FROM EXPERIENCE TO EXECUTION: STATE-CONDITIONED CONTROL FOR SMALL LANGUAGE MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

As language model agents move from answering questions to executing tasks, the substantial cost and data-exposure risks of remote inference make local small models an attractive deployment choice. Limited parameter capacity and reasoning ability, however, leave these models prone to accumulating errors in prerequisites, state tracking, and argument binding even in recurring workflows. Experience memory can provide the right procedure, but turning it into a current action still requires the model to resolve these local decisions. This difficulty motivates a control-theoretic approach that allocates decision freedom between the small model and its harness according to task state, assigning choices resolved by available evidence to external mechanisms. We propose state-conditioned control (SCC), which compiles trajectory relations into a shared procedural representation for state-specific guidance, pre-generation action restrictions, and direct execution of locally supported steps. With frozen weights, SCC exceeds the strongest of four offline memory adaptations on the evaluated ToolSandbox, ALFWorld, and WebShop suites (13.12 score points, 61.94 and 3.80 success-rate points) and reuses its experience and rules with two additional small models. Channel and cost analyses reveal task-dependent gains and guidance costs, supporting decision allocation as a design variable for small-model agent harnesses. Our agent implementation is available at `https://anonymous.4open.science/r/SCC-C6C3/`.

1 INTRODUCTION

Language model agents execute natural-language goals through tool calls and software operations (Yao et al., 2023; Agashe et al., 2025). Remote large models incur inference costs and network delays, while transmitting task context creates data-exposure risks (Wu et al., 2025a). External content can carry indirect prompt injections, and inter-agent messages can be intercepted or altered (Zhan et al., 2024; He et al., 2025). These pressures motivate local small models for resource-constrained, interaction-intensive, or data-sensitive applications (Belcak et al., 2025).

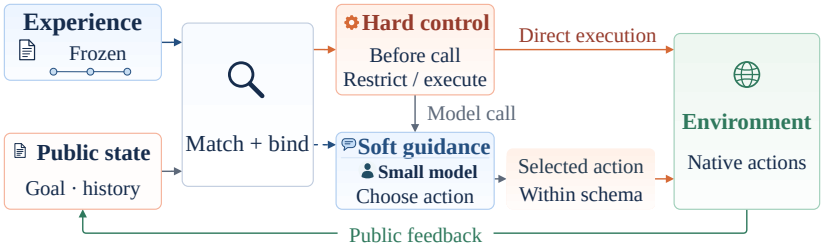

Figure 1: **Decision allocation in SCC.** Matched experience and public evidence support direct execution or a constrained model choice with optional guidance. Native feedback updates the state used for the next decision.

Local deployment requires reliable execution, yet limited capacity weakens small models' multi-step reasoning and long-horizon planning (Liu et al., 2024). Many tasks have reusable stages, tool inter-

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.

faces, and dependencies. Knowing the procedure, however, does not ensure timely execution. Small models may omit prerequisites, confuse returned identifiers with task descriptions, repeat completed operations, or stop prematurely (Kim et al., 2026). These errors change subsequent observations and divert familiar workflows, requiring continual progress assessment against available evidence.

Prior work improves reliability through hierarchical organization, memory evolution, procedural updates, and subtask-aligned retrieval (Kim et al., 2026; Ouyang et al., 2025; Fang et al., 2025; Shen et al., 2026). Retrieval supplies advice, but the model must still check conditions, identify unfinished steps, and bind current objects and arguments. Even after retrieving the right procedure, a weaker model may struggle with these local decisions and fail to turn experience into effective action. Yet some choices in recurring workflows can already be resolved from public observations and procedural dependencies. Others remain uncertain because the goal is ambiguous, an object is unknown, or the current state is not covered. *How can experience adjust the decisions that a small model must make independently while preserving the choices needed to resolve these uncertainties?*

Control theory provides a view of this allocation. Ashby's *law of requisite variety* relates effective regulation to the responses available to counter disturbances (Ashby, 1956). Agent cybernetics examines how feedback, state representation, and decision authority shape system capability (Wang et al., 2026; Eslami & Yu, 2026). We treat the model and its external execution framework (harness) as a joint regulator. When response conditions can be checked, the harness can perform the operation and leave interpretation and exploration to the model. This makes the state-dependent allocation of *decision freedom* a harness design variable.

We instantiate this idea through *state-conditioned control* (SCC), shown in Figure 1. SCC organizes trajectories from larger models or environment experts into stages, dependencies, and action bindings. Matching these structures against public observations identifies progress and unresolved choices. *Soft control* presents relevant state and procedural information. *Hard control* restricts available actions and lets the harness execute locally supported steps. The model retains search and goal interpretation, while native feedback updates state.

We evaluate this allocation in ToolSandbox (TS), ALFWorld (ALF), and WebShop (WS), which combine reusable procedures with dialogue, subgoal planning, and search. On the evaluated suites, SCC exceeds four fixed offline memory adaptations built from common expert sources. These comparisons concern the adapted systems rather than the original methods' full online capabilities. We also reuse the fixed experience and controller with Qwen3-4B and Gemma-3-4B-IT. Both exceed their memory baselines on TS and complete tasks in all three environments.

Channel ablations show that structured guidance improves execution, with additional hard-control gains supported on TS and ALF. Cost analysis reveals a tradeoff: guidance lengthens prompts, while better execution can reduce later decisions and failed-episode computation. Experience therefore affects both performance and cost.

Our contributions are as follows:

- **State-conditioned decision allocation.** We formulate the allocation of task-relevant choices between a frozen small model and its harness according to current public evidence, distinguishing model guidance from externally enforced restrictions and execution.
- **A shared procedural representation for soft and hard control.** SCC uses trajectory-derived dependencies and action bindings for public-state matching, progress tracking, and argument resolution. The resulting state supports both structured guidance and task-specific restrictions or local execution.
- **System-level evaluation and channel analysis.** Across three environments, we compare complete agents against four fixed offline memory adaptations, measure the contributions and costs of the two control channels, and assess reuse with two additional small models.

2 Background

2.1 Related Work

Recent advances in agents. Recent work optimizes agents at the system level. AgentSquare searches planning, reasoning, tool, and memory modules, AFlow optimizes code workflows us-

ing execution feedback, and Agent S uses experience for hierarchical planning and subtasks (Shang et al., 2025; Zhang et al., 2025; Agashe et al., 2025). MEM1 jointly learns memory compression and multi-turn reasoning (Zhou et al., 2026). These advances show that capability depends on interaction structure and information organization as well as the model. Small models still need to turn historical information into actions suited to the current state.

Limits of memory. A-Mem organizes memory through links and updates, while LightMem separates online use from offline consolidation to reduce maintenance costs (Xu et al., 2025; Fang et al., 2026). Agent Workflow Memory extracts reusable workflows from trajectories to guide subsequent actions (Wang et al., 2025). LongMemEval exposes difficulties in cross-session retrieval, temporal reasoning, and knowledge updates (Wu et al., 2025b). In the action-based tasks considered here, experience still requires applicability checks, object binding, and prerequisite validation before it becomes a current action. We examine how experience can participate in these decisions to improve execution reliability and reduce inference cost.

State and action knowledge. StateFlow separates state-machine process control from within-state task solving (Wu et al., 2024). KnowAgent uses action knowledge to constrain planning paths and guide self-learning (Zhu et al., 2025). Together with work on experience-guided execution (Wang et al., 2025), these methods establish state and process structure as agent design tools. SCC studies a shared procedural representation for public-state binding, guidance, and execution control.

2.2 Motivation

Ashby's law of requisite variety relates effective regulation to the responses needed to counter environmental variation. His discussion of supplementary regulation also describes the contribution of external mechanisms (Ashby, 1956). Together with work on feedback and decision authority in agent cybernetics (Wang et al., 2026; Eslami & Yu, 2026), this motivates treating the model and its harness as a joint regulator. Model judgment and external execution can both supply the responses the system needs, making the choices assigned to the model a harness design variable. We ask how shared procedural relations can allocate local decisions between the model and its harness, and how the resulting soft and hard channels jointly shape the complete agent's execution.

2.3 Problem Formulation

Inputs and outputs. In an action–observation model (Yao et al., 2023), task $q = (g, o_0, \mathcal{A}) \sim \mathcal{D}$ supplies a goal, initial observation, and public tool interfaces, drawn from task distribution $\mathcal{D}$. The agent interacts with a partially observed environment and produces an executed trajectory $\tau = (o_0, \tilde{a}_0, o_1, \ldots, o_T)$ and, when required, final answer y. Here $\tilde{a}_t$ is the executed action, o_t is the observation, and T is the termination time. Public history through time t is h_t. Native evaluation assigns $R(\tau, y; g)$.

Experience source and execution model. Reference agent π_E is a larger model or environment expert with stronger planning and execution on the target tasks. It supplies offline experience E_E containing observations, actions, and feedback, not all of which need be correct. Small model π_S is a resource-constrained deployment model responsible for online execution. Motivated by weaknesses in long-horizon reasoning and decisions (Liu et al., 2024), we use E_E to design an external mechanism H that improves the fixed small model's task performance.

Design space and fixed components. Following the design variables used in prior agent harness work (Zhang et al., 2025; Yang et al., 2025b), we allow changes to experience representation and retrieval, context organization, and execution control in H. Model parameters, environment transitions, native tool semantics, and scoring remain fixed. The mechanism uses task inputs, offline experience, and public interaction history, without online calls to a larger model or access to hidden evaluation information. These conditions define design space $\mathcal{K}$.

Decision freedom. For goal g, public history h_t, and mechanism H, $\mathcal{F}_S(g, h_t; H)$ denotes the task-relevant choices still assigned to the model and their available ranges. We call this the agent's *decision freedom*. Choices determined or executed by the external mechanism are excluded. This set-based notation describes who makes which choices. Model requests, automatic-action proportions, and constrained-decision proportions describe its implementation (Appendix D.3); they are not policy entropy.

Objective. Following ADAS in separating design space and evaluation (Hu et al., 2025), we maximize expected task performance under budget vector $B = (B_{\text{dec}}, B_{\text{act}}, B_{\text{out}}, B_{\text{mem}})$. Its components cap model decisions, native actions, output tokens per request, and injected memory tokens. For compliant mechanisms $\mathcal{K}_B \subseteq \mathcal{K}$ and $(\tau, y) \sim \text{Rollout}(\pi_S, H; q)$,

$$\max_{H \in \mathcal{K}_B} J(H), \qquad J(H) = \mathbb{E}_{q,\tau,y}[R(\tau, y; g)]\,. \tag{1}$$

We compare decision allocations and report request, token, and latency costs separately. Train-free means fixed weights; freezing offline experience is an additional protocol choice.

3 State-Conditioned Control

Figure 2 connects offline experience compilation to online state-conditioned control.

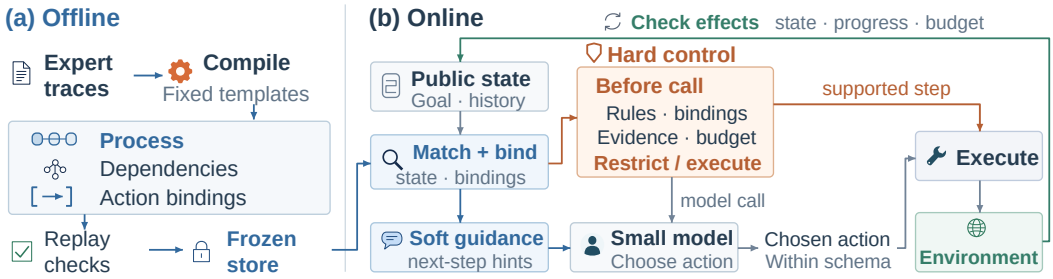

Figure 2: **Offline construction and online execution.** Fixed templates compile trajectory relations into reusable procedures. Public-state matching supplies soft guidance and pre-call hard control. A locally supported step executes directly; otherwise, the model chooses through the current action schema. Feedback updates progress and bindings.

3.1 Offline Experience Construction and State Modeling

Motivated by requisite variety, SCC allocates choices of subgoals, operations, arguments, and stopping by state. The harness uses trajectory relations to resolve supported local steps, leaving other judgments to the small model.

Extracting states and transitions from trajectories. Experience E_E records goals, observations, actions, arguments, and feedback. The Python compiler uses manually defined action grammars, state fields, binding templates, and execution checks. Trajectories supply retained steps, parameter sources, and order relations within this representation. It parses valid actions and results, replacing demonstrated objects and values with task slots or return references. If a later action uses an earlier result, its argument source becomes a dependency edge. The compiler also retains conservative state-change and stage ordering, rather than recovering a minimal causal graph. Each node records an operation and a checkable completion condition. Completed nodes, unmet dependencies, and public observations jointly determine the current state. Feedback confirms completion before the process advances. ALF procedures come from programmatic training experts. WS uses a predefined option parser and option-before-purchase rule admitted by training counterfactual checks.

The extracted experience has three layers. (1) The **process layer** retains reusable stages and their order. (2) The **state and dependency layer** records prerequisites, argument sources, and observable effects. (3) The **action layer** stores tools and argument templates whose values come from the current request, confirmed results, or constants. We write $M = \mathcal{C}(E_E)$ for compiler $\mathcal{C}$ and experience store M. For current goal g and public history h_t, state-matching function ψ_M yields $\hat{s}_t = \psi_M(g, h_t)$ for both control channels. Appendix C.1 details task-specific extraction, filtering, and adaptation.

3.2 PUBLIC-STATE MATCHING AND SOFT GUIDANCE

Let $x_t = (g, h_t, \hat{s}_t, M, b_t)$ denote the current context, where g is the task goal, h_t the public interaction history, $\hat{s}_t$ the current state, M the offline experience store, and b_t the remaining budget. The soft channel locates a procedure matching the task and current progress, checks outstanding dependencies, and resolves arguments from the request and public results. The harness assembles the process, progress, confirmed arguments, unmet conditions, and next-step suggestions into $u_t = \phi(x_t)$, where ϕ is a deterministic guidance constructor. The packet contains the current stage and its evidence, retaining a short dependency horizon for exact process matches. Fixed templates render these fields as JSON without additional generative-model calls. Appendix C.2 gives an end-to-end example.

When a decision is needed, the small model proposes $a_t \sim \pi_S(\cdot \mid g, h_t, \hat{s}_t, u_t)$. Guidance highlights current dependencies and arguments, directing attention toward unresolved choices. The model still decides whether to follow advice, which tools and arguments to use, and whether to try legal actions absent from the experience. Soft control therefore influences how the model uses its decision freedom by changing its evidence and action preferences while preserving its final choice. Disabling the channel sets $u_t = \varnothing$, leaving shared inputs and basic legality constraints unchanged.

Compared with retrieved-text memory, SCC moves applicability checks, progress tracking, and argument binding into the harness. Its guidance changes with completed steps and separates available operations from those with unmet prerequisites. Soft-only thus includes programmatic state extraction and guidance construction, rather than retrieval alone.

3.3 STATE-CONDITIONED HARD CONTROL

Soft guidance supplies evidence, but a small model may still omit prerequisites, choose the wrong object, or terminate too early. The hard channel combines experiential dependencies with adapter-defined repetition and termination guards. Extracted prerequisites determine when a step can proceed, argument references identify the current object, and completion conditions determine when restrictions can be released. These fields instantiate task-specific checking and execution templates. Executable experience passes source-success and replay checks, then runtime checks against current evidence. Hard control allocates decision authority in two ways.

(1) Restricting available actions. Before generation, the harness uses the current stage, outstanding dependencies, and confirmed objects to construct $\mathcal{A}_H(x_t) \subseteq \mathcal{A}_{\text{legal}}(x_t)$, where $\mathcal{A}_{\text{legal}}$ contains actions allowed by the native interface. A JSON action schema enforces this set, and the small model chooses among the remaining options. This directly narrows the choices available in the current state, excluding operations incompatible with the procedure while retaining unresolved decisions such as search. Task adapters also define repetition and termination guards triggered by current records. Missing evidence or an empty filtered set invokes adapter-specific fallbacks, detailed in Appendix C.4.

(2) Taking over locally supported steps. Before requesting a model proposal, the harness checks whether a pending operation and its arguments are supported by the current procedure, public evidence, and tool interface. A candidate must satisfy (1) a rule matching the current task and stage, (2) valid state facts and arguments resolved from the current request or confirmed results, and (3) tool availability and execution prerequisites. Rules requiring unique matches reject ambiguous bindings. These checks, the hard-channel switch, adapter support, and remaining budgets determine execution permission $c(x_t) \in \{0, 1\}$. Let $F(x_t)$ instantiate the supported native action and let $\mathcal{A}_t$ be the hard-filtered action set when enabled, or the native legal set otherwise. The evaluated control paths are

$$\begin{cases} \tilde{a}_t = F(x_t), & c(x_t) = 1 \quad \text{(no model call)}, \\ a_t \sim \pi_S(\cdot \mid x_t, u_t; \mathcal{A}_t), \quad \tilde{a}_t = a_t, & c(x_t) = 0. \end{cases} \tag{2}$$

Here $\mathcal{A}_t$ is enforced by the decoding schema, while u_t is optional guidance. Failure of the direct-execution gate returns the choice to the model and does not by itself disable applicable action restrictions. Fixed matching priorities select a process, then dependency order identifies its pending node. Appendix C.1 details arbitration. These are deterministic evidence checks, not calibrated confidence or guarantees of task value. TS combines automatic execution with termination restrictions; ALF

uses stage/object constraints, repetition filtering, and automatic execution; WS uses option and loop schema restrictions.

3.4 Native Feedback and the Execution Loop

Executing $\tilde{a}_t$ returns observation o_{t+1}. State update $\hat{s}_{t+1} = U(\hat{s}_t, \tilde{a}_t, o_{t+1})$ records confirmed objects and arguments and checks node completion. A completed step advances the dependency graph and refreshes guidance and permitted actions. Errors, missing effects, or unmet prerequisites leave the step unresolved and return decisions to the model; a proposal alone never advances progress. Automatic steps recheck the updated state and stop at their budget limit. Native termination ends the task.

4 Experiments

We ask four questions: **RQ1**, how does SCC compare with memory methods overall and across tasks (Section 4.2)? **RQ2**, can experience and rules be reused across small models (Section 4.3)? **RQ3**, what do soft and hard control contribute (Section 4.4)? **RQ4**, how does this allocation affect tokens and time (Section 4.5)?

4.1 Experimental Setup

Environments and scope. We evaluate reusable procedures with computable feedback in ToolSandbox dialogue/tool use (Lu et al., 2025), ALFWorld household operations (Shridhar et al., 2021), and WebShop shopping (Yao et al., 2022). The suites comprise 515 locally runnable TS scenarios selected by external-interface dependencies from 1,032, all 134 ALF `valid_unseen` games, and all 500 official WS test goals. These environments informed development and are not a new independent holdout. Appendix A details coverage, experience-source overlap, and sensitivity analyses.

Models and comparisons. Qwen3-8B (Yang et al., 2025a) is the main model; Qwen3-4B and Gemma-3-4B-IT (Gemma Team et al., 2025) assess reuse. Agent Memory Distillation (AMD), ReasoningBank (RB), MemP, and SASM are fixed offline memory adaptations sharing available expert sources, decoding, and native-action budgets. Comparisons with SCC use matching task sets across execution batches. The four channel configurations share each environment's core and initial states. Weights, experience, and rules are fixed within these runs. Appendix B specifies adaptations and provenance.

Metrics. We report native TS score, ALF success, and WS full-reward success and mean reward on 0–100 scales, plus small-model tokens and episode time. Paired 95% bootstrap intervals resample TS source templates or ALF/WS tasks, retaining repeats within tasks. N counts distinct tasks; WS has two executions per goal. Intervals describe task-sampling uncertainty.

4.2 RQ1: Overall and Task-Level Performance

SCC improves the complete agent system. SCC exceeds the strongest external memory adaptation by 13.12 TS score points, 61.94 ALF success-rate points, and 3.80 WS success-rate points, with positive paired intervals (Table 1). System gains combine changes in representation, retrieval, and execution; Section 4.4 isolates the additional hard channel within SCC. After excluding the 95 TS task IDs also present in the experience sources, SCC retains a 12.78-point gain over AMD on the remaining 420 scenarios (Appendix A.2). Full-credit cases increase from 50/515 under None to 85/515 under Hybrid (Appendix D.2).

Benefits vary with the decisions left to the model. ALF's large gains and the TS improvements on state dependencies are consistent with control helping when progress and local prerequisites can be identified. The pattern is not uniform: SCC trails a memory adaptation on TS Contacts and Queries and WS Grocery. Semantic interpretation, search, and the final answer still depend on model judgment. State clarity is thus a plausible explanation for heterogeneous gains, rather than an identified causal mediator.

Table 1: Qwen3-8B overall and task-level performance (0–100).

	ToolSandbox							ALFWorld						
Method	All	C	M	Q	R	S	I	All	E	P	Cl	Co	H	T
N	515	56	49	103	100	85	122	134	18	24	31	21	23	17
AMD	45.5	43.7	44.1	**75.8**	20.3	64.7	28.5	14.2	22.2	29.2	9.7	9.5	0.0	17.6
RB	41.6	44.1	45.1	63.2	33.5	69.1	8.2	12.7	22.2	29.2	6.5	9.5	4.3	5.9
MemP	39.8	**49.8**	47.5	53.4	23.0	67.2	15.3	20.1	22.2	37.5	12.9	19.0	21.7	5.9
SASM	41.2	43.0	43.9	58.9	27.7	59.7	22.5	16.4	33.3	41.7	3.2	4.8	13.0	5.9
SCC	**58.6**	44.7	**50.7**	74.8	**39.2**	**77.1**	**57.5**	**82.1**	**83.3**	**79.2**	**87.1**	**95.2**	**82.6**	**58.8**
Gain	+13.12 [6.67, 19.76]							+61.94 [53.73, 70.15]						

	WebShop						
Method	All	Rew	B	E	F	H	G
N	500	500	110	94	102	103	91
AMD	26.0	61.4	26.4	25.0	34.8	18.0	25.8
RB	28.0	61.8	28.2	24.5	40.7	14.1	**33.0**
MemP	28.8	62.5	26.8	25.0	38.2	21.4	**33.0**
SASM	25.0	55.2	26.4	23.9	37.7	12.6	24.2
SCC	**32.6**	**65.4**	**35.0**	**30.3**	**42.2**	**27.2**	27.5
Gain	+3.80 [0.40, 7.10]						

Gain compares SCC with AMD on TS and MemP on ALF/WS; brackets are paired 95% difference intervals. Category labels are defined in Appendix A.

4.3 RQ2: Reuse across Small Models

SCC reuses its fixed experience and controller with two further 4B models. Qwen3-4B scores 58.20 on TS and Gemma-3-4B-IT 48.21, exceeding their strongest memory adaptations by 16.50 and 19.42 points (Figure 3). For ALF and WS, we report absolute performance without matched 4B memory-baseline runs. Rankings vary: Qwen3-4B nearly matches Qwen3-8B on TS and leads on WS, whereas Qwen3-8B leads on ALF. Unresolved search and interpretation still depend on the model. Reuse without tuning on these results supports compatibility across models, rather than transfer to new procedures. Appendix E reports all outcomes and failure accounting.

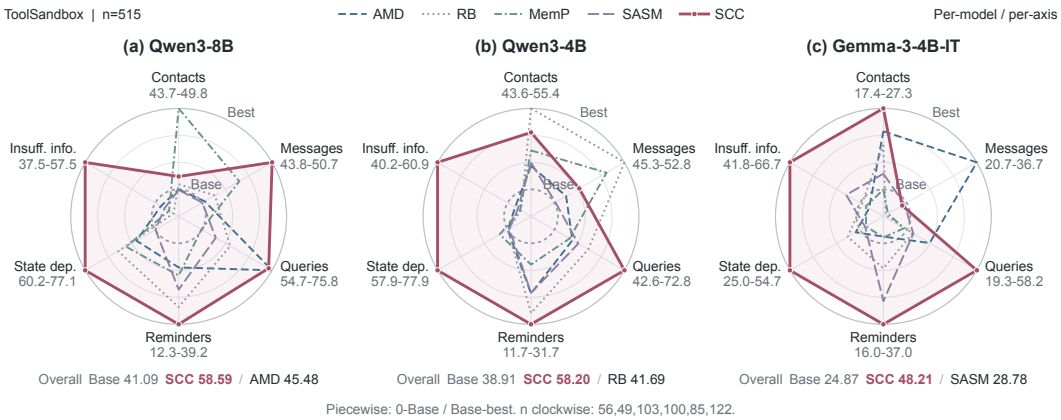

Figure 3: **Reuse across small models on ToolSandbox.** Each spoke maps Base to the inner quarter-ring and the best method to the outer ring; labels give the raw endpoints. All methods use the same 515 scenarios per model.

4.4 RQ3: Effects of Soft and Hard Control

We enable channels independently on the same tasks, initial states, and environment core, yielding None, Soft, Hard, and Hybrid (Appendix D).

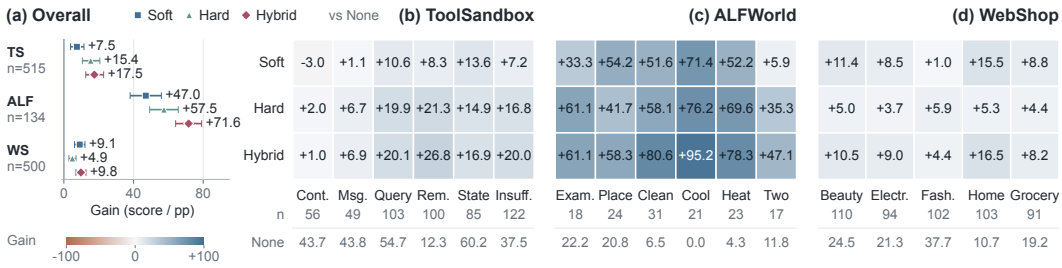

Figure 4: **Control gains over no control.** (a) Paired gains and 95% bootstrap intervals. (b–d) Category gains, task counts, and None scores. TS uses score points; ALF/WS use success-rate points.

Hard control contributes beyond no control. Hard-only improves all three primary metrics over None, with positive paired intervals (Figure 4a). Soft-only also improves all three, confirming that state extraction and guidance already carry substantial value. The measured hard channel includes adapter-specific stopping or repetition guards as well as experience conditions and, in TS/ALF, automatic execution. Its gain therefore measures this combined channel. WS illustrates an important distinction: Hard increases full-reward success from 22.80 to 27.70, while mean reward decreases from 60.30 to 58.92. Restricting choices can improve exact completion without improving partial goal satisfaction.

The additional value of hard control depends on the task. Hybrid exceeds Soft by 10.04 TS score points and 24.63 ALF percentage points. Adding guidance to Hard also improves all three primary metrics: 2.10 TS points, 14.18 ALF percentage points, and 4.90 WS percentage points, with positive paired intervals (Appendix D). Hybrid therefore outperforms either single channel on TS and ALF. On WS, guidance improves Hard, but adding hard control to Soft does not establish an additional success-rate gain. These comparisons estimate the complete channel effects, including adapter guards and the trajectories they produce.

4.5 RQ4: Token Consumption and Execution Time

Table 2 reports episode tokens, time, and outcome-normalized cost: total tokens including failures divided by ALF/WS successes or summed 0–1 TS scores. Parentheses show reductions from the lowest-cost memory baseline. This ratio is not expected retry-until-success cost. Times come from separate serving batches; Appendix F.1 reports all channel configurations.

Table 2: Episode tokens (k), time (s), and tokens per outcome (Norm., k).

Method	ToolSandbox			ALFWorld			WebShop		
	Tokens	Time	Norm.	Tokens	Time	Norm.	Tokens	Time	Norm.
AMD	19.25	13.08	42.33	28.32	19.57	199.75	6.47	**2.19**	24.88
RB	26.34	13.45	63.36	51.49	20.70	405.83	10.37	2.68	37.04
MemP	27.59	14.97	69.31	41.48	21.55	205.84	9.52	3.74	33.05
SASM	19.61	24.37	47.61	31.31	52.72	190.70	18.99	4.81	75.95
SCC	**4.17**	**4.48**	**7.12**	**14.53**	**13.56**	**17.70**	**6.10**	4.87	**18.71**
Reduction			(↓83.17%)			(↓90.72%)			(↓24.79%)

Hybrid improves outcome-normalized token efficiency on ALF. Compared with None, Hybrid uses 43.19% more input tokens per request but reduces model calls from 32.52 to 15.62 per episode

and total tokens from 21.31k to 14.53k. Relative to Hard, adding guidance raises episode tokens from 13.58k to 14.53k, while the higher success rate lowers tokens per success from 19.99k to 17.70k. Guidance therefore need not reduce per-episode computation to improve outcome-normalized efficiency.

The preferred configuration depends on the environment and cost metric. On TS, hard-only has the lowest tokens per score unit; adding guidance improves the score but increases this cost. On WS, hard-only raises success over no control while increasing requests and tokens per success. Guidance reduces this burden, but hybrid's observed 4.87 seconds exceeds soft-only's 2.34 seconds and the fastest memory baseline's time. More reliable execution can save failed attempts while adding prompt or action overhead; the balance depends on the procedure and observation length. TS simulator usage is separate. Including it, total input/output tokens fall 58.10% from None to Hybrid (Appendix F.1); this is not the reduction relative to AMD or a monetary or time saving.

5 Discussion and Limitations

Experience reuse beyond model weights. Trajectory experience contains reusable relations between prerequisites, actions, and effects. SCC places these relations in the harness and binds them to current observations, allowing procedural knowledge to support different models without updating their weights. Reuse concerns the conditions under which a step applies, rather than the objects or values in a demonstration. This division lets the harness resolve familiar dependencies while the model interprets the task and explores choices that experience leaves open.

Toward a systems view of harness design. Cybernetics, information theory, and general system theory offer complementary perspectives on harness design. Ashby's requisite variety concerns the responses available to the whole regulator (Ashby, 1956): a harness can supply supported responses while preserving judgment over unresolved choices. Information theory emphasizes uncertainty and representation (Shannon, 1948), raising the question of which evidence must reach the model and which can remain in external state. General system theory connects these choices through component interactions (von Bertalanffy, 1950). Information flow, decision allocation, and execution structure should therefore be optimized jointly. The SCC results suggest that harness design should consider where procedural decisions are made, as well as how much experience a model retrieves. They support assigning locally checkable dependencies to the harness while retaining model judgment for search and interpretation.

Limitations and future directions. SCC depends on identifiable states and manually adapted grammars, bindings, and guards. The suites informed development, and source-ID exclusions do not establish independent generalization. Current comparisons do not fully separate experience control from generic guards or a purely scripted policy. Independent tasks are needed to assess whether the state representation and decision allocation transfer beyond the evaluated procedures, including whether useful exploration is preserved when evidence is incomplete. Hybrid has the highest overall point estimates here, but does not significantly exceed Soft on WS or lead every task category.

6 Conclusion

SCC converts trajectory relations into state-conditioned guidance, restrictions, and execution without weight updates. It exceeds four offline memory adaptations on the evaluated suites and reuses its fixed design across two additional models. Ablations support additional hard-control gains on TS and ALF, with uncertain WS gains. These findings position the allocation between model judgment and external execution as a harness design variable governed by task and evidence.

AI USE STATEMENT

Generative AI assisted research hypotheses and conceptual framing, experimental design, implementation and debugging, and analysis of trajectories and results. It also assisted literature review, drafting, translation, and revision of text, figures, tables, and citations. Large language models supplied part of the offline experience, alongside programmatic experts; evaluated small models and the TS user simulator were experimental components. Outcomes use native environment scoring, without an LLM judge. The human authors are responsible for the final claims, attribution, code, and submission.

REPRODUCIBILITY STATEMENT

Sections 3–4.1 describe the method and protocol. Appendices A–F document sources, overlap, settings, implementation boundaries, results, and costs. The anonymous repository linked in the abstract supplies implementation and evaluation source code. The manuscript package supplies plotting and statistical inputs and a synthetic mechanism check. Full evaluation additionally requires benchmark and model installation, local configuration, and expert experience inputs.

ETHICS STATEMENT

Evaluations use benchmark environments without actions in real user accounts. Runtime controllers use public observations, not hidden targets or rewards; offline experts may use training-goal information. Local legality and evidence checks do not establish deployment safety or authorize consequential real-world actions. We report development exposure, evaluated scope, uncertainty, and failed-run costs.

REFERENCES

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Wang. Agent S: An Open Agentic Framework that Uses Computers Like a Human. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/394c7c30ea87b5c3521b4d9e9d419071-Paper-Conference.pdf.

W. Ross Ashby. *An Introduction to Cybernetics*. Chapman & Hall, London, 1956. URL https://ashby.info/Ashby-Introduction-to-Cybernetics.pdf.

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small Language Models are the Future of Agentic AI. arXiv:2506.02153, 2025. URL https://arxiv.org/abs/2506.02153.

Ali Eslami and Jiangbo Yu. A Control-Theoretic Foundation for Agentic Systems. arXiv:2603.10779, 2026. URL https://arxiv.org/abs/2603.10779.

Jizhan Fang, Xinle Deng, Haoming Xu, Ziyan Jiang, Yuqi Tang, Ziwen Xu, Shumin Deng, Yunzhi Yao, Mengru Wang, Shuofei Qiao, Huajun Chen, and Ningyu Zhang. LightMem: Lightweight and Efficient Memory-Augmented Generation. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a05b72653ec5b473732129829ae04195-Paper-Conference.pdf.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring Agent Procedural Memory. arXiv:2508.06433, 2025. URL https://arxiv.org/abs/2508.06433.

Gemma Team, Aishwarya Kamath, Johan Ferret, et al. Gemma 3 Technical Report. arXiv:2503.19786, 2025. URL https://arxiv.org/abs/2503.19786.

Pengfei He, Yuping Lin, Shen Dong, Han Xu, Yue Xing, and Hui Liu. Red-Teaming LLM Multi-Agent Systems via Communication Attacks. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025. URL https://aclanthology.org/2025.findings-acl.349/.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/36b7acf6f6010652b3f2a433774a66fe-Paper-Conference.pdf.

Taeil Kim, Kangsan Kim, and Sung Ju Hwang. Agent Memory Distillation: Empowering Small LLM Agents with Hierarchical Teacher Memory. arXiv:2608.07169, 2026. URL https://arxiv.org/abs/2608.07169.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A Stateful, Conversational, Interactive Evaluation Benchmark for LLM Tool Use Capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T. Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. ReasoningBank: Scaling Agent Self-Evolving with Reasoning Memory. arXiv:2509.25140, 2025. URL https://arxiv.org/abs/2509.25140.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992, 2019. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410/.

Yu Shang, Yu Li, Keyu Zhao, Likai Ma, Jiahe Liu, Fengli Xu, and Yong Li. AgentSquare: Automatic LLM Agent Search in Modular Design Space. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/0ae94013da7cd459402fd77874e09ee3-Paper-Conference.pdf.

Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x. URL https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

Kangning Shen, Jingyuan Zhang, Chenxi Sun, Wencong Zeng, and Yang Yue. Structurally Aligned Subtask-Level Memory for Software Engineering Agents. arXiv:2602.21611, 2026. URL https://arxiv.org/abs/2602.21611.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Ludwig von Bertalanffy. An Outline of General System Theory. *The British Journal for the Philosophy of Science*, 1(2):134–165, 1950. doi: 10.1093/bjps/I.2.134. URL https://doi.org/10.1093/bjps/I.2.134.

Xinrun Wang, Chang Yang, He Zhao, Zhuoyi Lin, and Shuyue Hu. The Agent Use of Agent Beings: Agent Cybernetics Is the Missing Science of Foundation Agents. arXiv:2605.10754, 2026. URL https://arxiv.org/abs/2605.10754.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Benlong Wu, Yuang Qi, Xiuwei Shang, Weiming Zhang, Nenghai Yu, and Kejiang Chen. MMPro: A decoupled perception-thinking-execution framework for secure GUI agent. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 4679–4687. Association for Computing Machinery, 2025a. doi: 10.1145/3746027.3755553. URL https://doi.org/10.1145/3746027.3755553.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d813d324dbf0598bbdc9c8e79740ed01-Paper-Conference.pdf.

Yiran Wu, Tianwei Yue, Shaokun Zhang, Chi Wang, and Qingyun Wu. StateFlow: Enhancing LLM Task-Solving through State-Driven Workflows. arXiv:2403.11322, 2024. URL https://arxiv.org/abs/2403.11322.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic Memory for LLM Agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 17577–17604. Curran Associates, Inc., 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/19909c36f51abc4856b4560aff3d36d6-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 Technical Report. arXiv:2505.09388, 2025a. URL https://arxiv.org/abs/2505.09388.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik A Chaudhari, George Karypis, and Huzefa Rangwala. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents. In *International Conference on Learning Representations*, 2025b. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/f2c6e459b95694a24ac69c469a4ee746-Paper-Conference.pdf.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, 2024. doi: 10.18653/v1/2024.findings-acl.624. URL https://aclanthology.org/2024.findings-acl.624/.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, XiongHui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating Agentic Workflow Generation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5492ecbce4439401798dcd2c90be94cd-Paper-Conference.pdf.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Pu Liang. MEM1: Learning to Synergize Memory and Reasoning for Efficient Long-Horizon Agents. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=XY8AaxDSLb.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Huajun Chen, and Ningyu Zhang. KnowAgent: Knowledge-Augmented Planning for LLM-Based Agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3709–3732. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.205. URL https://aclanthology.org/2025.findings-naacl.205/.

A Data Sources and Evaluation Scope

A.1 Experience Sources and Evaluated Suites

The study evaluates decision allocation in tasks with reusable procedures. The suites informed benchmark and model selection and refinements to guidance and action constraints, so they are not a new independent holdout. The comparisons assess the fixed final system on these suites; cross-model runs assess reuse with a different small model. Current SCC uses 200 available TS expert trajectories, 60 programmatic ALF training demonstrations, and 64 WS training goals. Executable structures retain 95 TS graphs with 195 nodes and 52 ALF graphs with 141 nodes. ALF demonstrations are training-expert programs, not LLM rollouts; the pool contains 52 successful and eight failed demonstrations. For WS, 62 goals have complete expert trajectories. The deduplicated baseline branch pool has 107 trajectories (75 successful, 32 failed). Native option-deletion checks yield 32 positive and two neutral pairs; these branches are not concatenated into fictitious natural correction traces. Training experts may use training-goal information, but runtime controllers cannot read hidden target identifiers or reward internals.

TS contains 515 locally executable scenarios selected by external-interface dependencies from the 1,032-scenario inventory. This selection does not constitute full-inventory evaluation. ALF uses all 134 `valid_unseen` games; training demonstrations come from the separate 3,553-game training inventory. WS uses all 500 official test goals and the complete product catalogue, with two executions per goal. Public-state fields and parsing grammars are adapted to each environment.

The main table uses six exclusive TS task strata (56 contacts, 49 messaging, 103 other queries, 100 reminders, 85 state-dependent, and 122 insufficient-information scenarios), six ALF types (18 examine, 24 place, 31 clean, 21 cool, 23 heat, and 17 two-object games), and five WS groups (110 beauty/personal care, 94 electronics, 102 fashion, 103 home/garden, and 91 grocery goals). WS groups are joined after evaluation using native catalogue order, deduplication, and shuffle seed 233. Hidden category metadata does not enter the agent or controller. These groupings support descriptive analysis, not category-specific model selection.

Table 1 uses the following category labels. TS: contacts (C), messaging (M), other queries (Q), reminders (R), state-dependent (S), insufficient information (I). ALF: examine (E), place (P), clean (Cl), cool (Co), heat (H), two objects (T). WS: mean reward (Rew), beauty/personal care (B), electronics (E), fashion (F), home/garden (H), grocery (G).

A.2 Source-ID Overlap and Sensitivity

The frozen TS source-task manifest identifies 95 evaluation IDs that also appear among experience sources. We exclude exactly these IDs, without selecting on outcomes, and recompute current SCC against the four archived memory adaptations on the remaining 420 scenarios. All paired identifiers and template labels match. The nonoverlap subset contains 79 template clusters; the overlap subset

contains 95. Their cluster counts need not sum to the full suite's 95 because variants of a template can occur in both subsets.

Table 3: Current TS sensitivity to exact source-ID overlap.

Subset	Comparator	SCC	Comparator	Gain [95% interval]
420 nonoverlap	AMD	56.54	43.75	$+12.78$ $[6.17, 19.61]$
	RB	56.54	40.30	$+16.23$ $[9.04, 23.72]$
	MemP	56.54	39.03	$+17.51$ $[10.04, 25.29]$
	SASM	56.54	39.58	$+16.95$ $[11.31, 22.85]$
95 overlap	AMD	67.69	53.11	$+14.58$ $[6.49, 22.82]$
	RB	67.69	47.20	$+20.49$ $[12.71, 28.68]$
	MemP	67.69	43.26	$+24.43$ $[16.96, 32.51]$
	SASM	67.69	48.29	$+19.40$ $[13.16, 26.29]$

Paired percentile intervals use 10,000 template-cluster resamples with seed 20260925, retaining scenario-weighted means. They describe sampling uncertainty within each subset, without multiplicity correction. Excluding exact source IDs preserves the comparison's direction, but related task templates and prior development exposure remain. This is a sensitivity analysis, not evidence of independent or unseen-family generalization.

B Settings and Adapted Baselines

The named comparisons share available expert sources, decoding, and native budgets for each task and small model. We freeze weights, experience, and rules; only public interaction state changes online. Inference uses RTX 4090 GPUs and vLLM (Kwon et al., 2023). A fixed Qwen3-8B user simulator serves all TS small models and does not score tasks. Native feedback uses no LLM judge. Budgets count native execution, including automatic interventions. Separate auxiliary model calls are retained in usage.

Table 4: Fixed execution settings.

Item	Setting
Main small model	Qwen3-8B (Yang et al., 2025a), frozen weights
Additional small models	Qwen3-4B and Gemma-3-4B-IT (Gemma Team et al., 2025), frozen controller and experience
Decoding	Temperature 0; seed 42; maximum 1,024 generated tokens per request
Templates	Qwen thinking disabled; Gemma native chat template
ToolSandbox budgets	20 small model action decisions; 40 native tool calls; original scenario-defined message limits
ALFWorld budget	50 native actions
WebShop budget	15 native actions
Automatic execution	TS/ALF: at most 8 consecutive steps; at most 2 applications per rule and controller instance
User simulator	Fixed Qwen3-8B on ToolSandbox; not a scoring judge
Cross-model serving	vLLM 0.8.5; 32,768 context; maximum one active sequence per service; eager execution; Gemma BF16
Compute	Four RTX 4090 GPUs; three or four services according to batch
Memory	Frozen MiniLM encoder; 2,048-token cap for named baselines

The Qwen3-4B revision is `1cfa9a7208912126459214e8b04321603b3df60c`. Model hashes, service arguments, and frozen configuration/source hashes are retained with the evaluation archive. The final cross-model evaluation reuses the fixed controller without outcome-dependent retries or further rule updates.

SCC guidance budget. TS checks that each structured packet contains fewer than 2,048 tokens; ALF checks at most 2,048. These checks reject oversized packets rather than clipping them. WS serializes the current page's option fields and pending selections without a separate guidance-token cap. The common native-action and output-token budgets remain fixed, but the memory constraint is therefore not identical for every adapter. Total measured input tokens include this guidance.

B.1 What the named adaptations implement

Method names in the main tables and plots denote these offline expert-transfer adaptations. Frozen parameters exclude training but do not require disabling memory updates or test-time scaling. Freezing experience is an additional controlled-protocol choice for comparing uses of common sources, not a substitute for evaluating the original methods with all key capabilities.

Table 5: Offline expert-transfer adaptations.

Method	Implemented retrieval and memory	Scope distinction
AMD	Workflow top-1; subtask similarity threshold 0.45 on the available initial request; same-function top-2 after errors.	No added online LLM task decomposer; frozen hierarchical store.
RB	Title/description/content lessons; semantic top-3 at the first turn.	No MaTTS or test-time memory writeback.
MemP	Procedural scripts and facts; one small model operation/object/constraint extraction per episode; AveFact top-3.	Facts extraction is an extra charged call; no online memory learning.
SASM	Functional stage filtering and semantic top-1; small model stage declarations inform the next retrieval.	Environment-adapted stage interface, without the original evolution pipeline.

All named baselines share the available expert sources and the fixed `all-MiniLM-L6-v2` encoder (Reimers & Gurevych, 2019), with a 2,048-token memory cap. In cross-model evaluation, a Qwen3-8B reference tokenizer applies the memory cap so that changing the small model does not change the retrieved content through different clipping. Actual usage is measured with each small model's native tokenizer. Tokens across different model families are not a common measure of compute. SCC matches normalized public goals and interfaces on TS, while ALF retrieves within a process type and rebinds the goal. Named baselines use their respective retrieval. System comparisons therefore differ in representation, routing, prompts, and execution. The current factorial comparison isolates channel-level changes within SCC, while the external comparisons assess complete systems.

B.2 Current Records and Comparison Provenance

The factorial evaluation contains 6,596 executions: four configurations on TS515 and ALF134 once, and WS500 twice. Current cross-model evaluation adds 3,298 SCC executions for two 4B models. Qwen3-8B main and cross-model summaries reuse the factorial evaluation's 1,649 hybrid records; they are not independent replications. External memory results are archived executions on matching task sets, not reruns inside the factorial core.

Table 6: Provenance of the reported comparisons.

Evidence	Fixed or shared	What differs
Current four configurations	Within-environment core, task IDs, initial states, decoding, sources and native budgets	Soft and hard switches; visited states and prompts
Current SCC vs memory methods	Task IDs, available sources, decoding and native budgets	Representation, routing, prompts, execution and serving batch
Cross-model SCC	Experience, controller and task sets	Small model and native template/tokenizer
Direct large-model reference	Evaluation task sets	Model, reasoning/output budget, inference service; one WS repeat

The paired initial-state hashes match in every paired group, with no missing episode records or unknown small-model usage. WS pairs retain repetition identifiers. Identical initialization and deterministic decoding do not ensure identical outputs across serving batches. The archived baselines also implement different retrieval and auxiliary-call interfaces. These facts delimit what each contrast measures.

B.3 Reproducibility Materials

The anonymous repository linked in the abstract provides SCC controller, experience-construction, benchmark-evaluation, and baseline-adapter source code, together with dependency and licence files. Running evaluations requires separate benchmark and model installations, local configurations, and expert experience inputs. The public code release does not bundle these inputs or individual evaluation trajectories. The manuscript package separately supplies aggregate and per-task statistical inputs, plotting scripts, and a synthetic mechanism check for compilation, current-state guidance, rebinding, and ambiguity rejection.

C Compilation and Runtime Semantics

The evaluated implementation uses project-written Python controllers and a standard-library control dispatcher. Environment adapters supply public-state parsing, dependency and binding checks, and native execution.

C.1 Manual Definitions and Automatic Extraction

The compiler and controller are implemented in Python. They translate trajectories into a predefined intermediate representation, rather than synthesizing arbitrary programs. Table 7 separates manual design from data-derived content. The same process graph supplies soft guidance and the hard channel's prerequisites, parameter sources, and outstanding subgoals. Executable rules therefore combine experience with a manually specified interpreter.

Manual components are fixed for each recorded evaluation. Automatic processing requires no additional generative-model call.

Table 7: Responsibilities in compilation.

Stage	Manually defined	Derived from trajectories or current feedback
Syntax and types	Action parsers, state fields, tool schemas, timestamp types, goal slots	Actions, arguments, returns, and parseable goal attributes
Dependencies	Templates for return references, request literals, date transforms, and ordering	Which node supplies a parameter and which demonstrated stages are retained
Admission	Success criteria, local replay checks, contradiction rejection	Source success labels, feasible bindings, retained subgoals
Soft guidance	JSON fields, serialization, and goal-slot templates	Completed nodes, current bindings, next verifiable step
Hard control	Matching, arbitration, interface checks, quotas	Whether public evidence supports a candidate and its actual execution effect

Return references and conservative ordering. In the tool environment, compilation removes terminal and error steps and deduplicates by action, arguments, and contemporaneous observation. For each parameter, the base rule scans prior returns in reverse order and selects the first return containing exactly one leaf with the same type and value. It stores the node and JSON path; uniqueness is not required across all historical returns. Booleans may remain constants. A string must match the public request directly, or it remains unresolved. For unresolved numbers, the extension tries exact numeric equality across numeric types, a number explicitly in the request, or a same-name UTC date field computed from a declared timestamp. Multiple candidate sources prevent binding at each stage. Edges follow these references and preceding state changes. They encode supported execution order, not a recovered minimal causal graph.

Successful traces do not directly establish valid rules. In the tool environment, graphs use the prefix for the first public user goal. The source must pass native success checks. Retained subgoals must be completable and bindable during replay of source observations, and the graph must not contain contradictory targets for a state field. Failed replay branches receive no execution authority. Of 200 sources, 142 pass native state validation; compilation retains 95 graphs, 195 nodes, and 119 goal nodes covering 45 distinct public goals. Replay establishes compatibility with the records, not correctness or necessity in every future state. The retained TS graphs average 2.05 nodes, and the 52 ALF graphs average 2.71. These are short procedural fragments: their completion can leave search, dialogue, or other task steps unresolved. Failed trajectories do not enter executable graphs. Whether memory baselines retain failed experience follows their adaptation definitions.

Household and page adapters. The household compiler retains successful, legal, effective demonstrated operations relevant to the goal object, treatment, and destination. Goal slots replace demonstrated object identifiers. Consecutive retained stages follow demonstration order; navigation routes are removed and the small model handles search. The shopping adapter uses a manually defined option parser and a rule to select a missing option before purchase. Training deletion pairs admit the rule only with at least five positive pairs and no negative pairs or execution errors. The observed 32 positive and two neutral pairs pass this threshold. Neither adapter discovers arbitrary control programs from experience.

Execution gates and arbitration. Execution permission combines state, binding, and interface checks. The tool adapter matches normalized public goals. When multiple graphs match, it orders them by source identifier and selects the first noncontradictory, interface-compatible graph. In stored order, it recursively selects the first missing dependency of an outstanding goal. The household adapter retrieves by semantic similarity within the same process type, breaking ties by source identifier. Direct execution requires one stage-compatible native candidate grounded in the current object binding and public observation. The current page adapter identifies uniquely matched missing options and restricts the schema; it does not fill them automatically. There is no model vote or calibrated confidence score for conflict resolution. Ambiguous local bindings abstain; multiple usable graphs follow fixed priorities.

Latest public queries determine tool state. Household completion requires legal execution and non-failure feedback. Completing a graph does not establish whole-task completion. Each automatic step rechecks state; errors, stale evidence, absent candidates, or exhausted budgets stop that segment. Logs cover actual execution and some blocked changes, but not every unmatched or unnecessary intervention, so they do not support a full frequency breakdown of abstention reasons.

C.2 A TRACE-TO-GUIDANCE-TO-EXECUTION EXAMPLE

The following uses the real node structure of the archived "remove my upcoming reminder" graph, with simplified timestamps and identifiers. It illustrates data flow, not a new expert trajectory or evaluation result. The supplementary program calls the original compiler on an explicitly synthetic fixture and then binds different runtime returns.

Demonstrated values become references to current execution results.

Table 8: Compiling a three-step experience.

Node	Demonstrated action and result	Compiled parameter	Dependency
0	Read current time; return 100.0	No argument	None
1	Search reminders from 100.0; return one record, ID `r7`	Lower bound ← return of node 0	0
2	Remove `r7`; operation succeeds	`reminder_id` ← node 1, `[0].reminder_id`	1

If current execution returns time 200.0 and the unique reminder `r9`, the next candidate removes `r9`, not the demonstrated `r7`. After the prerequisite nodes complete, the current-step constructor emits the following fields. The full packet also records public effects, available actions, and fixed instructions:

```
completed_steps: [0, 1]
current_step: {node: 2, operation: remove_reminder,
  dependencies: [1], action: {action: remove_reminder,
    arguments: {reminder_id: r9}}}
remaining_dependency_horizon: []
```

Soft mode gives this suggestion to the small model, which still chooses the call. Hard mode also checks the current interface, argument schema, and quota. If they pass, it can execute the action directly and write native feedback into the interaction history. Without a unique bindable record, node 2 has no execution authority and the small model handles the ambiguity. A completed removal alone does not establish conversation success. The current stopping policies are specified separately in Appendix C.4.

Minimal executable check. The supplementary material includes original Python modules, the archived graph, and clearly marked synthetic inputs for compilation, soft packets, rebinding, and ambiguity rejection without a model or benchmark service. This checks agreement between implementation and description; formal evaluation supports the environment-level results.

C.3 EXPERIENCE SCALE AND TASK-INSTANCE GENERALIZATION

Table 9 reports the distinct source tasks and trajectories retained by each method. The row labels give evaluation task counts; percentages divide retained source tasks by that count and express relative scale, not train–test overlap. The three environments contain 1,149 distinct evaluation task IDs. Two runs per WebShop goal yield 1,649 executions for each small-model method. Methods share the source pools in Appendix A, then apply their own selection and representation. For SCC, the counts refer to sources retained in executable graphs on TS/ALF and to the 34 paired intervention checks on WS, which comprise 68 trajectories from 28 training goals.

Table 9: Scale of retained experience sources across methods.

Environment	Metric	AMD	RB	MemP	SASM	SCC
ToolSandbox (515)	Tasks (%)	77 (15.0)	105 (20.4)	77 (15.0)	105 (20.4)	52 (10.1)
	Traces	142	200	142	200	95
ALFWorld (134)	Tasks (%)	52 (38.8)	60 (44.8)	52 (38.8)	60 (44.8)	52 (38.8)
	Traces	52	60	52	60	52
WebShop (500)	Tasks (%)	62 (12.4)	62 (12.4)	62 (12.4)	62 (12.4)	28 (5.6)
	Traces	75	107	75	107	68

The results support a degree of task-instance generalization within reusable procedures. SCC uses graphs from 52 ALF training tasks to act in 134 `valid_unseen` games, and dependency evidence from 28 WS training goals to support 500 test goals. In TS, 95 of the 515 evaluation IDs occur in the common source pool; excluding them leaves 420 scenarios on which SCC retains an advantage (Appendix A.2). The retained SCC graphs themselves come from 52 source tasks, 50 of which occur in evaluation, but the shared-pool exclusion gives all methods the same comparison scope. Together with fixed-controller reuse across small models, these results show that experience can support more instances than those retained as demonstrations. The evidence concerns reuse beyond the retained source instances within covered procedures; shared templates and prior development use, described in Appendix A, limit claims about unseen task families or independent holdout generalization.

C.4 Channel Semantics and Implemented Safeguards

None disables both SCC channels while retaining the environment's basic legal-action interface. Soft adds the current-step packet. Hard enables dynamic action schemas and supported automatic execution without that packet. Hybrid enables both. ALF's hard interface also includes a fixed notice to select an allowed action that advances the workflow; therefore the hard-only contrast is not a pure manipulation of decoding with absolutely no accompanying text. All four configurations use the same public observation interface, but their visited states and histories can diverge.

ALFWorld. The compiler retains 52 successful procedure graphs and 141 nodes from 60 training demonstrations. The current packet identifies the stage, held or bound object, publicly observed locations, and pending operations. It distinguishes inspected containers from locations merely visited and supplies fixed search, take, process, place, or use instructions. Hard schemas retain search operations and restrict object operations to the current stage, object type, binding, and required processor or destination. Unrecognized stages or an empty stage filter fall back to the native candidates; `FINISH` remains available. A repeated-action guard detects A, A or A, B, A and removes recent actions where alternatives remain. If that removal empties the list, the guard restores stage-compatible choices. The inherited proactive controller can execute up to eight uniquely supported steps before another small model decision. These grammars, templates, fallbacks, and repetition heuristics are manually implemented.

ToolSandbox. The source pool has 200 trajectories, compiling to 95 replay-checked graphs with 195 nodes. Current public goals and interfaces determine graph applicability. The soft packet exposes the current verified step and a window of at most three subsequent nodes, with their dependencies but without inventing future argument bindings. No additional workflow memory is supplied in any of the four configurations. Proactive execution can take up to eight supported steps. The schema implementation can retain the pending operation and public read operations, but all observed mask events in these runs are termination guards: 227 completed-plan cases plus 168 stable-repeat cases for hard, and 227 plus 194 for hybrid. A completed verified plan with no pending goal, or two identical public calls and identical observations when no graph controller is available, restricts the next choice to `FINISH`. That choice triggers a charged model final answer and may be followed by a user reply. A stable repeat is a heuristic, not proof of native task success. Thus the measured TS effect combines automatic execution, stopping policy, and guidance; it cannot be attributed to intermediate candidate pruning alone.

WebShop. The packet reports the current public page and uniquely matched pending options. Hard schemas exclude buying while required options remain unresolved, reject incompatible options or changes to a correctly selected option, and block repeated clicks under the same short-cycle pattern.

When cycle filtering removes all candidates and search is unavailable, cycle-blocked choices can be restored while dependency constraints remain. Search strings remain open when a search bar is available. The option-before-purchase rule uses the previously collected 32 positive and two neutral native counterfactual pairs. The parser and guards are manually specified. No proactive execution or reactive replacement occurs in these records: the observed hard mechanism is schema restriction. Hidden target identifiers, reward internals, and category metadata are not supplied to the controller.

C.5 Execution Pseudocode

	State-conditioned supervisor loop
1	Load fixed experience and adapter; initialize public state and budgets.
2	Read observation, update confirmed facts, and invalidate stale bindings.
3	If hard is enabled and the adapter supports a uniquely executable pending step, execute it, charge its native cost, and return to step 2 (up to eight automatic steps).
4	If soft is enabled, serialize the current-state guidance packet.
5	Construct native legal candidates; if hard is enabled, apply dependency and adapter guards with the documented fallback.
6	Request and charge a model action using that schema and any guidance.
7	Execute the selected action, charge native use, and record its actual result.
8	Update effects and pending dependencies. Continue from step 2 until native termination or budget exhaustion; errors return unresolved decisions to the model.

Figure 5: Common information flow; adapters instantiate state fields, guards, and supported automatic steps. WS has no automatic branch in the evaluated implementation.

D Current Outcomes, Statistics, and Decision Allocation

Within each environment, the four configurations use a shared frozen core, source pool, task identifiers, reset conditions, and decoding settings. Automatic steps consume the same native-action budget. All small-model requests, including final answers where applicable, count toward resource use. No online expert, LLM judge, memory writeback, or parameter update is added. Appendix B gives fixed budgets and Appendix C.4 gives the actual channel operations.

D.1 Outcomes and Paired Uncertainty

TS uses the native mean score on a 0–100 scale. Native score equal to one is reported separately as full credit, not complete conversational success. ALF and WS use native binary success; WS reward is an additional graded metric. Table 10 retains all four configurations.

Table 10: Unified four-configuration outcomes.

Configuration	TS score	TS full	ALF count	ALF %	WS count	WS %	WS reward
None	41.09	50/515	14/134	10.45	228/1000	22.80	60.30
Soft	48.56	66/515	77/134	57.46	319/1000	31.90	64.53
Hard	56.50	77/515	91/134	67.91	277/1000	27.70	58.92
Hybrid	58.59	85/515	110/134	82.09	326/1000	32.60	65.36

Paired 95% percentile intervals use 10,000 bootstrap resamples with seed 20260922. TS resamples 95 source-template clusters and retains the scenario-weighted mean. ALF resamples 134 games; WS resamples 500 goals with both repeats kept together. Differences are computed before rounding. These intervals describe task or template sampling, not repeated-seed stability or development-selection uncertainty; no multiplicity correction is applied. The factorial interaction is hybrid minus soft minus hard plus none.

Table 11: Paired score and success-rate differences.

Contrast	TS score	ALF pp	WS pp
Soft − None	$+7.47\ [+3.84, +11.55]$	$+47.01\ [+38.06, +55.97]$	$+9.10\ [+6.10, +12.10]$
Hard − None	$+15.41\ [+10.59, +20.66]$	$+57.46\ [+49.25, +65.67]$	$+4.90\ [+2.90, +7.00]$
Hybrid − None	$+17.51\ [+12.57, +22.83]$	$+71.64\ [+64.18, +79.10]$	$+9.80\ [+6.90, +12.70]$
Hybrid − Hard	$+2.10\ [+0.40, +4.17]$	$+14.18\ [+7.46, +21.64]$	$+4.90\ [+2.20, +7.60]$
Hybrid − Soft	$+10.04\ [+5.90, +14.53]$	$+24.63\ [+17.16, +32.84]$	$+0.70\ [-0.30, +1.70]$
Interaction	$-5.37\ [-8.89, -2.26]$	$-32.84\ [-43.28, -22.39]$	$-4.20\ [-6.50, -2.00]$

Hard control improves performance over None in all three environments. Hybrid has the highest aggregate point estimates, but this does not imply a uniformly best action policy. On TS full credit, hybrid minus hard is only 1.55 pp with interval $[0.00, 3.55]$. On WS success, hybrid minus soft is 0.70 pp with interval $[-0.30, 1.70]$. All three interaction estimates are negative, indicating subadditivity on the reported score or success-rate scale. This does not identify mechanism overlap. In ALF, the additive prediction from the rounded single-channel rates is approximately 114.92%, above the attainable maximum. The metric bound therefore forces a negative interaction even at perfect Hybrid success. Paired task transitions give a more direct account of which outcomes change. ALF hybrid gains 22 successful games and loses three relative to hard. Figure 4 shows the corresponding task-group variation; those subgroups are descriptive, not separately tuned controllers.

D.2 TOOLSANDBOX OUTCOME DEFINITIONS

TS mean score is continuous native credit, including final-response matching. Full credit means native score exactly one. State completion requires every applicable non-final-text state milestone and has a denominator of 393 scenarios. A logged normal termination means `agent_finished` or `official_conversation_end`; it does not imply correctness of the final response or the whole task. Table 12 keeps these quantities separate.

Table 12: Current TS score, full credit, state completion, and logged termination.

Config.	Score	Full credit /515	State /393	Normal end /515
None	41.09	50	145	235
Soft	48.56	66	189	222
Hard	56.50	77	197	468
Hybrid	58.59	85	202	460

The remaining termination records are message-limit endings (280, 292, 47, and 55 respectively) plus one model/runtime error in Soft. Normal termination is especially sensitive to the stopping guard, so its increase must not be reported as the increase in completed tasks.

D.3 OBSERVED DECISION ALLOCATION

A mask event is a small model decision where the permitted set differs from the base schema. An automatic action is a native step executed without a small model proposal. Neither count is policy entropy. Table 13 reports totals; masked-decision denominators are all schema-construction opportunities, and episode counts require at least one effective mask. No post-proposal replacement occurs in these runs. ALF's mean fraction of finite candidates removed, including `FINISH` and unfiltered decisions, is 6.86% for hard and 6.44% for hybrid. These describe different visited states, not an isolated causal effect of a fixed cardinality reduction. WS search arguments remain open, so finite candidate counts do not characterize its full action space.

Table 13: Observed allocation of action decisions.

Env.	Config.	Masked decisions	Episodes	Automatic	Native actions
TS	Hard	395/1508	395/515	444	2155
TS	Hybrid	421/1700	421/515	444	2355
ALF	Hard	1789/2603	122/134	397	2996
ALF	Hybrid	1288/2093	123/134	484	2575
WS	Hard	996/4745	327/1000	0	4745
WS	Hybrid	711/3783	297/1000	0	3783

D.4 Task Categories and Paired Gains

Figures 6 and 7 supplement Table 1 with paired gains and marginal category intervals. The six TS strata, six ALF task types, and five WS product groups cover all evaluated tasks. Intervals use 10,000 percentile-bootstrap resamples with seed 20260922. TS resamples source templates with all interface variants, ALF resamples games, and WS resamples goals with both executions. Category intervals describe marginal means, whereas gain intervals use within-task paired differences. These task-sampling intervals are not estimates of cross-seed variance and have no multiplicity correction.

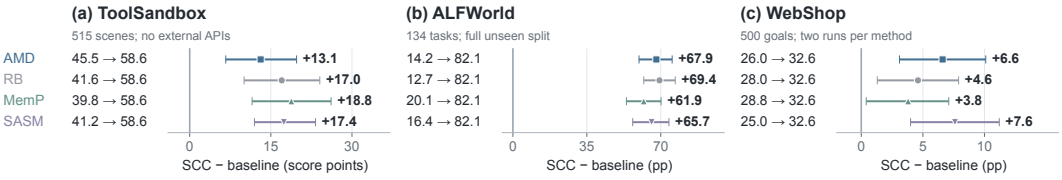

Figure 6: **Paired gains over memory baselines.** Points and bars show SCC minus each baseline with 95% bootstrap intervals.

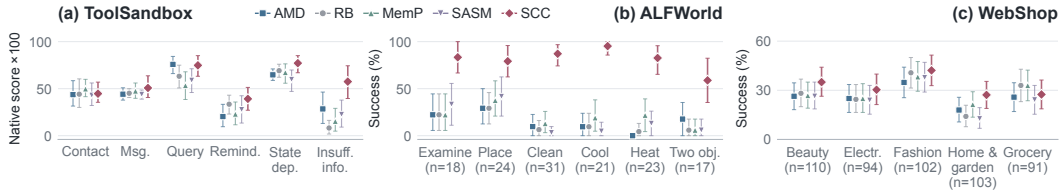

Figure 7: **Task-category results.** Points and bars show means and marginal 95% bootstrap intervals. Category means match Table 1.

E Current Cross-Model and Expert Comparisons

E.1 Frozen Reuse across Three Environments

The current hybrid is evaluated with Qwen3-4B and Gemma-3-4B-IT on all three environments, adding 3,298 episodes. Each model runs TS515 and ALF134 once and WS500 twice. Qwen3-8B reuses 1,649 hybrid records from the factorial evaluation. The experience, rules, task sets, temperature 0, seed 42, 1,024-token output limit, and native budgets are fixed. Qwen thinking is disabled; Gemma uses its native template. TS always uses Qwen3-8B for user simulation. The audit matches initial-state hashes, ALF retrieved-graph hashes, and WS training-evidence hashes to the 8B reference. No outcome-dependent retries are used.

Only hybrid was newly evaluated on the 4B small models. Existing fixed TS memory-baseline records are reused for Figure 3, with task identifiers paired; ALF and WS have no corresponding 4B baseline reruns. The latter therefore report absolute SCC performance. Main-table 8B comparisons likewise pair current SCC with the frozen baseline archive. Differences in system components are

part of the method comparison; all four channel settings themselves share the same environment core.

Table 14: Current SCC across small models.

Small model	TS score	TS full	ALF count	ALF %	WS count	WS %	Reward
Qwen3-8B	58.59	85/515	110/134	82.09	326/1000	32.60	65.36
Qwen3-4B	58.20	85/515	88/134	65.67	388/1000	38.80	65.99
Gemma-3-4B-IT	48.21	101/515	65/134	48.51	297/1000	29.70	57.25

Gemma's WS denominator retains 66 terminal format failures, two HTTP 400 request rejections, and six worker interruptions without a response. The first two categories may involve constrained-decoding compatibility; they are not all attributed to semantic reasoning. These failure records can leave token usage unknown, so they do not support complete cross-model cost comparisons.

E.2 Radar Scale

Figure 3 uses a separate piecewise radial mapping for each model and category. Raw scores from zero to the corresponding no-memory Base score occupy radii zero to one quarter; scores from Base to the best displayed method occupy the remaining radius. Spoke labels report the Base and best scores. Base uses the current None configuration for Qwen3-8B and the recorded no-memory references for the two 4B models. The mapping preserves order within each spoke, but distances and polygon areas do not compare absolute gains across spokes or models. Overall summaries use raw task-weighted scores.

E.3 Paired Main and Cross-Model Differences

Table 15 reports current SCC minus each fixed memory adaptation. TS intervals resample source templates, ALF games, and WS goals retaining both repeats. There are 10,000 resamples with seed 20260922, no multiplicity correction, and no interpretation as seed variance. The TS memory comparisons span separate archived execution batches.

Table 15: Paired gains over memory adaptations.

Small model / env.	Baseline	Gain	95% interval
8B / TS	AMD	$+13.12$	$[6.67, 19.76]$
8B / TS	RB	$+17.02$	$[10.06, 24.05]$
8B / TS	MemP	$+18.79$	$[11.54, 26.14]$
8B / TS	SASM	$+17.41$	$[11.97, 23.27]$
8B / ALF	AMD	$+67.91$	$[59.70, 75.39]$
8B / ALF	RB	$+69.40$	$[61.94, 76.87]$
8B / ALF	MemP	$+61.94$	$[53.73, 70.15]$
8B / ALF	SASM	$+65.67$	$[56.72, 73.88]$
8B / WS	AMD	$+6.60$	$[3.10, 10.10]$
8B / WS	RB	$+4.60$	$[1.30, 7.90]$
8B / WS	MemP	$+3.80$	$[0.40, 7.10]$
8B / WS	SASM	$+7.60$	$[4.00, 11.20]$
Qwen4B / TS	AMD	$+17.93$	$[9.97, 25.93]$
Qwen4B / TS	RB	$+16.50$	$[9.17, 24.19]$
Qwen4B / TS	MemP	$+20.53$	$[13.42, 27.74]$
Qwen4B / TS	SASM	$+19.30$	$[12.89, 25.73]$
Gemma4B / TS	AMD	$+22.86$	$[16.08, 29.74]$
Gemma4B / TS	RB	$+22.54$	$[15.05, 29.75]$
Gemma4B / TS	MemP	$+29.66$	$[23.04, 36.22]$
Gemma4B / TS	SASM	$+19.42$	$[13.16, 25.60]$

E.4 Direct Large-Model Reference

The DeepSeek `deepseek-v4-pro` reference uses thinking with high reasoning effort and at most 8,192 output tokens per request. Small models use at most 1,024 output tokens, with Qwen thinking disabled. The large model directly executes all 515 TS scenarios, 134 ALF games, and 500 WS goals once. Small models use the same task sets, with two executions per WS goal. All terminal failures remain in the denominator, including 51 TS, 29 ALF, and 11 WS model-output failures. ALF and WS experience also includes programmatic experts, so this reference is not the sole source of SCC's experience.

Table 16: Direct large model and Qwen3-8B SCC.

Environment	Large model	SCC	Difference	95% interval
ToolSandbox	78.05	58.59	-19.45	$[-27.64, -11.14]$
ALFWorld	61.19	82.09	$+20.90$	$[10.45, 30.60]$
WebShop	37.20	32.60	-4.60	$[-9.10, 0.00]$

TS reports native score times 100; ALF/WS report success percentages. Differences are SCC minus large model. Paired intervals use 10,000 bootstrap resamples with seed 20260922, over TS source templates, ALF games, or WS goals. WS retains both small model runs and the single large model run within each sampled goal. These intervals describe task-sampling uncertainty, not repeated large model runs.

On ALF, SCC completes 110 games versus the large model's 82, while the large model leads on TS. WS distinguishes graded reward from success: SCC receives reward 65.36 versus 61.28 but completes fewer goals. These results concern the full agent systems, not general superiority of the smaller underlying model.

Qwen3-4B SCC exceeds the large-model point estimate by 4.48 ALF points $[-7.46, 15.67]$ and 1.60 WS points $[-3.00, 6.20]$; neither establishes superiority. Gemma's primary point estimates are lower in all three environments (Table 14). Evidence for higher SCC success is strongest for Qwen3-8B on ALF. Remote large-model API latency is not an equal-hardware comparison with local small models.

F Resource Accounting

F.1 Resource and Decision-Allocation Accounting

All episode means include failures. Costs cover small-model input/output tokens, requests, and rollout time; TS simulator usage is separate. Token/outcome divides total tokens by summed TS scores on 0–1 or ALF/WS successes, not expected retry-until-success cost. Cross-batch times are observational. Table 17 uses thousands of tokens and seconds.

Table 17: Small model costs in the unified evaluation.

Env.	Config.	Calls	Input	Output	Total	Time	Token/outcome
TS	None	9.39	11.602	0.370	11.97	11.04	29.14
TS	Soft	9.84	14.526	0.371	14.90	11.37	30.68
TS	Hard	3.89	2.911	0.117	3.03	3.95	5.36
TS	Hybrid	4.24	4.041	0.133	4.17	4.48	7.12
ALF	None	32.52	20.813	0.496	21.31	23.45	203.96
ALF	Soft	30.31	27.660	0.451	28.11	24.75	48.92
ALF	Hard	19.43	13.295	0.280	13.58	15.71	19.99
ALF	Hybrid	15.62	14.313	0.219	14.53	13.56	17.70
WS	None	3.65	5.465	0.053	5.52	2.21	24.20
WS	Soft	3.80	6.191	0.055	6.25	2.34	19.58
WS	Hard	4.75	9.108	0.071	9.18	6.21	33.14
WS	Hybrid	3.78	6.045	0.055	6.10	4.87	18.71

The TS user simulator adds the separate totals in Table 18. Its waiting time is included in rollout time, but its tokens are not small model tokens.

Table 18: Additional TS user-simulator usage.

Config.	Calls	Input tokens	Output tokens	HTTP seconds
None	239	261,237	4,542	125.07
Soft	236	257,803	4,729	134.53
Hard	493	544,069	9,740	248.58
Hybrid	485	535,382	9,706	246.99

Including all small-model and user-simulator input/output tokens, TS None uses 6,430,992 tokens and Hybrid 2,694,663, a 58.10% reduction. The denominator is the None configuration, not AMD. This token comparison is neither a monetary saving nor a latency estimate; simulator calls actually increase from 239 to 485. Small-model main comparisons and the simulator should not be combined selectively.

F.2 Accounting Scope

Online costs exclude source collection, memory extraction, CPU compilation, and engineering. Compilation is deterministic and uses no generative calls; construction cost and amortization are unmeasured. Formal failures retain their outcomes and known usage, including two output failures among 4,000 WS baseline runs. Unknown Gemma usage prevents a complete cross-model cost ranking (Appendix E). Tokens, API charges, and engineering effort use different units.