C}_\omega^\pi = \{h : \hat{\omega}_{t,h}^\pi = \omega\}$, then $p_t(\omega \mid \pi) = \sum_{h \in \mathcal{C}_\omega^\pi} b_t(h)$. If ω were observed, the hypothetical posterior would be

$$b_t^{\pi,\omega}(h) = \frac{b_t(h)\mathbf{1}\{h \in \mathcal{C}_\omega^\pi\}}{p_t(\omega \mid \pi)}. \tag{25}$$

The expected entropy reduction is

$$\mathrm{IG}_t(\pi) = \mathrm{H}[b_t] - \sum_\omega p_t(\omega \mid \pi)\, \mathrm{H}[b_t^{\pi,\omega}] \tag{26}$$

$$= -\sum_h b_t(h) \log b_t(h) + \sum_\omega \sum_{h \in \mathcal{C}_\omega^\pi} b_t(h) \log \frac{b_t(h)}{p_t(\omega \mid \pi)} \tag{27}$$

$$= -\sum_\omega p_t(\omega \mid \pi) \log p_t(\omega \mid \pi) = \mathrm{H}[p_t(\cdot \mid \pi)]. \tag{28}$$

The entropy is taken over complete observable traces rather than by summing independent per-step entropies. Once the first result has separated the mechanisms, repeating the same deterministic check does not provide the same information again.

B.3 Online Bayes Update as VFE Minimization

Let $\ell_t(h) = p_{t,h}(y_{t+1} \mid a_t)$ be the likelihood read from the saved contract, and let $q(h)$ be any candidate posterior. Define the local VFE

$$\mathcal{F}_t^{\mathrm{online}}(q) = -\mathbb{E}_{q(h)}[\log \ell_t(h)] + D_{\mathrm{KL}}(q \| b_t). \tag{29}$$

Let $Z_t = \sum_h b_t(h)\ell_t(h) > 0$ and define $b_{t+1}(h) = b_t(h)\ell_t(h)/Z_t$. Expanding Eq. 29 gives

$$\mathcal{F}_t^{\mathrm{online}}(q) = \sum_h q(h) \log \frac{q(h)}{b_t(h)\ell_t(h)} \tag{30}$$

$$= \sum_h q(h) \log \frac{q(h)}{b_{t+1}(h)} - \log Z_t \tag{31}$$

$$= D_{\mathrm{KL}}(q \| b_{t+1}) - \log Z_t. \tag{32}$$

The minimum is attained at $q = b_{t+1}$, proving Eq. 6. In the deterministic case, a contradicted mechanism has $\ell_t(h) = 0$ and receives zero posterior mass. If $Z_t = 0$, the observed result lies outside the support of the current Rule, so the posterior is undefined within that model and structural revision is required.

B.4 Proof of the Local Builder VFE Proposition

For a fixed memory version $\mathcal{K}$, the replay likelihood of mechanism h is

$$L_D(h, \mathcal{K}) = \prod_{i=1}^N p_{\mathcal{K}}(y_i \mid x_i, a_i, h). \tag{33}$$

This product is a chain-rule factorization, not an i.i.d. assumption: each x_i contains the observable history available before a_i. The local generative model for the replay is

$$P(D, h \mid \mathcal{K}) = b_{0,\mathcal{K}}(h) L_D(h, \mathcal{K}), \qquad P(D \mid \mathcal{K}) = Z_D(\mathcal{K}) = \sum_h b_{0,\mathcal{K}}(h) L_D(h, \mathcal{K}). \tag{34}$$

Place an exponential prior on the predictive text required by the local memory,

$$P(\mathcal{K}) \propto \exp[-\lambda L_{\mathrm{tok}}(\mathcal{K})]. \tag{35}$$

For a variational distribution $q(h)$, the negative expected log joint plus entropy is, up to a proposal-independent normalizer,

$$\mathcal{F}_B(q, \mathcal{K}; D) = \mathbb{E}_q[\log q(h) - \log b_{0,\mathcal{K}}(h) - \log L_D(h, \mathcal{K})] - \log P(\mathcal{K}) \tag{36}$$

$$= -\mathbb{E}_q[\log L_D(h, \mathcal{K})] + D_{\mathrm{KL}}(q \| b_{0,\mathcal{K}}) + \lambda L_{\mathrm{tok}}(\mathcal{K}), \tag{37}$$

which is the first line of Eq. 7. Define the replay posterior

$$b_D(h \mid \mathcal{K}) = \frac{b_{0,\mathcal{K}}(h) L_D(h, \mathcal{K})}{Z_D(\mathcal{K})}. \tag{38}$$

Substitution yields

$$\mathcal{F}_B(q, \mathcal{K}; D) = D_{\mathrm{KL}}(q \| b_D(\cdot \mid \mathcal{K})) - \log Z_D(\mathcal{K}) + \lambda L_{\mathrm{tok}}(\mathcal{K}). \tag{39}$$

The KL term is minimized at $q = b_D$, giving the second line of Eq. 7. Subtracting the minimized scores for $\mathcal{K}_{\mathrm{new}}$ and $\mathcal{K}_{\mathrm{old}}$ gives Eq. 8.

C Stochastic Mechanisms

A stochastic contract allows multiple observable outcomes even after the local mechanism is fixed. The Worker–Builder decomposition is unchanged, but a mechanism now specifies a normalized conditional law rather than a single label. This section states the mathematical and algorithmic changes; a small service example is included only to illustrate the resulting quantities.

C.1 Stochastic Predictive Contracts

For each mechanism h, observable history $\rho_{\ell-1} = ((a_1, y_1), \ldots, (a_{\ell-1}, y_{\ell-1}))$, and admissible next action a_ℓ, the contract stores

$$p_{t,h}(y_\ell \mid \rho_{\ell-1}, a_\ell, x_t), \quad \text{where } p_{t,h}(y_\ell \mid \rho_{\ell-1}, a_\ell, x_t) \geq 0, \quad \sum_{y_\ell \in \mathcal{Y}(a_\ell)} p_{t,h}(y_\ell \mid \rho_{\ell-1}, a_\ell, x_t) = 1. \tag{40}$$

The result set $\mathcal{Y}(a_\ell)$ and its observation criteria are fixed before execution. The probabilities are part of the Rule version or its context-specific instantiation; they are not estimated from the frequency with which an LLM samples different textual answers.

A short policy maps each visible history to the next action, $a_\ell = \pi(\rho_{\ell-1})$, and terminates according to its stated stopping condition. For a complete trace $\omega = ((a_1, y_1), \ldots, (a_m, y_m))$, the mechanism-conditioned path probability is

$$p_{t,h}(\omega \mid \pi) = \prod_{\ell=1}^{m} p_{t,h}(y_\ell \mid \omega_{<\ell}, a_\ell, x_t), \tag{41}$$

where $\omega_{<\ell}$ is the visible prefix before step ℓ. No conditional-independence assumption is required: history-dependent effects are represented through the conditioning prefix. The mixture prediction remains

$$p_t(\omega \mid \pi) = \sum_h b_t(h) p_{t,h}(\omega \mid \pi). \tag{42}$$

For the short horizons used by PACT-Agent, the numerical module constructs the finite outcome tree exactly, propagates each mechanism's branch probabilities, and merges identical complete traces before scoring.

C.2 Expected Information Gain and Policy Scoring

For any stochastic contract, the posterior that would follow from observing a complete trace is

$$b_t^{\pi,\omega}(h) = \frac{b_t(h) p_{t,h}(\omega \mid \pi)}{p_t(\omega \mid \pi)}. \tag{43}$$

The expected mechanism information gain can be computed equivalently as

$$\mathrm{IG}_t(\pi) = \sum_{\omega} p_t(\omega \mid \pi) D_{\mathrm{KL}}(b_t^{\pi,\omega} \| b_t) \tag{44}$$

$$= \mathrm{H}[p_t(\cdot \mid \pi)] - \sum_{h} b_t(h)\, \mathrm{H}[p_{t,h}(\cdot \mid \pi)]. \tag{45}$$

The second entropy removes outcome variability that would remain even if the mechanism were known. Consequently, an action is epistemically valuable only when different mechanisms assign different distributions to its possible traces; high outcome entropy by itself is not information gain. The operational objective in Eq. 5 is otherwise unchanged. The deterministic model is the special case in which every $p_{t,h}(\cdot \mid \pi)$ is a point mass and the conditional-entropy term vanishes.

C.3 Online Belief Revision

After executing a_t and grounding the result as y_{t+1}, the mechanism posterior is

$$b_{t+1}(h) = \frac{b_t(h) p_{t,h}(y_{t+1} \mid a_t, x_t)}{\sum_{h'} b_t(h') p_{t,h'}(y_{t+1} \mid a_t, x_t)}. \tag{46}$$

Unlike deterministic elimination, a mechanism remains possible after an unlikely result whenever it assigned that result positive probability. The same online VFE derivation in Eq. B.3 applies with a non-binary likelihood.

C.4 Builder Revision under Stochastic Likelihoods

The Builder objective does not change. For each replay record, the likelihood is the probability assigned to the actual result under its pre-action context, and

$$L_D(h, \mathcal{K}) = \prod_{i=1}^{N} p_{\mathcal{K}}(y_i \mid x_i, a_i, h). \tag{47}$$

The evidence $Z_D(\mathcal{K})$, posterior q^*, and acceptance condition are then computed by Eq. 7 and 8. A patch may change a conditional probability table, introduce a history-dependent state transition, or add a new mechanism; any such change creates a new Rule version. If a mechanism contains uncertain continuous parameters, the finite implementation either represents a small set of parameterizations as separate candidates or augments the latent posterior to include those parameters. The experiments considered here use explicit finite candidates, so probability parameters are fixed within a Rule version and revised only through the Builder gate.

C.5 Algorithmic Requirements

A stochastic contract must satisfy four additional requirements. First, every mechanism supplies a normalized distribution over mutually exclusive grounded labels for each reachable action and history. Second, the policy tree branches only on visible results; it cannot condition on the hidden mechanism. Third, path probabilities are propagated through the complete finite tree before expected utility and information gain are computed. Fourth, the saved probability model is reused for online updating and Builder replay; the LLM may not rewrite probabilities after observing the result. These checks preserve the same prospective-prediction discipline as the deterministic contract.

C.6 Illustrative Service Example

Suppose a read-only `Fetch` call returns `ok`, `timeout`, or `other`. Two mechanisms have current belief $b_0(h_S) = 0.6$ and $b_0(h_B) = 0.4$ and predict

Mechanism	p(`ok`)	p(`timeout`)	p(`other`)
h_S: normal service	0.90	0.08	0.02
h_B: busy service	0.30	0.68	0.02

The mixture prediction is $(0.66, 0.32, 0.02)$. Its entropy is approximately 0.7171 nat, while the belief-weighted conditional entropy is approximately 0.5057 nat, so Eq. 45 gives $\mathrm{IG}_0(\texttt{Fetch}) \approx 0.2114$ nat. If the call returns `timeout`, Eq. 46 yields

$$b_1(h_S) = \frac{0.6 \times 0.08}{0.32} = 0.15, \qquad b_1(h_B) = \frac{0.4 \times 0.68}{0.32} = 0.85. \tag{48}$$

The normal-service mechanism is down-weighted rather than eliminated because it assigned timeout a nonzero probability. The example illustrates the distinction between uncertainty over mechanisms and irreducible randomness within a known mechanism; the equations above define the general procedure independently of this particular service.

D Method Details

D.1 Structure of Semantic Memory

The storage layer follows a graph-backed memory design Han et al. (2024); Guo et al. (2024). Each record contains natural-language text, a stable identifier, an embedding, and typed dependency edges. We retain three record classes because they separate what the Worker must ground, predict, and execute:

- **Entities ($\mathcal{V}_E$):** observable objects, locations, fields, and state interpretations needed to bind a raw return to a contract label;
- **Rules ($\mathcal{V}_R$):** scoped environment mechanisms, candidate hypotheses, beliefs, versions, and evidence provenance;
- **Skills ($\mathcal{V}_S$):** reusable procedures whose conditional steps cite the Rules on which they depend.

We use directed semantic edges to preserve how operational knowledge composes. Every edge stores its type, direction, provenance, and active version range. The graph used for community detection is an undirected weighted projection of these active edges; edge-type weights are fixed for an experiment and give direct operational dependencies more influence than incidental co-occurrence.

To preserve higher-level structure as memory grows, we construct a weighted undirected projection of the active record graph and recursively apply Leiden community detection (Han et al., 2024). This produces partitions $\mathcal{C}^{(0)}, \ldots, \mathcal{C}^{(L)}$, from fine leaf communities to progressively coarser parents. Each active record has one primary assignment at every available level, while boundary links allow neighboring reports to reference records that connect their scopes. Each community c has a report $\sigma(c)$ that summarizes its scope, principal Entities, established Rules, reusable Skills, unresolved mechanism questions, cross-community dependencies, and source record IDs. Leaf reports are generated from member records and their internal and boundary edges; a parent report is generated from its child reports and inter-child links. The hierarchy therefore offers a concise route from broad task context to the exact records needed for action, following the local-to-global organization of graph-based RAG (Han et al., 2024).

D.2 Rule Schema

A practical Rule record can be serialized as follows. The exact storage backend is not part of the method, but the version, hypotheses, numerical belief, and evidence provenance must remain distinct fields.

Listing 1: Illustrative Rule schema.

```
rule_id: record_persistence
version: 1
name: Record Persistence Mechanism
base_entity: "[[Record Edit Page]]"
scope:
  - The target record and edit permission have been verified.
  - ReadBack reads the stored field, not the unsaved input value.
  - ReadBack does not discard a pending edit.
status: unresolved                # unresolved | resolved
needs_revision: false

description: >
```

```
  It is not yet known whether Save directly persists the edited value
  or whether a later Confirm action is required.

hypotheses:
  h0: Save directly persists the edited value.
  h1: Save creates a pending edit; Confirm is required.

initial_belief: {h0: 0.7, h1: 0.3}
belief:         {h0: 0.7, h1: 0.3}
default_hypothesis: null

evidence:
  - event_id: task08/action12
    kind: online_test            # online_test | retrospective_fit
    contract_ref: C_0

contract_refs: [C_0]
related_nodes:
  - "[[Record Edit Page]]"
  - "[[Confirm Action]]"
```

`initial_belief` records the proposal prior for the version. `belief` is the only source of current mechanism weights. Natural-language terms such as "likely" are never parsed into probabilities. Adding a mechanism, changing its causal meaning, changing a stochastic probability table, or changing its scope creates a new version.

D.3 Predictive Contract Schema

Listing 2: Illustrative deterministic predictive contract.

```
contract_id: C_0
rule_ref: record_persistence/v1
mechanism_test: true
belief_snapshot: {h0: 0.7, h1: 0.3}

context:
  object: record_A
  original_value: Draft
  requested_value: Published
  completed_actions: [Save]
  not_yet_executed: [Confirm]

applicability:
  - Target record and edit permission are verified.
  - ReadBack returns the stored field value.
  - Confirm is authorized.

outcome_criteria:
  ReadBack:
    new:   "Target status is explicitly Published."
    old:   "Target status is explicitly Draft."
    other: "Target status is visible but has another value."
  Confirm:
    ack:   "The tool explicitly reports that Confirm completed."
    other: "The tool explicitly reports another result."

subgoal:
  criterion: "Read the target record and explicitly observe Published."

first_action_predictions:
  ReadBack: {h0: new, h1: old}
  Confirm:  {h0: ack, h1: ack}

candidate_plans:
  piA: "ReadBack, then stop."
  piB: "Confirm, ReadBack, then stop."
  piC: "ReadBack; if old, Confirm and ReadBack again."

predicted_traces:
  piA:
    h0: {steps: [[ReadBack, new]], subgoal_satisfied: true}
    h1: {steps: [[ReadBack, old]], subgoal_satisfied: false}
  piB:
    h0: {steps: [[Confirm, ack], [ReadBack, new]], subgoal_satisfied: true}
    h1: {steps: [[Confirm, ack], [ReadBack, new]], subgoal_satisfied: true}
  piC:
    h0: {steps: [[ReadBack, new]], subgoal_satisfied: true}
```

```
    h1: {steps: [[ReadBack, old], [Confirm, ack],
                 [ReadBack, new]], subgoal_satisfied: true}
```

The contract may retain natural-language explanations, but the numerical module reads only the version reference, hypothesis IDs, belief snapshot, result labels, action IDs, trace grouping, Boolean subgoal flags, and step counts. The contract is saved before the environment call. Its content is immutable after execution.

D.4 Retrieval and Context Assembly

The semantic-memory backend uses a hybrid retrieval pipeline. Community summaries provide a coarse map. A lightweight Retriever reads the goal, current observation, recent history, and referenced node names, then issues semantic or exact-name queries. Retrieved nodes are expanded through wiki references and Skill dependencies. A Filter reranks the expanded set before it is inserted into the Worker context.

Our reference configuration retains up to five nodes of each type after graph expansion and before final filtering. Query and node descriptions are embedded with a fixed embedding model. The exact embedding model and similarity threshold are implementation parameters rather than elements of the FEP objective. The following constraints are method-critical:

- A selected Rule is returned with its complete hypothesis set and weights.
- A Skill cannot be returned without any Rule whose state-dependent condition is necessary to interpret that Skill.
- Pending contracts are inserted directly and cannot be removed by retrieval or filtering.
- Historical Rule versions are hidden from normal planning unless they are explicitly requested for audit or revision.
- Retrieval relevance does not update mechanism belief.

D.5 Text Complexity and Versioning

Token complexity is measured with a fixed tokenizer. The counted closure includes all new or modified text that is necessary to make the local prediction: Rule description, scope, hypothesis definitions, stochastic tables, and newly required Entity definitions. Moving the same mechanism text into another referenced node does not avoid the cost. Raw observations, immutable contracts, and audit logs are not counted as predictive model text because they are evidence rather than reusable model parameters.

The accepted patch is stored as a new Rule version. The current version points to the accepted belief and status. Prior versions and their contracts remain available for audit, but only the current committed version is exposed to ordinary retrieval. This avoids carrying old and new knowledge simultaneously through future Worker trajectories.

D.6 Default Limits and Tie Breaking

The reference implementation uses two or three mechanism hypotheses, at most three candidate short policies, and a maximum predicted horizon of three environment calls. These limits control context length and combinatorial growth; they are not claimed to be optimal. If policies have equal $\overline{G}_t$, the solver chooses the policy with fewer expected environment calls, followed by a stable policy ID. If a Rule has one positive-weight hypothesis, its mechanism information gain is zero, but it may still guide task completion through its predicted trace.

E Experiment Details

E.1 Implementation Details

We separate structural memory construction from final evaluation. For each benchmark, the Worker first interacts with a designated building split while the Builder revises $\mathcal{K}$. After building, Entity, Rule,

Table 5: Build- and test-time tokens (k) on WebArena with Gemini-2.5-pro, averaged over 684 tasks.

Methods	Build-time Tokens(k)	Test-time Tokens(k)	Success Rate(%)
ReAct	-	155.8	46.7
AutoManual	196.3	174.4	54.7
ReasoningBank	–	188.6	53.9
PACT-Agent	231.4	178.2	64.7

and Skill text is frozen; mechanism beliefs may still be updated by later contracts. On **WebArena**, the building split contains 228 tasks, stratified across the five sites (52 Shopping, 52 Admin, 52 GitLab, 47 Reddit, and 25 Multi-site) and shared with the re-runs of AutoManual, ExpeL, and MemRL. A separate 120-task probe (32/32/32/20/4 from the leftover tasks of the same sites) is used only for the learning-curve plots in Sec. 4. For **AndroidWorld**, seed 1 instantiates one building task from each of the 116 templates, whereas the official seed-0 instantiations form the test set. For **ALFWorld**, we follow AutoManual by drawing six training tasks from each of the six task categories (36 in total) and evaluate on all 135 tasks in `eval_out_of_distribution`.

The Builder uses Gemini-2.5-pro in every experiment. The Worker backbone is Gemini-2.5-pro, Gemini-2.5-flash, GPT-4o-mini, or Qwen3-32B as specified by each table. Lightweight retrieval and filtering calls use Gemini-2.5-flash, including the Qwen3-32B Worker runs, so that only the decision model changes. Dense retrieval embeds both queries and memory-node descriptions with `text-embedding-3-large`. Candidate nodes whose cosine similarity is below 0.4 are removed before reranking and dependency expansion.

Numerical Module and Hyperparameters. All benchmarks share the same hyperparameters. In the contracted EFE (Eq. 5), we set the subgoal utility $\beta = 2$ and the per-call cost $\alpha = 0.1$. Each contract compares at most three mechanisms and three short policies, and each policy contains at most three environment calls; policies exceeding the remaining step budget are discarded before scoring. In the Builder gate (Eq. 7), the description cost is $\lambda = 0.01$ nat per token, counted with a fixed tokenizer (`cl100k_base`) over the local predictive closure. A rejected patch is returned to the Builder with its score decomposition, and up to three proposals are allowed for each local mechanism question. A mechanism is promoted to `resolved` once a prospective contract has discriminated its candidates and its posterior exceeds 0.95. All numerical operations are deterministic and take negligible time compared with LLM calls.

Repeated Runs. Each PACT-Agent result is averaged over three independent runs. Every run rebuilds $\mathcal{K}$ from scratch under a re-sampled building order within the stratified split and then evaluates the frozen graph on the full test set, so the reported standard deviation reflects LLM sampling, task ordering, and memory evolution. Baselines are taken from their original papers when the backbone, task set, and protocol are identical (ReasoningBank and AWM on the 684-task WebArena subset; the specialized AndroidWorld systems in Table 3). Otherwise they are re-run in our harness with the same Worker backbone and building tasks. For MemRL we keep its episodic memory and value update and replace only the backbone and task split. JitRL is discussed only in related work, because its public WebArena evaluation uses the 165-task Lite split.

Token accounting. Tab. 5 separates construction and evaluation traffic. Prospective tests make building more expensive than AutoManual. At test time, the Worker still writes a contract when a retrieved Rule is unresolved, but most selected policies are short and already resolved, so the test-time count stays close to the other memory agents. ReasoningBank spends more test-time tokens because it continues to write memory on the evaluation stream.

E.2 Additional Experimental Results

Statistical Reliability. Tab. 6 complements the mean±std in the main text with task-level bootstrap intervals. All improvements over the strongest same-protocol baseline are significant at $p < 0.05$ by a paired bootstrap over tasks. The wider interval on AndroidWorld reflects its small test set (116 tasks, one task $\approx 0.86\%$). All three AndroidWorld runs still exceed both ReasoningBank (73.7%) and AgentProg (78.0%).

Do Contracts Carry the Decisions? Since the numerical module computes EFE and VFE exactly given a contract, the reliability of PACT-Agent hinges on whether the LLM can write valid contracts

Table 6: Statistical reliability of the main results. We report the per-run success rates and the 95% bootstrap confidence interval over task instances (pooled across three runs).

Benchmark	Backbone	Runs	PACT-Agent	95% CI
WebArena	Gemini-2.5-flash	54.8 / 55.8 / 54.7	55.1	[51.7, 58.5]
WebArena	Gemini-2.5-pro	64.2 / 65.9 / 64.0	64.7	[61.4, 68.0]
AndroidWorld	Gemini-2.5-pro	80.2 / 82.0 / 80.9	81.0	[74.4, 87.4]
ALFWorld	GPT-4o-mini	94.1 / 92.6 / 92.6	93.1	[89.2, 96.4]
ALFWorld	Qwen3-32B	96.3 / 97.0 / 95.6	96.3	[93.1, 98.6]

Table 7: Diagnostics of predictive contracts and the VFE gate, collected during building (averaged over three runs). Grounding accuracy is measured on 300 randomly sampled result labels per benchmark, annotated by two authors. Prospective accuracy is the fraction of first-action predictions that match the grounded result in later tasks.

Statistic	WebArena (flash)	AndroidWorld (pro)	ALFWorld (Qwen3-32B)
Worker steps with a mechanism contract (%)	31.6	34.2	25.8
Contracts passing the structural check (%)	94.8	93.1	97.4
Accuracy of grounded result labels (%)	94.1	92.0	97.9
Rule suspensions ($Z_t = 0$) per 100 contracted actions	7.4	9.6	3.4
Builder patches accepted by the VFE gate (%)	62.1	54.8	78.2
Prospective accuracy of `resolved` mechanisms (%)	92.8	91.0	96.1
Prospective accuracy of retrospective-only mechanisms (%)	74.1	69.4	81.6
Final memory size (k tokens)	71.8	41.2	16.8

and ground their results. Tab. 7 shows that contracts are formed at roughly one in three Worker steps on the two GUI benchmarks, and less often on ALFWorld, where many household Rules are resolved early. Over 93% of the contracts pass the structural check. Grounding is hardest on AndroidWorld (92.0%), which matches the higher suspension rate: a pop-up or delayed redraw often matches no saved label, so the Rule is handed to the Builder instead of being given an arbitrary posterior. Prospectively validated mechanisms predict later outcomes far more accurately than those supported only by retrospective fitting (92.8% vs. 74.1% on WebArena; 96.1% vs. 81.6% on ALFWorld with Qwen3-32B). This gap is the empirical reason for keeping VFE admission and `resolved` promotion apart in Sec. 3.4.

Robustness to Observation Noise. Deterministic contracts eliminate a mechanism after a single contradicting label. We inject synthetic noise by replacing each grounded label with a random wrong label with probability ϵ, and compare this default with the stochastic extension of Appendix C, where each mechanism assigns probability $1 - \epsilon_0$ to its predicted label and spreads $\epsilon_0 = 0.1$ over the others. Tab. 8 shows that deterministic contracts still hold at $\epsilon = 0.05$, which is already above the grounding error in Tab. 7, but fall more sharply at $\epsilon = 0.20$. Stochastic contracts lose only 2.8 points over the same range, because a contradicted mechanism is down-weighted rather than removed. At $\epsilon = 0$ they need slightly more tests to separate the candidates, which is why the clean result is 0.7 points lower. We therefore keep deterministic contracts as the default and use the stochastic extension when the observation channel is known to be noisy.

Sensitivity to Hyperparameters. Tab. 9 varies the description cost λ and the per-call cost α around their defaults. Performance stays within 1.0% for $\lambda \in [0.005, 0.02]$ and $\alpha \in [0, 0.2]$. A large λ rejects useful mechanisms and shrinks $\mathcal{K}$ too far; a large α suppresses multi-step tests and leaves more Rules unresolved after 36 building tasks. Because β only matters relative to the information gain and to α, we keep it at 2 on all benchmarks.

Case Study. Consider the Admin task "set product `P` to `Enabled` and make it visible in the storefront". The retrieved Rule *catalog-visibility* holds two mechanisms with belief $(0.55, 0.45)$: h_0, saving the product attribute is enough; h_1, visibility changes only after a separate `Publish` on the storefront tab. After the Worker writes `Enabled` and clicks `Save`, it compares leaving the page, opening the storefront tab immediately, and "reload the product, then publish only if the storefront still hides it". Under h_0 the reload returns `visible`; under h_1 it returns `hidden`. The reload therefore carries $\mathrm{H}(0.55, 0.45) = 0.688$ nat at the cost of one call, whereas publishing first completes the task under both mechanisms but yields no information. The numerical module selects the reload-first policy. The observation is grounded as `hidden`, the belief becomes $(0, 1)$, and a later

Table 8: Robustness to grounding noise on ALFWorld (GPT-4o-mini). Each grounded label is replaced by a random wrong label with probability ϵ.

Contract	$\epsilon = 0$	0.05	0.10	0.20
Deterministic	**93.1**	91.6	89.4	85.2
Stochastic	92.4	**92.1**	**91.3**	**89.6**

Table 9: Sensitivity to hyperparameters on ALFWorld (GPT-4o-mini). Tests: mechanism tests per building task.

λ	0	0.005	**0.01**	0.02	0.05
SR (%)	91.1	92.6	**93.1**	92.4	89.6
$\lvert\mathcal{K}\rvert$ (k tok.)	26.8	20.1	16.8	13.4	9.1
α	0	0.05	**0.1**	0.2	0.5
SR (%)	92.2	92.8	**93.1**	92.1	89.9
Tests	2.8	2.4	2.1	1.5	0.8

`Publish` completes the subgoal. Because the test was prospective, the Rule is marked `resolved` and later catalog tasks skip the extra check.

E.3 Prompts Used in Our Method

Listing 3: System prompt for the Worker.

```
[Role]
You are the Worker in an Active Inference Guided Agent. Complete the user's task while
    producing
an auditable predictive contract for each environment action. You interpret
natural-language mechanisms and observations. A separate numerical module
will update mechanism beliefs and choose among the candidate policies that
you describe. Do not replace the required predictions with a verbal claim
that one action is best.

[Inputs]
You receive:
1. Goal and admissible environment actions.
2. Current raw observation and grounded action history.
3. Retrieved Entity, Rule, and Skill nodes from the current memory version.
4. The pending predictive contract from the previous executed action, if any.
5. The raw return produced by that executed action.

A retrieved Rule contains its complete mechanism hypotheses, numerical belief,
scope, status, and evidence summary. Retrieval similarity is not evidence.
The pending contract is immutable and must be reviewed before you construct
the next contract.

[Required output]
Return one structured object with the following blocks in this order:

A. review_previous
B. semantic_state
C. next_contract

A. review_previous
If there is a pending contract that has not been processed:
- Read its outcome criteria and the actual executed first action.
- Locate the exact evidence in the new raw return.
- Output one declared result label.
- Use "unobserved" only when the target object or field cannot be verified.
- If the target is verified and a definite unexpected value is visible, use
  "other" and preserve the concrete value.
- State which old hypotheses are compatible with the observed label according
  to the saved prediction. Do not invent a new prediction after seeing the
  result and do not change the old contract.
If no pending contract exists, set this block to null.

B. semantic_state
Give a compact grounded state with:
1. action_result: what the last real action changed or failed to change;
2. current_layout_or_object_state: the current task-relevant state;
3. critical_analysis: the remaining task requirement, applicable memory, and
   any scope condition that is satisfied or violated.
Cite retrieved nodes by their exact names when they materially support the
analysis. Do not restate irrelevant page content.

C. next_contract
First decide whether a local mechanism uncertainty materially changes the next
action. If not, set mechanism_test=false and output one ordinary admissible
action with a visible completion criterion.
```

```
If mechanism_test=true, provide:

1. rule_ref
- Reuse one retrieved Rule version when applicable.
- If no Rule covers the uncertainty, propose one new version with a shared
  observable scope and 2 or 3 genuinely competing mechanisms.
- For every new mechanism, output a positive initialization score. Scores are
  relative proposal preferences, not calibrated probabilities.

2. applicability
- List all observable conditions needed for the predictions.
- Do not assume an unknown permission, object identity, page mode, or prior
  action result. Observe it first or include it in the mechanism model.

3. outcome_criteria
- Define mutually exclusive result labels using facts that the next Worker can
  read from the tool return.
- Include "other" for a definite result outside the expected named values.
- "unobserved" is reserved for an inability to perform the check and is not a
  mechanism outcome.

4. subgoal
- Define one common, visible success criterion for every candidate policy.
- Do not change the success standard to favor a policy.

5. candidate_plans
- Propose 2 or 3 short observation-contingent policies, normally no more than
  3 environment calls per predicted branch.
- Branch only on results that would actually be visible during execution.
- If two mechanisms have produced the same visible history, the policy must
  choose the same next action for both.
- Respect authorization, irreversibility, action budget, and environment
  syntax. Inadmissible plans must not be proposed for numerical scoring.

6. predicted_traces
For every candidate policy and every positive-weight mechanism:
- Predict the complete action-result trace under that mechanism.
- In the default deterministic setting, provide exactly one result for each
  action and one complete trace per mechanism.
- Mark subgoal_satisfied as true or false using the common subgoal criterion.
- Shared action prefixes must have identical predictions across policies.
- Treat the mechanism as true while predicting its trace, regardless of its
  current belief weight.
- Do not estimate LLM generation probabilities and do not sample yourself
  repeatedly to create a probability table.

Use a stochastic result distribution only when the Rule explicitly states that
the same mechanism can yield multiple outcomes under the same conditions. In
that case, output a normalized distribution over the declared labels and state
any conditional independence or state-transition assumptions.

7. first_action_predictions
- Repeat the predictions needed to evaluate the first environment action of
  each plan. These entries will be saved before execution and used for the
  next posterior update.

[Important separation of roles]
- You may propose mechanisms, plans, predictions, and observable criteria.
- You may note that an observed result conflicts with an old hypothesis.
- You do not normalize posterior beliefs.
- You do not compute entropy, information gain, Expected Free Energy, or the
  final numerical policy ranking.
- You do not edit the committed memory during task execution.
```

Listing 4: System prompt for the mechanism Builder.

```
[Role]
You are the mechanism Builder in an Active Inference Guided Agent. After one task trajectory
     has
finished, propose a local semantic-memory revision that better explains a
fixed set of action-result records without adding unnecessary predictive text.
A separate numerical module will compute model evidence, token complexity,
and the VFE difference. You propose a model; you do not decide that it should
be accepted.

[Inputs]
You receive:
1. A fixed replay set D. Every row contains only the context visible before an
   executed action, the executed action, the fixed result criterion, and the
   actual result.
```

```
2. The currently committed local Rule version and its pre-replay belief
   snapshot.
3. Required Entity definitions and dependent Skills.
4. Immutable online predictive contracts and evidence provenance.
5. Optional score feedback from a rejected proposal: contradicted replay rows,
   change in negative log evidence, added token count, and total delta VFE.

All proposals in this revision attempt are scored on the same D and against the
same committed old version. Do not remove difficult rows or narrow the scope
only to exclude a contradiction.

[Task]
Return a local patch and a replay-prediction table.

1. Diagnose the mechanism question
- Identify one local environmental mechanism that the replay records test.
- Separate an incorrect mechanism from a violated scope condition or a mistaken
  object interpretation.
- Use the raw contracts to distinguish prospective online evidence from facts
  noticed only after the trajectory.

2. Propose one local patch
Allowed operations are add, replace, merge, compress, or delete.
- Define a reusable mechanism and observable scope. Do not merely paraphrase
  the sequence of results in D.
- Give 2 or 3 competing hypotheses when uncertainty remains.
- If the hypothesis set is new or changed, assign every hypothesis a positive
  initialization score. Freeze these scores before replay scoring.
- If the mechanism is stochastic, provide a normalized conditional result
  distribution and all history-dependent conditions needed to use it.
- Include only Entity definitions that are necessary to interpret the scope or
  result labels.

3. Replay the old and proposed versions
For each row i in D and each hypothesis in the relevant version, answer:
"Given only x_i and the executed action a_i, what result did this hypothesis
predict before y_i occurred?"
- Use the same actual result label for the old and new versions.
- Do not use y_i as an input to the prediction.
- Reuse a saved online prediction when the Rule version and context match.
- A prediction reconstructed after the event must be labeled replay_prediction
  and must not overwrite an online contract.
- One hypothesis must explain the complete replay set; do not choose a different
  hypothesis independently for each row.

4. State dependencies and optional Skill changes
- List any Entity or Rule text required by the proposed predictions.
- Propose a Skill edit only if it follows from the accepted mechanism model.
- A Skill depending on an unresolved Rule must preserve a verification step or a
  conditional branch.

5. Evidence provenance
- Mark knowledge proposed after seeing D as retrospective_fit.
- Never label it as an online_test and never fabricate a pre-action contract.

[Response format]
Return:
- patch_operation
- proposed_rule_version
- initialization_scores
- old_replay_predictions
- new_replay_predictions
- required_dependencies
- optional_skill_patch
- evidence_provenance
- concise rationale tied to the failed or explained records

[After score feedback]
If the numerical module rejects the proposal:
- Keep the replay set and old version fixed.
- Address the named zero-likelihood records.
- Remove redundant mechanism text or unsupported branches.
- Do not change only the belief field or assert that the rule is useful.
- A successful revision must improve predictive evidence enough to justify its
  added text, or must preserve evidence while reducing text.
```

Listing 5: Prompt template for Skill maintenance after a Rule decision.

```
[Role]
Update reusable Skill nodes after local Rule revisions have been accepted.
```

```
The accepted Rule versions, statuses, and evidence provenance are authoritative.

[Guidelines]
1. Extract a reusable procedure only from an actually successful trajectory.
2. State the precondition and target task type.
3. Replace instance-specific identifiers with variables.
4. Cite the Rule nodes that justify conditional steps.
5. If a required Rule is resolved, the Skill may use its default mechanism within
   the declared scope.
6. If a required Rule is unresolved, retain an observation check or conditional
   branch. Do not convert the highest-weight hypothesis into an unconditional
   instruction.
7. Reuse existing subskills and declare dependencies rather than duplicating
   their bodies.
8. Do not store a local single-entity action as a Skill when it is already
   represented by an Entity description and Rule.
9. Record the successful trajectory and accepted Rule versions as evidence.
```

E.4 Prompting and Numerical Control

The prompts make mechanism proposals and observation criteria explicit, but prompts are not the final decision procedure. In particular, the Worker prompt does not ask the language model to balance "high value" and "high uncertainty" through an unobserved internal judgment. The contract externalizes the predictions needed for that comparison. Likewise, the Builder prompt does not ask whether a memory update feels counterfactually useful. It asks for a model and replay predictions, after which Eq. 8 determines admission.