When Not to Imitate: Boundary-Aware Skill Memory for Reliable Tool-Use LLM Agents

Zihan Lin[1,2,3*] **Zhenyu Chen**[3,4*] **Jiawen Wei**[1*] **Xiaohan Wang**[1†] **Jie Cao**[3]
Jiajun Chai[1] **Wei Lin**[1] **Guojun Yin**[1] **Ran He**[2,3†]

[1] Meituan
[2] School of Advanced Interdisciplinary Sciences, University of Chinese Academy of Sciences
[3] MAIS&NLPR, Institute of Automation, Chinese Academy of Sciences
[4] Zhongguancun Academy

[*]Equal contribution. [†]Corresponding authors.

wangxiaohan17@meituan.com ran.he@ia.ac.cn

Abstract

Extracting skills from past successes is critical for the efficient evolution of Large Language Model (LLM) agents. Prevailing agent self-evolution paradigms typically rely on a core assumption: equipping LLMs with skill memories derived from successful trajectories will monotonically improve their problem-solving capabilities. However, probe analyses reveal that extracting skills solely from successful trajectories traps the model in a **Skill Imitation Trap**. For tasks that resemble past successes but require different tools, retrieving more skills paradoxically increases the model's confidence in wrong tool calls—procedure skills raise the wrong-tool margin by 47% over a memory-free baseline. To overcome this limitation, we propose **Boundary-Aware Skill Memory** (BASM), which augments each skill with explicit boundary fields—applicability conditions, risk cues, avoidance rules, and recovery notes. These fields transform each retrieved skill from an unconditional action template into state-conditioned guidance: the agent applies the skill when its conditions hold, suppresses inapplicable tool calls when they do not, and issues targeted repairs when execution fails. Across three agent benchmarks and four model scales, BASM consistently outperforms success-distilled skill-memory baselines: it improves task success rate by up to 23.8% on AppWorld, accuracy by up to 5.0% on BFCL, and reduces attack success rate by 4.6% on AgentDojo, while simultaneously reducing average AppWorld steps by up to 6.6% relative to the memory-free baseline.

1 Introduction

Large language model (LLM) agents are increasingly deployed in complex, interactive environments that demand multi-step tool use, API orchestration, and long-horizon planning (Yao et al., 2022; Lin et al., 2026a; Xi et al., 2025; Lin et al., 2026b; Lu et al., 2026a; Chen et al., 2025). A central challenge in building such agents is enabling them to improve over time without constant human supervision. Skill acquisition, which enables agents to extract, store, and reuse procedural knowledge from past experience, has emerged as a critical mechanism for agent self-evolution (Wang et al., 2023; Zhu et al., 2023; Zhao et al., 2024; Lin et al., 2025). By distilling successful trajectories into reusable skill memories, agents can avoid redundant exploration and transfer solutions across structurally similar tasks (Wang et al., 2026c; Mi et al., 2026).

Existing approaches have explored how skills are represented, acquired, retrieved, and evolved across a range of settings. Early agents such as Voyager and ExpeL compressed successful trajectories into reusable programs or verbal lessons (Wang et al., 2023; Zhao et al., 2024); recent work has extended this to online skill induction, RL-driven distillation, and cross-agent library transfer (Wang et al., 2025; Tang et al., 2025; Lu et al., 2026b; Wang et al., 2026a,b). Despite their varied perspectives, these approaches largely treat success-distilled skills as positive transfer: once retrieved as relevant, a skill is presumed useful, while its validity boundary remains implicit.

This omission matters when semantic relevance diverges from decision validity: a retrieved skill appears relevant but lacks boundary cues for whether its action pattern still applies. By probing model behavior at varying retrieval depths, we identify a failure mode we term the **Skill Imitation Trap**

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 24166–24182
October 24-29, 2026 ©2026 Association for Computational Linguistics

(Figure 2): the model's preference for a semantically similar but inapplicable tool increases as more success-distilled skills are retrieved, causing skill memory to underperform a memory-free baseline in inapplicable states. Mechanistically, attention analysis pinpoints the cause: when a retrieved skill matches the current query, success-distilled memory shifts the model's attention toward procedural content rather than boundary cues, treating the retrieved skill as an unconditional action template. This teaches the model to focus on how a past task was solved, with no mechanism to assess whether the present state actually meets the conditions for that solution.

To overcome the Skill Imitation Trap, we propose **Boundary-Aware Skill Memory** (BASM), a skill-memory method that augments procedural knowledge with explicit boundary conditions. Figure 1 illustrates how BASM transports boundary evidence from offline extraction to online retrieval and decision-time checking. Unlike success-distilled skill memory, BASM treats a retrieved skill as state-contingent evidence rather than an unconditional action template, enabling the agent to apply the skill when its assumptions hold, suppress risky tool calls when they do not, and repair local failures when execution deviates. Probe analyses confirm that boundary fields function as a state-conditioned attention modulator: the model consults boundary content more in risky and repair states than in safe ones. Moreover, same-prompt knockout raises the wrong-tool margin by $+3.45$ logits and shifts 69.4% of risky-state samples toward the success-only baseline, confirming that boundary fields are a causal anti-imitation channel. Across BFCL, AppWorld, and AgentDojo on four model scales, BASM consistently outperforms success-distilled skill memory baselines: it improves task success rate by up to 23.8% on AppWorld and accuracy by up to 5.0% on BFCL; it reduces attack success rate by 4.6% on AgentDojo while reducing average AppWorld steps by up to 6.6% relative to the memory-free baseline; and it achieves Pareto improvements in both task utility and safety on Qwen3-14B.

Our contributions are as follows:

1. **Skill Imitation Trap probe analyses.** We identify and characterize the Skill Imitation Trap through mechanism-level probe analyses. Using wrong-tool margin measurements, attention attribution, and same-prompt knockout, we show that success-distilled skill memory amplifies wrong-tool confidence in risky states (wrong-tool margin increases by 47% relative to memory-free baseline) and routes its imitation effect through procedural attention at decision tokens. Critically, ablating boundary-span attention raises the wrong-tool margin by $+3.45$ logits and shifts 69.4% of risky-state samples toward the success-only baseline, confirming that boundary fields serve as a causal anti-imitation channel.

2. **Boundary-Aware Skill Memory design.** We propose BASM, a seven-slot boundary-aware skill schema—goal, procedure, tools, applicability conditions, risk cues, avoidance rules, and recovery notes—together with a budgeted retrieval-and-formatting pipeline, a boundary checker for prompt-side enforcement, and a runtime repair module for stateful environments. This design transforms retrieved skills from unconditional action templates into boundary-aware records that guide when a skill should be applied, suppressed, or locally repaired.

3. **Broad experimental validation.** We evaluate BASM across three complementary benchmarks (BFCL, AppWorld, AgentDojo) and four model scales spanning 8B to 397B parameters. BASM consistently improves over success-distilled skill-memory baselines: AppWorld success rate improves by up to 23.8%; BFCL accuracy improves by up to 5.0%; AgentDojo attack success rate drops by 4.6%; average AppWorld steps fall by up to 6.6% relative to the memory-free baseline; and Qwen3-14B achieves Pareto improvements in both utility and safety. Attention, logit, and knockout probes provide a coherent mechanistic account of these gains.

2 Related Work

Self-Evolution Agents. LLM tool use has evolved from early demonstrations—the Reason-Act-Observe loop of (Yao et al., 2022), self-supervised API invocation in Toolformer (Schick et al., 2023), and large-scale API mastery in ToolLLM (Qin et al., 2024)—to a central paradigm in agent research, yet these approaches treat each task independently without accumulating reusable knowledge. Skill memory emerged to address this limitation. (Wang et al., 2023) and (Zhu et al., 2023) pioneered skill libraries that store success-

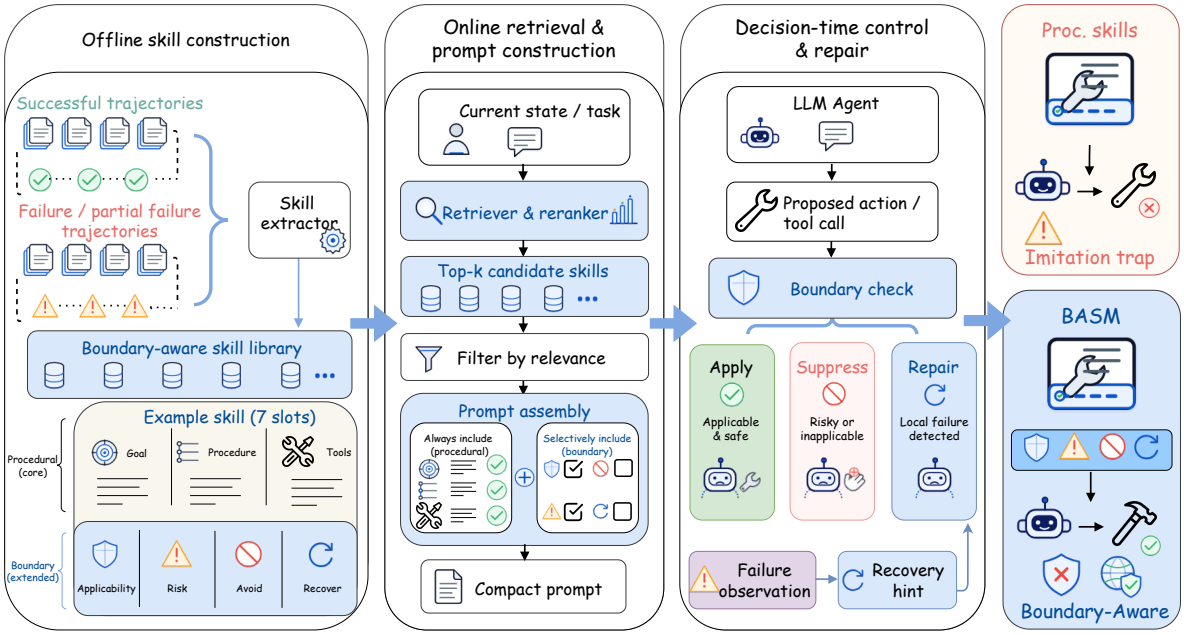

Figure 1: **Overview of BASM.** BASM builds a boundary-aware skill library from logged trajectories, storing each skill as a procedural core—goal, procedure, and tools—with boundary fields covering applicability, risk, avoidance, and recovery. At inference, BASM retrieves candidate skills, selectively exposes boundary evidence under a fixed token budget, and checks proposed tool calls to decide whether to apply, suppress, or repair. This transforms skill reuse from blind imitation into validity-scoped transfer, suppressing wrong-tool preference while preserving efficient reuse and local recovery.

ful behaviors as executable code or text-based knowledge for compositional reuse. (Zhao et al., 2024) and (Wang et al., 2024) distilled experience into natural-language insights and reusable workflows without parameter updates. (Shinn et al., 2023) pursued episode-level verbal reflection rather than a persistent skill library. More recent work has extended the paradigm along orthogonal axes: program-based online induction (Wang et al., 2025), hierarchical skill knowledge bases (Wang et al., 2026a), RL-driven skill internalization (Lu et al., 2026b), and cross-framework memory sharing (Tang et al., 2025). Despite their varied perspectives, all these approaches share a common premise: skills distilled from successful experience, when retrieved as relevant, should benefit the current decision. This positive-transfer assumption leaves the boundary of a skill under-specified, and can break down in practice—recent work has noted that retrieved skills sometimes cause models to over-imitate procedures rather than adapt them to novel tasks (Wang et al., 2026a), an empirical signal consistent with the Skill Imitation Trap we identify through mechanism-level analysis in Section 3.

Skill Retrieval and Applicability. BASM's retrieval module builds on RAG (Lewis et al., 2020; Jeong et al., 2024) and demonstration retrieval for in-context learning (Rubin et al., 2022; Lu et al., 2022). Standard retrieval scores measure relevance, not applicability: neither retrieved documents nor procedurally similar skills carry an explicit signal for when retrieved content should be suppressed (Yoran et al., 2023; Asai et al., 2024). BASM addresses this by augmenting each skill with boundary fields and incorporating a boundary relevance score into reranking and prompt formatting, so that retrieved skills function as validity-scoped records rather than unconditional action templates. The idea of conditioning skill execution on explicit preconditions has a precedent in reinforcement learning: (Konidaris et al., 2014) formalized skill applicability via initiation sets within the options framework (Sutton et al., 1999). BASM transplants this boundary-conditioning idea to LLM agents, replacing hand-crafted symbolic predicates with natural-language fields extracted from agent trajectories and enforced at retrieval, formatting, and execution time.

3 Probe Analysis

We use probe analysis to establish two claims: success-distilled skill memory can increase wrong-tool preference in inapplicable states, and BASM counteracts this effect through boundary-aware attention at the decision token. The analysis first diagnoses how retrieval depth affects wrong-tool preference, then shows how boundary evidence is read, how it suppresses wrong-tool logits, and how attention knockout tests the causal role of the boundary path.

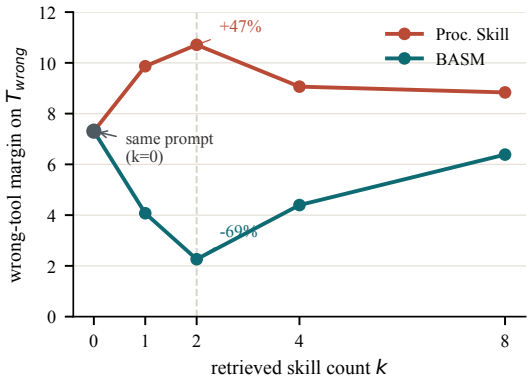

Figure 2: Success-distilled skill memory (**Proc. Skill**) amplifies wrong-tool preference with retrieval depth; BASM reverses this trend.

3.1 Diagnostic Setup

Let x_t denote the prompt prefix immediately before the agent generates its next action, and let d_t be the first generated structural token of that action. In BFCL, d_t is the first token of the function name; in stateful environments, it is the first content token of the next code, tool, or action step. We partition decision points into three task-state buckets: T_{ok} (the retrieved skill matches the correct tool and should be applied), T_{wrong} (a semantically similar skill is retrieved but is inapplicable and should be suppressed), and T_{repair} (a local failure has occurred and the agent should repair rather than repeat).

The central diagnostic is the wrong-tool margin. Let $\mathcal{Y}_{\text{wrong}}$ be the first tokens of semantically similar but incorrect function names, and let $\mathcal{Y}_{\text{ctrl}}$ contain the first tokens of the correct function and abstention tokens. We define

$$m_{\text{wrong}}(x_t) = \max_{y \in \mathcal{Y}_{\text{wrong}}} \ell_t(y) - \max_{y \in \mathcal{Y}_{\text{ctrl}}} \ell_t(y). \quad (1)$$

A positive value means that the model locally prefers an incorrect tool over the correct or abstention alternatives. We measure this margin at d_t, before environment feedback can intervene. This logit-level diagnostic exposes the model's immediate tool preference, even when later repair steps might correct the final action.

3.2 Retrieval Trap

Figure 2 shows that retrieving more success-distilled skills can amplify, rather than correct, risky tool choices. At $k = 0$, both memory configurations produce the same prompt, so their wrong-tool margins coincide at 7.31. Once retrieved skills are added, the curves diverge: success-distilled skill memory rises to 10.71 at $k = 2$, whereas BASM falls to 2.26. This reversal shows that additional procedural evidence can strengthen preference for a semantically similar but invalid tool, challenging the monotonicity assumption behind success-distilled skill memory (Wang et al., 2023; Zhao et al., 2024). The design implication is direct: a skill should encode not only how to execute a procedure, but also when that procedure should not be used.

3.3 Mechanism Analysis

Boundary attention. We first test whether the model reads boundary fields at the decision point. For a retrieved skill, let P be the token span of its procedure field and B be the union of its boundary fields: applicability conditions, risk cues, avoidance rules, and recovery notes. For attention layer l and head h, we compute the length-normalized attention from the decision token to a span X:

$$\alpha_X^{(l,h)}(x_t) = \frac{1}{|X|} \sum_{i \in X} A_{d_t,i}^{(l,h)}, \qquad \eta^{(l,h)}(x_t) = \frac{\alpha_B^{(l,h)}(x_t)}{\alpha_P^{(l,h)}(x_t) + \epsilon}. \quad (2)$$

The ratio η measures how much the decision token attends to boundary evidence relative to procedural content. The first panel of Figure 3 shows a state-dependent pattern: under BASM, the ratio is low in T_{ok} (0.15), rises in T_{wrong} (1.23), and peaks in T_{repair} (1.81). Same-length controls with neutral padding or extra procedure text do not reproduce this pattern. Thus, the shift is not explained by prompt length or formatting; it reflects the model's use of boundary evidence when the retrieved procedure becomes risky or requires repair.

Wrong-tool suppression. The second panel of Figure 3 links this attention pattern to the action distribution. Its y-axis reports wrong-tool preference

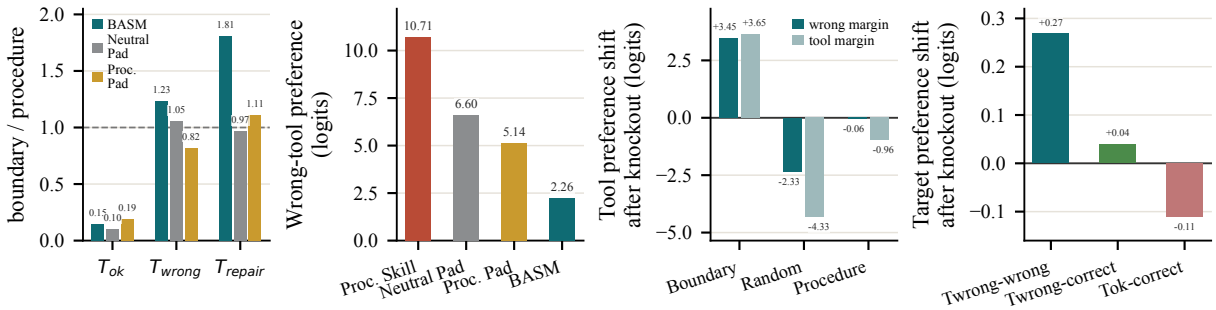

Figure 3: **Boundary fields suppress wrong-tool preference through a state-conditioned attention mechanism.** BASM reduces the wrong-tool margin from 10.71 to 2.26, a suppression that padding controls cannot explain. Knocking out boundary-span attention restores the imitation-trap direction. T_{ok}: applicable state; T_{wrong}: inapplicable state; T_{repair}: post-failure state

in logits: larger values mean stronger preference for a semantically similar but invalid tool. Proc. Skill yields a high wrong-tool preference of 10.71 on T_{wrong} examples. Same-length controls reduce it to 6.60 and 5.14, showing that additional context can partially soften the action distribution. BASM lowers the value further to 2.26, well below both controls. This gap shows that boundary evidence provides a suppression signal beyond length and formatting, allowing the model to reject invalid procedural imitation while retaining the retrieved skill as contextual evidence.

3.4 Attention Knockout

The third and fourth panels of Figure 3 test whether boundary attention is causally responsible for this suppression. We keep the prompt tokens fixed and ablate only the attention from the decision token d_t to the boundary span B. All bars report the signed shift after knockout, $\Delta m = m_{\text{knockout}} - m_{\text{BASM}}$, so positive values indicate that removing an attention path increases the corresponding target preference. On Qwen3-8B T_{wrong} examples, boundary knockout raises wrong-tool preference by $+3.45$ logits and the broader tool preference by $+3.65$ logits. Random same-length and procedure-span knockouts do not produce the same shift, showing that the effect is specific to boundary access rather than a generic attention perturbation. The fourth panel repeats the diagnostic on Qwen3-14B: T_{wrong}–wrong measures invalid-tool preference in inapplicable states, T_{wrong}–correct measures correct-tool preference in the same states, and T_{ok}–correct measures correct-tool preference when the retrieved skill is applicable. Boundary knockout selectively increases T_{wrong}–wrong ($+0.27$), leaves T_{wrong}–correct nearly unchanged ($+0.04$), and slightly decreases T_{ok}–correct (-0.11). This pattern shows that BASM is not imposing a general penalty on tool use; instead, boundary attention specifically suppresses invalid imitation while largely preserving preference for correct actions. Extended evidence is reported in Appendix B.

Table 1: **BFCL results on Qwen3-32B and Qwen3.5-397B-A17B.** We report accuracy across four multi-turn subsets. Best in **bolded** and second are underlined.

Method	Base	Long Context	Miss Func	Miss Param	Overall
Qwen3-32B					
Base	52.0	41.5	45.5	35.0	43.50
Proc. Skill	**60.5**	41.5	49.0	36.0	46.75
App. Skill	60.0	41.5	47.0	**37.0**	46.38
BASM	57.5	**42.5**	**51.0**	36.5	**46.88**
Qwen3.5-397B-A17B					
Base	62.0	51.5	50.0	**44.5**	52.00
Proc. Skill	**64.0**	**56.5**	53.0	40.0	53.38
App. Skill	63.0	55.0	50.0	38.5	51.63
BASM	**64.0**	55.5	51.5	44.0	**53.75**

Table 2: **AppWorld results on Qwen3-32B and Qwen3.5-397B-A17B.** BASM achieves the highest AppWorld task success rate among the tested methods.

Model	Method	Success	Avg. Steps
Qwen3-32B	Base	62.50	12.46
	Proc. Skill	66.07	**11.61**
	App. Skill	60.12	12.03
	BASM	**76.19**	11.71
Qwen3.5-397B-A17B	Base	35.12	13.50
	Proc. Skill	52.98	12.64
	App. Skill	52.98	**12.61**
	BASM	**54.76**	**12.61**

4 Method

BASM is a boundary-aware skill-memory pipeline for tool-use agents. Given an offline set of logged trajectories, it constructs a boundary-aware skill library, retrieves relevant skills for a new decision state, formats them under a fixed context budget,

and optionally performs local repair in stateful environments.

4.1 Skill Schema

Formally, let $\mathcal{D}$ be a skill library and let $\rho_{\mathcal{D}}$ be a retrieval-and-formatting operator that maps a task state x_t to an in-context memory block. A skill-conditioned agent samples the next action from

$$a_t \sim \pi_\theta(\cdot \mid x_t, \rho_{\mathcal{D}}(x_t)). \tag{3}$$

Proc. Skill improves the procedural information in $\rho_{\mathcal{D}}$, but does not constrain when the retrieved procedure should be applied. BASM changes the library object and the formatting operator so that the prompt exposes an explicit boundary signal alongside procedural content. The boundary signal is realized through a structured seven-slot skill schema. BASM represents each reusable behavior as a seven-slot boundary-aware skill:

$$s = (g,\, \pi,\, \mathcal{T},\, \mathcal{A},\, \mathcal{R},\, \mathcal{N},\, \mathcal{F}), \tag{4}$$

where g is the skill goal, π is the procedural specification, $\mathcal{T}$ is the tool set, $\mathcal{A}$ denotes applicability conditions, $\mathcal{R}$ denotes risk cues, $\mathcal{N}$ denotes avoidance rules, and $\mathcal{F}$ denotes recovery notes. The first three fields constitute the procedural component, while the last four collectively form the boundary component $B(s)$. We define the boundary text of a skill as

$$B(s) = \mathcal{A}(s) \oplus \mathcal{R}(s) \oplus \mathcal{N}(s) \oplus \mathcal{F}(s), \tag{5}$$

where $\oplus$ denotes ordered text composition. This decomposition reflects the distinct functional roles of the two components: π enables the agent to execute a known procedural pattern, whereas $B(s)$ provides the contextual evidence necessary to determine whether that pattern is applicable.

Each boundary slot serves a distinct role: applicability conditions specify when the procedure is safe to invoke; risk cues identify observations that render a semantically similar skill unsafe; avoidance rules enumerate actions or parameters to suppress under particular states; and recovery notes prescribe one-step corrections after local failures. Together, these fields give each retrieved skill an explicit validity scope, which the pipeline enforces at retrieval, formatting, and execution time (§4.2).

4.2 Method Design

Skill library construction. BASM constructs the skill library offline from logged trajectories. Let $\mathcal{K}^+$ be the set of successful trajectories and $\mathcal{K}^-$ be failed or partially failed trajectories, and let $\mathcal{U}$ be the set of all extraction anchors derived from these trajectories. Extraction is anchored at two granularities: step-level extraction derives functional skills from plan steps, while tool-level extraction derives atomic skills from concrete tool invocations and their associated observations. For an anchor u, the extractor receives a successful context τ_u^+ and, when available, a nearby failure context τ_u^-, then emits a candidate skill

$$\hat{s}_u = f_{\text{ext}}(u,\, \tau_u^+,\, \tau_u^-). \tag{6}$$

The successful context supplies $g, \pi, \mathcal{T}$; the failure context is used only as boundary evidence for $\mathcal{R}, \mathcal{N}, \mathcal{F}$.

The raw extractor output is passed through a deterministic validation operator ψ. This operator enforces strict JSON parsing, required fields, list-valued boundary slots, non-empty risk cues and avoidance rules, and tool-schema validity. Candidate skills that fail validation after retry are discarded, ensuring that library construction degrades gracefully rather than halting on malformed extractions. The resulting library is

$$\mathcal{D}_{\text{BASM}} = \left\{\, \psi(\hat{s}_u) \mid u \in \mathcal{U},\ \psi(\hat{s}_u) \neq \varnothing \,\right\}. \tag{7}$$

This precision-first construction prioritizes boundary specificity over recall. Appendix A reports the fixed retrieval settings, benchmark-specific budgets, and retained-library statistics needed to reproduce the pipeline.

Retrieval and prompt formatting. At inference time, BASM retrieves skills with a multi-view score that combines semantic relevance, tool overlap, and step/goal overlap. For decision state x_t and skill s, let $e(\cdot)$ be a text embedding function, v_s be the concatenation of the goal and procedure fields of s, $\mathcal{T}_{x_t}$ be the set of tools referenced in the current state, and $J(\cdot,\cdot)$ be Jaccard overlap over tool names. The retrieval score is

$$\begin{aligned}\text{score}(x_t, s) = {} & \lambda_{\text{sem}} \cos\big(e(x_t), e(v_s)\big) \\ & + \lambda_{\text{tool}}\, J(\mathcal{T}_{x_t}, \mathcal{T}_s) \\ & + \lambda_{\text{step}}\, \text{overlap}(x_t, g_s, \pi_s),\end{aligned} \tag{8}$$

where g_s and π_s denote the goal and procedure fields of skill s, and $\lambda_{\text{sem}}, \lambda_{\text{tool}}, \lambda_{\text{step}}$ are weighting coefficients. The top-k retrieved set is

$$\mathcal{S}_k(x_t) = \text{TopK}_{s \in \mathcal{D}_{\text{BASM}}}\, \text{score}(x_t, s). \tag{9}$$

The prompt formatter operates under a fixed token budget, motivated by the probe finding (Figure 2) that indiscriminate retrieval amplifies rather than mitigates wrong-tool preference. For each retrieved skill, the procedural component $g, \pi, \mathcal{T}$ is always exposed. Let $r_B(x_t, s)$ denote the boundary relevance score measuring the overlap between the current state x_t and the boundary fields of skill s. Boundary fields are exposed only when $r_B(x_t, s)$ exceeds a threshold γ. The formatted memory block is

$$\rho_{\mathcal{D}}(x_t) = \mathrm{Proc}(\mathcal{S}_k) \oplus \mathrm{Compress}_{\beta}\Big(\{B(s) \mid s \in \mathcal{S}_k,\, r_B(x_t, s) > \gamma\}\Big), \tag{10}$$

where β is the maximum boundary-token budget.

Boundary-conditioned execution. BASM separates prompt-side guidance from runtime control. The boundary checker compares the proposed tool call a_t against retrieved boundary fields, blocking or revising calls that violate applicability, match an avoidance rule, or repeat a known failed pattern. Let $\mathcal{S}_k^*(x_t, a_t)$ denote the subset of retrieved skills whose tools or goals match a_t.

$$G(x_t, a_t, \mathcal{S}_k) = \prod_{s \in \mathcal{S}_k^*(x_t, a_t)} \mathbb{I}\big[a_t \not\models \mathcal{N}(s) \;\wedge\; x_t \models \mathcal{A}(s)\big], \tag{11}$$

where $\mathbb{I}[\cdot]$ is the indicator function. In practice, G is evaluated via deterministic string/schema checks first, with LLM-assisted judgment reserved for ambiguous natural-language conditions.

Local failure repair. Recovery notes $\mathcal{F}$ deliver their primary benefit through a runtime loop in stateful environments. In single-turn settings, $\mathcal{F}$ steers tool selection at inference time; no execution loop exists in which a repair action can be issued. In multi-turn stateful settings, $\mathcal{F}$ additionally drives a repair loop: when observation o_{t+1} signals a local failure, the repair module retrieves a targeted hint

$$h_t = \mathrm{Repair}(x_t,\, a_t,\, o_{t+1},\, \mathcal{S}_k), \tag{12}$$

and injects it into the next decision step in place of full skill re-retrieval.

5 Experiments

5.1 Setup

Benchmarks. We evaluate on three benchmarks covering complementary capability tiers. **BFCL v3** (Patil et al., 2025) tests multi-turn function calling (800 instances across *base*, *long_context*, *miss_func*, and *miss_param* subsets); the miss_func and miss_param subsets directly instantiate the Skill Imitation Trap by requiring the agent to reject semantically similar but inapplicable tool calls. **AppWorld** (Trivedi et al., 2024) evaluates stateful multi-step task completion via Python-style API calls in a persistent app environment (168 *test_normal* tasks); it is the primary testbed for BASM's runtime repair loop. **AgentDojo v1.2.2** (Debenedetti et al., 2024) measures the utility–safety tradeoff under *tool_knowledge* prompt-injection attacks across four task suites, reporting Utility (↑) and Attack Success Rate (↓).

Table 3: **Results on Qwen3 variants.** BASM turns skill memory from procedural reuse into controlled transfer.

Benchmark	Base	Proc. Skill	App. Skill	BASM
Qwen3-8B				
BFCL Acc. (↑)	34.13	33.88	35.25	**38.88**
AppWorld SR (↑)	29.76	33.33	33.33	**36.31**
AgentDojo Utility (↑)	**40.99**	31.51	31.61	35.93
AgentDojo ASR (↓)	8.01	10.01	10.54	**5.90**
Qwen3-14B				
BFCL Acc. (↑)	39.88	37.38	41.00	**42.50**
AppWorld SR (↑)	33.33	46.43	53.57	**57.14**
AgentDojo Utility (↑)	29.93	31.40	31.61	**33.51**
AgentDojo ASR (↓)	3.79	6.01	6.11	**2.95**

Table 4: **Component ablation on Qwen3-8B.** *Base*, *Proc. Skill*, *App. Skill*, and *BASM* follow Table 3. *Prompt-BASM* uses BASM-style boundary prompting without the runtime repair loop; *Repair-Only* uses the runtime repair loop without boundary-enriched prompts; *BASM − Avoid* removes avoidance rules; *BASM − Recovery* removes recovery notes.

Mode	BFCL (↑)	AppWorld (↑)	Utility (↑)	ASR (↓)
Base	34.13	29.76	**40.99**	8.01
Proc. Skill	33.88	33.33	31.51	10.01
App. Skill	35.25	33.33	31.61	10.54
Prompt-BASM	38.38	29.17	31.93	10.54
Repair-Only	34.13	25.60	30.87	7.17
BASM	**38.88**	**36.31**	35.93	**5.90**
BASM − Avoid	38.00	**36.31**	30.87	6.95
BASM − Recovery	37.00	33.93	31.82	6.85

Models and baselines. We evaluate **Qwen3-8B**, **Qwen3-14B**, **Qwen3-32B**, and **Qwen3.5-397B-A17B** (Yang et al., 2025) as the agent backbones. Thinking mode is disabled for the controlled Qwen3-8B/14B/32B comparisons and all mechanism probes, while the Qwen3.5-397B-A17B evaluation uses thinking enabled. We set temperature to 0.001. Within each benchmark, all configurations share the same retrieval substrate and evaluation pipeline. Full implementation settings are reported

in Appendix A.

We compare four memory configurations. **Base** is the memory-free agent. **Proc. Skill** retrieves the goal, procedure, and tools of successful past trajectories, instantiating the procedural-memory view of skills as reusable execution knowledge distilled from experience (Wang et al., 2023; Zhao et al., 2024; Wang et al., 2025, 2026a). **App. Skill** augments Proc. Skill with applicability conditions only, isolating the classical idea that a reusable skill should specify when it can be applied, as in action preconditions and option initiation sets (Konidaris et al., 2014). **BASM** is the full boundary-aware schema with applicability conditions, risk cues, avoidance rules, and recovery notes.

5.2 Main Results

Table 3 supports three aggregate conclusions. First, the Skill Imitation Trap is visible at benchmark scale: Proc. Skill underperforms Base on BFCL for both Qwen3-8B and Qwen3-14B, confirming that procedural memory alone can amplify wrong-tool preference in inapplicable states. Second, BASM consistently overcomes this trap, outperforming the strongest skill-memory baselines on BFCL and AppWorld across both model sizes. Third, AgentDojo reveals a capacity-dependent safety–utility pattern: on Qwen3-8B, BASM achieves the largest ASR reduction among memory methods while incurring only a minor utility cost; on Qwen3-14B, it achieves Pareto improvements, yielding both the highest utility and the lowest ASR among all memory configurations.

Tables 1 and 2 extend these results to larger models. On Qwen3-32B, BASM achieves the best aggregate BFCL accuracy, with its largest margin on Miss Func (the subset most directly instantiating the Skill Imitation Trap); on AppWorld it reaches 76.19% success, improving over Proc. Skill (66.07%), while reducing average steps by 6.0% relative to Base (12.46 to 11.71). On Qwen3.5-397B-A17B, the ranking is preserved: BASM reaches 53.75 on BFCL versus 53.38 for Proc. Skill, and 54.76% AppWorld success versus 52.98%, with average steps reduced by 6.6% relative to Base (13.50 to 12.61). These lower average-step counts relative to Base indicate that BASM's gains do not require longer interaction trajectories: boundary fields make retrieved skills more targeted, and the repair module supports local recovery when execution deviates—consistent with the mechanism established in Section 3.

5.3 Ablation Study

Prompt-BASM nearly matches BASM on BFCL, so boundary-conditioned prompting explains most of the function-calling gain; Repair-Only stays at Base on BFCL (34.13%) and drops below it on AppWorld (25.60% vs. 29.76%), showing that runtime repair without boundary-conditioned prompts provides no benefit on single-step tasks and can actively disrupt multi-step execution without proper applicability guidance. On AppWorld, however, the ranking flips: Prompt-BASM falls below Base (29.17% vs. 29.76%), while BASM reaches 36.31%. The two components are therefore complementary: boundary prompting sharpens tool-choice precision, and runtime repair carries benefit in stateful multi-step recovery.

Removing avoidance rules lowers BFCL by 0.88 points and leaves AppWorld unchanged, but it also reduces AgentDojo utility from 35.93 to 30.87 and raises ASR from 5.90 to 6.95. This indicates that avoidance rules contribute not only to function-selection precision but also to safety-sensitive suppression in adversarial tool-use settings. Removing recovery notes has a broader performance effect: BFCL falls by 1.88 points and AppWorld by 2.38 points; AgentDojo utility decreases to 31.82, and ASR rises to 6.85. Recovery notes therefore influence prompt-side tool-choice logits and stateful recovery behavior, consistent with the probe findings in Section 3.3. In contrast, adding applicability conditions alone (App. Skill) improves BFCL over Proc. Skill by 1.37 points but raises ASR by 0.53 points, suggesting that partial boundary specification can inadvertently increase susceptibility to injected instructions that satisfy applicability criteria on the surface.

6 Conclusion

This work reveals a failure mode in skill-memory agents that goes beyond procedural knowledge gaps: without explicit validity boundaries, retrieved skills can amplify wrong-tool preference rather than suppress it. BASM closes this gap by equipping each skill with explicit boundary fields—covering applicability, risk, avoidance, and recovery—enforced at retrieval, prompt formatting, and execution to transform skill reuse from unconditional imitation into state-conditioned decision-making. Evaluated across BFCL, AppWorld, and AgentDojo at model scales from 8B to 397B, BASM consistently outperforms success-distilled

skill-memory baselines: task success improves by up to 23.8% on AppWorld; function-calling accuracy by up to 5.0% on BFCL; attack success rate drops by 4.6% on AgentDojo; and average AppWorld steps decrease by up to 6.6% relative to the memory-free baseline. Knockout probes causally attribute wrong-tool suppression to boundary-span attention, not to prompt length or added context; ablations confirm that boundary-conditioned prompting and runtime repair provide complementary benefits—showing that the key to reliable skill reuse is not more procedures, but explicit validity boundaries around each one.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.

Hao Chen, Zhexin Hu, Jiajun Chai, Haocheng Yang, Hang He, Xiaohan Wang, Wei Lin, Luhang Wang, Guojun Yin, et al. 2025. Toolforge: A data synthesis pipeline for multi-hop search without real-world apis. *arXiv preprint arXiv:2512.16149*.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. 2024. Agentdojo: A dynamic environment to evaluate prompt injection attacks and defenses for llm agents. *Advances in Neural Information Processing Systems*, 37:82895–82920.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050.

George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. 2014. Constructing symbolic representations for high-level planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Li Wang, Xiaodong Lu, Wei Lin, Ran He, and Guojun Yin. 2026a. Resrl: Boosting llm reasoning via negative sample projection residual reinforcement learning. *arXiv preprint arXiv:2605.00380*.

Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Guojun Yin, Wei Lin, and Ran He. 2026b. Rest: Reshaping token-level policy gradients for tool-use large language models. In *The Fourteenth International Conference on Learning Representations*.

Zihan Lin, Xiaohan Wang, Hexiong Yang, Jiajun Chai, Jie Cao, Guojun Yin, Wei Lin, and Ran He. 2025. Awpo: Enhancing tool-use of large language models through adaptive integration of reasoning rewards. *arXiv preprint arXiv:2512.19126*.

Xiaodong Lu, Xiaohan Wang, Jiajun Chai, Guojun Yin, Wei Lin, Zhijun Chen, Yu Luo, Fuzhen Zhuang, Yikun Ban, and Deqing Wang. 2026a. Contextual rollout bandits for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.08499*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026b. Skill0: In-context agentic reinforcement learning for skill internalization. *arXiv preprint arXiv:2604.02268*.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. 2026. Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents. *arXiv preprint arXiv:2602.01869*.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pages 9695–9717.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2655–2671.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. 2025. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16022–16076.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, et al. 2026a. Skillx: Automatically constructing skill knowledge bases for agents. *arXiv preprint arXiv:2604.04804*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Li Wang, Xiaodong Lu, Xiaohan Wang, Yikun Ban, Jiajun Chai, Wei Lin, Tianhao Peng, and Guojun Yin. 2026b. When self-belief misleads: Active label acquisition for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2605.25864*.

Li Wang, Xiaohan Wang, Xiaodong Lu, Zipeng Zhang, Jinyang Wu, Jiajun Chai, Wei Lin, and Guojun Yin. 2026c. Implicit hierarchical grpo: Decoupling tool invocation from execution for tool-integrated mathematical reasoning. *Preprint*, arXiv:2605.18500.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. 2025. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. In *Proceedings*

of the AAAI Conference on Artificial Intelligence, volume 38, pages 19632–19642.

Xizhou Zhu, Yuntao Chen, Hao Tian, Chenxin Tao, Weijie Su, Chenyu Yang, Gao Huang, Bin Li, Lewei Lu, Xiaogang Wang, et al. 2023. Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory. *arXiv preprint arXiv:2305.17144*.

Limitations

This work focuses on text-based tool-use agents in benchmarked environments where task goals, observations, and tool feedback can be expressed in natural language. While this setting covers a broad class of practical agent tasks, future work could study boundary-aware skill memory in richer interactive settings, such as multimodal agents, dynamically changing tool APIs, or long-running workflows with delayed feedback. In addition, our current implementation represents boundary conditions as explicit natural-language fields, which makes the mechanism interpretable and easy to inspect; an interesting direction is to explore more compact or adaptive boundary representations that can be selected under tighter context budgets. Finally, our evaluation emphasizes autonomous tool-use performance, safety under adversarial instructions, and interaction efficiency; extending the analysis to human-agent collaboration or personalized tool preferences may further clarify how boundary-aware memories should be adapted across users and deployment settings. As with any method that improves autonomous tool-use agents, broader deployment may raise risks if agents are connected to high-impact tools without appropriate permissioning, audit logging, or human approval, although BASM itself is designed to make tool use more state-aware and easier to inspect. We use publicly available benchmarks and environments under their respective licenses and terms of use, and any released code, prompts, or evaluation scripts associated with this work will include explicit license information; third-party artifacts will not be redistributed beyond what their original licenses permit.

Artifact Use

We use existing benchmarks and environments, including AppWorld, BFCL, and AgentDojo, only for their intended research and evaluation purposes and in accordance with their documented access conditions. Artifacts produced by this work, such as code, prompts, evaluation scripts, and derived skill memories, are intended to support research on tool-use agents; any derived artifacts based on benchmark trajectories should remain within research contexts and should not be redistributed or deployed in ways that exceed the original artifacts' licenses or terms of use.

We do not collect real user data in this work; our experiments use publicly available benchmark

environments and model-generated trajectories for research evaluation. Before releasing any derived artifacts, such as prompts, trajectories, or skill memories, we screen them for personally identifying information, credentials, API keys, unique user identifiers, and offensive content using automated pattern checks and manual inspection of representative samples. Any detected sensitive fields are removed, anonymized, or excluded from release, and benchmark-derived artifacts are shared only when consistent with the original access conditions. We use the official evaluation partitions provided by each benchmark and do not introduce new train/dev/test splits or a new human-collected dataset; the reported statistics therefore correspond to the benchmark instances used for evaluation. We use the official evaluation scripts for BFCL v3, AppWorld, and AgentDojo v1.2.2 with their default evaluator settings unless otherwise stated. We do not use additional NLP preprocessing, normalization, or text-overlap packages such as NLTK, SpaCy, or ROUGE; model choices, evaluated subsets, and metrics are reported in Section 5, with complete retrieval, decoding, and environment budgets in Appendix A. This work does not involve human-subject data collection, participant recruitment, paid annotation, user studies, or intervention with human participants. Therefore, participant-facing instructions, recruitment and payment procedures, data-consent procedures, and ethics review board approval are not applicable. AI assistants were used for language editing, polishing, formatting suggestions, and drafting support for some non-technical explanatory text. All research ideas, method design, experiments, results, analysis, and final manuscript content were reviewed, verified, and approved by the authors; AI assistants were not used to generate experimental results or replace author judgment.

A Implementation and Reproducibility Details

This appendix records the fixed construction, retrieval, decoding, and environment settings used in the reported experiments. It contains no additional evaluation results; supplementary mechanism analyses are reported in Appendix B.

A.1 Inference and Environment Settings

Table 5 summarizes the shared retrieval configuration and benchmark-specific budgets. The retrieval weights and similarity threshold are fixed across benchmarks rather than tuned separately for each evaluation. Character and token limits are retained in their original units because they constrain different stages of the pipeline.

Table 5: **Fixed retrieval, decoding, and environment settings.**

Setting	Value
Retrieval score	$0.65\, s_{\text{semantic}} + 0.20\, s_{\text{tool}} + 0.15\, s_{\text{step/goal}}$
Similarity / embedding	Threshold 0.45; `text-embedding-3-large`
Decoding	Temperature 0.001; thinking disabled for Qwen3-8B/14B/32B and mechanism probes, enabled for Qwen3.5-397B-A17B
BFCL	Top-$k = 3$; 1,600 retrieved-skill characters
AppWorld	Top-$k = 3$; 15 steps; 1,400 generation tokens; 4 repair attempts; observation/repair caps 1,800/1,200 characters; 800 repair tokens
AgentDojo	Top-$k = 2$; 320 skill characters; 44,000 message characters; 850 tool-output characters
Acceptance / checking	Success threshold 0.999; discard malformed candidates after retry; deterministic string/schema checks before LLM-assisted adjudication

Within each benchmark, Base, Proc. Skill, App. Skill, and BASM use the same evaluator and decoding configuration. The memory-using variants additionally share the same source pool, retriever, similarity threshold, and context budget; they differ only in the skill fields and control components defined in Sections 4.1–4.2. We use the official BFCL v3, AppWorld, and AgentDojo v1.2.2 evaluators and introduce no additional NLP preprocessing, normalization, or text-overlap scoring.

A.2 Skill-Library Construction and Provenance

The library follows the SkillX construction pipeline (Wang et al., 2026a), extended with the seven-slot boundary schema in Section 4.1. For AppWorld and BFCL, one shortest successful trajectory per source task supplies the positive procedural context. Failed or partially failed contexts are used only to derive boundary evidence and are never inserted as positive demonstrations. When no failure context is available for an extraction anchor, the builder uses deterministic fallback boundary clauses rather than dropping the anchor.

Extraction operates at two granularities. Functional skills are derived from plan steps, whereas atomic skills are anchored to concrete tool calls and their observations. AgentDojo uses benchmark task ground truth and tool schemas as construction anchors rather than positive source trajectories. Every candidate passes strict JSON parsing, required-field and list-type checks, tool-schema

Table 6: **Retained skill-library statistics.** AgentDojo is schema-anchored.

Benchmark	Anchors	Functional	Atomic	Total
AppWorld	82	1,065	834	1,899
BFCL	118	649	617	1,266
AgentDojo	97	97	74	171

validation, and boundary-specificity/conflict filters. Candidates that still fail after retry are discarded; malformed extractions therefore reduce library recall rather than entering the retrieval index. Every retained record has non-empty boundary fields.

For AppWorld and BFCL, the anchor counts equal the numbers of positive source trajectories. AgentDojo uses no positive trajectory source; its 97 anchors come from task ground truth and tool schemas. Proc. Skill, App. Skill, and BASM are constructed from the same benchmark-specific source pool and retrieval substrate, so the internal comparisons vary the exposed memory representation and boundary-aware controls rather than source availability.

B Additional Experiments

This appendix expands the probe evidence behind the Skill Imitation Trap. The main text shows the central reversal: success-distilled memory does not simply provide helpful experience; under semantically similar but inapplicable states, it becomes an imitation prior that pulls the model toward the retrieved procedure. BASM changes the role of memory. Instead of storing a bare success template, it stores a procedure together with its boundary of validity. The probes below make this mechanism visible at four levels: decision-token attention, next-action logits, same-prompt intervention, and realized call behavior. Together, they support a single interpretation: boundary fields act as a state-conditioned attention modulator that preserves useful procedural recall in T_{ok} while suppressing overconfident imitation in T_{wrong} and guiding recovery in T_{repair}.

B.1 Mechanism Replication and Logit Evidence

Figure 4: boundary fields are read when the state becomes risky. Figure 4 tests whether the added boundary slots are merely present in the prompt or are actually used at the decision point. The results indicate that, under BASM, the boundary/procedure attention ratio is low in T_{ok}, rises substantially in T_{wrong}, and rises again in T_{repair}. This pattern is consistent with the intended boundary-aware behavior. In a safe state, the retrieved skill should behave like a normal reusable procedure, so attention can remain concentrated on the procedural recipe. In a risky state, however, the same procedure becomes dangerous because semantic similarity is no longer equivalent to applicability. The model therefore shifts attention toward applicability conditions, risk cues, avoidance rules, and recovery notes. This state-dependent shift is the core of boundary-aware memory: BASM does not make the agent uniformly conservative; it makes the agent consult the boundary exactly when imitation becomes unsafe.

The two equal-length controls sharpen this interpretation. Irrelevant padding and procedure-padding controls preserve much of the prompt length and schema position, but they do not reproduce the same risk-sensitive boundary/procedure profile. This rules out the simplest alternative explanation that the effect is caused by adding more tokens, slowing down the model, or changing the formatting template. The 14B curve follows the same qualitative pattern as the 8B curve, showing that the phenomenon is not an idiosyncrasy of a single backbone. The magnitude need not be identical across model sizes: larger models can distribute contextual evidence across more internal pathways. What matters for the mechanism is the invariant ordering—safe states read less boundary information, while risky and repair states read more.

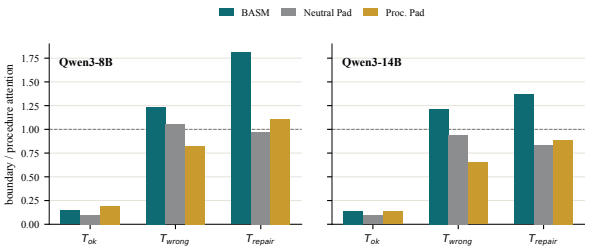

Figure 4: **Boundary attention replication.** Length-normalized boundary/procedure attention across task-state buckets for Qwen3-8B and Qwen3-14B. Under BASM, decision-token attention shifts from the procedure toward boundary fields precisely in risky and repair states, while equal-length controls fail to reproduce this state-conditioned pattern.

Figure 5: boundary attention changes the action distribution. Figure 5 connects the attention shift to the model's next-action preference. On T_{wrong}, Proc. Skill produces a high wrong-tool

margin of 10.71, meaning that the retrieved but currently invalid tool has become a strong local attractor. Equal-length controls partially reduce this margin (6.60 for irrelevant padding and 5.14 for procedure padding), which is expected because extra context and schema regularity can mildly dilute a retrieved procedure. BASM, however, drives the margin down to 2.26. The difference between 5.14 and 2.26 is the important part: once length and formatting are controlled, explicit boundary semantics still provides a much stronger suppression signal.

The right panel reaches the same conclusion with a broader tool-vs-control margin. BASM reduces this margin from 14.24 under Proc. Skill to 5.20, while the two controls remain substantially higher. Thus, BASM is not only discouraging one mislabeled function token; it is lowering the model's general tendency to commit to an inappropriate tool action in a state where the retrieved skill should be rejected. This directly supports the central claim: the failure mode is not lack of skill knowledge, but lack of a boundary around skill reuse.

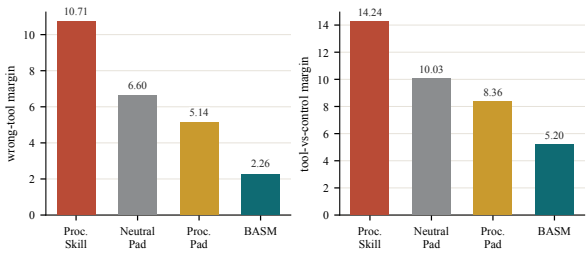

Figure 5: **Detailed T_{wrong} logit margins.** BASM reduces both the specific wrong-tool margin and the broader tool-vs-control margin more than equal-length controls, showing that boundary semantics suppresses the risky action attractor rather than merely changing prompt length or formatting.

B.2 Same-Prompt Knockout Evidence

Figure 6: removing boundary attention reopens the imitation trap. Figure 6 provides the strongest intervention evidence because the prompt tokens are kept fixed. We only mask attention from the decision token to the boundary span, leaving the rest of the memory block untouched. Under this same-prompt intervention, the wrong-tool margin rises by $+3.45$ logits, and the broader tool margin rises in the same direction. In other words, once the decision token can no longer read the boundary fields, the model moves back toward the success-distilled behavior that BASM was designed to prevent. The recovery analysis makes the effect more interpretable: 69.4% of risky-state samples move toward the Proc. Skill baseline after boundary knockout, while the broader tool-margin gap is recovered by 41.0%.

The controls are equally important. A random same-length mask does not produce the same effect, and masking procedure-span attention does not restore the imitation trap. This means the result is not a generic consequence of deleting context or damaging attention. The specific path from the decision token to the boundary span is what counteracts procedural imitation. The knockout therefore turns the boundary fields from a plausible prompt-engineering addition into a mechanistically active component: they are not merely ancillary prompt annotations, but an anti-imitation channel that keeps retrieved experience from becoming an unconditional action template.

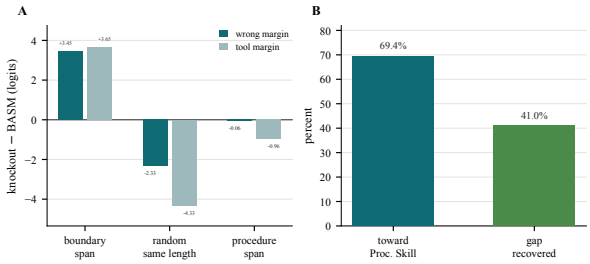

Figure 6: **Boundary knockout controls.** Same-prompt attention knockout identifies the boundary span as the causal anti-imitation channel. Masking boundary attention pushes 69.4% of risky-state samples toward the Proc. Skill baseline, while random same-length and procedure-span masks fail to restore the imitation pattern.

Figure 7: the larger model follows the same causal direction. Figure 7 repeats the intervention on Qwen3-14B. The effect is smaller than on Qwen3-8B but follows the same direction: boundary knockout increases the T_{wrong} wrong-tool margin, produces only a small change in the T_{wrong} correct-tool margin, and moves the T_{ok} correct-tool margin in the opposite direction. This selectivity is consistent with a boundary mechanism: the boundary path matters most when the retrieved procedure is tempting but invalid, rather than acting as a blanket penalty against tool use.

We interpret the smaller magnitude as a scale-dependent version of the same mechanism. A larger model may integrate boundary evidence through more distributed contextual routes, so blocking a single span-level path perturbs the final logits less strongly than in the 8B model. Even so,

the intervention still moves the risky margin in the predicted direction and suppresses the boundary-attention path, confirming that the causal signature of boundary-aware retrieval survives a change in model scale.

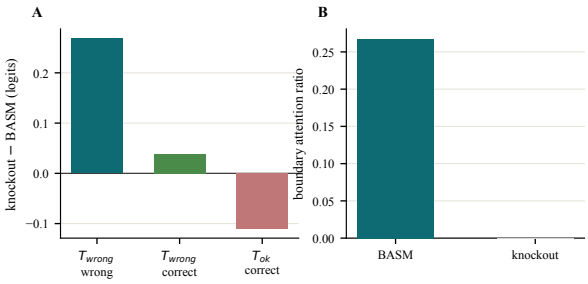

Figure 7: **Qwen3-14B lightweight knockout.** Boundary knockout again moves the T_{wrong} wrong-tool margin upward while suppressing the boundary-attention path. The smaller magnitude is consistent with a more distributed implementation in the larger model, but the causal direction matches the 8B intervention.

B.3 Where the Boundary Signal Appears

Figure 8: boundary reading is concentrated, not diffuse. Figure 8 plots the per-layer/per-head $\log_2$ boundary/procedure ratio for BASM in T_{wrong} at $k = 2$. The map is not uniform. Several regions, especially in middle and later layers, assign substantially more decision-token attention to boundary spans than to procedure spans. This pattern is theoretically meaningful. If BASM were only improving behavior through global prompt length or a uniform caution prior, the attention map would be broadly smooth. Instead, the model appears to recruit a subset of attention heads that preferentially read the boundary when the retrieved procedure is semantically close but unsafe.

This head-level structure also explains why span-level knockout is effective. The boundary signal is not spread evenly across all tokens; it is routed through identifiable attention pathways that can be disrupted. These heads behave like a state-conditioned modulator: they do not erase the retrieved procedure, but they change how much the decision token should trust that procedure under the current state.

Figure 9: candidate heads specialize in the risky-state transition. Figure 9 ranks heads by the increase in boundary/procedure log-ratio from T_{ok} to T_{wrong}. The strongest heads show large positive shifts, meaning they change behavior when the same kind of retrieved skill moves from a safe state

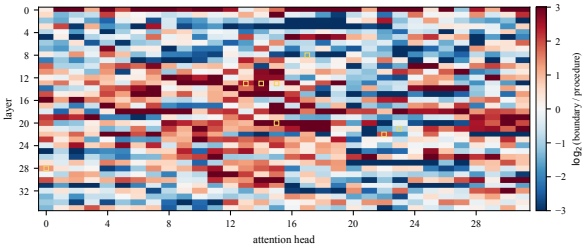

Figure 8: **Head-level boundary attention.** Per-layer/per-head $\log_2$(boundary/procedure) attention for BASM in T_{wrong} at $k = 2$. Boundary reading is concentrated in identifiable heads rather than uniformly spread across the prompt, consistent with a state-conditioned attention modulator.

to a risky one. This is a more precise signature than high boundary attention alone. A head that always attends to boundary text could simply be responding to formatting. A head whose boundary attention increases specifically from T_{ok} to T_{wrong} is instead sensitive to the state transition that defines the Skill Imitation Trap.

These candidate heads provide a concrete mechanistic locus for BASM's effect. The main causal claim remains at the span level, because span-level knockout directly tests the information channel used by the decision token. The head ranking goes one step deeper: it shows that the boundary channel is not an abstract metaphor, but an observable pattern in the model's attention computation. This strengthens the interpretation of BASM as a boundary-aware memory object rather than a longer prompt template.

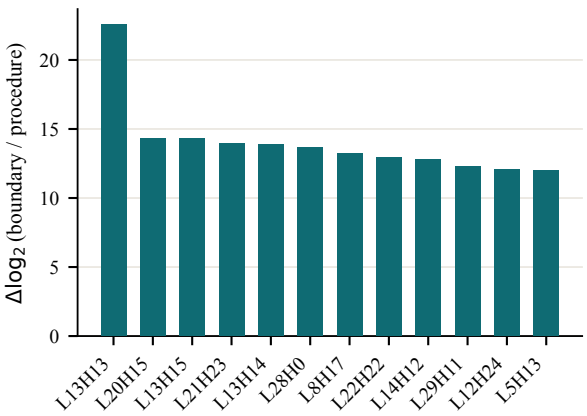

Figure 9: **Candidate boundary heads.** Heads ranked by the increase in boundary/procedure log-ratio from T_{ok} to T_{wrong}. The largest shifts identify attention heads that become boundary-seeking exactly when procedural imitation becomes risky.

B.4 Boundary Mechanism, Behavior, and Control Comparisons

Figure 10: the effect is dynamic attention, not a static hidden-state label. Figure 10 reports a task-disjoint linear probe for skill applicability over residual-stream states. The probe does not reveal a stable, high-AUC applicability feature that is uniquely stronger under BASM. This result should not be read as a weakness of the method. Instead, it rules out a simpler and less interesting explanation: BASM is not merely injecting an easy linear label into the hidden state. The boundary signal is used dynamically at the decision token through attention to the retrieved skill text.

The Skill Imitation Trap occurs because retrieval gives the model a compelling procedure before the model has checked whether the current state satisfies the procedure's preconditions. BASM solves this by keeping the boundary available as contextual evidence and letting the decision token read it when the state demands it. A static residual-stream classifier is therefore not the expected mechanism. The negative linear-probe result is consistent with a stronger claim: boundary-aware skill memory changes the computation at the moment of action selection, rather than simply changing a global representation of the example.

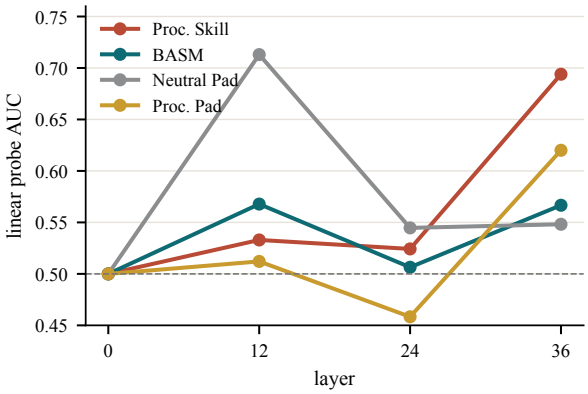

Figure 10: **Residual-stream probe control.** Task-disjoint linear probes do not isolate a stable applicability vector. This supports the view that BASM works through dynamic decision-token attention to boundary text rather than through a static hidden-state label artifact.

Figure 11: the trap appears in realized candidate choices. Figure 11 verifies that the logit-margin story is not only a continuous-score artifact. As retrieval depth increases, Proc. Skill keeps the wrong-call rate high in T_{wrong}, showing the behavioral version of the Skill Imitation Trap: the more successful procedures the model sees, the more it is pulled toward calling a semantically related but invalid tool. BASM sharply lowers the wrong-call rate at the riskiest retrieval depths, especially around the depths where the logit probe shows the strongest trap.

The correct-call curve should be interpreted together with the wrong-call curve. In T_{wrong}, the desired behavior is often not simply "call another tool"; it may be to avoid the retrieved tool, choose a control action, or wait for a repair path. Therefore, the key behavioral quantity is whether the model continues to execute the tempting wrong tool. By reducing wrong calls without relying on a generic no-tool bias, BASM demonstrates that boundary fields reshape the agent's action preference in the exact region where success-distilled memory is most dangerous.

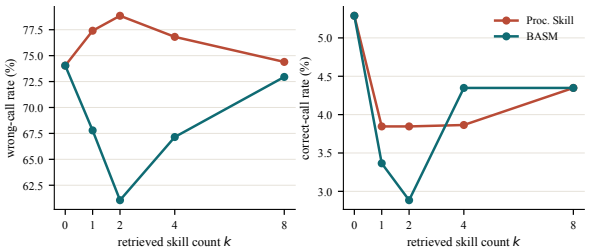

Figure 11: **Recall-depth call rates.** The Skill Imitation Trap appears in realized candidate choices, not only in logits. Proc. Skill keeps wrong calls high near risky retrieval depths, whereas BASM suppresses those wrong calls by conditioning skill reuse on boundary evidence.

Figure 12: equal-token pairing isolates boundary semantics. Figure 12 compares BASM with a strong equal-token procedure-padding control at the paired-example level. This is a demanding control because procedure padding still contains task-relevant procedural text rather than empty filler. On BFCL, the paired wins are balanced between BASM and padding, while many examples are jointly solved or jointly failed. This observation does not undermine the mechanism result; instead, it clarifies the class of problems addressed by boundary information. BFCL contains many cases where extra procedural evidence can help with ordinary function selection, but the probe results show that when the retrieved procedure is plausible yet unsafe, boundary semantics is what suppresses the wrong-tool margin.

The AppWorld paired result is especially informative because the environment is stateful and

multi-step. There, BASM wins more examples than procedure padding, matching the role of recovery notes and avoidance rules in long-horizon interaction. The paired analysis therefore supports a more precise conclusion: successful experience is valuable, but successful experience alone is incomplete. In states that require checking applicability, detecting risk cues, or repairing a local failure, a skill must carry its own boundary. BASM supplies that missing boundary, turning memory from a source of overconfident imitation into a source of controlled transfer.

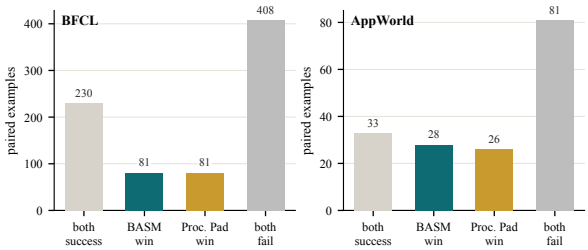

Figure 12: **Equal-token control paired status.** Pairing against procedure padding isolates the value of boundary semantics under an equal context budget. BASM and padding trade wins on BFCL, while BASM is slightly ahead on AppWorld, consistent with the claim that boundary fields matter most when procedural memory must be state-conditioned.

Appendix takeaway. The appendix strengthens the main mechanism chain. Figure 4 shows that boundary text is read more in risky and repair states. Figure 5 shows that this reading suppresses wrong-tool preference beyond equal-token controls. Figures 6 and 7 show that removing boundary attention pushes the model back toward imitation. Figures 8 and 9 show that the signal is routed through identifiable attention heads. Figure 10 rules out a static linear-feature explanation, and Figures 11–12 connect the probe analysis to realized calls and paired outcomes. Taken together, these results support a precise conclusion: more memory is not automatically better. Memory becomes reliable only when a retrieved skill also tells the agent when not to imitate it.