FROM INTERVENTION TO SKILL: EVOLVING SMALL–LARGE LANGUAGE MODEL COLLABORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Small–large language model collaboration improves inference efficiency by assigning routine decisions to a smaller model and invoking a stronger model when additional capability is needed. Existing approaches primarily optimize this allocation for the current interaction, leaving the value of a large-model intervention largely transient. We study whether past interventions can instead enable collaboration to evolve. We introduce Intervention-to-Skill (I2S), which converts execution-supported large-model interventions into reusable skills. I2S localizes the decision opportunity at which stronger assistance is requested, grounds its effect in environment feedback, and uses trajectory context to identify a reusable procedural residual. The resulting skills persist across tasks and evolve with new evidence, allowing the effective small-model policy to improve while both acting models remain fixed. Across ALFWorld, WebShop, and ScienceWorld with Qwen3-8B paired with Qwen3-32B, DeepSeek-V4-Flash, and Gemini-2.5-Flash, I2S consistently improves task quality over strong routing and skill-based baselines. Compared with SkillGen, task-quality gains reach 27.1 score points and weighted token usage is reduced by up to 63.2%. As skills accumulate, task quality increases while large-model involvement decreases, and acquired skills are repeatedly reused in later small-model decisions. These results establish small–large collaboration as a cumulative adaptation process in which previous assistance reshapes the future division of labor between models. Code is available in the Supplementary Material.

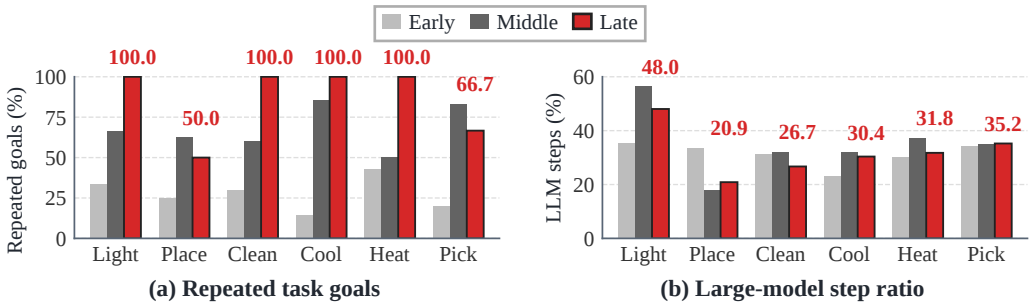
1 INTRODUCTION

Language model agents increasingly solve interactive tasks through iterative loops of reasoning, action, and environment feedback (Yao et al., 2023; Wang et al., 2024). Large language models provide strong language understanding and reasoning capabilities, but relying on them for every step of a long interaction trajectory incurs substantial inference cost (Chen et al., 2024; Lin et al., 2023). A practical alternative is small–large language model collaboration: a smaller model serves as the default executor, while a stronger model is invoked only when additional capability is needed, which we refer to as an *intervention* (Lin et al., 2023; Narayan et al., 2025). Since interventions are costly, the central challenge is to carefully invoke the larger model only where it provides sufficient value.

Existing work on small–large collaboration has largely explored how to allocate computation across models. These methods operate at two levels: request-level methods, such as cascades and routers, choose which model handles an entire request (Chen et al., 2024; Ding et al., 2024; Ong et al., 2025; Dekoninck et al., 2025), whereas step-level methods, such as confidence-aware approaches, make the same decision within a reasoning or interaction trajectory (Chuang et al., 2025; Zhang et al., 2026a; Kapoor et al., 2026; Zhang et al., 2026c). However, these methods optimize allocation only within the current interaction, and the benefit does not persist once that interaction ends. When a similar procedural difficulty appears again, the large model may therefore need to be invoked again for the same kind of assistance. An ideal collaborative system should not only allocate assistance efficiently, but also learn from the assistance it has already received. We therefore ask: *Can past interventions become reusable knowledge that reduces repeated large-model assistance on future tasks?*

The above issue becomes particularly important when related tasks recur over deployment. As shown in Figure 1, in the late stage, 86.1% of tasks on average repeat a goal seen earlier in the same task stream, yet large-model reliance does not decline correspondingly: even in the late stage, the large

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.



(a) Repeated task goals **(b) Large-model step ratio**

Figure 1: **Illustration of repeated tasks and large-model reliance in small–large language model collaboration.** Results are shown for AutoMix (Aggarwal et al., 2024) with Qwen3-8B+32B (Yang et al., 2025) on ALFWorld (Shridhar et al., 2021). Panel (a) shows the fraction of tasks whose goals have appeared earlier. Panel (b) shows the fraction of decision steps executed by the large model.

model still executes 32.2% of decision steps on average. Such recurrence creates opportunities to reuse knowledge from earlier assistance. Importantly, the reusable content is often not the intervention action itself, but an ordering relation, verification condition, or recovery rule revealed by that intervention. We characterize this reusable structure as a *procedural residual*. Capturing such residuals allows an intervention to contribute beyond the task on which it was originally needed.

Motivated by this gap, we introduce Intervention-to-Skill (I2S), which turns large-model interventions that produce observable environment feedback into reusable skills. The small-model proposal and large-model intervention identify the decision step at which stronger assistance was requested. Execution feedback establishes the local effect of the executed intervention, while the completed trajectory places that effect within the surrounding task procedure. Together, these signals allow I2S to identify the corresponding procedural residual and abstract it into a skill with observable applicability, an ordered procedure, and outcome verification. These skills persist across tasks and evolve as new evidence becomes available. Both acting models remain fixed throughout deployment, while accumulated skills change the effective small-model policy.

We evaluate I2S on ALFWorld, WebShop (Yao et al., 2022), and ScienceWorld (Wang et al., 2022), with Qwen3-8B as the small model and various large-model backends, including Qwen3-32B, DeepSeek-V4-Flash (DeepSeek-AI, 2026), and Gemini-2.5-Flash (Comanici et al., 2025). Experiments across these environments and large-model backends show that I2S consistently improves task quality over strong routing and skill-based baselines. Compared with SkillGen, the largest task-quality gain reaches 27.1 points and weighted token usage is reduced by as much as 63.2%. Beyond performance, the deployment trajectories show a progressive shift toward higher task quality and lower large-model involvement as skills accumulate. Acquired skills are repeatedly reused in later small-model decisions, while confidence calibration further improves selective intervention decisions. These results support a view of small–large collaboration as a cumulative adaptation process in which previous assistance changes the future division of labor between models.

In summary, our contributions are as follows:

- We formulate evolving small–large collaboration over ordered deployment streams, where procedural knowledge acquired from earlier interventions changes the effective small-model policy across tasks while both acting models remain fixed.
- We introduce I2S, which turns large-model interventions with observable environment feedback into reusable procedural skills by combining proposal–intervention contrast, execution feedback, trajectory context, and evidence-supported refinement across tasks.
- We evaluate evolving collaboration across interactive environments and large-model backends, demonstrating improved task quality and efficiency together with persistent cross-task skill reuse and reduced dependence on repeated large-model assistance.

2 Related Work

Efficient Small–Large Model Collaboration. Small–large model collaboration improves inference efficiency by allocating computation across models with different capability and cost profiles. Cascading and routing methods decide whether a request should be handled by a smaller or stronger model (Chen et al., 2024; Ding et al., 2024; Aggarwal et al., 2024; Ong et al., 2025; Dekoninck et al., 2025; Zhao et al., 2024b). More recent work extends this idea to finer-grained decisions through confidence-aware routing, adaptive escalation, and step-level coordination in reasoning and interactive tasks (Chuang et al., 2025; Zhang et al., 2026a; Kapoor et al., 2026; Zhang et al., 2026c). These methods primarily improve how model capacity is allocated during the current interaction. I2S studies a complementary dimension in which a large-model intervention can also create skills that remain available to future interactions.

Procedural Memory and Skill Learning. Language agents can also improve across tasks by retaining reusable experience outside the model parameters. Prior work stores verbal feedback and distilled experience (Shinn et al., 2023; Zhao et al., 2024a), organizes recurring interaction patterns into workflows or programmatic skills (Wang et al., 2025b;a), and develops structured procedural memories with mechanisms for retrieval, refinement, verification, and skill synthesis (Fang et al., 2026; Cao et al., 2026; Mi et al., 2026; Suzgun et al., 2026; Ma et al., 2026). Related systems further show that accumulated experience can support transfer and adaptation across tasks (Wang et al., 2024; Zheng et al., 2025; Ouyang et al., 2026; Yu et al., 2026). I2S acquires reusable skills from selectively triggered small–large interventions grounded in execution evidence, using each intervention to identify a reusable procedural residual.

Large-to-Small Capability Transfer. Knowledge distillation transfers capabilities from stronger models to smaller ones through parameter optimization using teacher supervision (Hinton et al., 2015; Hsieh et al., 2023; Gu et al., 2024; Ko et al., 2024). Recent approaches improve this transfer using model-generated rationales, student trajectories, and adaptive teacher supervision (Hsieh et al., 2023; Agarwal et al., 2024; Jang et al., 2026). On-policy distillation further reduces teacher–student distribution mismatch by training on states or outputs generated by the student itself (Agarwal et al., 2024; Zhang et al., 2026b; Jang et al., 2026), while related work explores stronger-to-weaker transfer in interactive and agentic settings (Nourzad & Joe-Wong, 2026; Yao et al., 2026). I2S retains the large-to-small transfer objective while keeping both acting models fixed during deployment. The knowledge exposed by large-model interventions is transferred through external skills that remain available to later small-model decisions. In this sense, I2S connects selective model collaboration with reusable skill acquisition, allowing stronger assistance to influence future small-model behavior without online parameter updates.

3 Problem Formulation

Deployment stream and skill state. We study small–large collaboration over an ordered stream of interactive tasks $x_{1:T} = (x_1, \ldots, x_T)$, where related tasks may share reusable procedural knowledge. A small language model M_s acts as the default executor, while a larger model M_l provides selective assistance. At the start of task t, the system carries an external skill state $K_t = \{\omega_i\}_{i=1}^{N_t}$ containing skills acquired from earlier collaboration. Both acting models remain fixed throughout deployment. Given the current task, interaction history $h_{t,j}$, and skill state, the effective small-model policy is

$$\pi_{s,t}^{\mathrm{eff}}\big(\cdot \mid x_t, h_{t,j}\big) = M_s\big(\cdot \mid x_t, h_{t,j}, K_t\big). \tag{1}$$

The collaboration can therefore evolve without changing the acting model parameters. What changes across deployment is the skill state available to the small-model path.

Execution and adaptation timescales. Execution and adaptation have distinct timescales. During task t, the skill state K_t remains fixed while the system interacts with the environment and records a trajectory τ_t and intervention events E_t. Cross-task adaptation occurs only after the task ends

$$(\tau_t, E_t) = \mathrm{Execute}_\pi\big(x_t, M_s, M_l, K_t\big), \tag{2}$$

$$K_{t+1} = \mathrm{Update}_\pi\big(K_t, \tau_t, E_t\big). \tag{3}$$

Within-task decisions therefore use a fixed snapshot of the available skill state. After each task, I2S uses the collected intervention evidence to acquire or refine skills for future tasks. Section 4 specifies how I2S turns supported intervention evidence into such updates.

Evolving collaboration objective. Conventional routing allocates model capacity given the skills currently available. Evolving collaboration also changes the skill state that will condition later decisions. Let $q_t(\pi)$ denote task quality and $w_t(\pi)$ the complete-stream weighted token usage attributed to task t. For a target quality level Q_{target}, we characterize efficient collaboration by

$$\min_{\pi \in \Pi} \frac{1}{T} \sum_{t=1}^{T} w_t(\pi) \qquad \text{s.t.} \qquad \frac{1}{T} \sum_{t=1}^{T} q_t(\pi) \geq Q_{\text{target}}. \tag{4}$$

The evaluation in Section 5 instantiates w_t using the complete-stream weighted token measure. Routing improves allocation under the current skill state, while evolving collaboration can change that state so that assistance acquired earlier reshapes the future division of labor between models.

4 Intervention-to-Skill

4.1 Framework Overview

Intervention-to-Skill (I2S) turns selectively acquired large-model assistance into reusable skills that can improve future collaboration. Figure 2 illustrates three I2S stages. During a task, action-conditioned selective intervention identifies small-model proposals that require stronger assistance. After task completion, trajectory-aware skill induction extracts the reusable procedural relation exposed by supported interventions. Across tasks, streamwise skill evolution carries the resulting skills forward and updates them as new evidence becomes available. Both acting models remain fixed during deployment, while accumulated skill state changes the effective small-model policy.

4.2 Action-Conditioned Selective Intervention

At decision step j of task t, let

$$u_{t,j} = (x_t, o_{t,j}, h_{t,j}, K_t)$$

denote the visible decision context. The small model first proposes $a^s_{t,j} \sim M_s(\cdot \mid u_{t,j})$. I2S evaluates the confidence of this specific proposal before deciding whether stronger assistance is required.

A lightweight confidence adapter is trained before deployment on disjoint small-model rollouts with `CONFIDENT` and `UNCERTAIN` labels derived from localized environment evidence. The adapter assigns the proposal confidence score

$$c_{t,j} = p_\phi\big(\texttt{CONFIDENT} \mid u_{t,j}, a^s_{t,j}\big).$$

Deployment uses the fixed threshold $\gamma = 0.99$. Large-model assistance is triggered when proposal confidence falls below this threshold or when an execution-side recovery condition is observed

$$d_{t,j} = \mathbb{I}\left[c_{t,j} < \gamma \ \vee \ f_{t,j} = 1\right], \tag{5}$$

where $f_{t,j}$ summarizes observable recovery signals. When $d_{t,j} = 0$, the small-model proposal is executed. When $d_{t,j} = 1$, the large model receives the current visible context and returns an intervention action $a^l_{t,j}$. Appendix A details confidence calibration, scoring, and the controller.

The proposal-first design also localizes the procedural gap exposed by an intervention. The pair $(a^s_{t,j}, a^l_{t,j})$ identifies the decision opportunity at which stronger assistance is requested and records whether the intervention confirms or changes the small-model proposal. For pre-execution interventions, the small-model proposal serves as a localization signal, while the executed large-model action and resulting environment transition provide the observable evidence used for skill induction.

4.3 Trajectory-Aware Intervention-to-Skill Induction

An intervention can reveal reusable procedural structure beyond the action that resolves the current decision. I2S combines three information sources. The small-model proposal and intervention action

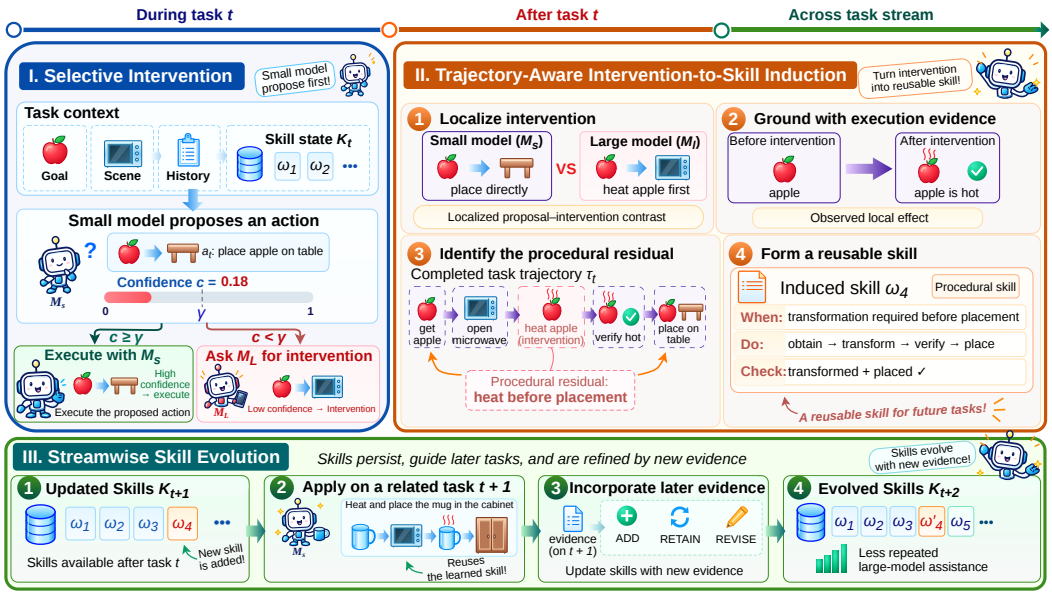


Figure 2: **Overview of Intervention-to-Skill (I2S).** Selective intervention identifies decisions that benefit from stronger assistance. Execution evidence grounds the local effect of an intervention, while trajectory context reveals the reusable procedural relation behind it. The resulting skill enters the reusable skill state and can be reused or refined on later tasks.

localize the relevant decision opportunity, execution feedback establishes the effect of the executed assistance, and the completed trajectory places that effect within the surrounding task procedure.

After executing $a^l_{t,j}$, I2S records an intervention anchor

$$e_{t,j} = \left(u_{t,j}, a^s_{t,j}, a^l_{t,j}, \xi_{t,j}\right), \tag{6}$$

where $\xi_{t,j}$ records the observed local effect of the executed intervention. An anchor becomes eligible for induction when the intervention produces a recognized environment transition. Recovery anchors may additionally include an executed small-model failure from the same decision opportunity. Let $\widetilde{E}_t$ denote the eligible anchors collected during task t.

The completed trajectory τ_t supplies the context needed to interpret these local effects. Earlier states can identify prerequisites that hold before an intervention, while later states can reveal ordering relations and verification conditions associated with its outcome. Intervention evidence determines what happened locally, and the trajectory places that event within a reusable procedure.

We call the reusable missing relation exposed by this evidence a *procedural residual*. A residual may encode a prerequisite, an ordering dependency, a verification condition, or a recovery relation. I2S abstracts source-specific entities into semantic roles and represents the resulting procedure as

$$\omega = \langle I_\omega, P_\omega, O_\omega \rangle, \tag{7}$$

where I_ω specifies observable applicability, P_ω specifies an ordered semantic procedure, and O_ω specifies the expected local outcome and its verification. The resulting skill captures the reusable subprocedure supported by the intervention while abstracting away source-specific details.

After task completion, I2S induces a candidate update from the completed trajectory, eligible anchors, and current skill state

$$\Omega_t = \mathrm{Induce}\left(\tau_t, \widetilde{E}_t, K_t\right). \tag{8}$$

Candidate applicability conditions, semantic operations, role bindings, and outcome checks are validated against their cited intervention evidence before admission. This grounding keeps newly acquired skills tied to observed execution effects while allowing trajectory context to recover the reusable relation connecting them. The full evidence representation, skill schema, induction procedure, and admission rules are provided in Appendix A.

4.4 STREAMWISE SKILL EVOLUTION

Skills remain available beyond their source task. At the start of each task, the active portion of K_t is supplied to the small-model path. The actor checks observable applicability and binds semantic roles to the current task before using a skill. Within each task, this state is fixed (Section 3).

After each task, new evidence can update the skills available to subsequent tasks

$$K_{t+1} = \mathrm{Update}_{\mathrm{I2S}}\big(K_t, \Omega_t\big). \tag{9}$$

An admitted update has three possible outcomes. ADD introduces a skill for a procedural residual not yet represented in the state. RETAIN preserves an existing skill when the new evidence supports content already represented. REVISE updates the evidence-implicated fields of an existing skill while preserving compatible procedural structure.

This update process closes the I2S loop. Selective intervention identifies where stronger capability is needed. Trajectory-aware induction converts supported assistance into reusable procedural knowledge. Streamwise evolution then makes that knowledge available to later decisions and incorporates new evidence as deployment continues. Collaboration can therefore evolve through earlier assistance while the parameters of both acting models remain fixed.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmarks and models. We evaluate I2S on ALFWorld, WebShop, and ScienceWorld. Qwen3-8B serves as the small actor and is paired separately with Qwen3-32B, DeepSeek-V4-Flash, and Gemini-2.5-Flash as the large-model backend. Evaluation follows ordered deployment streams in which the skill state persists across related tasks while the acting model parameters remain fixed. Full task construction, model configurations, and interaction budgets are provided in Appendix B.

Comparisons and token accounting. We compare I2S with Small-Only and Large-Only as capacity endpoints, AutoMix as a routing baseline, and ExpeL, Cheatsheet, and SkillGen as persistent experience or skill baselines. All methods use matched task sets and task orders within each backend. Task quality follows the native benchmark metric. We measure complete-stream weighted token usage as $W = L + 0.25S$, where L and S are charged large-model and small-model output tokens. The small-model weight of 0.25 follows the 8B-to-32B model-size ratio of the local Qwen pair and is kept fixed across all settings as a common normalization. The accounting includes model calls used for action generation, confidence assessment, large-model assistance, and cross-task skill updates. Baseline adaptations and accounting details are given in Appendices B.3 and B.4.

Research questions. Evaluation addresses four questions. **RQ1.** How does I2S affect task quality and complete-stream weighted token usage? **RQ2.** How does small–large collaboration change as skills accumulate over deployment? **RQ3.** Do acquired procedural skills persist and enter later small-model decisions? **RQ4.** Which intervention and skill mechanisms support evolving collaboration?

5.2 OVERALL PERFORMANCE AND TOKEN USAGE

I2S consistently improves task quality across collaboration settings. Table 1 shows that I2S achieves the highest or tied-highest score in all nine benchmark–backend settings. The gains are substantial in several cases. With Qwen3-32B on ScienceWorld, I2S reaches 73.9 compared with 58.0 for AutoMix and 62.3 for SkillGen. With Gemini-2.5-Flash, the corresponding scores are 83.3, 57.5, and 56.2. These results show that persistent skill acquisition improves the effective collaboration policy across different large-model backends.

The quality gains are usually accompanied by lower weighted token usage. Figure 3 compares task quality and W jointly. Relative to AutoMix, I2S uses fewer weighted tokens in seven of nine settings while achieving higher task quality in all nine. Relative to SkillGen, I2S uses fewer weighted tokens in eight of nine settings while again improving task quality in all nine. AutoMix captures selective routing, while SkillGen uses persistent skills with a different acquisition mechanism.

Table 1: **Task quality and complete-stream weighted token usage.** Seven methods share task sets and order within each backend. L, S, and W are reported in thousands of output tokens per task.

Method	ALFWorld				WebShop				ScienceWorld			
	Score↑	L↓	S↓	W↓	Score↑	L↓	S↓	W↓	Score↑	L↓	S↓	W↓
Qwen3-8B + Qwen3-32B												
Small-Only	67.9	0.0	34.4	8.6	56.3	0.0	5.4	1.4	46.5	0.0	36.7	9.2
Large-Only	85.1	13.7	0.0	13.7	57.9	3.4	0.0	3.4	60.3	11.5	0.0	11.5
AutoMix	76.9	6.1	31.6	14.0	54.6	2.1	7.1	3.9	58.0	7.9	26.2	14.4
ExpeL	89.6	5.1	23.5	11.0	54.0	3.0	8.8	5.2	46.8	8.7	30.8	16.4
Cheatsheet	79.9	7.2	31.0	14.9	47.6	4.5	10.2	7.1	56.1	10.5	35.0	19.2
SkillGen	82.8	6.4	33.0	14.7	60.8	2.8	9.1	5.1	62.3	10.1	28.5	17.3
I2S	**90.3**	5.0	13.6	**8.3**	**64.1**	1.3	3.2	**2.1**	**73.9**	8.9	8.1	**10.9**
Qwen3-8B + DeepSeek-V4-Flash												
Small-Only	67.9	0.0	36.7	9.2	56.3	0.0	4.5	1.1	51.9	0.0	32.7	8.2
Large-Only	98.5	7.1	0.0	7.1	71.1	3.1	0.0	3.1	87.0	29.0	0.0	29.0
AutoMix	94.0	6.0	17.6	10.4	67.5	1.9	6.5	**3.5**	72.2	22.1	20.5	27.2
ExpeL	99.3	2.9	14.6	6.5	67.0	3.8	6.5	5.4	87.6	12.3	12.6	15.5
Cheatsheet	**100.0**	7.4	11.3	10.2	69.8	9.3	6.7	11.0	83.6	8.8	8.3	**10.9**
SkillGen	97.8	5.1	17.3	9.4	69.9	4.6	9.7	7.0	87.8	19.6	18.5	24.3
I2S	**100.0**	3.8	9.1	**6.0**	**75.2**	5.9	2.8	6.5	**91.0**	30.1	6.0	31.6
Qwen3-8B + Gemini-2.5-Flash												
Small-Only	70.9	0.0	33.3	8.3	56.6	0.0	4.9	1.2	49.5	0.0	34.7	8.7
Large-Only	85.8	1.1	0.0	1.1	66.8	0.3	0.0	0.3	53.5	1.1	0.0	1.1
AutoMix	84.3	0.5	26.6	7.1	65.1	0.2	6.1	1.7	57.5	0.5	21.7	6.0
ExpeL	82.1	0.6	28.1	7.7	67.7	0.2	6.1	1.7	61.9	0.6	26.6	7.2
Cheatsheet	87.3	0.5	23.4	6.3	64.0	0.4	7.1	2.1	62.5	0.7	24.7	6.9
SkillGen	85.8	1.0	35.3	9.8	67.7	0.5	8.7	2.7	56.2	1.0	27.5	7.8
I2S	**94.0**	0.8	11.3	**3.6**	**75.5**	0.5	2.6	**1.1**	**83.3**	1.2	11.3	**4.0**

Improvements over both comparators support combining selective large-model intervention with reusable skills. Appendix C.1 details token composition and interaction statistics.

5.3 Evolution Across Deployment

Collaboration shifts toward higher task quality with less large-model involvement as deployment progresses. Figure 4 tracks I2S as skills accumulate across the task stream. At the aggregate level, later windows reach higher task scores while both large actions per task and L-Step decline from their midstream levels. The same broad pattern appears across the benchmark averages, although individual benchmark–backend trajectories retain their own variation. The collaboration policy evolves over deployment as reusable skills accumulate, allowing a larger rate of the interaction to be carried by the small-model path while maintaining or improving task quality.

The changing division of labor is central to streamwise adaptation. Because both acting models remain fixed during deployment, this shift is driven by skills acquired earlier in the stream and supplied to later small-model decisions. Earlier large-model interventions can therefore alter the operating point of future collaboration by contributing reusable skills. This turns selective assistance from a repeated per-task allocation decision into a cumulative adaptation process across the deployment stream. Appendix B.4 details the temporal aggregation and token accounting for Figure 4.

5.4 Persistent Skill Reuse

Procedural skills persist beyond the intervention that created them and are reused across later tasks. Figure 5 shows that the skill state continues to evolve through ADD, RETAIN, and REVISE updates throughout deployment. More importantly, admitted skills are repeatedly referenced after their source task, often across multiple later tasks. Repeated later references show that admitted skills function as reusable procedural knowledge across tasks. This persistence makes earlier large-model assistance available to the small-model path as deployment progresses.

Reused skills enter later autonomous small-model decisions. Across the main evaluation streams, 675 of 856 later skill-citing decisions are executed by the small model without large-model takeover. Among these executions, 519 receive a native environment acknowledgement aligned with the cited

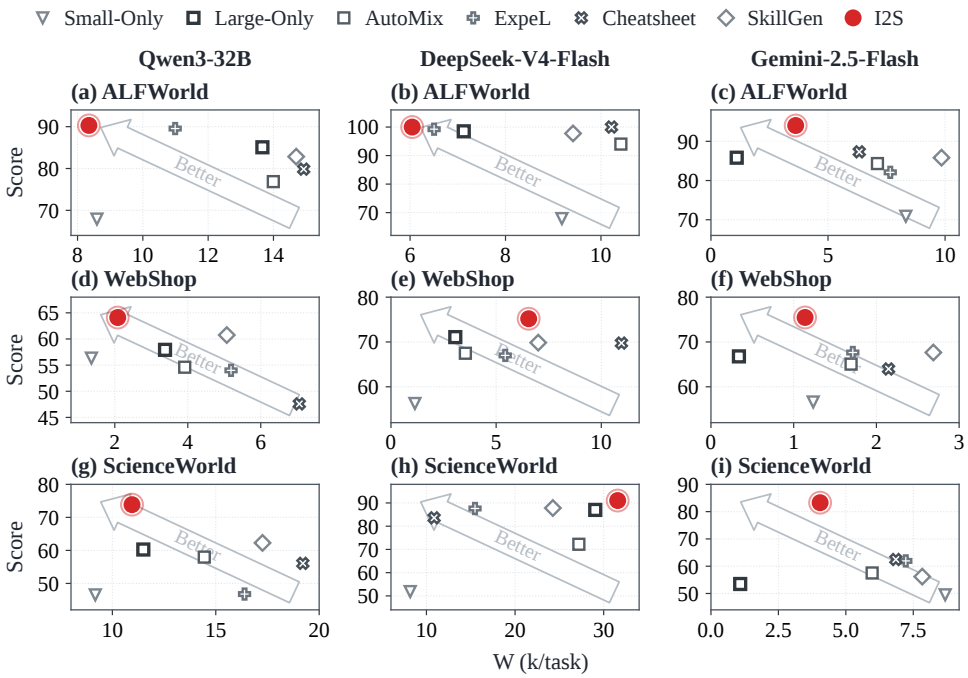


Figure 3: **Quality and weighted token usage.** Each panel compares the seven methods under one benchmark and large-model backend. Score retains the native benchmark metric and W denotes complete-stream weighted output tokens per task in thousands. Red rings identify I2S and arrows indicate the direction of higher quality and lower token usage.

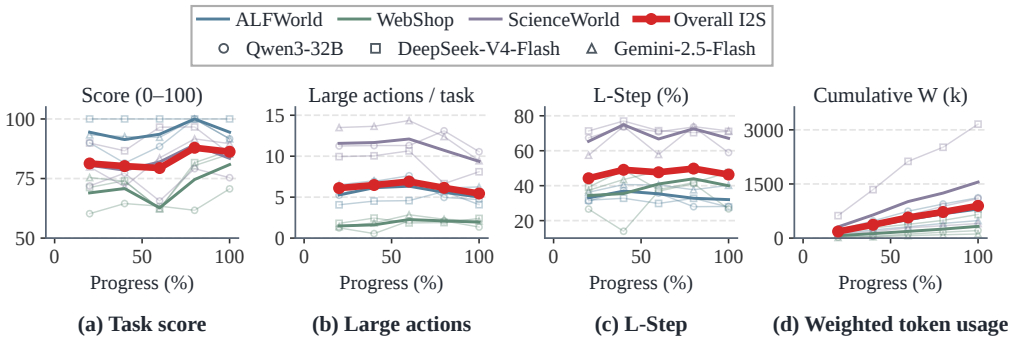


Figure 4: **The performance analysis on I2S over the deployment stream.** Faint lines represent individual benchmark–backend settings, medium lines average the three large-model backends within each benchmark, and the thick red line averages all nine settings.

skill operation and expected outcome. Acquired skills are thus retrieved, used in executed small-model actions. Benchmark-level reuse patterns appear in Appendix C.2, with representative skill acquisition, revision, and cross-task reuse trajectories in Appendix C.3.

5.5 Mechanism Analysis

Confidence calibration sharply improves proposal-level action assessment. Figure 6 shows that the calibrated scorer better separates supported and unsupported small-model proposals. ROC–AUC increases from 0.6488 to 0.9558, while ECE (Guo et al., 2017) decreases from 24.88% to 1.63%. The risk–coverage curve (Geifman & El-Yaniv, 2017) shows the same improvement across acceptance levels, allowing a larger fraction of small-model proposals to be accepted at low error. Calibrated proposal confidence thus better informs when large-model intervention is needed.

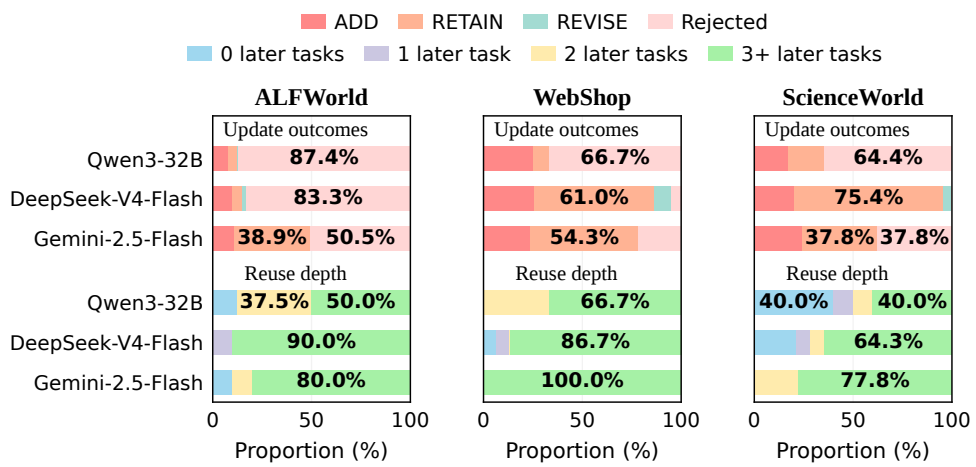
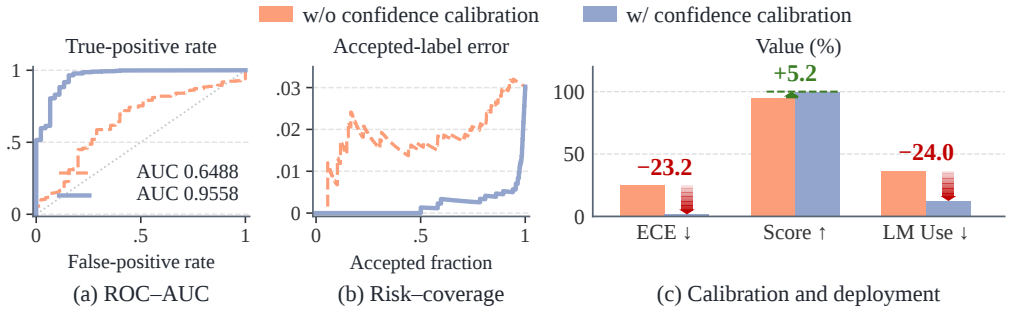


Figure 5: **Statistics of skill-state updates and cross-task reuse.** Upper bars show the distribution of candidate skill updates across ADD, RETAIN, REVISE, and rejection outcomes. Lower bars show the reuse depth of admitted skills. Columns correspond to the three benchmark environments, and rows correspond to the three large-model backends.



(a) ROC–AUC (b) Risk–coverage (c) Calibration and deployment

Figure 6: **Confidence calibration and deployment behavior.** All panels use the matched ALFWorld configuration with Qwen3-8B and DeepSeek-V4-Flash. Panels (a) and (b) evaluate proposal confidence. Panel (c) reports calibration error, task score, and large-model use.

Confidence calibration improves task quality while reducing large-model intervention. Across the full matched ALFWorld deployment, calibration increases task score by 5.22 percentage points while reducing LM Use by 24.01 percentage points. Confidence calibration therefore improves the intervention signal and changes the resulting division of labor between the two models. Appendix C.4 reports the corresponding matched ablation and token breakdown. Together with the persistent reuse results in Section 5.4, these findings identify two complementary mechanisms behind I2S. Confidence calibration provides the signal for deciding when stronger assistance is needed, while persistent skills carry useful information from that assistance into future decisions.

6 Conclusion

We study small–large language model collaboration as a process that can evolve through its own past assistance. I2S converts execution-supported large-model interventions into reusable skills, allowing knowledge acquired during one task to influence later small-model decisions while the acting models remain fixed. Across three interactive environments and multiple large-model backends, I2S achieves strong task quality and efficiency, while deployment trajectories show a broader shift toward higher task scores and less large-model involvement as skills accumulate. Acquired skills persist across tasks and enter later small-model actions, while confidence calibration helps determine when stronger assistance is still needed. Together, these results move small–large collaboration beyond repeated model allocation toward cumulative adaptation, where past interventions reshape the future division of labor between models.

Ethics Statement

This work studies the efficiency and adaptation of small–large language model collaboration in established interactive benchmarks. The experiments use pretrained language models and benchmark environments and do not involve a human-subject study or the collection of new personal data. The proposed method changes how model assistance is allocated and reused during deployment, while the underlying acting models remain fixed.

Reproducibility Statement

We provide detailed experimental and implementation information to support reproducibility. Appendix A specifies confidence calibration, intervention evidence, skill induction, state updates, and the complete I2S procedure. Appendix B describes the benchmark streams, model configurations, interaction budgets, baseline adaptations, evaluation metrics, and complete-stream token accounting. Appendix C reports additional results, skill-reuse analyses, representative skill-evolution cases, and mechanism ablations, while Appendix D provides the runtime prompt templates, benchmark-specific interfaces, and output schemas used in the experiments. Within each large-model backend, all compared methods use matched task identities and task orders. The code is available in the Supplementary Material.

AI Use Statement

Generative AI tools were used to assist with language polishing, manuscript editing, coding implementation, and the preparation of figures and tables. All AI-assisted material was reviewed and incorporated by the authors, who retain full responsibility for the technical content and final presentation of the paper.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 16803–16822. Association for Computational Linguistics, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route LLMs with confidence tokens. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 10859–10878. PMLR, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. arXiv preprint arXiv:2606.19348, 2026.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12987–13010. PMLR, 2025.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, 2024.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502. Association for Computational Linguistics, 2026.

Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8003–8017. Association for Computational Linguistics, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Ijun Jang, Jewon Yeom, Juan Yeo, Hyunggyu Lim, and Taesup Kim. Stable on-policy distillation through adaptive target reformulation. *arXiv preprint arXiv:2601.07155*, 2026.

Vansh Kapoor, Aman Gupta, Hao Chen, Anurag Beniwal, Jing Huang, and Aviral Kumar. TRIM: Hybrid inference via targeted stepwise routing in multi-step reasoning tasks. In *International Conference on Learning Representations*, 2026.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. DistiLLM: Towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 24872–24895. PMLR, 2024.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. SwiftSage: A generative agent with fast and slow thinking for complex interactive tasks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 23813–23825, 2023.

Yuchen Ma, Yue Huang, Han Bao, Haomin Zhuang, Swadheen Shukla, Michel Galley, Xiangliang Zhang, and Stefan Feuerriegel. SkillGen: Verified inference-time agent skill synthesis. *arXiv preprint arXiv:2605.10999*, 2026.

Qirui Mi, Zhijian Ma, Mengyue Yang, Yisen Wang, Haoxuan Li, Haifeng Zhang, and Jun Wang. ProcMEM: Learning reusable procedural memory from experience via non-parametric PPO for LLM agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Avanika Narayan, Dan Biderman, Sabri Eyuboglu, Avner May, Scott Linderman, James Zou, and Christopher Ré. Cost-efficient collaboration between on-device and cloud language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45682–45719. PMLR, 2025.

Narjes Nourzad and Carlee Joe-Wong. MIRA: Memory-integrated reinforcement learning agent with limited LLM guidance. In *International Conference on Learning Representations*, 2026.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations*, 2025.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long 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. In *International Conference on Learning Representations*, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, 2026. Association for Computational Linguistics.

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

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025a.

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, 2025b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

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.

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.

Yuhang Yao, Zeyu Wang, Wanyi Chen, Tongyun Yang, Yuhang Han, Jie Xiao, Chengke Bao, Tianyi Zhao, Lynn Ai, Eric Yang, and Tianyu Shi. MERA: Model evolution and routing with skill adaptation for agentic systems at scale. *arXiv preprint arXiv:2608.10333*, 2026.

Simon Yu, Gang Li, Weiyan Shi, and Peng Qi. PolySkill: Learning generalizable skills through polymorphic abstraction for continual learning. In *International Conference on Learning Representations*, 2026.

Chuang Zhang, Zizhen Zhu, Yihao Wei, Bing Tian, Junyi Liu, Henan Wang, Wang Xavier, and Yaxiao Liu. Confidence-calibrated small-large language model collaboration for cost-efficient reasoning. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational*

Linguistics (Volume 1: Long Papers), pp. 4480–4501. Association for Computational Linguistics, 2026a.

Dongxu Zhang, Zhichao Yang, Sepehr Janghorbani, Jun Han, Andrew Ressler, II, Qian Qian, Gregory D. Lyng, Sanjit Singh Batra, and Robert E. Tillman. Fast and effective on-policy distillation from reasoning prefixes. *arXiv preprint arXiv:2602.15260*, 2026b.

Yuxin Zhang, Mengxue Hu, Zheng Lin, Xiaoyi Fan, Fan Xie, Zihan Fang, Jing Yang, Wenjun Zhu, Zhiwen Chen, Chengfei Lv, and Zhe Chen. Hera: Learning long-horizon coordination for device-cloud collaborative LLM agents. *arXiv preprint arXiv:2605.24598*, 2026c.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (17):19632–19642, 2024a.

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. Eagle: Efficient training-free router for multi-LLM inference. *arXiv preprint arXiv:2409.15518*, 2024b.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Appendix

Appendix Contents

A Method Details **15**
A.1 Confidence Calibration for Selective Intervention 15
A.2 Intervention Evidence and Skill Induction 16
A.3 Skill State and Streamwise Evolution 16
A.4 Complete I2S Procedure 17

B Experimental Protocol and Implementation **18**
B.1 Evaluation Scope and Deployment Streams 18
B.2 Models and Collaboration Protocol 18
B.3 Baselines and Matched Comparisons 19
B.4 Metrics and Complete-Stream Accounting 19
B.5 Diagnostic Protocols 20

C Additional Experimental Results **21**
C.1 Complete-Stream Efficiency 21
C.2 Cross-Task Skill Reuse 22
C.3 Skill Evolution Cases 22
C.4 Ablation and Confidence Calibration 29

D Prompt Templates and Runtime Interfaces **30**

E Discussion **33**

A METHOD DETAILS

This appendix expands the I2S framework in Section 4 into its training, execution, induction, and state-update procedures. For task t, τ_t denotes the completed visible trajectory, $\widetilde{E}_t$ the eligible intervention anchors, and K_t the skill state supplied at task start. Algorithms A.1 and A.2 summarize the complete streamwise procedure.

A.1 CONFIDENCE CALIBRATION FOR SELECTIVE INTERVENTION

Proposal confidence. The confidence scorer estimates whether a proposed small-model action is a supported next step under the current visible decision context. Each calibration example is (u_i, a_i^s, z_i), where u_i contains the task, observation, admissible actions, interaction history, and supplied skill state, and $z_i \in \{\texttt{CONFIDENT}, \texttt{UNCERTAIN}\}$. `CONFIDENT` examples come from locally supported actions on successful trajectories. `UNCERTAIN` examples come from explicit execution failures and localized recovery events. The calibration set retains decisions with unambiguous local evidence.

The adapter is trained on 564 examples from 60 ALFWorld (Shridhar et al., 2021) tasks and validated on 513 examples from 26 disjoint tasks. Training updates the LoRA (Hu et al., 2022) parameters of Qwen3-8B (Yang et al., 2025) and keeps the actor backbone fixed. The resulting adapter is then used for proposal assessment on all three evaluation environments. The training objective is

$$\mathcal{L}_{\text{conf}} = -\frac{1}{N}\sum_{i=1}^{N} \log p_\phi\big(z_i \mid u_i, a_i^s\big). \tag{10}$$

Inverse-frequency class weights are normalized to unit mean sample weight. Table A.1 gives the optimization configuration.

Table A.1. Confidence adapter training configuration.

Model and training input		**Optimization**	
Backbone	Qwen3-8B	Optimizer	AdamW
Target modules	$q_{\text{proj}}, v_{\text{proj}}$	Learning rate	10^{-4}
Rank / scale	8 / 16	Weight decay	0.01
Dropout	0.05	Gradient clipping	1.0
Precision	bfloat16	Gradient accumulation	4
Input limit	8,192 tokens	Microbatch decisions	1
Random seed	7,311	Epochs / checkpoint	3 / final

The deployment input limits and fixed confidence threshold are specified in Appendix B.2, and the exact request templates are given in Appendix D.

Confidence scoring and intervention. At deployment, the scorer evaluates the complete `CONFIDENT` and `UNCERTAIN` label sequences. Let

$$\ell^+ = \log p_\phi\big(\texttt{CONFIDENT} \mid u, a^s\big), \qquad \ell^- = \log p_\phi\big(\texttt{UNCERTAIN} \mid u, a^s\big).$$

The normalized proposal confidence score is

$$c = \frac{\exp(\ell^+)}{\exp(\ell^+) + \exp(\ell^-)}. \tag{11}$$

Each decision begins with a small-model proposal. Proposals with $c_{t,j} \geq \gamma$ continue on the small-model path. Lower confidence or a parsing failure triggers large-model assistance while intervention budget remains. Execution feedback can also trigger recovery from the latest visible state. ALFWorld additionally uses a non-progress check based on recurring observation, action, and admissible-action sets, while goal-directed returns to a known source or destination remain permissible.

A large-model intervention may execute a bounded continuation from the triggering decision opportunity. The benchmark-specific intervention limits are specified in Appendix B.2. Throughout execution, the controller records the visible context, proposed and executed actions, environment feedback, and declared skill references.

A.2 Intervention Evidence and Skill Induction

Intervention anchors. I2S grounds skill acquisition in observed action–environment transitions. An eligible intervention anchor links an executed large-model action to a recognized change in the visible environment. Recovery anchors additionally connect a localized small-model failure with the supported intervention that follows at the same decision opportunity.

The recognized effects follow the native feedback of each benchmark. ALFWorld provides navigation, opening and closing, acquisition, placement, and object-transformation signals. WebShop (Yao et al., 2022) provides search results, product-page access, and acknowledgements of selected options. ScienceWorld (Wang et al., 2022) provides room arrival, container access, content revelation, and device activation. Each anchor records the executed operation, visible role bindings, and observed local outcome used by the induction procedure.

Procedural residual induction. After task completion, I2S combines the eligible anchors $\widetilde{E}_t$, completed trajectory τ_t, and current skill state K_t to produce

$$\Omega_t = \mathrm{Induce}\big(\tau_t, \widetilde{E}_t, K_t\big). \tag{12}$$

Repeated trajectory fragments are represented through a shared lossless reference pool, preserving their original content within the request budget.

The anchors establish the local effect of stronger assistance, while the completed trajectory supplies the task stage, entity bindings, and observed ordering needed to interpret that effect procedurally. The inducer identifies the corresponding *procedural residual*, such as a prerequisite, ordering dependency, verification condition, or recovery relation. Multiple compatible anchors can jointly support a partial procedure when they establish a consistent semantic relation.

Role abstraction and skill representation. I2S abstracts source-specific entities into semantic roles that can be rebound on later tasks. Each induced skill has the form

$$\omega = \langle I_\omega, P_\omega, O_\omega \rangle\,, \tag{13}$$

where I_ω specifies observable applicability, P_ω specifies an ordered semantic procedure, and O_ω specifies the expected local outcome and its verification. Each skill also carries a stable identity, content version, supporting anchors, and guarded fallback information. During execution, the actor binds semantic roles to the current task and realizes the next applicable operation through benchmark-native actions.

Evidence-grounded admission. Admission verifies that the proposed applicability conditions, role bindings, semantic operations, expected outcomes, and revision scope are supported by the cited anchors. A supported residual that is new to the state produces ADD. New evidence for existing content produces RETAIN, and evidence that refines an existing skill produces REVISE. The induction instruction, response schema, and validation interface are provided in Appendix D.

A.3 Skill State and Streamwise Evolution

The skill state tracks each skill's stable identity and current evidence-supported version. ADD creates a new skill identity. RETAIN preserves the current content and attaches additional support. REVISE records an updated content version under the same identity.

Capacity limits are specified in Appendix B.2. When the active-state limit is reached, older cards leave the supplied context first. Their identities, versions, and supporting evidence remain part of the recorded skill history.

At task start, the active cards are copied into a fixed task-local snapshot. The actor checks observable applicability, binds semantic roles to the current task, and records the identity and version of any skill that informs its action. The same snapshot is used throughout the task. Post-task skill induction incorporates newly collected evidence into the state supplied to task $t + 1$.

Algorithm A.1 Intervention-to-Skill over an ordered task stream

Input: Ordered tasks $x_{1:T}$; fixed models M_s, M_l; confidence scorer C_ϕ; threshold γ; action limits.
Output: External skill states $K_1, \ldots, K_{T+1}$ and recorded interactions.

$K_1 \leftarrow \varnothing$
for $t = 1, \ldots, T$ **do**
  $\bar{K} \leftarrow \mathrm{Snapshot}(K_t)$; $\tau_t \leftarrow \varnothing$; $\widetilde{E}_t \leftarrow \varnothing$
  $s \leftarrow \mathrm{Reset}(x_t)$; initialize the action-budget ledger b
  while task t is active and its environment budget remains **do**
    $u \leftarrow \mathrm{Context}(x_t, s, \bar{K}, \tau_t)$
    $a^s \leftarrow \mathrm{ParseAction}(M_s(u))$
    $g \leftarrow (a^s = \perp)$ **or** $[C_\phi(u, a^s) < \gamma]$
    if g and the large-model task budget remains **then**
      $\chi \leftarrow \mathrm{Intervene}(M_l, u, b)$
    else
      $\chi \leftarrow \mathrm{ExecuteOrStop}(a^s, b)$
      if $\mathrm{RecoveryTriggered}(\chi)$ and assistance remains **then**
        $\chi \leftarrow \chi \,\|\, \mathrm{Recover}(M_l, x_t, \bar{K}, \tau_t \,\|\, \chi, b)$
      end if
    end if
    $\tau_t \leftarrow \tau_t \,\|\, \chi$; $s \leftarrow \mathrm{LastVisibleState}(\chi)$
    $\widetilde{E}_t \leftarrow \widetilde{E}_t \cup \mathrm{EligibleAnchors}(\chi)$
    Record the proposal, declared skill references, and all model-call usage
  end while
  $K_{t+1} \leftarrow \mathrm{AdmitUpdate}(\tau_t, \widetilde{E}_t, K_t)$
end for
return $K_{1:T+1}$ and the recorded interactions

Algorithm A.2 ADMITUPDATE: induction and state admission

Input: Trajectory τ; eligible anchors E; state K; fixed inducer M_l; induction limits.
Output: Skill state available to the next task.

if $E = \varnothing$ **then return** K
$q \leftarrow \mathrm{PackLosslessly}(\mathrm{Visible}(\tau), E, \mathrm{Active}(K))$
if the complete request q exceeds its input budget **then return** K
$\Omega \leftarrow \mathrm{Induce}(M_l, q)$; charge the request to its source task
if Ω has no well-formed candidate **then return** K
$e \leftarrow \mathrm{CheckEvidence}(\Omega, E)$
if e is invalid **then return** K
$\omega \leftarrow \mathrm{ValidateContent}(\Omega, e, K)$
if ω is invalid **then return** K
$o \leftarrow \mathrm{ResolveUpdate}(\Omega, \omega, K)$
if $o =$ ADD **then**
  $K' \leftarrow \mathrm{Add}(K, \omega)$
else if $o =$ REVISE **then**
  $K' \leftarrow \mathrm{Revise}(K, \omega)$
else
  $K' \leftarrow K$
end if
Attach e to the admitted identity and record the update provenance
return $\mathrm{EnforceCapacity}(K')$

A.4 COMPLETE I2S PROCEDURE

Algorithm A.1 summarizes streamwise execution under a fixed task-start skill snapshot. Algorithm A.2 summarizes post-task induction, evidence validation, and state update.

B EXPERIMENTAL PROTOCOL AND IMPLEMENTATION

This appendix specifies the evaluation design underlying the main results.

B.1 EVALUATION SCOPE AND DEPLOYMENT STREAMS

We evaluate I2S in three environments that expose complementary forms of procedural decision making. ALFWorld (Shridhar et al., 2021) emphasizes grounded household manipulation with recurring action dependencies. WebShop (Yao et al., 2022) evaluates goal-directed web interaction through search, inspection, option selection, and purchase decisions. ScienceWorld (Wang et al., 2022) requires multi-step scientific procedures in which intermediate observations determine later actions. Together, these benchmarks span embodied, web, and scientific interaction, with different action spaces, feedback signals, and opportunities for cross-task procedural reuse.

The evaluation contains 334 tasks organized into 15 scenario streams. Each stream groups tasks that share a broad procedural structure, creating repeated opportunities for acquired skills to become relevant while preserving variation in goals, entities, and observations.

- **ALFWorld** uses all 134 official `valid_unseen` tasks across six families: Examine in Light with 18 tasks, Pick and Place with 24, Clean and Place with 31, Cool and Place with 21, Heat and Place with 23, and Pick Two and Place with 17. These families cover navigation, object acquisition, transformation, and placement under different procedural requirements.
- **WebShop** uses 100 test tasks from Beauty, Electronics, Fashion, Garden, and Grocery, with 20 tasks per domain. The five domains preserve a common interaction interface while varying products, attributes, and goal constraints.
- **ScienceWorld** uses 100 test tasks from Plant Growth, Melting Point with a known substance, Melting Point with an unknown substance, and Conductivity Testing, with 25 tasks per family. These families provide diverse scientific procedures with distinct ordering and verification requirements.

Tasks are evaluated in a fixed order within each scenario. Skill state persists across tasks in the same stream and is reinitialized at the start of the next scenario. All compared methods receive the same ordered task identities within a backend, providing matched deployment streams for quality, model-use, and token comparisons.

WebShop uses the full catalog, human goals, and upstream retrieval backend. ScienceWorld uses the official text parser with the `easy` simplification and compact valid-action presentation. Confidence-adapter training is described separately in Appendix A.1.

B.2 MODELS AND COLLABORATION PROTOCOL

Qwen3-8B (Yang et al., 2025) serves as the small actor throughout the evaluation. We pair it with Qwen3-32B (Yang et al., 2025), DeepSeek-V4-Flash (DeepSeek-AI, 2026), and Gemini-2.5-Flash (Comanici et al., 2025). The Qwen pairing evaluates collaboration within one model family at different scales, while the DeepSeek and Gemini pairings evaluate collaboration across model families. Proposal confidence is assessed by the frozen calibrated Qwen3-8B described in Appendix A.1.

For the local Qwen configuration, the small and large actors use 32,768-token input limits and 8,192-token output limits. Confidence scoring uses a 16,000-token input limit. Post-task skill induction uses a 24,576-token input limit and reserves up to 16,384 output tokens. Actor and induction requests enable model thinking and use temperature 0.6, top-p 0.95, top-k 20, and presence penalty 1.5. Deployment uses the fixed confidence threshold $\gamma = 0.99$.

Interaction budgets. The action budgets reflect the typical interaction length of each benchmark. ALFWorld permits at most 50 environment actions, 16 large-model actions per task, and four large-model actions per intervention. WebShop uses limits of 15, eight, and three. ScienceWorld uses limits of 100, 30, and four. Large-model and small-model actions share the same environment horizon.

Visible interaction history is bounded at 24,000 estimated tokens for ALFWorld, 12,000 for WebShop, and 20,000 for ScienceWorld. The active skill state contains at most six cards and 2,800 estimated tokens, with up to 900 estimated tokens and eight semantic operations per card. Post-task skill induction receives the complete lossless induction input within its configured request budget.

B.3 Baselines and Matched Comparisons

The comparison set is designed to separate model capacity, selective routing, and persistent cross-task knowledge.

- **Small-Only** is the standalone small-model endpoint and uses Qwen3-8B as the sole actor throughout each task.
- **Large-Only** is the standalone large-model endpoint and uses the assigned large-model backend throughout each task.
- **AutoMix** (Aggarwal et al., 2024) provides the routing comparison. It uses its own entailment and self-verification mechanism with threshold 0.50 to choose between the small and large actors.
- **ExpeL** (Zhao et al., 2024a) represents persistent experience-based memory. Our adaptation maintains an insight bank from the latest eight task outcomes and updates it every four tasks through the method's add, agreement, edit, and removal operations.
- **Cheatsheet** (Suzgun et al., 2026) represents editable cross-task guidance. It updates the persistent guidance after every task through up to three schema-valid add, revise, or remove edits.
- **SkillGen** (Ma et al., 2026) provides the closest persistent-skill comparison. Every fourth task, it induces a candidate from recent successful and failed trajectories. Candidate quality is evaluated through paired replay under candidate-present and candidate-absent conditions, and a positive success gain admits the skill. One refinement round is available for a second candidate.

Persistent-state baselines retain their original acquisition and update logic within the shared deployment protocol. SkillGen replay is included in its complete-stream weighted token usage, while primary task quality is computed from the deployment stream itself.

Matched mechanism controls. The full-ALFWorld mechanism study evaluates three matched configurations on the same 134 tasks, task order, and seeds. Full I2S uses confidence calibration and reusable skills. *w/o confidence calibration* uses the corresponding uncalibrated proposal scorer with the same intervention and skill-update pipeline. *w/o skills & calibration* combines the uncalibrated scorer with an empty skill state throughout deployment. These controls separate the contribution of confidence calibration from the contribution of reusable skills.

B.4 Metrics and Complete-Stream Accounting

The evaluation measures three complementary aspects of collaboration. Task quality captures endpoint performance, large-model involvement characterizes the division of labor during interaction, and weighted token usage measures the model computation consumed across the complete deployment stream.

Task quality. Score follows the native benchmark metric. ALFWorld uses task completion percentage, WebShop reward is multiplied by 100, and ScienceWorld reports its native 0–100 score. Benchmark results average task-level scores over the reported stream.

Large-model involvement. L-Step measures the fraction of executed environment actions produced by the large model. For task t, let H_t denote all executed environment actions and ℓ_t the subset produced by the large model. We compute

$$\text{L-Step} = 100\frac{\sum_t \ell_t}{\sum_t H_t}.$$

LM Use complements L-Step in the matched mechanism study by measuring the fraction of small-model proposal opportunities that trigger large-model intervention. A single intervention can execute multiple large-model actions, so the two metrics capture intervention frequency and executed large-model share at different levels.

Weighted token usage. Let L and S denote charged large-model and small-model output tokens. Complete-stream weighted token usage is

$$W = L + 0.25S.$$

The coefficient 0.25 follows the 8B-to-32B model-size ratio of the local Qwen pair and is fixed across all backends as a common normalization.

For task t,

$$L_t = \sum_{r \in \mathcal{R}_t^l} L_r^{\text{out}}, \qquad S_t = \sum_{r \in \mathcal{R}_t^s} S_r^{\text{out}}, \tag{14}$$

and $W_t = L_t + 0.25S_t$. Reported per-task usage averages W_t across the stream.

The accounting follows backend-reported output tokens and includes small-model action generation, confidence assessment, large-model intervention and recovery, post-task skill induction, baseline state updates, and auxiliary verification when applicable. Provider-accounted reasoning output is counted once as part of the reported output usage.

Streamwise aggregation. Skill state evolves independently within the six ALFWorld, five WebShop, and four ScienceWorld scenarios. For Figure 4, each scenario is divided into five chronological windows according to normalized within-scenario progress. Benchmark curves average the three large-model backends, and the overall curve gives equal weight to the nine benchmark–backend settings. Cumulative W sums all charged model usage attributed to the completed prefix.

B.5 Diagnostic Protocols

Repeated-goal analysis. Figure 1 uses AutoMix with Qwen3-8B and Qwen3-32B on ALFWorld. Within each task-family stream, a task is counted as repeated when its canonical goal has appeared earlier in the same family. Early, Middle, and Late denote consecutive thirds of each family stream. Large-model reliance is the fraction of executed decision steps produced by the large model within the same stage.

Cross-task skill reuse. Appendix C.2 further traces how cited skills enter later execution. Unassisted execution counts skill-citing decisions carried out by the small model autonomously. Verified local effects count those executions followed by a native environment acknowledgement aligned with the cited skill operation and expected outcome. Representative acquisition, revision, and reuse trajectories are reported in Appendix C.3.

Confidence diagnostics. Figure 6 uses the matched ALFWorld configuration with Qwen3-8B and DeepSeek-V4-Flash. Proposal-level diagnostics are computed on 1,490 labeled deployment contexts from the full 134-task stream. ROC–AUC (Fawcett, 2006) measures discrimination between supported and unsupported proposals, and the risk–coverage curve (Geifman & El-Yaniv, 2017) traces accepted-proposal error as the confidence threshold varies.

Expected calibration error (Guo et al., 2017) uses 15 equal-width confidence bins,

$$\text{ECE} = \sum_b \frac{n_b}{N} \left| \bar{p}_b - \bar{z}_b \right|.$$

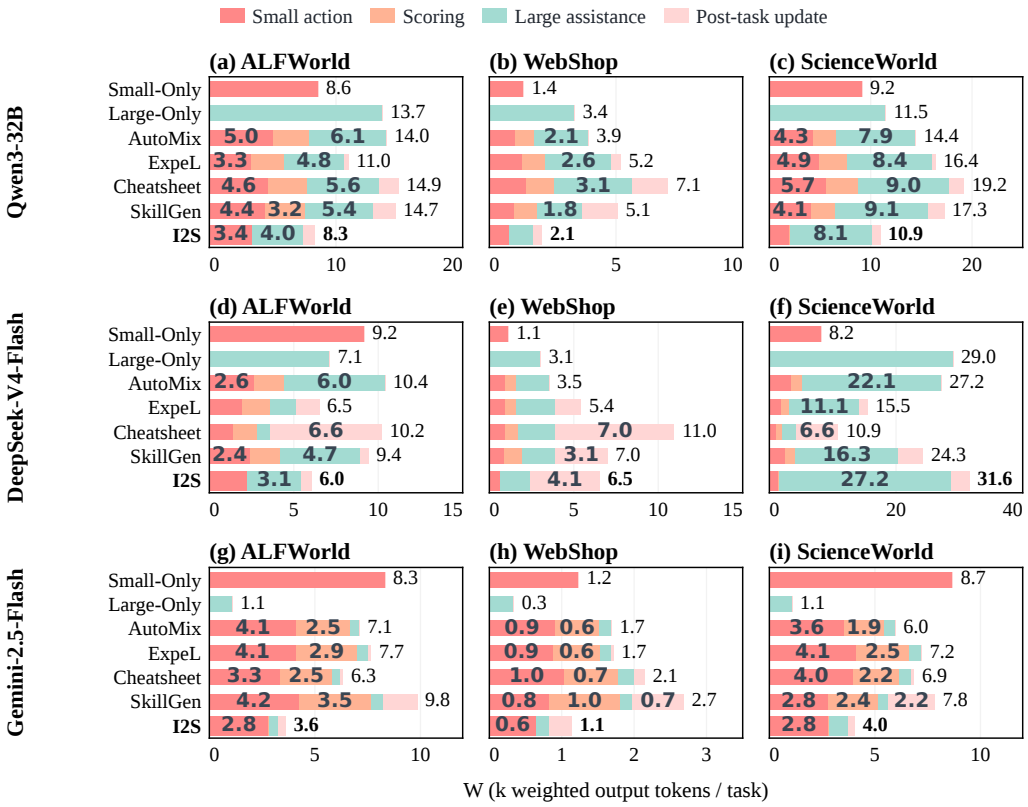


Figure C.1: **Complete-stream weighted token composition.** Bars decompose W into small-model action generation, confidence assessment, large-model assistance, and post-task updates for every method, benchmark, and backend. Bar-end labels report W in thousands of weighted output tokens per task.

C Additional Experimental Results

This appendix provides the detailed evidence underlying the main experimental findings. We first decompose complete-stream model usage, then examine cross-task skill reuse at the scenario level, illustrate representative skill lifecycles, and finally report the matched mechanism ablations and confidence diagnostics.

C.1 Complete-Stream Efficiency

Figure C.1 decomposes weighted token usage into the model calls that occur during deployment. Post-task skill induction is charged alongside online action generation and intervention.

Across the nine benchmark–backend settings, I2S uses fewer weighted tokens than AutoMix in seven settings and fewer than SkillGen in eight, while improving task quality in all nine comparisons.

Table C.1 complements token accounting with interaction-level statistics. Steps capture trajectory length, while Large steps and Large calls separate executed large-model actions from the requests used by the collaboration mechanism. The resulting profiles show that similar weighted-token totals can arise from different combinations of trajectory length, intervention frequency, and post-task computation.

Table C.1: **Execution and large-model interaction statistics.** All entries are mean counts per task. Steps denotes executed environment actions, Large steps the subset executed by the large model, and Large calls all large-model requests used for assistance or post-task updates.

Method	ALFWorld			WebShop			ScienceWorld		
	Steps	Large steps	Large calls	Steps	Large steps	Large calls	Steps	Large steps	Large calls
Qwen3-8B + Qwen3-32B									
Small-Only	27.74	0.00	0.00	5.80	0.00	0.00	31.10	0.00	0.00
Large-Only	21.82	21.82	21.82	6.50	6.50	6.50	16.03	16.03	16.03
AutoMix	24.99	7.98	7.98	7.31	3.48	3.48	23.66	10.32	10.32
ExpeL	20.69	6.37	6.60	8.18	3.83	4.08	25.44	11.10	11.34
Cheatsheet	21.69	6.57	7.57	8.96	4.28	5.28	28.29	12.38	13.38
SkillGen	22.89	7.19	8.78	6.52	2.85	4.31	23.46	11.45	12.47
I2S	18.22	5.89	6.68	4.90	1.47	1.74	16.74	11.50	12.11
Qwen3-8B + DeepSeek-V4-Flash									
Small-Only	28.99	0.00	0.00	5.10	0.00	0.00	27.87	0.00	0.00
Large-Only	12.56	12.56	12.60	5.80	5.80	5.82	11.89	11.89	12.27
AutoMix	17.07	5.79	5.81	6.53	2.96	2.97	18.21	7.89	8.09
ExpeL	13.91	3.12	3.35	6.29	3.05	3.32	11.95	4.76	5.12
Cheatsheet	11.45	1.91	2.91	6.23	2.84	3.87	8.69	2.36	3.36
SkillGen	16.12	5.25	5.61	6.22	2.69	4.86	13.80	5.83	8.01
I2S	14.77	4.59	5.37	4.82	2.08	2.70	12.55	9.08	9.89
Qwen3-8B + Gemini-2.5-Flash									
Small-Only	27.35	0.00	0.00	5.45	0.00	0.00	30.57	0.00	0.00
Large-Only	20.64	20.64	20.64	5.87	5.87	5.87	20.66	20.66	20.66
AutoMix	21.60	7.05	7.05	6.25	2.74	2.79	20.93	9.19	9.22
ExpeL	23.26	8.35	8.59	6.00	2.54	2.85	22.69	9.78	10.04
Cheatsheet	20.00	6.82	7.82	7.00	3.48	4.56	20.28	8.31	9.35
SkillGen	21.86	7.46	10.07	6.40	3.04	5.27	16.16	7.48	11.96
I2S	16.81	6.54	7.41	4.64	2.10	2.70	19.36	12.67	13.51

C.2 Cross-Task Skill Reuse

The aggregate reuse analysis in Section 5.4 is further resolved by task family and large-model backend in Figure C.2. Skills are formed and reused across all three environments, showing that cross-task reuse extends across different interaction structures.

Observed later execution. Across the main evaluation streams, 675 of 856 later skill-citing decisions (78.9%) are executed by the small model without large-model takeover. Among these autonomous executions, 519 (76.9%) receive a native environment acknowledgement aligned with the cited skill operation and expected outcome. A further 136 executions (20.1%) increase the native task score at the same step. Together, these measurements trace acquired skills from later retrieval to autonomous execution and observable environment response.

C.3 Skill Evolution Cases

The following cases illustrate the lifecycle summarized by Figure 5. Each example follows a skill from intervention evidence to an ADD, RETAIN, or REVISE update and then to later small-model use. The examples are drawn from ALFWorld, WebShop, and ScienceWorld deployments with Qwen3-8B and DeepSeek-V4-Flash. Task text, actions, environment feedback, and stored skill fields are reproduced from the deployment logs, with marked excerpts for long WebShop observations.

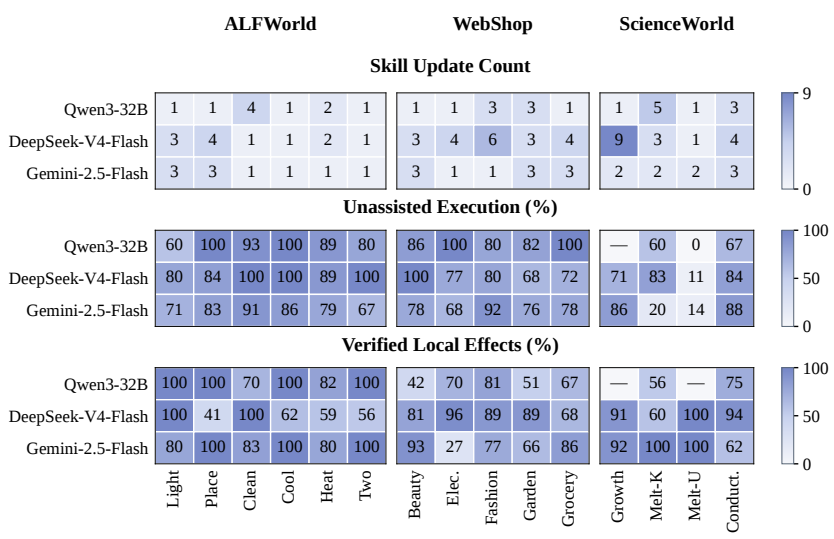


Figure C.2: **Skill formation and later execution.** Columns group task families by benchmark and each family repeats the three large-model backends. Skill Update Count reports ADD and REVISE outcomes. Unassisted Execution is the percentage of later skill-citing decisions executed by the small model without takeover, and Verified Local Effects is the percentage of those executions followed by a native acknowledgement aligned with the cited skill operation and outcome. Melt-K and Melt-U denote the known- and unknown-substance melting-point families.

Example C.1. ALFWorld — acquisition skill and later autonomous reuse

Source task	put some peppershaker on drawer.
Intervention	**Small proposal (withheld):** `inventory` **Large action:** `go to countertop 3` **Feedback:** You arrive at countertop 3. On the countertop 3, you see a apple 1, a bread 1, a butterknife 2, a butterknife 1, a cup 2, a egg 2, a houseplant 1, a knife 3, a knife 2, a pan 2, a peppershaker 1, and a vase 3. **Large action:** `take peppershaker 1 from countertop 3` **Feedback:** You pick up the peppershaker 1 from the countertop 3.
State update	**ADD.** The admitted acquisition skill is reproduced below.
Applicability	`["transfer_task(target,destination)", "not_holding(target)", "visible(target,candidate)"]`
Procedure	`[{"op": "ACQUIRE", "args": ["target", "candidate"]}]`
Outcome check	`["holding(target)"]`
Fallback	`{"when": ["visible(target,candidate)", "not_holding(target)"], "procedure": [{"op": "INSPECT", "args": ["candidate"]}, {"op": "ACQUIRE", "args": ["target", "candidate"]}], "outcome_check": ["holding(target)"]}`
Later reuse	**Task:** put a pencil in shelf. **Small action (skill cited; no takeover):** `take pencil 1 from desk 1` **Feedback:** You pick up the pencil 1 from the desk 1. **Native score at this step:** 0→0. **Task success:** yes; native task score 1/1. Large-model actions over the whole task: 0/8.

Example C.2. ALFWorld — revision and repeated reuse

Revision task	put a saltshaker in drawer.
Observed state	You open the cabinet 2. The cabinet 2 is open. In it, you see a plate 1, a saltshaker 1, and a soapbottle 1.
Intervention	**Small proposal (withheld):** `go to cabinet 2` **Large action:** `take saltshaker 1 from cabinet 2` **Feedback:** You pick up the saltshaker 1 from the cabinet 2. **Large action:** `go to drawer 3` **Feedback:** You arrive at drawer 3. The drawer 3 is open. In it, you see nothing.
Before	`[{"op": "ACQUIRE", "args": ["target", "candidate"]}]`
After	**REVISE.** `[{"op": "ACQUIRE", "args": ["target", "candidate"]}, {"op": "VERIFY", "args": ["holding(target)"]}]`
Other fields	Applicability, outcome check, and fallback remain exactly as in Example C.1.
Later reuse A	**Task:** put a peppershaker in drawer. **Small action (skill cited; no takeover):** `take peppershaker 1 from countertop 1` **Feedback:** You pick up the peppershaker 1 from the countertop 1. **Native score at this step:** 0→0. **Task success:** yes; native task score 1/1. Large-model actions over the whole task: 0/12.
Later reuse B	**Task:** put some soapbottle on toilet. **Small action (skill cited; no takeover):** `take soapbottle 2 from cabinet 3` **Feedback:** You pick up the soapbottle 2 from the cabinet 3. **Native score at this step:** 0→0. **Task success:** yes; native task score 1/1. Large-model actions over the whole task: 4/9.

Example C.3. ALFWorld — placement skill with task-score progress

Source task	put a saltshaker in drawer.
Intervention	**Small proposal (withheld):** `move saltshaker 1 to drawer 1` **Large action:** `go to drawer 1` **Feedback:** You arrive at drawer 1. The drawer 1 is open. In it, you see a creditcard 1. **Large action:** `move saltshaker 1 to drawer 1` **Feedback:** You move the saltshaker 1 to the drawer 1.
State update	**ADD.** The admitted placement skill is reproduced below:
Applicability	`["transfer_task(target,destination)", "holding(target)", "at(destination)"]`
Procedure	`[{"op": "PLACE", "args": ["target", "destination"]}, {"op": "VERIFY", "args": ["placed(target,destination)"]}]`
Outcome check	`["placed(target,destination)"]`
Fallback	`{"when": ["holding(target)", "placement_unconfirmed(target,destination)"], "procedure": [{"op": "NAVIGATE", "args": ["destination"]}, {"op": "PLACE", "args": ["target", "destination"]}, {"op": "VERIFY", "args": ["placed(target,destination)"]}], "outcome_check": ["placed(target,destination)"]}`
Later reuse	**Task:** put some soapbottle on toilet. **Small action (skill cited; no takeover):** `move soapbottle 2 to toilet 1` **Feedback:** You move the soapbottle 2 to the toilet 1. **Native score at this step:** 0→1. **Task success:** yes; native task score 1/1. Large-model actions over the whole task: 4/9.
Task context	This is the same later task as Example C.2, reuse B. The small model first cites the revised acquisition skill, navigates to the toilet, then cites this placement skill. The final placement increases the native task score; earlier steps include large-model assistance.

Example C.4. WebShop — option-selection skill and retention

Source task	i want a highly pigmented lip gloss that is in the color r901, and price lower than 40.00 dollars
Intervention	**Small proposal (withheld):** `click[r901]` **Large action:** `click[r901]` **Feedback (excerpt):** You have clicked r901. [...] `[clicked button] r901 [clicked button_]` [...]
State update	**ADD.** The large model executes the same option proposed by the small model.
Applicability	`["item_details(item)"]`
Procedure	`[{"op": "SELECT_OPTION", "args": ["option"]}]`
Outcome check	`["option_selected(option)"]`
Fallback	`{"when": ["option_selected_unconfirmed(option)"], "procedure": [{"op": "SELECT_OPTION", "args": ["option"]}], "outcome_check": ["option_selected(option)"]}`
Later task	i'm looking for a lip pencil high pigmented for long lasting and melrose place color, and price lower than 40.00 dollars
New evidence	**Small proposal (withheld):** `click[buy now]` **Large action:** `click[melrose place]` **Feedback (excerpt):** You have clicked melrose place. [...] `[clicked button] melrose place [clicked button_]` [...]
State update	**RETAIN.** All four operative fields remain unchanged after this additional supported option-selection example.

Example C.5. WebShop — repeated option-selection reuse

Available skill	The option-selection skill in Example C.4, with its operative fields unchanged. Each action below explicitly cites this skill.
Later reuse A	**Task:** i need a long lasting 6.76 fl oz bottle of l'eau d'issey, and price lower than 90.00 dollars **Small action (skill cited; no takeover):** `click[6.76 fl oz (pack of 1)]` **Feedback (excerpt):** You have clicked 6.76 fl oz (pack of 1). [...] `[clicked button] 6.76 fl oz (pack of 1) [clicked button_]` [...] **Native score at this step:** 0→0. **Task success:** yes; native task score 1/1. Large-model actions over the whole task: 0/4.
Later reuse B	**Task:** i want a body wash that is dermatologist tested. it should have a cucumber and aloe scent, and price lower than 30.00 dollars **Small action (skill cited; no takeover):** `click[cucumber + aloe]` **Feedback (excerpt):** You have clicked cucumber + aloe. [...] `[clicked button] cucumber + aloe [clicked button_]` [...] **Native score at this step:** 0→0. **Task success:** yes; native task score 1/1. Large-model actions over the whole task: 2/8.
Later reuse C	**Task:** i would like a 34 piece set of some long lasting press on nails, and price lower than 30.00 dollars **Small action (skill cited; no takeover):** `click[34 piece set]` **Feedback (excerpt):** You have clicked 34 piece set. [...] `[clicked button] 34 piece set [clicked button_]` [...] **Native score at this step:** 0→0. **Task success:** no; native task score 0.75/1. Large-model actions over the whole task: 0/4.
Observed scope	The environment acknowledges the selected option in all three later tasks. Two tasks reach full reward, while the third reaches 0.75 with no large-model actions.

Example C.6. ScienceWorld — device activation across later tasks

Source task	Your task is to grow a peach plant from seed. Seeds can be found in the bathroom. First, focus on a seed. Then, make changes to the environment that grow the plant until it reaches the reproduction life stage.
Intervention	**Small proposal (withheld):** `wait` **Large action:** `activate sink` **Feedback:** The sink is now activated.
State update	**ADD.** The admitted device-activation skill is reproduced below:
Applicability	`["device_active_unconfirmed(device)"]`
Procedure	`[{"op": "ACTIVATE_DEVICE", "args": ["device"]}]`
Outcome check	`["device_active(device)"]`
Fallback	`{"when": ["device_active_unconfirmed(device)"], "procedure": [{"op": "ACTIVATE_DEVICE", "args": ["device"]}], "outcome_check": ["device_active(device)"]}`
Later reuse A	**Task:** Your task is to grow a peach plant from seed. Seeds can be found in the workshop. First, focus on a seed. Then, make changes to the environment that grow the plant until it reaches the reproduction life stage. **Small action (skill cited; no takeover):** `activate sink` **Feedback:** The sink is now activated. **Native score at this step:** 33→33. **Task success:** no; native task score 34/100. Large-model actions over the whole task: 23/28.
Later reuse B	**Task:** Your task is to grow a peach plant from seed. Seeds can be found in the living room. First, focus on a seed. Then, make changes to the environment that grow the plant until it reaches the reproduction life stage. **Small action (skill cited; no takeover):** `activate sink` **Feedback:** The sink is now activated. **Native score at this step:** 36→36. **Task success:** yes; native task score 100/100. Large-model actions over the whole task: 11/15.

Table C.2: **Matched ALFWorld mechanism ablations.** All configurations use Qwen3-8B and DeepSeek-V4-Flash on the same 134 tasks, task order, and seeds. Δ reports the change from each ablation to Full I2S in the favorable direction. Weighted output tokens use $W = L + 0.25S$, and k denotes 1,000 tokens.

Metric	Full I2S	w/o confidence calibration		w/o skills & calibration	
		Value	Δ	Value	Δ
Task quality					
Score ↑	**100.00**	94.78	+5.22	95.52	+4.48
Intervention					
LM Use (%) ↓	**12.35**	36.36	+24.01	30.89	+18.53
Interventions/task ↓	**1.43**	2.84	+1.41	2.51	+1.09
Output tokens (k/task)					
Large ↓	**3.761**	7.915	+4.154	5.027	+1.266
Small ↓	**9.133**	7.440	−1.693	7.148	−1.985
Weighted ↓	**6.044**	9.775	+3.731	6.814	+0.770

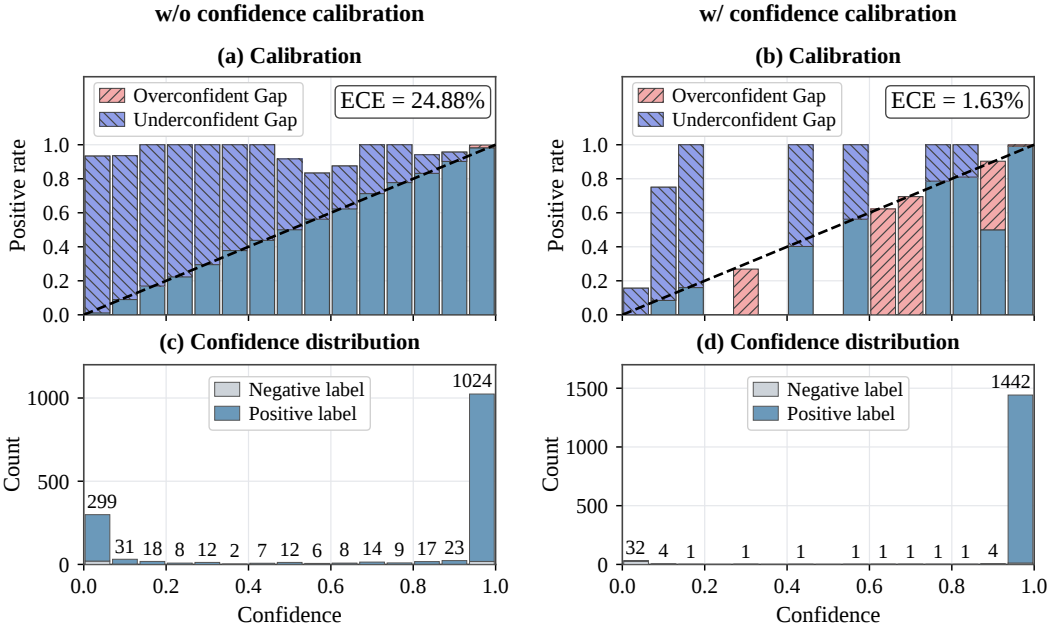


Figure C.3: **Proposal confidence calibration.** Columns compare Qwen3-8B without and with the confidence adapter on the same 1,490 labeled deployment contexts from 134 ALFWorld tasks with DeepSeek-V4-Flash. Panels (a,b) show reliability diagrams and panels (c,d) show label-conditioned confidence distributions. ECE decreases from 24.88% to 1.63% using 15 equal-width confidence bins.

C.4 Ablation and Confidence Calibration

Table C.2 expands the matched full-ALFWorld comparison used in Section 5.5. The controls separate the effect of confidence calibration from the combined system that also carries skills across tasks.

Calibration improves the deployment operating point. Relative to the configuration without confidence calibration, full I2S raises ALFWorld score by 5.22 points while reducing LM Use by 24.01 percentage points. Interventions fall from 2.84 to 1.43 per task, and large-model output drops from 7.915k to 3.761k tokens per task. After small-model usage is included, weighted token usage decreases from 9.775k to 6.044k per task. The configuration without skills and confidence calibration also has a lower score, higher LM Use, and higher weighted token usage than full I2S.

Calibration concentrates supported proposals at high confidence. The detailed confidence distributions complement the ROC–AUC and risk–coverage results in Figure 6. After calibration, locally supported proposals concentrate near high confidence, while unsupported proposals move toward the low-confidence region. The resulting ECE reduction from 24.88% to 1.63% aligns proposal confidence with the local execution evidence used by the intervention controller. Therefore, better calibrated action assessment reduces unnecessary large-model intervention, while reusable skills continue to support later decisions.

D Prompt Templates and Runtime Interfaces

This appendix documents the runtime interfaces used by I2S for small-model action generation, proposal confidence assessment, and post-task skill induction. The prompt panels reproduce the natural-language instructions used in the experiments. Their wording is preserved exactly, with only line breaks and JSON indentation adjusted for readability. Benchmark-specific instructions extend the shared actor prompt with the native action grammar of each environment.

Actor and confidence prompts. At each decision step, the actor receives the current task, visible environment state, interaction history, and active skill snapshot. It proposes one benchmark-native action and records any skill identities that directly inform the decision. The confidence scorer then evaluates this proposal under the same visible decision context.

The benchmark-specific action constraints in Prompts D.4–D.6 are appended to the shared actor instruction. The same confidence instruction is used for adapter training and deployment, and the proposal confidence score is defined in Equation 11.

Skill induction. After task completion, skill induction receives the completed visible trajectory, eligible intervention anchors, and the active skill state from the source task. These inputs provide the execution evidence and trajectory context used to identify and compile a reusable procedural residual. The complete induction instruction further specifies the evidence fields, revision rules, benchmark-native vocabulary, and serialization requirements.

Benchmark-specific instructions. The shared actor interface is paired with benchmark-specific constraints that map semantic skill guidance to executable environment actions. These instructions define the native action grammar and grounding rules used in ALFWorld, WebShop, and ScienceWorld.

Runtime interfaces. Structured actor and induction outputs record skill use, evidence provenance, and cross-task updates consistently. The actor returns the three fields defined in Prompt D.1. `analysis` contains a concise decision rationale, `action` contains exactly one benchmark-native action, and `skill_refs` records the identity and version of each supplied skill that directly informs the proposal. The controller parses the action before environment execution and stores the declared skill references with the corresponding decision opportunity.

Prompt D.1. Shared actor instruction

You control an agent in an interactive environment. Complete the given task efficiently.

Use only the task instruction, observations, interaction history, and action guidance supplied in the conversation. Produce exactly one next environment action.

Return exactly one JSON object with analysis, action, and skill_refs fields:

```
{
  "analysis":  "one concise sentence",
  "action":  "the next environment action",
  "skill_refs":  []
}
```

Use standard JSON escaping; do not put a backslash before an apostrophe. Do not use Markdown fences, XML tags, extra keys, or text outside the JSON object.

Treat different complete names as distinct entities. Never substitute a shorter or similar visible name for an entity explicitly named by the task; when that named entity is not currently visible, choose a grounded navigation or inspection action instead.

When a supplied reusable procedure actually informs your proposed action, declare its exact skill_id and integer version in skill_refs, e.g. [{"skill_id":"skill-0001","version":1}]. Otherwise use []. Only reference procedures present in this context. This declaration is an analysis record and never an environment command.

Before choosing an action, read any supplied skill card and check its observable applicability against the current state. When an applicable card supplies the procedural relation used for this action, briefly mention that relation in analysis and cite its exact skill ID/version in skill_refs. If no card informs the action, return an empty skill_refs list. Do not cite a skill just because it is present.

Prompt D.2. Confidence instruction: training and deployment

Assess whether the proposed next environment action is reliable and useful in the supplied visible state. Judge only this action, not whether the entire task is solved. Ground exact object identity, destination, possession and ordering in the task, current observation, admissible actions, and interaction history. Admissibility alone does not establish useful progress. A repeated information action whose result is already known may fail to make progress; a valid acquisition or return to a required source can still be necessary. Use supplied procedural guidance only when its observable conditions hold. Do not invent hidden locations or assume a proposed action was executed.

Return exactly `CONFIDENT` when the proposal is a supported useful next action, otherwise `UNCERTAIN`. These are the only two reliability labels. Do not explain the answer.

Prompt D.3. Skill induction instruction: opening excerpt

Induce one compact, reusable semantic procedural skill after the completed task, using eligible localized intervention evidence. This is ONE joint evidence-interpretation and induction call. Output a JSON object with exactly two keys: `evidence_interpretation` and `skill_proposal`. Both may be null if no supported addition/refinement exists. Prefer a short useful supported relation; a skill need not solve the entire task.

Scientific meaning: I=observable applicability, P=ordered semantic procedure, O=observable verification. Semantic operators are guidance for an acting model, NEVER literal environment commands. No final success, hidden evaluator, future task, or remembered object location may support a skill. The counterfactual outcome of a withheld SMALL proposal is UNKNOWN, even if a teacher command has an observed effect. An immediate navigation anchor supports arrival/state observation only; it does not establish that an alternative failed or that a search order is optimal. Explain the procedural relation the observation supports, without claiming causal performance benefit beyond it.

Read the completed chronological trajectory as context. Only `eligible_anchors/local_evidence` support new procedural content. `lossless-reference-pool-v1 $ref` fields expand to exact fragments, not summaries. Every claimed new operation and outcome must be directly supported by cited local action/observation pairs. Do not turn a small observation into a complete generic task recipe. The induction should FIRST identify a missing prerequisite, order, observation/verification, or recovery relation, THEN construct a skill realizing that relation. Use one consistent role binding across all cited anchors in a proposal; cite a small sufficient subset.

Prompt D.4. ALFWorld: native action constraints

ALFWorld uses lower-case environment commands. Select exactly one currently admissible action, retaining its exact entity names and numbers. Primitive capabilities include look, inventory, go to RECEPTACLE, open RECEPTACLE, close RECEPTACLE, take OBJECT from RECEPTACLE, move OBJECT to RECEPTACLE, clean OBJECT with TOOL, heat OBJECT with APPLIANCE, cool OBJECT with APPLIANCE, use OBJECT, examine OBJECT, examine RECEPTACLE, slice OBJECT with TOOL, and help. Availability depends on the current state; the supplied admissible action list is authoritative. The task instruction's word "put" is realized by the admissible "move ... to ..." command. "search", "put", "place", "find", "navigate", and procedural role placeholders are not literal environment commands. External skills describe applicability, an ordered procedure, and verification. Resolve their roles from this task's own instruction and visible observations. To realize high-level inspection, select an admissible go to command for a visible receptacle, use its arrival observation, and open it only if the observation and admissible actions permit. Inventory checks use inventory; object acquisition uses take ... from ...; placement uses move ... to .... Execute only the next applicable primitive. A skill cannot override current admissibility or establish a hidden object's location. After a rejection, use the new visible feedback and admissible actions to realize the next step. Never emit a sentence from a skill as a command.

Semantic skill realization: a card's uppercase operators, predicate strings and role names are non-executable guidance. Bind roles to this task's current entities; determine which observable conditions hold now. Skip steps whose outcome is already visibly established. Choose the next useful native action from the CURRENT admissible list, preserving the complete string exactly. Never output NAVIGATE(...), LOCATE(...), INSPECT(...), VERIFY(...), a predicate, or a procedural sentence as action. For INSPECT use the arrival/opening observation already obtained before spending another action. Declaring a skill means its applicable relation actually informed this proposed action, not merely that a skill was shown.

Prompt D.5. WebShop: native action constraints

WebShop accepts only search[query] and click[visible text] actions. On a search page, replace <your query> with your own concise query. On other pages, copy a currently visible click target exactly. Opening a currently visible product candidate from search results is a reversible inspection step, not a purchase commitment. Do not request help solely because its attributes require product-page inspection; when the exact candidate click target is visible and no visible fact contradicts the current request, open it and verify on the product page. Before buying, click every requested product option; seeing an option on the page does not select it. In text_rich observations, only [clicked button] marks a selected option; plain [button] marks an unselected option. Immediately before buying, inspect the current observation for every value requested in the task. If any requested value is still shown as plain [button], click that value first; click[buy now] is not yet allowed. Never infer selection merely from availability or earlier reasoning. If you open Description, Features, or Reviews, use click[< prev] to return to the product page. Purchasing is performed only with click[buy now].

Prompt D.6. ScienceWorld: native action constraints

ScienceWorld provides either exact current actions or action patterns below. When exact actions are shown, copy one of them verbatim and do not construct an unlisted action. When a shown action contains OBJ, replace each OBJ with an object or location name grounded in the task instruction, current observation, previous environment responses, or reusable context. Do not output OBJ itself. The standard visible observation includes the action response, current room, and inventory. If the rendered state is ambiguous, use a currently listed inspection action. If the environment explicitly asks you to choose a numbered action, use one of the displayed numbers exactly.

E DISCUSSION

I2S turns selective assistance into cumulative adaptation. Conventional small–large collaboration uses stronger models to resolve difficult decisions as they arise. I2S extends this interaction by converting supported interventions into reusable skills that remain available to later tasks. Earlier assistance can therefore change how the small model acts in the future, even though the parameters of both models remain fixed.

reusable skills extend the value of an intervention beyond its source task. An intervention creates immediate value by resolving the current difficulty and longer-term value when it reveals a procedure that can be reused later. The streamwise results are consistent with this pattern. As skills accumulate, later tasks achieve stronger performance with less large-model involvement.

Selective intervention and reusable skills play complementary roles. Confidence calibration provides the signal for deciding when stronger capability is needed, while skill acquisition preserves the reusable knowledge exposed by that assistance. The two mechanisms operate at different timescales. Intervention improves the current decision, and skills improve later decisions when the same procedural structure reappears. Their combination allows I2S to progressively shift more work to the small-model path while retaining access to the large model when stronger capability is still required.