BEYOND SUCCESSFUL EXPERIENCE REUSE: INDUCING REVISION KNOWLEDGE FROM EVOLVING EXPERIENCE FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Experience-based learning enables LLM agents to accumulate and reuse knowledge from prior interactions, but previously successful procedures may become inapplicable as contexts evolve. We study *revision knowledge induction*: recovering structured knowledge about when acquired procedures remain valid or require revision. We introduce *Contrastive Evolution Attribution (CEA)*, which factorizes evolutionary experience through structured contrasts to identify revision-relevant contextual factors and recover the corresponding state-to-procedure mapping. To evaluate this capability, we develop APPLYBOUNDARY, a controlled diagnostic benchmark for revision knowledge recovery and downstream procedure selection under unseen contexts with correlated contextual changes. Experiments show that CEA improves both revision-factor identification and state-to-procedure mapping recovery over experience-based approaches that use the same evolutionary evidence without contrastive factorization. A small-scale validation on documented real-world system changes provides preliminary evidence that similar revision patterns can also appear in practical settings. Further analysis shows that controlled-success evidence helps identify revision-relevant changes, while effectively utilizing recovered knowledge for future decisions remains a separate challenge.

1 INTRODUCTION

Large language model (LLM) agents are increasingly expected to improve through experience by accumulating and reusing knowledge from past interactions, including memories, workflows, trajectories, and reusable procedures. Recent studies have shown that such experience reuse can improve agent capabilities in long-horizon planning, tool usage, and autonomous task completion (Zhao et al., 2024; Wang et al., 2024; 2025).

Existing approaches have made substantial progress in enabling agents to store, retrieve, and refine successful procedures through memory-based reflection (Park et al., 2023; Zhong et al., 2024; Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024; Zhou et al., 2024) and explicit procedural representations (Lu et al., 2026; Zhao et al., 2024). These methods provide effective mechanisms for learning *what* procedures agents should execute in previously encountered or related situations. However, effective experience reuse also requires understanding *when* previously acquired procedures remain applicable. Existing methods often leave the contextual knowledge governing procedure validity implicit: retrieving similar past experiences does not reveal whether their underlying conditions still hold or when a learned procedure should be trusted or revised.

This limitation becomes critical in dynamic environments where task requirements, environment states, or external constraints evolve. Consider an agent interacting with an external service or software platform whose procedure becomes ineffective after a documented system update. Because multiple contextual factors may change simultaneously, the failure alone does not reveal which factor affects procedure validity or whether the original procedure remains applicable under certain conditions. Blind reuse may cause repeated failures, whereas discarding previous procedures may remove knowledge that remains applicable under unchanged conditions. Figure 1 illustrates this ambiguity.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

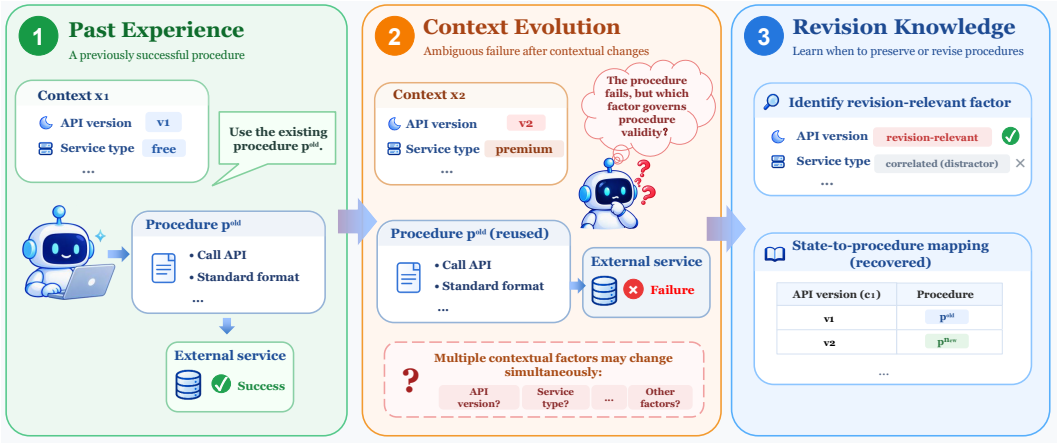


Figure 1: A motivating example of revision knowledge induction. Contextual evolution changes procedure validity, requiring agents to learn the applicability conditions of existing procedures.

In this work, we study **revision knowledge induction**, where agents recover structured knowledge about procedure applicability from evolving experience. Specifically, the goal is to identify revision-relevant contextual factors and recover the corresponding state-to-procedure mapping. Unlike learning new procedures from scratch, revision knowledge induction focuses on understanding when existing procedures should be preserved or revised as environments change.

A fundamental question therefore remains:

When a previously successful procedure becomes ineffective, how can an agent determine whether it should be preserved or revised under future contexts?

Our key insight is that **behavioral evolution provides richer evidence for learning procedure applicability than isolated failures**. By comparing executions before and after behavioral changes, agents can distinguish revision-relevant changes from correlated contextual variations and recover the conditions under which different procedures should be applied.

Based on this insight, we propose **Contrastive Evolution Attribution (CEA)**, a contrastive framework for inducing revision knowledge from evolving experience. CEA factorizes evolutionary observations through structured contrasts to recover revision knowledge about procedure applicability. The original and revised procedures are obtained from prior experience rather than generated by CEA, which instead extracts the knowledge governing when each procedure should be used.

To evaluate this capability, we develop APPLYBOUNDARY, a diagnostic benchmark for LLM agents. Each task contains an evolving experience trajectory and unseen contexts where the revision-relevant factor conflicts with correlated distractor factors. The benchmark separately evaluates revision knowledge recovery and downstream procedure selection. We additionally validate the formulation on a small set of documented real-world system changes.

Experiments show that CEA more accurately recovers revision knowledge than approaches that reuse evolutionary experience without explicitly organizing evidence for revision. Further analysis reveals that evolutionary contrasts provide important signals for identifying revision-relevant changes, and that recovered revision knowledge provides useful guidance for downstream procedure selection while effective utilization remains a distinct challenge.

Our contributions are summarized as follows:

- **Problem formulation.** We formalize *revision knowledge induction* for experience-based LLM agents, focusing on when acquired procedures remain applicable or require revision under contextual evolution.
- **Method and benchmark.** We introduce **Contrastive Evolution Attribution (CEA)**, a structured evidence factorization framework, and APPLYBOUNDARY, a diagnostic benchmark for revision knowledge recovery and downstream procedure selection.

- **Empirical analysis.** We show that contrastive evidence factorization improves structured revision knowledge recovery, provide preliminary validation on documented real-world system changes, and analyze revision knowledge recovery and downstream utilization as separate evaluation axes.

2 Related Work

Experience reuse, memory, and reusable skills. A growing body of work studies how LLM agents accumulate and reuse knowledge from previous interactions. Long-term memory mechanisms enable agents to store, retrieve, and reflect on past experiences (Park et al., 2023; Zhong et al., 2024; Packer et al., 2023), while recent approaches further investigate memory organization, evolution, and evaluation (Xu et al., 2025; Maharana et al., 2024; Wu et al., 2025). Beyond storing interaction histories, experience can be transformed into reusable skills, workflows, and tools. ExpeL extracts transferable insights from prior tasks, Voyager maintains executable skills through environment interaction, and Agent Workflow Memory learns reusable workflows for long-horizon tasks (Zhao et al., 2024; Wang et al., 2024; 2025). Related efforts also explore large-scale tool learning and agent evaluation (Qin et al., 2024; Liu et al., 2024). These approaches focus on acquiring, organizing, and reusing effective knowledge or procedures. However, they do not explicitly model the contextual conditions under which a previously successful procedure remains applicable or requires revision. In contrast, our work studies revision knowledge induction by recovering applicability knowledge from evolving experience.

Feedback, failure, and self-correction. Another line of research investigates how LLM agents improve through feedback and failed attempts. Reflexion uses verbal reflections from previous failures, Self-Refine performs iterative self-feedback refinement, and CRITIC incorporates external feedback for correction (Shinn et al., 2023; Madaan et al., 2023; Gou et al., 2024). Recent methods further combine search, reinforcement learning, or program-guided verification for improving self-correction (Zhou et al., 2024; Kumar et al., 2025; Song et al., 2025). These methods demonstrate the value of failures as improvement signals, but mainly use them to correct or refine behaviors. Our work differs by treating failure as structured evolutionary evidence for inducing knowledge about when existing procedures should be preserved or revised.

Causal reasoning and contextual adaptation. Recent work studies causal reasoning capabilities of large language models, including causal discovery, interventional reasoning, and counterfactual inference (Kiciman et al., 2024; Jin et al., 2023). Related studies explore counterfactual explanations of model behavior (Feder et al., 2021; Kaushik et al., 2020). These works focus on causal reasoning or explaining model behavior, rather than inducing reusable applicability knowledge from evolving agent experience. Our work instead investigates how evolutionary experience can be factorized to recover revision knowledge for future procedure adaptation.

3 Method

We study *revision knowledge induction* for LLM agents under contextual evolution. Given experience where a previously successful procedure becomes ineffective and a successful recovery execution is observed, our goal is to recover the applicability boundary that determines when an acquired procedure remains valid or requires revision. Specifically, we formulate revision knowledge as identifying the contextual factor governing procedure applicability and recovering the corresponding state-to-procedure mapping. We introduce *Contrastive Evolution Attribution (CEA)*, an evidence factorization framework that structures evolutionary experience through complementary contrasts to recover revision knowledge. Instead of directly summarizing past experiences, CEA factorizes evolutionary evidence to identify revision-relevant contextual factors and recover the corresponding state-to-procedure mapping, forming a structured revision knowledge representation. The induced revision knowledge can then be reused for procedure selection under unseen contexts without replaying the original evolutionary trajectory. Figure 2 summarizes the resulting two-stage inference setup.

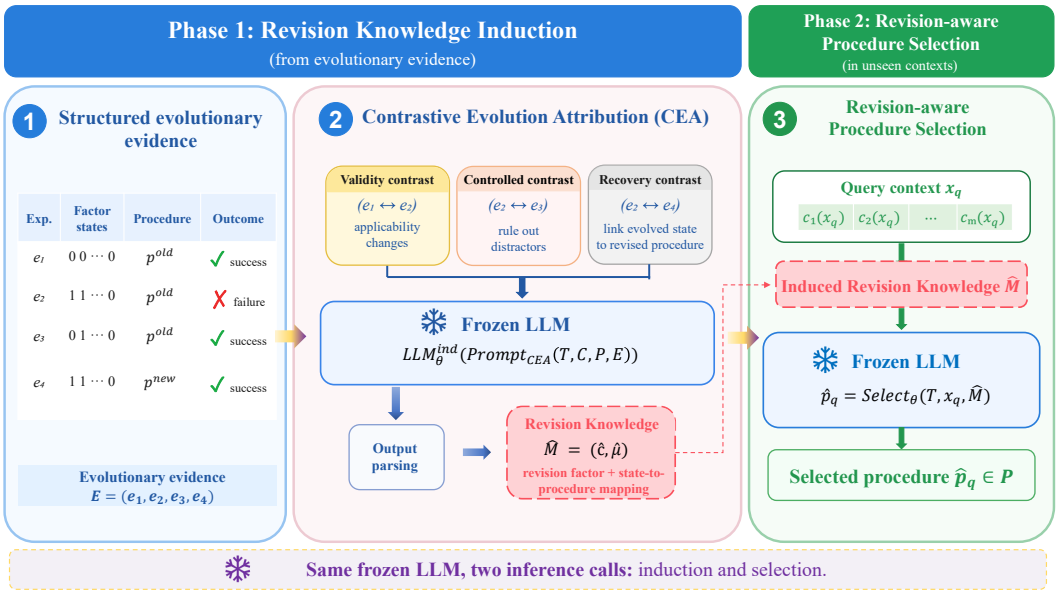


Figure 2: Overview of the two-stage revision knowledge induction setup. CEA first induces structured revision knowledge representation from evolutionary evidence, which is then used for procedure selection under unseen contexts.

3.1 Problem Formulation

Let $\mathcal{X}$ denote the context space and $\mathcal{P} = \{p^{(0)}, p^{(1)}\}$ denote the set of registered procedures. Each task provides an ordered list of m candidate contextual factors $\mathcal{C} = (c_1, \ldots, c_m)$, where each factor $c_i : \mathcal{X} \to \mathcal{S}_i$ maps a context to its corresponding state. These factors represent observable contextual dimensions available to the agent, while CEA identifies the factor whose state determines procedure applicability.

Let $c^* = c_j \in \mathcal{C}$ denote the revision-relevant factor. The target revision knowledge is defined as

$$Z_{\text{rev}} = (c^*, \mu^*), \qquad \mu^* : \mathcal{S}_j \to \mathcal{P}, \tag{1}$$

where μ^* maps each state of the revision-relevant factor to the applicable procedure.

For simplicity, we consider a binary revision setting where $\mathcal{S}_j = \{0, 1\}$. The correspondence between factor states and procedures is task-dependent and is not determined by the temporal order of procedures. Therefore, revision knowledge requires recovering both the revision-relevant factor and its state-dependent procedure assignment. We focus on factor-conditioned revision induction under an observable contextual factor space, where attribution over candidate factors is separated from open-ended factor discovery.

3.2 Contrastive Evolution Attribution

Evolutionary evidence. Let $\mathcal{Y} = \{\text{success}, \text{failure}\}$ denote the outcome space. An evolutionary experience is represented as $e = (x, p, y) \in \mathcal{X} \times \mathcal{P} \times \mathcal{Y}$.

For each task, p^{old} denotes the procedure used before revision, and p^{new} denotes the procedure that restores successful execution after contextual evolution. The evolutionary evidence is factorized as

$$\begin{aligned} e_1 &= (x_1, p^{\text{old}}, \text{success}), \quad e_2 = (x_2, p^{\text{old}}, \text{failure}), \\ e_3 &= (x_3, p^{\text{old}}, \text{success}), \quad e_4 = (x_2, p^{\text{new}}, \text{success}). \end{aligned} \tag{2}$$

Here, p^{old} and p^{new} represent temporal roles in the evolution trajectory rather than fixed assignments to specific factor states.

CEA factorizes evolutionary evidence into complementary evidence relations, including validity, controlled, and recovery relations. The validity relation $e_1 \leftrightarrow e_2$ captures whether procedure applicability changes after contextual evolution. The controlled relation $e_2 \leftrightarrow e_3$ provides evidence for distinguishing the revision-relevant contextual factor from correlated variations by comparing executions with controlled differences. The recovery relation $e_2 \leftrightarrow e_4$ provides evidence for recovering the procedure that restores successful execution under the evolved context. Together, these relations enable CEA to recover the revision-relevant factor and its state-to-procedure mapping.

Revision knowledge induction. Based on these factorized evidence relations, CEA induces the revision structure. Let $\hat{j} \in \{1, \ldots, m\}$ denote the predicted position of the revision-relevant factor, where $\hat{c} = c_{\hat{j}}$. CEA further induces the state-to-procedure mapping $\hat{\mu} : \mathcal{S}_{\hat{j}} \to \mathcal{P}$. The recovered revision knowledge is represented as

$$\hat{Z}_{\text{rev}} = (\hat{c}, \hat{\mu}), \tag{3}$$

which has the same form as the target revision knowledge.

LLM-based realization. We instantiate CEA with a frozen LLM as the inference backbone for structured revision knowledge induction. Given task specification T, candidate contextual factors $\mathcal{C}$, procedures $\mathcal{P}$, and the CEA-factorized evolutionary evidence E, the LLM predicts the revision knowledge:

$$(\hat{c}, \hat{\mu}) = \text{LLM}_\theta^{\text{ind}}\left(\text{Prompt}_{\text{CEA}}(T, \mathcal{C}, \mathcal{P}, E)\right). \tag{4}$$

The induced revision knowledge is stored as revision knowledge representation $\hat{M} = (\hat{c}, \hat{\mu})$, where $\hat{c}$ denotes the identified revision-relevant factor and $\hat{\mu}$ denotes the recovered state-to-procedure mapping.

Implementation details regarding output formatting and structural constraints are provided in Appendix B. The resulting revision knowledge representation can be reused for future contexts without replaying the original evolutionary trajectory.

3.3 Revision-Aware Procedure Selection

The induced revision knowledge specifies the applicability boundary of existing procedures, but applying such knowledge to a new context requires an additional decision step. Therefore, we separate revision knowledge induction from revision-aware procedure selection. For an unseen query context x_q, the agent conditions on the induced revision knowledge representation $\hat{M}$ rather than replaying the full evolutionary trajectory. Procedure selection is performed through a second inference call:

$$\hat{p}_q = \text{Select}_\theta(T, x_q, \hat{M}), \qquad \hat{p}_q \in \mathcal{P}. \tag{5}$$

Given the ground-truth revision structure, the oracle applicable procedure is

$$p_q^* = \mu^*(c^*(x_q)). \tag{6}$$

Therefore, downstream evaluation measures whether the induced revision knowledge representation can be effectively utilized for procedure selection under unseen contexts. The concrete learner-visible realization of this interface is described in Section 4.

Table 1: **Task domains and representative revision scenarios in APPLYBOUNDARY.**

Domain	Reusable Procedure	Revision Trigger
Tool/API Usage	API interaction procedure	Interface or format change
Environment Adaptation	Environment-dependent procedure	Environmental state change
Interaction Policy	Rule-following procedure	Policy or constraint change
Workflow Adaptation	Multi-step procedure	Dependency change

4 APPLYBOUNDARY: A DIAGNOSTIC BENCHMARK FOR REVISION KNOWLEDGE

4.1 BENCHMARK CONSTRUCTION

APPLYBOUNDARY instantiates the binary revision setting defined in Section 3.1 as a controlled diagnostic benchmark. Each task program contains two registered procedures, candidate contextual factors, and benchmark-defined revision knowledge $Z_{\text{rev}} = (c^*, \mu^*)$, where one factor is revision-relevant and the others serve as distractors. Evolutionary trajectories follow Equation 2, where correlated contextual changes create ambiguity in attributing procedure applicability changes. This enables evaluation of whether agents recover when acquired procedures remain valid or require revision.

In the current instantiation, APPLYBOUNDARY uses binary state encoding, where states 0 and 1 denote the two states of the revision-relevant factor. The semantic correspondence between states and procedures is defined by each task program rather than a fixed global rule, enabling evaluation of state-dependent applicability without relying on procedure ordering.

To evaluate applicability reasoning beyond surface correlations, we introduce unseen *boundary–distractor disagreement* cases, where the revision-relevant factor and correlated distractors imply different procedure choices. Correct selection requires applying the induced state-to-procedure mapping. The two procedures are presented as randomized options A and B across queries to prevent option-position shortcuts.

4.2 TASK DOMAINS AND SCENARIO INSTANTIATION

Although different domains exhibit different surface semantics, they share the same revision pattern: contextual evolution changes procedure applicability and requires selecting between alternative procedures. We instantiate APPLYBOUNDARY across four representative forms of procedure evolution: tool/API usage, environment adaptation, interaction policy, and workflow adaptation. Table 1 summarizes the instantiated domains, while detailed task statistics and examples are provided in Appendix A.

5 EXPERIMENTAL SETUP

We evaluate revision knowledge induction under a controlled experimental protocol. All main experiments use the same frozen LLM backbone and inference configuration without parameter updates. Details of model configuration, prompt templates, output validation, and additional experimental analyses are provided in Appendix B. Additional backbone evaluations on GPT-5 and DeepSeek-V3 are reported in Appendix C.

We construct task-aligned controls to evaluate different forms of experience reuse. B0 uses no prior experience and measures the base procedure selection capability without external knowledge. B1 provides successful experience only, evaluating direct reuse of previously effective procedures. B2 provides evolutionary experience without explicitly organizing revision-relevant evidence. B3 provides structured evolutionary observations without explicit contrastive factorization, isolating the contribution of evidence organization. B4 is the primary controlled-evidence baseline: it receives the same evolutionary observations, evidence ordering, task information, and output format as CEA, but presents the evidence without the complementary contrast structure defined by CEA. Therefore, the

Table 2: **Evaluation controls for experience reuse and revision knowledge induction.**

Method	Experience	Evidence Organization	Purpose
B0	None	None	Base capability
B1	Successful experience	None	Direct reuse
B2	Evolutionary experience	None	Evolutionary evidence reuse
B3	Structured evolutionary evidence	Generic structure	Evidence organization control
B4	Full evolutionary evidence	Non-contrastive organization	Controlled evidence baseline
CEA	Full evolutionary evidence	Contrastive relations	Revision knowledge induction

CEA–B4 comparison isolates the contribution of contrastive evidence factorization under controlled evidence conditions. Table 2 summarizes the evaluation settings.

We evaluate revision knowledge recovery and downstream procedure selection. For revision knowledge recovery, Revision Factor Identification Accuracy measures whether the revision-relevant contextual factor is correctly identified, i.e., $\hat{c} = c^*$. State-to-Procedure Mapping Accuracy measures exact recovery of the state-to-procedure mapping, requiring $\hat{\mu}(0) = \mu^*(0)$ and $\hat{\mu}(1) = \mu^*(1)$ simultaneously. For downstream evaluation, we report procedure selection accuracy on unseen boundary–distractor disagreement cases, where the revision-relevant factor and correlated distractors imply different procedure choices. The task program is used as the primary inferential unit, with query-level outcomes aggregated within each program. The CEA–B4 comparison is the primary comparison under controlled evidence conditions.

We further evaluate predefined CEA variants by removing or perturbing individual components of evolutionary evidence: **CEA_NO_E3**, **CEA_NO_FAILURE**, **CEA_NO_RECOVERY**, **CEA_NO_OUTCOME**, and **CEA_SHUFFLED**. These variants analyze the contribution of different evidence components to revision knowledge induction. We additionally conduct two diagnostic analyses: **Ground-Truth Revision Knowledge Selection**, which provides benchmark-defined revision knowledge to the frozen LLM selection interface, and **Ground-Truth Mapping Decoding**, which directly applies the benchmark-defined state-to-procedure mapping. Statistical comparisons are paired at the task-program level with Holm correction applied across predefined hypothesis tests.

6 RESULTS

We evaluate revision knowledge induction from two perspectives: (1) recovery of revision knowledge Z_{rev} from evolutionary experience, and (2) downstream procedure selection under unseen contexts. Unless otherwise specified, all comparisons follow the predefined evaluation protocol. Invalid induction outputs are counted as failures in downstream evaluation, while recovery metrics report valid-output conditional accuracy to separately analyze structural recovery quality.

6.1 REVISION KNOWLEDGE CAN BE RECOVERED FROM EVOLUTIONARY EVIDENCE

Table 3 reports revision factor identification and state-to-procedure mapping accuracy. CEA achieves the best recovery performance on both components of Z_{rev}. Compared with B4, which receives the same evolutionary evidence without contrastive factorization, CEA improves revision factor identification from 0.8750 to 0.9407 and mapping recovery from 0.5333 to 0.6271. B3 performs similarly to B2, indicating that generic structuring of evolutionary observations alone provides limited benefit. The improvement from B3 to B4 suggests that organizing evidence through applicability-related relations provides additional attribution signals. The gap between factor identification and mapping recovery indicates that recovering applicability conditions and corresponding procedure assignments remain distinct challenges, which is further supported by the error decomposition in Appendix B.5.

6.2 CONTRIBUTION OF CONTROLLED EVIDENCE IN REVISION KNOWLEDGE INDUCTION

We analyze the contribution of different evolutionary evidence components using predefined CEA ablations. Since downstream procedure selection relies on the induced revision knowledge, we use selection accuracy as a functional measure of evidence contribution. Removing the controlled-

Table 3: **Revision knowledge recovery performance.** Accuracy is computed over valid revision knowledge induction outputs to measure structural recovery quality. Valid-output rates are reported in Appendix B.4.

Method	Revision Factor Identification Accuracy	State-to-Procedure Mapping Accuracy
B2	0.7250	0.4833
B3	0.7436	0.4667
B4	0.8750	0.5333
CEA	**0.9407**	**0.6271**

Table 4: **CEA ablation results on downstream procedure selection accuracy.**

Variant	Procedure Selection Accuracy ↑
CEA (full)	**0.8438**
CEA_NO_FAILURE	0.7882
CEA_NO_RECOVERY	0.6875
CEA_NO_OUTCOME	0.6493
CEA_NO_E3	0.5243
CEA_SHUFFLED	0.4931

success observation e_3 reduces accuracy from 0.8438 to 0.5243, with a paired mean difference of 0.31944 across 24 task programs (95% CI: $[0.21181, 0.42708]$; $p_{\text{adj}} = 1.192 \times 10^{-5}$). This indicates that controlled evidence provides important attribution signals for distinguishing revision-relevant changes from correlated variations. Other ablations further show the contribution of evolutionary evidence: removing failure, recovery, and outcome information reduces accuracy to 0.7882, 0.6875, and 0.6493, respectively, while shuffling controlled correspondence yields the lowest accuracy (0.4931). These results suggest that both informative evidence and its organization are important for revision knowledge induction.

6.3 Revision Knowledge Recovery and Procedure Selection Are Distinct Capabilities

We next evaluate whether recovered revision knowledge improves downstream procedure selection on unseen boundary–distractor disagreement cases. CEA achieves comparable accuracy to B4 (0.8438 vs. 0.8472). Across 24 paired task programs, the difference is -0.00347 with a 95% bootstrap CI of $[-0.06944, 0.06250]$ and is not significant after Holm correction ($p_{\text{adj}} = 0.5354$). These results indicate that improved revision knowledge recovery does not directly translate into higher end-to-end selection accuracy, suggesting that knowledge recovery and decision-time utilization remain distinct evaluation axes. Interestingly, generic structuring alone reduces downstream selection performance relative to unstructured evolutionary experience, suggesting that structure is not inherently beneficial unless it organizes evidence around procedure applicability.

6.4 Diagnostic Analysis of Knowledge Utilization

To analyze the gap between revision knowledge recovery and downstream utilization, we conduct two post-hoc diagnostics. Providing benchmark-defined revision knowledge while keeping the downstream selection interface unchanged improves procedure selection accuracy from 0.8438 to 0.9167, while directly decoding the state-to-procedure mapping achieves 1.0000 accuracy. These diagnostics reveal that revision knowledge induction and downstream knowledge utilization represent different evaluation axes.

6.5 Summary of Predefined Hypothesis Tests

We evaluate four predefined hypotheses covering revision knowledge recovery, evidence contribution, and downstream utilization. H1a and H1b test whether CEA improves revision-factor identification and state-to-procedure mapping over B4, which uses the same evolutionary evidence without contrastive factorization. H2 evaluates the contribution of controlled-success evidence, while H3

Table 5: **Downstream procedure selection accuracy on unseen boundary–distractor disagreement cases.**

Method	Procedure Selection Accuracy ↑
B0	0.5000
B1	0.6944
B2	0.7361
B3	0.5799
B4	**0.8472**
CEA	0.8438

Table 6: **Knowledge utilization diagnostics.**

Diagnostic Condition	Procedure Selection Accuracy ↑
CEA-induced Revision Knowledge	0.8438
Ground-Truth Revision Knowledge	0.9167
Ground-Truth Mapping Decoding	1.0000

tests whether improved revision knowledge recovery translates into downstream procedure selection. Table 7 summarizes the results. Overall, CEA improves revision knowledge recovery, whereas downstream performance also depends on effective utilization of the recovered knowledge.

Table 7: **Summary of predefined hypothesis tests.** Effects are paired task-program differences with bootstrap 95% CIs and Holm-adjusted p-values. Recovery metrics use valid induction outputs; downstream metrics count invalid outputs as failures.

H	Endpoint	Comparator	CEA	Comp.	Δ	95% CI / p_{adj}	Outcome
H1a	Revision Factor Identification	B4	0.9407	0.8750	+0.0657	[0.0104,0.1215]/0.0312	Improved recovery
H1b	State-to-Procedure Mapping Accuracy	B4	0.6271	0.5333	+0.0938	[0.0187,0.1719]/0.0284	Improved recovery
H2	Controlled-Success Evidence	CEA_NO_E3	0.8438	0.5243	+0.3195	[0.21181,0.42708]/1.192×10^{-5}	Evidence contribution observed
H3	Downstream Procedure Selection	B4	0.8438	0.8472	-0.0035	[-0.06944,0.06250]/0.5354	No aggregate improvement

6.6 Small-Scale Real-World Case Validation

We further validate CEA on three documented real-world system changes. CEA recovers the revision-relevant factor in all three cases and the corresponding state-to-procedure mapping in two. These results provide preliminary evidence that similar revision patterns occur in practical system evolution, while the remaining mapping error illustrates the difficulty of more complex platform changes. Full details are provided in Appendix D.

7 Conclusion

This work studies *revision knowledge induction* for experience-based LLM agents: recovering when previously acquired procedures remain applicable or require revision as contexts evolve. We introduce **Contrastive Evolution Attribution (CEA)**, which factorizes evolving experience into complementary evidence relations to induce structured revision knowledge. Experiments on ApplyBoundary show that CEA improves procedure applicability recovery over approaches using the same evolutionary evidence without contrastive factorization. In three documented real-world cases, CEA recovers the revision-relevant factor in all cases and the corresponding mapping in two. Further analysis shows that revision knowledge recovery and downstream utilization remain distinct challenges. More broadly, our results suggest that adaptation under contextual evolution requires representing not only revised procedures, but also the applicability boundaries that determine when each procedure should be used. Overall, experience-based agents benefit not only from reusing successful procedures, but also from explicitly modeling the conditions under which they should be reused or revised. Current evaluation focuses on factor-conditioned boundaries with observable contextual factors; extending revision knowledge induction to open-ended factor discovery and continuous applicability boundaries remains future work.

Ethical Considerations

This work evaluates LLM agents under controlled benchmark settings and does not involve private user data or human subjects. The proposed method aims to improve the reliability of experience-based agents by enabling them to better identify when previously acquired procedures remain applicable or require revision. As with other LLM-based systems, the resulting agents may still make incorrect decisions due to model limitations, and human oversight remains necessary in high-stakes applications.

AI use statement

Generative AI tools were used for limited assistance in literature discovery, manuscript writing refinement, and code-related discussions. Specifically, GPT-5.5 was used to assist with literature search, related-work exploration, and language refinement. Qwen and Claude were used for code discussions and debugging support during implementation. The research ideas, problem formulation, method design, experimental design, analysis, and conclusions were developed and conducted by the authors. All AI-assisted outputs were reviewed and verified by the authors, who take full responsibility for the final content of this work, including all scientific claims, analyses, code, and experimental artifacts.

Reproducibility statement

We provide detailed descriptions of the benchmark construction, experimental protocol, evaluation metrics, and statistical analysis in the paper and appendix. The released materials include the benchmark specification, experimental configurations, evaluation scripts, and analysis code required to reproduce the reported results, subject to external model provider availability and access constraints.

References

Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. Causalm: Causal model explanation through counterfactual language models. *Comput. Linguistics*, 47(2):333–386, 2021. doi: 10.1162/COLI_A_00404. URL https://doi.org/10.1162/coli_a_00404.

GitHub. Token authentication requirements for git operations. https://github.blog/security/application-security/token-authentication-requirements-for-git-operations/, 2020. Updated February 7, 2022; accessed 2026-09-25.

Google Chrome. Manifest v3. https://developer.chrome.com/docs/extensions/develop/migrate/what-is-mv3, a. Accessed 2026-09-25.

Google Chrome. Migrate to a service worker. https://developer.chrome.com/docs/extensions/develop/migrate/to-service-workers, b. Accessed 2026-09-25.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: large language models can self-correct with tool-interactive critiquing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Sx038qxjek.

Zhijing Jin, Yuen Chen, Felix Leeb, Luigi Gresele, Ojasv Kamal, Zhiheng Lyu, Kevin Blin, Fernando Gonzalez, Max Kleiman-Weiner, Mrinmaya Sachan, and Bernhard Schölkopf. Cladder: assessing causal reasoning in language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. Learning the difference that makes A difference with counterfactually-augmented data. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Sklgs0NFvr.

Emre Kiciman, Robert Osazuwa Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=mqoxLkX210.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D. Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M. Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal M. P. Behbahani, and Aleksandra Faust. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=CjwERcAU7w.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Yuxing Lu, Yicheng Chen, Shanchan Wu, and Sercan Ö. Arik. Procedural graphs: Self-evolving execution structures for LLM agents, 2026. URL https://arxiv.org/abs/2609.09153.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 13851–13870. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.747. URL https://doi.org/10.18653/v1/2024.acl-long.747.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. doi: 10.48550/ARXIV.2310.08560. URL https://doi.org/10.48550/arXiv.2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In Sean Follmer, Jeff Han, Jürgen Steimle, and Nathalie Henry Riche (eds.), *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, UIST 2023, San Francisco, CA, USA, 29 October 2023- 1 November 2023*, pp. 2:1–2:22. ACM, 2023. doi: 10.1145/3586183.3606763. URL https://doi.org/10.1145/3586183.3606763.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances*

in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Slack. The files.upload method is retiring, to be replaced by sequenced web api methods. https://api.slack.com/changelog/2024-04-a-better-way-to-upload-files-is-here-to-stay, 2024. Accessed 2026-09-25.

Xiaoshuai Song, Yanan Wu, Weixun Wang, Jiaheng Liu, Wenbo Su, and Bo Zheng. Progco: Program helps self-correction of large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 944–959. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.ACL-SHORT.73. URL https://doi.org/10.18653/v1/2025.acl-short.73.

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. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=pZiyCaVuti.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19724–19731. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In Ruslan

Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

A BENCHMARK DETAILS

We use the final APPLYBOUNDARY benchmark introduced in the main paper. The benchmark contains 40 task programs spanning 10 semantic domains, and each task program is instantiated with three surface realizations, resulting in 120 surface-level instances. Since different realizations correspond to the same underlying task program, all evaluations are aggregated at the task-program level. The benchmark is divided into development, validation, and frozen evaluation splits, where all splits are disjoint at the task-program level. Table 8 summarizes the benchmark statistics.

Table 8: Benchmark split statistics.

Split	Task Programs	Surface Instances
Development	8	24
Validation	8	24
Frozen evaluation	24	72
Total	40	120

Each task program contains two registered procedures, candidate contextual factors, and a binary applicability boundary determined by the revision-relevant factor. One contextual factor governs procedure applicability, while other factors serve as correlated distractors. Following Equation 2, each task contains four evolutionary observations: an initial successful execution, a failure after contextual evolution, a controlled successful execution, and a successful recovery using the revised procedure. This construction creates an attribution challenge because the failure observation alone cannot determine which contextual factor governs the applicability change. For downstream evaluation, unseen contexts are constructed by combining revision-relevant and distractor factor states. Boundary–distractor disagreement cases contain conflicting signals between these factors and therefore require applying the induced revision knowledge rather than relying on surface correlations. The ordering of contextual factors and the presentation of procedures as options A/B are randomized across surface realizations to prevent positional shortcuts. The semantic relationship between contextual states and procedures is determined by each task program rather than by fixed option positions, ensuring that mapping recovery requires identifying the applicability relation instead of exploiting state or option ordering.

B ADDITIONAL EXPERIMENTAL DETAILS

B.1 MODEL CONFIGURATION

All main experiments use the same frozen LLM backbone without parameter updates. The model is accessed through an OpenAI-compatible inference interface with deterministic decoding (temperature $= 0$). We use `qwen3.7-max` as the primary backbone model for the main evaluation. No external tools, retrieval modules, or search systems are used. The maximum generation length is set to 2048 tokens. All methods in the main evaluation share the same inference configuration and differ only in the provided experience representation and evidence factorization.

Additional backbone evaluations use GPT-5 and DeepSeek-V3 under the same task programs, prompts, output schema, and decoding configuration to examine the robustness of revision knowledge induction across different LLM families (Appendix C).

Structured Revision Knowledge Induction Template

`[Task Description]`
task description

`[Candidate Contextual Factors]`
c_1: factor description
c_2: factor description
...
c_m: factor description

`[Registered Procedures]`
Procedure $p^{(0)}$:
procedure description
Procedure $p^{(1)}$:
procedure description

`[Evolutionary Evidence]`
e_1: original procedure succeeds before evolution
e_2: original procedure fails after contextual evolution
e_3: original procedure succeeds under controlled context
e_4: revised procedure restores success under evolved context

`[Factorized Evidence Relations]`
For CEA, the evolutionary trajectory is represented through:

- **Validity relation:** captures the transition from successful execution to failure after contextual evolution.
- **Controlled relation:** captures evidence for attributing procedure applicability changes to a specific contextual factor while controlling correlated variations.
- **Recovery relation:** captures the procedure that restores successful execution under the evolved context.

`[Structured Revision Knowledge Representation Output]`
Revision Factor:
The index of the contextual factor governing procedure applicability.
State-to-Procedure Mapping:
The recovered mapping from contextual states to applicable procedures.
Explanation:
An auxiliary natural-language justification of the inferred applicability condition.

Figure 3: Generic structured induction template. CEA additionally provides the factorized evidence relations shown above.

B.2 Prompt Templates and Revision Knowledge Representation

The induction stage receives a structured representation containing the task description, candidate contextual factors, registered procedures, and evolutionary evidence. All structured induction methods share the same task information and output schema. For CEA, the evolutionary trajectory is further represented through complementary evidence relations corresponding to validity, controlled, and recovery analysis. The model then induces the revision-relevant contextual factor and the associated state-to-procedure mapping from this structured representation. Figure 3 illustrates the generic induction input template used in our experiments.

The induced revision knowledge representation is represented as a structured record containing the recovered applicability knowledge and an auxiliary explanation field. The representation components are summarized in Table 9.

Table 9: Structured revision knowledge representation. Evaluation focuses on the first two components; explanation is retained only as auxiliary information for interpretability.

Component	Description
Revision Factor	The identified contextual factor governing procedure applicability.
State-to-Procedure Mapping	The recovered mapping from the states of the revision-relevant factor to applicable procedures. The state identifiers are internal representations, and their semantic correspondence is defined by each task program.
Explanation	An auxiliary natural-language justification describing the inferred applicability condition from evolutionary evidence.

Revision-Aware Procedure Selection Template

`[Revision Knowledge Representation]`
induced revision knowledge representation
`[Current Context]`
query context
`[Procedure Options]`
Option A:
procedure description
Option B:
procedure description
`[Output Requirement]`
Output only:
A or B.

Figure 4: Generic template for downstream procedure selection.

Procedure Selection Input. For downstream procedure selection, the model receives the induced revision knowledge representation and a learner-visible query context. The two candidate procedures are presented as randomized options A and B, and the model outputs only the selected option. Figure 4 illustrates the generic selection input format.

The explanation component is retained as auxiliary information during downstream selection but excluded from revision knowledge recovery metrics, which evaluate only the recovered revision factor and state-to-procedure mapping.

B.3 Output Validation

Induction outputs are validated according to the predefined schema. Outputs with missing fields, invalid factor indices, or incomplete procedure mappings are treated as invalid induction results and cannot generate usable revision knowledge representations. Valid outputs are distinguished from invalid schema outputs to separate formatting failures from revision knowledge recovery errors.

For downstream selection, the model must output exactly one of the two presented procedure options (A or B); other outputs are treated as incorrect selections.

B.4 Valid Output Statistics

To distinguish schema validity from revision knowledge recovery quality, we report the valid-output rates of different induction methods. A valid output must satisfy the predefined structured output schema, including a valid revision factor and a complete state-to-procedure mapping. Schema validity only indicates that the output can be evaluated structurally and does not guarantee correct

Table 10: Valid revision knowledge induction rates.

Method	Valid Output Rate ↑
B2	0.9250
B3	0.9417
B4	0.9833
CEA	1.0000

Table 11: **Error decomposition of CEA revision knowledge recovery.** Proportions are computed over valid induction outputs and sum to one.

Outcome	Proportion
Correct factor and correct mapping	0.6271
Correct factor but incorrect mapping	0.3136
Incorrect factor	0.0593

revision knowledge recovery. Table 10 summarizes the proportion of valid induction outputs for each method.

B.5 Error Decomposition of Revision Knowledge Recovery

To further analyze the gap between revision factor identification and state-to-procedure mapping recovery, we decompose CEA's valid induction outputs into three mutually exclusive categories. The decomposition distinguishes failures in identifying the revision-relevant factor from failures in recovering the corresponding procedure mapping after the correct factor has been identified.

Most remaining recovery errors occur after the revision-relevant factor has been correctly identified: 0.3136 of valid outputs identify the correct factor but fail to recover the complete state-to-procedure mapping, whereas only 0.0593 fail at factor identification. This indicates that factor identification and mapping recovery represent different levels of difficulty.

C Backbone Generalization Analysis

To examine whether the effectiveness of contrastive evidence organization depends on a specific backbone model, we evaluate CEA and B4 using additional LLM backbones, including GPT-5 and DeepSeek-V3. We select these backbones because they represent different model families from the primary Qwen backbone, covering both proprietary and open-source LLMs. These additional evaluations examine whether the observed gains generalize beyond the primary backbone. All models use the same task programs, prompts, output schema, and deterministic decoding settings.

Table 12 shows that CEA consistently improves revision factor identification and state-to-procedure mapping recovery over the controlled B4 baseline across different backbone families. Although backbone models differ in their absolute recovery performance, the relative improvements of CEA over B4 remain consistent across models. These results suggest that the effectiveness of contrastive evidence organization is not limited to a single backbone model and remains robust across different LLM families.

D Real-World Case Validation

Motivation. ApplyBoundary is intentionally designed as a controlled diagnostic benchmark that isolates revision knowledge induction from open-ended agent interaction. To examine whether the revision patterns studied in the benchmark also arise in practical systems, we additionally construct a small set of validation cases from documented real-world platform changes. These cases are derived from publicly available documentation and changelogs rather than from synthetically specified environment transitions.

Table 12: **Revision knowledge recovery across different LLM backbones.** CEA is compared with the controlled B4 baseline under the same evaluation protocol. Δ denotes the improvement of CEA over B4.

Backbone	Revision Factor Identification			State-to-Procedure Mapping		
	B4	CEA	Δ	B4	CEA	Δ
Qwen3.7-Max	0.8750	0.9407	+0.0657	0.5333	0.6271	+0.0938
GPT-5	0.9000	0.9573	+0.0573	0.5874	0.6859	+0.0985
DeepSeek-V3	0.8581	0.9250	+0.0669	0.5167	0.6143	+0.0976

For each case, we identify (i) a procedure that was valid before a documented system change, (ii) a contextual factor whose state changed, and (iii) a revised procedure required under the evolved system. The revision-relevant factors are identified manually from the documented system changes before evaluation, following the formulation of the main benchmark. Therefore, this validation evaluates revision knowledge induction under provided candidate factors rather than open-ended factor discovery.

We then instantiate the same revision-induction formulation used in the main experiments and evaluate whether CEA recovers the revision-relevant contextual factor and the corresponding state-to-procedure mapping. This evaluation is intended as a small-scale validation of the applicability of our formulation to naturally occurring system evolution, rather than as a comprehensive real-world agent benchmark.

D.1 Case Construction

GitHub authentication policy migration. GitHub discontinued account-password authentication for authenticated Git operations on GitHub.com and required token-based authentication or SSH instead (GitHub, 2020). This change provides a natural instance of procedure revision under an evolving interaction constraint. Before the policy change, password-based authentication constituted a valid procedure for authenticated Git operations. After the change, the same procedure was no longer valid, while authentication using a personal access token or SSH remained supported.

We identify the revision-relevant contextual factor as *authentication policy*. The original state corresponds to a policy under which password authentication is accepted, whereas the evolved state corresponds to a policy requiring token- or key-based authentication. The two registered procedures are: p_0 = password-based authentication, p_1 = token/SSH-based authentication. The target revision knowledge associates the original authentication-policy state with p_0 and the evolved state with p_1.

Slack file-upload API migration. Slack announced the retirement of the legacy `files.upload` Web API method and instructed applications to migrate to a sequenced upload procedure using `files.getUploadURLExternal` followed by `files.completeUploadExternal` (Slack, 2024). This case represents API evolution in which the underlying task of uploading a file to Slack remains unchanged while the valid interaction procedure changes.

We identify the revision-relevant contextual factor as *file-upload API interface*. The original state corresponds to an interface supporting the legacy single-call upload method, while the evolved state corresponds to the interface requiring the new sequenced upload mechanism. The registered procedures are: p_0 = `files.upload`, and p_1 = `files.getUploadURLExternal` $\rightarrow$ `files.completeUploadExternal`. The target revision knowledge therefore maps the legacy interface state to p_0 and the evolved interface state to p_1.

Chrome extension platform migration. Chrome's extension platform introduced Manifest V3 as the successor to Manifest V2, with migration requiring changes to both extension manifests and execution mechanisms (Google Chrome, a;b). Among these changes, Manifest V3 replaces background pages with extension service workers and requires corresponding modifications to background exe-

Table 13: **Documented real-world revision cases.** Each case is derived from an externally documented platform change and instantiated using the revision-knowledge formulation in the main paper.

Case	Domain	Revision-Relevant Factor	Procedure Revision
GitHub authentication	Interaction Policy	Authentication policy	Password → token/SSH
Slack file upload	Tool/API Usage	Upload API interface	Legacy upload → sequenced upload
Chrome Manifest migration	Workflow Adaptation	Platform version	Background page → service worker

Table 14: **CEA on documented real-world revision cases.** Factor recovery evaluates whether CEA identifies the documented contextual factor governing procedure applicability, while mapping recovery evaluates whether it recovers the corresponding state-to-procedure assignment.

Case	Factor Recovery	Mapping Recovery
GitHub authentication	Correct	Correct
Slack file upload	Correct	Correct
Chrome Manifest migration	Correct	Incorrect

cution logic. This provides a real system-evolution case in which a previously valid implementation workflow must be revised to remain compatible with an updated platform.

We identify the revision-relevant contextual factor as *extension platform version*. The original state corresponds to Manifest V2, while the evolved state corresponds to Manifest V3. For the background execution component, the registered procedures are: p_0 = background-page-based execution, p_1 = service-worker-based execution. The target revision knowledge maps the Manifest V2 state to p_0 and the Manifest V3 state to p_1.

D.2 Evaluation Protocol

For each real-world case, we construct an evolutionary trajectory following the same revision pattern used in the main benchmark. The initial observation demonstrates successful execution of the original procedure under the original system state. A subsequent observation represents the documented system change under which the original procedure becomes invalid or unsupported. Additional contextual factors are included as distractors so that the model must attribute the procedure change to the documented revision-relevant factor rather than simply reproduce all observed changes. Finally, the documented replacement procedure provides recovery evidence under the evolved system state.

CEA receives the task description, candidate contextual factors, registered procedures, and evolutionary evidence in the same structured format used in the main experiments. We evaluate two components of revision knowledge: whether CEA identifies the documented revision-relevant factor and whether it recovers the corresponding state-to-procedure mapping.

Discussion. CEA correctly identifies the documented revision-relevant factor in all three cases and recovers the corresponding state-to-procedure mapping in two of the three cases. The remaining error occurs in the Chrome Manifest migration case, where the platform transition involves multiple interacting implementation changes rather than a single isolated procedural revision.

These results provide preliminary evidence that the revision patterns studied in ApplyBoundary also appear in documented real-world system evolution. At the same time, the cases remain a small-scale validation rather than a comprehensive real-world benchmark, and the mapping error highlights the additional difficulty of recovering procedure assignments under complex system changes involving multiple coupled modifications.