SUPERVISING AGENT EVOLUTION WITH PREFIX POTENTIALS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent evolution through continued interaction and parameter updates requires informative supervision. However, environmental feedback is often sparse and concentrated at terminal outcomes, offering limited guidance for intermediate decisions. Typically, existing approaches alleviate this sparsity via separately trained evaluators or task-specific reference structures to construct step-level supervision. Instead, we propose **A**gent **S**elf-supervision through **C**ompletion-potential **E**stimation and **N**eighboring-prefix **D**ifferencing (ASCEND), which derives this fine-grained supervision from assessments by the evolving agent. Concretely, the evolving agent estimates a prefix potential at each step through an additional scoring call, assessing task-completion prospects based on the task objective and the actions and observations available up to that step. Differences between adjacent prefix potentials yield step-level signals, which are normalized, accumulated from each step onward, and combined with terminal outcome feedback to guide parameter updates. Evaluations on long-horizon interactive tasks and multi-hop reasoning demonstrate that ASCEND outperforms representative evolution baselines, with improvements extending to additional benchmarks without further adaptation. Further analysis shows that prefix potentials correlate with final task outcomes and that differences between adjacent prefix potentials align with independently assessed task progress. Code is available at `Anonymous`.

1 INTRODUCTION

Recent autonomous agents demonstrate remarkable proficiency in executing complex tasks across diverse interactive environments (Yao et al., 2023; Wang et al., 2024; Trivedi et al., 2024; Xie et al., 2024). Despite increasingly capable autonomous execution, deployed agents remain largely fixed, whereas real-world operating environments, task requirements, and interaction patterns typically continue to change post-deployment (Luo et al., 2025; Fang et al., 2025; Jaglan & Barnes, 2025; Kaymak et al., 2025). For example, updated software interfaces and novel operational constraints frequently demand capabilities not covered during initial agent construction (Patil et al., 2024; Ye et al., 2024). Bridging the gap between static deployment and open-ended operational demands motivates a paradigm shift toward **agent evolution**, wherein an *evolutionary strategy* organizes continuous adaptation according to current agent capabilities and accumulated environmental interactions (Gao et al., 2026; Fang et al., 2025). Through successive adaptation, the agent can progressively improve environmental fitness and accumulate capabilities throughout deployment.

Once an evolutionary update is initiated, a central challenge is converting accumulated interaction evidence into informative learning signals. Concretely, the agent requires guidance on intermediate decisions throughout multi-step task execution, whereas environmental feedback is often sparse and concentrated at terminal outcomes (Kumar et al., 2025). Such coarse feedback indicates eventual task success or failure, but provides limited guidance on the learning value of intermediate behaviors, as illustrated in Figure 1. Existing methods mitigate this sparsity by constructing fine-grained step-level supervision, but often introduce additional information dependencies. One line of work relies on auxiliary evaluators, including process evaluators or learned critics, to assess intermediate behaviors (Zhai et al., 2025; Zhang et al., 2026c; Wang et al., 2025a). Another line derives supervision from internally constructed evidence (Zelikman et al., 2022; Xia et al., 2025; Huang et al., 2025), with recent methods further building semantic outcome references or cross-trajectory comparison structures (Hu et al., 2026a; Feng et al., 2026). Across these approaches, fine-grained

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

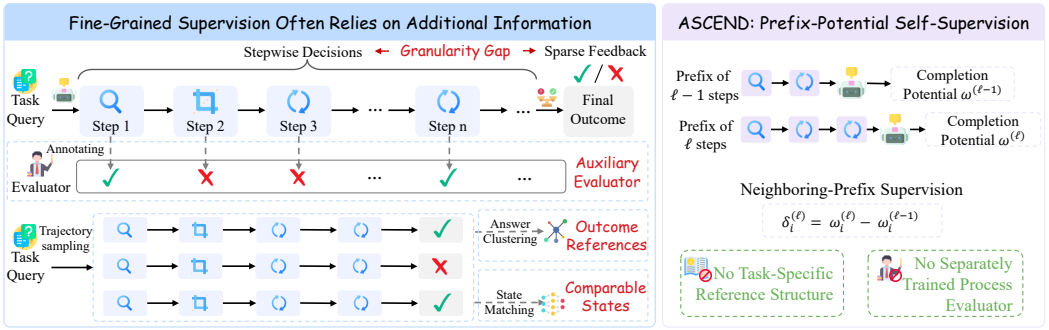


Figure 1: Agent evolution faces a granularity gap between stepwise decisions and sparse trajectory-level feedback. Existing fine-grained supervision often relies on auxiliary evaluators or task-specific reference structures. ASCEND derives step-level signals from adjacent prefix-potential differences through additional scoring calls to the evolving agent, without a separately trained process evaluator.

supervision often depends on auxiliary evaluation or task-specific reference construction beyond the available interaction evidence. This motivates us to study an information-constrained setting for evolutionary supervision, where fine-grained signals are derived from the evolving agent and available interaction evidence. We therefore ask a critical question: *how can an evolutionary strategy construct informative fine-grained supervision under these information constraints?*

To address this research question, we propose **A**gent **S**elf-supervision through **C**ompletion-potential **E**stimation and **N**eighboring-prefix **D**ifferencing (**ASCEND**), a supervision construction strategy for parametric evolution, as illustrated in Figure 1. ASCEND employs the same agent for task execution and prefix-potential estimation, with an additional scoring prompt applied to each partial execution history in a separate call. This score assesses task-completion prospects based on the task objective and the actions and observations available so far. The execution history captures changes in task state and newly acquired evidence, providing a basis for assessing how task-completion prospects change as execution proceeds. Differences between adjacent prefix potentials provide step-level signals, which are normalized, accumulated from each step onward, and combined with terminal outcome feedback to guide parameter updates. The method requires no separately trained process evaluator, explicit answer clustering, or cross-trajectory state matching for process supervision construction.

Empirical evaluations across long-horizon stateful interaction and multi-hop reasoning demonstrate the effectiveness, generality, and quality of ASCEND supervision. On APPWorld (Trivedi et al., 2024), augmenting terminal-outcome supervision with prefix-potential supervision improves task success rate from 44.64% to 50.00% while reducing average execution steps from 18.83 to 18.22, demonstrating the benefit of fine-grained prefix-potential supervision beyond trajectory-level feedback. On multi-hop reasoning, ASCEND achieves the highest macro-average F1 of 51.73% across three benchmarks, with improvements retained on additional evaluation benchmarks, indicating that the constructed supervision generalizes beyond stateful interaction. Supervision analyses further show that prefix potentials estimated at later execution stages correlate more strongly with final outcomes, with Spearman correlation rising from 0.052 to 0.448 on APPWorld and from 0.011 to 0.341 on 2Wiki (Ho et al., 2020). Human audits further show that neighboring-prefix signals identify step-level progress with 92.0% and 100.0% precision, supporting their role as informative supervision.

Our contributions are summarized as follows:

(i) We study how to construct fine-grained supervision for agent evolution under explicit information constraints, drawing on the evolving agent and interaction evidence without a separately trained process evaluator or task-specific reference structures (Section 3).

(ii) Under this formulation, we propose **ASCEND**, a supervision construction strategy for parametric evolution. ASCEND estimates task-completion potential directly from visible interaction prefixes and transforms neighboring potential changes into fine-grained supervision, while retaining environmental outcomes as complementary task-level feedback (Section 4).

(iii) We evaluate ASCEND across long-horizon stateful environments and multi-hop reasoning. ASCEND outperforms representative evolution methods and supports cross-benchmark transfer, while analyses substantiate the quality and value of prefix-potential supervision (Section 5).

2 Related Work

Autonomous Agents. Recent language-model-powered agents demonstrate increasing capabilities in reasoning (Yao et al., 2023; Wei et al., 2026), task planning and decomposition (Li et al., 2025; Sun et al., 2026), and external tool interaction (Shen et al., 2023; Patil et al., 2024; Wang et al., 2025b), supporting task execution in long-horizon operational environments (Trivedi et al., 2024), web navigation (Zhou et al., 2024), and computer interaction (Xie et al., 2024). However, many agents remain fixed after deployment, limiting adaptation to changing tools, interfaces, and emerging tasks (Luo et al., 2025; Ye et al., 2024) and motivating agent evolution (Fang et al., 2025).

Agent Evolution. Agent evolution supports continuous adaptation during deployment, with an evolutionary strategy organizing adaptation according to current agent capabilities and accumulated interactions (Gao et al., 2026; Fang et al., 2025). Approaches differ in how acquired capabilities are retained. *Parametric evolution* refines model weights through interaction-driven optimization (Feng et al., 2026; Zhou et al., 2025; Kumar et al., 2025; Zhai et al., 2025; Zhang et al., 2026c), while *externalized evolution* retains capabilities outside model parameters, in memories (Shinn et al., 2023; Wei et al., 2025; Xu et al., 2025), skills (Bao et al., 2026; Yan et al., 2026), or tools and harnesses (Xiong et al., 2026; Luo et al., 2026). Hybrid methods coordinate both pathways (Zhai et al., 2025; Hebbar et al., 2026; Ji et al., 2026). Constructing informative supervision from interaction evidence remains challenging. ASCEND addresses this problem within parametric evolution.

Supervision for Agent Evolution. Evolutionary updates require informative supervision, yet sparse terminal feedback offers limited guidance for intermediate decisions (Kumar et al., 2025; Wang et al., 2025a). Auxiliary-evaluator approaches assess intermediate behaviors through teachers, process evaluators, or critics (Zhai et al., 2025; Wang et al., 2025a; Yuan et al., 2026; Zhang et al., 2026c;a). Other approaches derive signals through reflection (Madaan et al., 2023; Lu et al., 2023), generation consensus (Huang et al., 2025; Xia et al., 2025; Zuo et al., 2026), answer-based information gain (Hu et al., 2026b), or cross-trajectory state comparisons (Feng et al., 2026; Liang et al., 2026). Recent works such as RePro and SIOP construct process rewards from adjacent score differences. RePro (Ma et al., 2026) initializes progress assessment through external demonstration warmup, then combines retrospective self-assessment of complete trajectories and outcomes with online-retrospective consistency training. SIOP (Hu et al., 2026a) tracks changes in prefix-conditioned support for answer clusters constructed from sampled rollouts. ASCEND assesses completion prospects from visible prefixes without retrospective warmup, consistency training, or answer-derived references, integrating terminal feedback separately for parametric evolution.

3 Problem Formulation

This section formalizes the supervision in agent evolution, examining how interaction evidence guides agent updates. We characterize agent evolution and evolutionary strategy in Section 3.1. We formulate fine-grained evolutionary supervision under information constraints in Section 3.2.

3.1 Agent Evolution and Evolutionary Strategy

We consider **agent evolution** as a continuous adaptation process throughout deployment and continued interaction, aiming to improve environmental fitness and accumulate capabilities under evolving operational demands. Conventional agent deployment typically emphasizes capability preparation before release, while agent evolution extends adaptation into the deployment lifecycle through successive interaction and update cycles. During task execution, the agent continuously interacts with the operating environment and accumulates observations, actions, and available evaluative feedback as interaction evidence for subsequent adaptation.

Formally, continuous adaptation is driven by an *evolutionary strategy* $\mathcal{S}$, which characterizes how adaptation is organized under the current evolutionary context. At evolution round t, let A_t denote the current agent, E_t denote the operating environment, and $\mathcal{H}_t$ denote the collection of interaction histories accumulated before the current round. The evolutionary transition is formulated as:

$$A_{t+1} = \mathcal{S}(A_t, E_t, \mathcal{H}_t). \tag{1}$$

Different methods may instantiate distinct evolutionary processes through different forms of $\mathcal{S}$. Across successive evolution rounds, the agent follows an evolutionary trajectory $A_0 \rightarrow A_1 \rightarrow$

$\cdots \rightarrow A_T$. Once evolution is activated, supervision construction determines how available interaction evidence and feedback are transformed into learning signals for subsequent adaptation.

3.2 Evolutionary Supervision Construction

Available supervision may originate from environmental outcomes, oracle labels, auxiliary evaluators, or signals constructed by the evolving agent. For a task τ_i with query q_i, let its execution history be $H_i = (q_i, a_i^{(1)}, o_i^{(1)}, \ldots, a_i^{(L_i)}, o_i^{(L_i)})$, where $a_i^{(\ell)}$ and $o_i^{(\ell)}$ denote the action and observation at step ℓ, with $H_i^{(\ell)} = (q_i, a_i^{(1)}, o_i^{(1)}, \ldots, a_i^{(\ell)}, o_i^{(\ell)})$ representing the visible interaction prefix up to step ℓ. Successful execution requires both task-level alignment and appropriate intermediate decisions. A terminal outcome r_i^{out} provides task-level feedback on eventual completion, yet offers **limited resolution** for assessing individual behaviors. Fine-grained supervision can complement this signal with behavior-aligned guidance throughout execution.

We formalize supervision construction as a mapping $\mathcal{G}$ from the current agent, task execution, trajectory-level outcome, and auxiliary supervision resources $\mathcal{R}_i$ to step-aligned learning signals:

$$\boldsymbol{\delta}_i = \mathcal{G}\left(A_t, H_i, r_i^{\text{out}}, \mathcal{R}_i\right) = \left(\delta_i^{(1)}, \ldots, \delta_i^{(L_i)}\right), \tag{2}$$

where $\mathcal{R}_i$ denotes auxiliary supervision resources such as oracle annotations, separately trained process evaluators, or task-specific reference structures. Different strategies may depend on subsets of this information, and r_i^{out} may or may not participate in fine-grained signal construction. This formulation separates information available within the evolution process from auxiliary resources introduced for fine-grained supervision. We focus on the setting $\mathcal{R}_i = \emptyset$, with process supervision derived through scoring calls to the current agent and terminal feedback supplied separately.

This supervision problem applies across different forms of agent evolution. *Parametric evolution* provides a direct setting for studying fine-grained supervision, as behavior-aligned signals can directly guide model parameter updates. We therefore focus on this setting. Specifically, ASCEND instantiates the formulated setting by deriving step-level learning signals $\delta_i^{(\ell)}$ directly from the evolving agent A_t and visible interaction prefixes $H_i^{(\ell)}$ without a separately trained process evaluator, while retaining terminal outcomes r_i^{out} as complementary task-level supervision.

4 Methodology

To address the evolutionary supervision problem formulated in Section 3, we propose ASCEND, which constructs fine-grained supervision from the evolving agent and interaction evidence without a separately trained process evaluator. Its overall workflow is illustrated in Figure 2. At each evolution round, the current agent assesses completion prospects from visible interaction prefixes and constructs step-aligned supervision from neighboring potential changes. The framework proceeds through completion-potential estimation, neighboring-prefix supervision construction, and multi-granularity supervision integration, with the detailed implementation provided in Appendix A.

Completion-Potential Estimation. Terminal outcomes provide task-level supervision, while fine-grained supervision requires assessing how intermediate behaviors contribute toward the task objective. An isolated action provides limited evidence as its effect depends on the preceding state. Recent studies show that intrinsic signals elicited from the evolving model can support learning without separately trained reward models or verifiers (Zhao et al., 2026; Hu et al., 2026a). ASCEND estimates a *prefix potential* for each interaction prefix $H_i^{(\ell)}$, comprising the actions and observations available up to step ℓ. This potential reflects the model assessment of task-completion prospects given the task objective and execution history. The score summarizes estimated completion prospects, with no direct interpretation as a completion percentage or calibrated success probability.

Starting from the query-only prefix $H_i^{(0)} = (q_i)$, the current agent assesses each prefix without access to subsequent interactions or terminal evaluation scores. Direct scalar decoding summarizes the assessment in a single value, whereas bin log-probabilities preserve relative support across potential levels (Kadavath et al., 2022). ASCEND therefore predicts over $N_{\text{bin}} = 6$ ordered bins $\{b_k\}_{k=1}^{N_{\text{bin}}}$ with numerical anchors $v(b_k) = 0.2(k-1)$, corresponding to $\{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. We com-

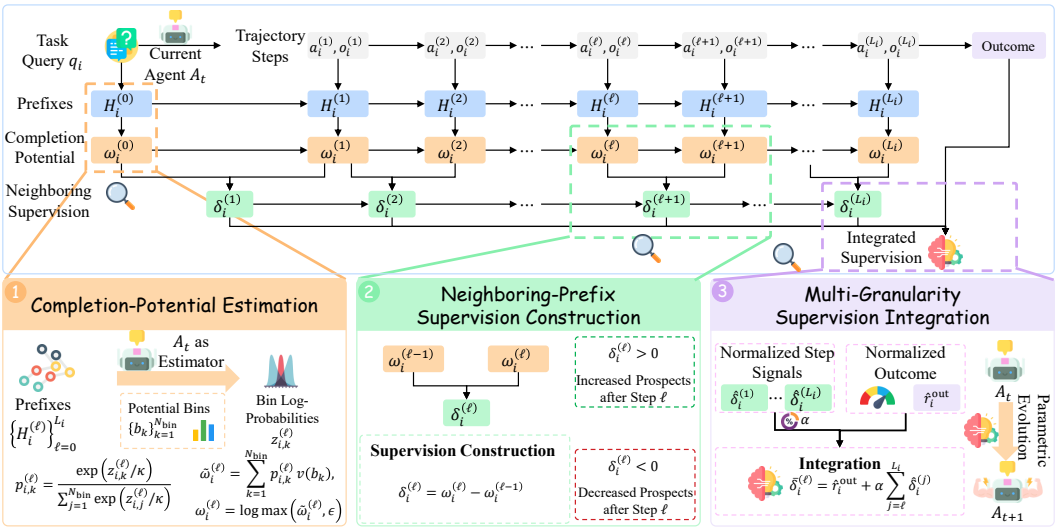


Figure 2: ASCEND constructs task-aligned fine-grained supervision from visible interaction prefixes. Starting from rollout trajectories, the evolving agent estimates prefix completion potentials, neighboring potential changes provide step-level supervision, and normalized signals are integrated with available terminal outcomes for subsequent parametric evolution.

pute the probability-weighted score $\tilde{\omega}_i^{(\ell)}$ and adopt its log-space representation $\omega_i^{(\ell)}$ for differencing. Above the numerical floor ϵ, adjacent differences correspond to log-ratios that express relative changes in the assessed score. Given bin log-probabilities $z_{i,k}^{(\ell)}$, the prefix potential is computed as:

$$p_{i,k}^{(\ell)} = \frac{\exp(z_{i,k}^{(\ell)}/\kappa)}{\sum_{j=1}^{N_{\text{bin}}} \exp(z_{i,j}^{(\ell)}/\kappa)}, \quad \tilde{\omega}_i^{(\ell)} = \sum_{k=1}^{N_{\text{bin}}} p_{i,k}^{(\ell)} v(b_k), \quad \omega_i^{(\ell)} = \log \max\left(\tilde{\omega}_i^{(\ell)}, \epsilon\right), \tag{3}$$

where $\tilde{\omega}_i^{(\ell)}$ is the expected completion potential at prefix $H_i^{(\ell)}$, and $\omega_i^{(\ell)}$ denotes its log-space representation. The temperature κ controls the sharpness of the categorical distribution, while ϵ maintains numerical stability. The implementation refines bin probabilities before expectation, as detailed in Appendix A.2, with assessment prompts provided in Appendix E.1.

Neighboring-Prefix Supervision Construction. To construct step-level supervision, ASCEND compares the prefix potentials before and after each step. The difference measures the change in assessed task-completion prospects associated with that step. For step ℓ, we define:

$$\delta_i^{(\ell)} = \omega_i^{(\ell)} - \omega_i^{(\ell-1)} = \log \frac{\max(\tilde{\omega}_i^{(\ell)}, \epsilon)}{\max(\tilde{\omega}_i^{(\ell-1)}, \epsilon)}. \tag{4}$$

A positive $\delta_i^{(\ell)}$ indicates an increase in assessed completion prospects, while a negative value indicates a decrease. These differences serve as proxy signals of task progress based on model assessments. Before clipping and normalization, the adjacent differences satisfy $\sum_{\ell=1}^{L_i} \delta_i^{(\ell)} = \omega_i^{(L_i)} - \omega_i^{(0)}$, recovering the net change in prefix potential over the trajectory.

Multi-Granularity Supervision Integration. These neighboring-prefix differences yield fine-grained behavioral supervision, while terminal outcomes provide complementary task-level feedback. Since neighboring-prefix differences and terminal outcomes operate at different granularities and numerical scales, ASCEND clips the potential differences and normalizes the process and outcome signals separately within each task group, with details provided in Appendix A.4. Let $\hat{\delta}_i^{(\ell)}$ and $\hat{r}_i^{\text{out}}$ denote the normalized step-level signal and terminal outcome, respectively. When terminal feedback is available, the two normalized signals are integrated as:

$$\bar{\delta}_i^{(\ell)} = \hat{r}_i^{\text{out}} + \alpha \sum_{j=\ell}^{L_i} \hat{\delta}_i^{(j)}. \tag{5}$$

Here, α weights fine-grained supervision. The suffix term accumulates normalized process signals from the current step onward, while the terminal signal provides task-level guidance.

Together, ASCEND constructs task-aligned fine-grained supervision from visible interaction evidence and integrates it with available terminal feedback for parametric evolution. Appendix A.6 characterizes conditional signal stability and the stepwise structure of integrated supervision.

5 EXPERIMENTS

We examine how ASCEND guides agent evolution. Following the setup in Section 5.1, we ask: **RQ1**: How effective is ASCEND in stateful environments (Section 5.2)? **RQ2**: Does its supervision generalize across interaction forms and support cross-benchmark transfer (Section 5.3)? **RQ3**: How informative is this supervision, and how does it contribute to evolution (Section 5.4)?

5.1 EXPERIMENTAL SETUP

Foundation Model. We adopt Qwen3.5-9B (Qwen Team, 2026) as the foundation model throughout our experiments, enabling direct comparison under a consistent model capacity.

Benchmarks. We conduct experiments in two forms of sequential interaction. For stateful agent execution, we adopt APPWorld (Trivedi et al., 2024), where the agent interacts with applications through multi-step operations. For multi-hop reasoning, we evolve on 2WikiMultiHopQA (2Wiki) (Ho et al., 2020) and additionally evaluate the evolved agent on HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022) without further adaptation to assess transferable capability gains beyond the evolution environment. Benchmark details are provided in Appendix B.1.

Baselines. On APPWorld (Trivedi et al., 2024), we compare ASCEND with Reflexion (Shinn et al., 2023), MemP (Fang et al., 2026), ReasoningBank (Ouyang et al., 2026), and SkillMentor (Bao et al., 2026), along with Vanilla GRPO (Shao et al., 2024), covering both non-parametric adaptation and parametric evolution. For multi-hop reasoning, we compare ASCEND against Vanilla GRPO (Shao et al., 2024) and representative methods with self-induced or environment-provided supervision, including TTRL (Zuo et al., 2026), EMPO (Zhang et al., 2026b), IGPO (Wang et al., 2026), and SIOP (Hu et al., 2026a). Baseline details are provided in Appendix B.2.

Metrics. For APPWorld, we report task success rate (SR) as the primary metric and average execution steps (Avg. Steps) as an efficiency measure. For parametric methods, we report the highest SR within the evolution budget and the corresponding Avg. Steps. For multi-hop reasoning, we report exact match (EM) and F1 across all benchmarks at the evolution round selected by the highest 2Wiki F1 within 4 evolution rounds. Additional evolution and diagnostic metrics are introduced in the corresponding analyses, with definitions provided in Appendix B.3.

Implementation Details. We instantiate parametric evolution with GRPO, which translates ASCEND supervision into agent updates, as detailed in Appendix A.5. We perform 8 evolution rounds for APPWorld and 4 for multi-hop reasoning, with learning rates of 2×10^{-6} and 1×10^{-6} and $\alpha = 0.4$ and 0.2, respectively. More implementation details are provided in Appendix B.4.

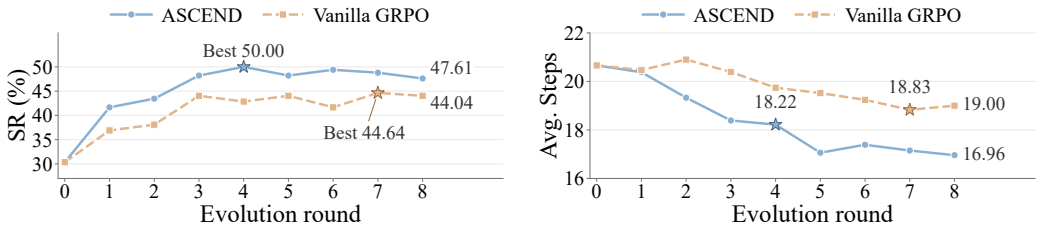
5.2 EFFECTIVENESS OF EVOLUTIONARY SUPERVISION

To investigate **RQ1**, we evaluate ASCEND on APPWorld (Trivedi et al., 2024), a representative stateful interaction environment. Table 1 compares ASCEND with non-parametric and parametric evolution baselines and summarizes their feedback utilization. For parametric methods, we perform 8 evolution rounds and report the best task success rate (SR) with the corresponding average execution steps (Avg. Steps). Figure 3 further tracks SR and Avg. Steps throughout evolution for ASCEND and Vanilla GRPO. From the results, we draw the following findings:

(i) ASCEND achieves the strongest overall evolution performance. As shown in Table 1, ASCEND reaches the best task success rate of 50.00% within 8 evolution rounds. Compared with the base model, ASCEND improves task success rate from 30.35% to 50.00% while reducing average execution steps from 20.66 to 18.22. It also exceeds the strongest non-parametric baseline, SkillMentor, improving task success from 41.07% to 50.00% with slightly fewer steps (18.50 vs. 18.22), and shows larger gains over Reflexion, MemP, and ReasoningBank. Overall, ASCEND delivers the highest task success rate and interaction efficiency among the compared methods.

Table 1: Performance on APPWorld with Qwen3.5-9B. We summarize evolution type, feedback utilization, task success rate (SR), and average execution steps (Avg. Steps) for the methods. For parametric evolution, we report the best SR and corresponding Avg. Steps over 8 rounds. Δ denotes changes from the base model, with ΔSR in percentage points. Darker shading indicates better column rank, and bold marks the optimal results. ASCEND achieves the highest SR with the fewest steps, supporting prefix-potential supervision for agent evolution in stateful environments.

Method	Type	Feedback Utilization	SR ↑	ΔSR ↑	Avg. Steps ↓	ΔSteps ↓
Base Model	None	–	30.35	–	20.66	–
Reflexion	Non-parametric	Self-reflection	31.55	+1.20	20.80	+0.14
MemP	Non-parametric	Procedure distillation	32.74	+2.39	20.60	-0.06
ReasoningBank	Non-parametric	Reasoning distillation	33.33	+2.98	20.60	-0.06
SkillMentor	Non-parametric	Corrective skill distillation	41.07	+10.72	18.50	-2.16
Vanilla GRPO	Parametric	Outcome supervision	44.64	+14.29	18.83	-1.83
ASCEND	Parametric	Outcome + prefix-potential supervision	**50.00**	**+19.65**	**18.22**	**-2.44**



(a) Task Success Rate.

(b) Average Execution Steps.

Figure 3: Evolution dynamics of ASCEND and Vanilla GRPO on APPWorld over 8 rounds. SR (left) and Avg. Steps (right) start from the shared base model. Stars mark best-SR rounds. ASCEND maintains higher SR and requires fewer steps from round 2 onward, indicating effective evolution.

(ii) Prefix-potential supervision provides effective fine-grained guidance. Compared with Vanilla GRPO, which relies on terminal outcome supervision, ASCEND improves the best task success rate from 44.64% to 50.00% while reducing the corresponding average steps from 18.83 to 18.22. This controlled improvement indicates that prefix-potential supervision complements sparse terminal feedback with more informative fine-grained learning signals.

(iii) Prefix-potential supervision provides sustained benefits throughout evolution. As shown in Figure 3, ASCEND maintains higher task success rates across the evolution rounds and requires fewer execution steps from round 2 onward. After 8 rounds, the evolved agent still achieves a higher task success rate than Vanilla GRPO (47.61% vs. 44.04%) with fewer steps (16.96 vs. 19.00). Although both trajectories fluctuate after peak SR, the persistent performance margin supports the value of prefix-potential supervision across the evaluated rounds under the same evolution budget. Longer-horizon self-assessment reliability and potential safeguards are discussed in Appendix D.

Overall, the results demonstrate that ASCEND constructs effective fine-grained supervision that consistently improves parametric evolution in stateful interaction.

5.3 Generality Beyond Stateful Interaction

To investigate **RQ2**, we examine whether ASCEND supervision remains effective beyond stateful interaction and supports transferable capability gains. We perform evolution on 2Wiki and evaluate the evolved agent on 2Wiki (Ho et al., 2020), HotpotQA (Yang et al., 2018) and MuSiQue (Trivedi et al., 2022), with no further adaptation on the latter two. Table 2 compares ASCEND with methods relying on self-induced or environment-provided outcome feedback, reporting EM, F1, and their macro averages. From these results, we draw the following findings:

(i) Prefix-potential supervision remains beneficial for multi-hop reasoning. ASCEND improves 2Wiki from 38.84% to 63.49% in EM and from 54.68% to 72.88% in F1 over the base model. Compared with Vanilla GRPO, it further improves EM from 62.21% to 63.49% and F1 from 71.61% to 72.88%. These improvements indicate that fine-grained supervision from interaction prefixes remains beneficial when interaction shifts from stateful environment operation to multi-hop reasoning.

Table 2: Multi-hop reasoning performance with Qwen3.5-9B. Outcome Feedback distinguishes self-induced from environment-provided terminal signals, and Macro Avg. averages the three benchmarks. Darker shading indicates better rank within each feedback category, with bold marking category-best results. ASCEND achieves the highest macro-average performance, supporting the generality and transferability of prefix-potential supervision beyond stateful interaction.

Method	Outcome Feedback	2Wiki		HotpotQA		MuSiQue		Macro Avg.	
		EM ↑	F1 ↑	EM ↑	F1 ↑	EM ↑	F1 ↑	EM ↑	F1 ↑
Base Model	–	38.84	54.68	36.15	53.83	4.38	13.04	26.46	40.52
TTRL	Self-induced	**40.00**	**55.75**	36.52	54.37	4.13	13.33	26.88	41.15
EMPO	Self-induced	37.56	54.02	35.94	53.74	4.47	**13.80**	25.99	40.52
SIOP	Self-induced	39.86	55.66	**37.17**	**55.14**	**4.55**	13.56	**27.19**	**41.45**
Vanilla GRPO	Environment	62.21	71.61	49.38	64.16	6.08	14.61	39.22	50.13
IGPO	Environment	60.11	70.85	46.75	63.37	6.54	**16.63**	37.80	50.28
ASCEND	Environment	**63.49**	**72.88**	**50.33**	**65.73**	**7.07**	16.57	**40.30**	**51.73**

Table 3: Potential-outcome alignment on trajectories collected for evolutionary updates. We report outcome correlation, pre-final ranking, separation, and selection gain. Prefix potentials become more outcome-aligned as interaction evidence accumulates and provide informative trajectory ranking.

Benchmark	Outcome Correlation			Outcome Ranking		Separation & Selection	
	ρ_1 ↑	ρ_{pre} ↑	$\Delta\rho$ ↑	$\text{AUROC}_{\text{pre}}$ ↑	C_{task} ↑	Outcome Gap ↑	Selection Gain ↑
APPWorld	0.052	0.448	+0.397	0.684	0.725	0.411	+0.100
2Wiki	0.011	0.341	+0.330	0.639	0.644	0.365	+0.071

Table 4: Step-level alignment of applied potential differences $\delta_i^{(\ell)}$ with human-audited action progress. We report precision, recall, no-change alignment, and AUROC on assessable steps. The signals show informative fine-grained supervision across both benchmarks.

Benchmark	Valid / Sampled	Progress		No Change		AUROC ↑
		Precision ↑	Recall ↑	Non-Reward ↑	Neutral ↑	
APPWorld	99 / 100	92.0	58.2	80.0	5.0	0.827
2Wiki	100 / 100	100.0	79.1	100.0	100.0	0.797

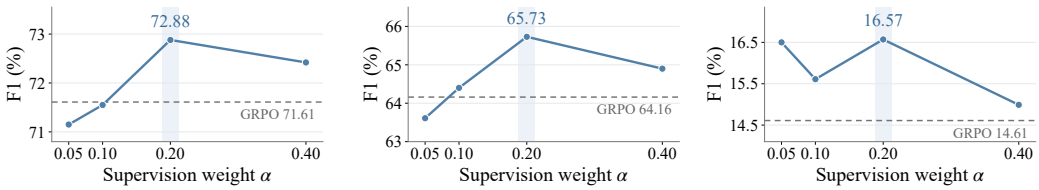
(ii) Environment-provided outcomes offer task-level supervision, while prefix potentials provide complementary fine-grained guidance. Methods relying on self-induced outcome signals achieve macro-average F1 between 40.52% and 41.45%, close to the base model at 40.52%. Methods with environment-provided outcomes exceed 50%, showing stronger reasoning evolution in this setting. Among these methods, ASCEND outperforms Vanilla GRPO and IGPO, reaching the highest macro-average EM and F1 of 40.30% and 51.73%. This supports combining environmental outcomes for task-level alignment with prefix-potential supervision for fine-grained learning.

(iii) Evolutionary supervision supports transferable capability gains. Although evolution is performed only on 2Wiki, ASCEND maintains improvements on HotpotQA and MuSiQue without further adaptation. Compared with Vanilla GRPO, F1 increases from 64.16% to 65.73% on HotpotQA and from 14.61% to 16.57% on MuSiQue, with corresponding EM improvements on both benchmarks. These gains indicate that the supervision constructed during evolution supports capabilities that remain beneficial beyond the evolution environment.

Overall, ASCEND provides generalizable evolutionary supervision across interaction forms and supports transferable capability gains beyond the evolution environment.

5.4 Analysis of Prefix-Potential Supervision

To address **RQ3**, we examine the quality and practical value of prefix-potential supervision through detailed alignment analyses and its contribution to agent evolution.



(a) 2Wiki. (b) HotpotQA. (c) MuSiQue.

Figure 4: Sensitivity to α. Each setting reports results at its highest-2Wiki-F1 round among 4 rounds. Dashed lines denote GRPO with identical selection. $\alpha = 0.2$ performs best across benchmarks.

Supervision Alignment. We analyze *potential-outcome alignment* on eligible trajectories with derived potentials across evolution rounds. Table 3 reports Spearman correlations between final outcomes and potentials at $H_i^{(1)}$ and $H_i^{(L_i-1)}$, denoted by ρ_1 and ρ_{pre}, respectively. At the pre-final prefix, $\text{AUROC}_{\text{pre}}$ measures whether potentials rank full-score trajectories above non-full-score trajectories, while C_{task} measures ranking agreement between potentials and final outcomes within the same task. Outcome Gap compares mean outcomes between the top and bottom potential quartiles. Selection Gain measures the task-averaged outcome improvement from selecting the highest-potential trajectory over random selection. We then assess *step-level alignment* through human judgments of 100 sampled steps per benchmark. Table 4 reports positive-signal precision, progress recall, non-reward and neutral rates for no-change actions, and AUROC for ranking progress above no-change actions, excluding uncertain judgments. Detailed metric definitions are provided in Appendix B.3. From these results, we draw the following findings:

(i) Prefix potentials align with final outcomes and support trajectory ranking. Spearman correlation increases from 0.052 after the first action to 0.448 before the final action on APPWorld, and from 0.011 to 0.341 on 2Wiki. Pre-final AUROC reaches 0.684 and 0.639, and within-task concordance reaches 0.725 and 0.644, respectively. Positive Outcome Gaps and Selection Gains further support trajectory selection. Detailed round-wise results and analyses are provided in Appendix C.3.

(ii) Potential differences align with task progress. Human-audited positive-signal precision reaches 92.0% on APPWorld and 100.0% on 2Wiki, with progress AUROC of 0.827 and 0.797. Progress recall reaches 58.2% and 79.1%, and no-change non-reward rates reach 80.0% and 100.0%, respectively. Audit-source comparisons and case studies appear in Appendices C.2 and C.4.

Contribution to Agent Evolution. We vary $\alpha \in \{0.05, 0.1, 0.2, 0.4\}$ in multi-hop reasoning to assess the contribution of prefix-potential supervision beyond outcome feedback. Figure 4 reports F1 across the three benchmarks at the round attaining the highest 2Wiki F1 within 4 rounds. ASCEND exceeds Vanilla GRPO in 9 of the 12 weight-benchmark combinations, including all 4 weights on MuSiQue. Setting $\alpha = 0.2$ achieves the highest F1 on each benchmark among the evaluated weights, while increasing it to 0.4 reduces performance. These findings support the value of prefix-potential supervision across multiple configurations and the importance of balancing its contribution. More detailed results and analysis are provided in Appendix C.1.

Overall, these analyses substantiate the quality and value of prefix-potential supervision.

6 Conclusion

This work studies fine-grained supervision construction for agent evolution under information constraints. We formulate the setting where behavior-level supervision is constructed without a separately trained process evaluator or task-specific reference structures, and propose ASCEND based on completion-potential estimation and neighboring-prefix differencing. Across evolution rounds, ASCEND constructs fine-grained supervision from the current agent and interaction evidence, integrating local changes in completion prospects with terminal environmental feedback. Experiments across long-horizon stateful interaction and multi-hop reasoning demonstrate effective parametric evolution, generality across interaction forms, and transferable capability gains beyond the evolution environment. Further analyses demonstrate that prefix potentials capture task-aligned information and their neighboring differences provide informative local progress signals. These findings establish prefix potentials as an effective source of fine-grained supervision for agent evolution.

AI USE STATEMENT

We adopt AI assistants for language polishing and writing assistance. Additionally, GPT-6 Astra provided supplementary action-progress judgments for the audit-source comparison in Appendix C.2, with human judgments serving as the primary audit evidence in the paper. All research ideas, methodological design, experiments, analysis, and conclusions were developed and verified by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work studies supervision construction for agent evolution on public benchmarks. APPWorld experiments involve simulated application interactions, and human annotation is limited to assessing recorded agent actions and observations. We encourage responsible application of the method, with appropriate human oversight and safeguards for consequential actions.

REPRODUCIBILITY STATEMENT

Section 4 and Section 5.1 describe ASCEND and the experimental protocol. Appendix A and Appendix B provide the details of supervision construction, optimization procedure, baseline implementations, metric definitions, and evaluation settings. Appendix E outlines the prompt templates, while Appendix C reports supplementary results and diagnostic analyses.

REFERENCES

Xiaoyi Bao, Yuanzhen Xie, Yunzhi Tan, Jinghang Gu, Zhongqing Wang, Chu-Ren Huang, Bo Hu, and Zang Li. Skillmentor: Llm agent self-evolution via learning blind-spot diagnosis. *arXiv preprint arXiv:2607.27360*, 2026.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, et al. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems. *arXiv preprint arXiv:2508.07407*, 2025.

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, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Huanang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Prannay Hebbar, Yogendra Manawat, Samuel Verboomen, Alesia Ivanova, Selvam Palanimalai, Kunal Bhatia, and Vignesh Baskaran. Sia: Self improving ai with harness & weight updates. *arXiv preprint arXiv:2605.27276*, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Senkang Hu, Yong Dai, Xudong Han, Zhengru Fang, Yuzhi Zhao, Sam Tak Wu Kwong, and Yuguang Fang. Self-induced outcome potential: Turn-level credit assignment for agents without verifiers. *arXiv preprint arXiv:2605.04984*, 2026a.

Senkang Hu, Yong Dai, Yuzhi Zhao, Yihang Tao, Yu Guo, Zhengru Fang, Sam Kwong, and Yuguang Fang. Optimizing agentic reasoning with retrieval via synthetic semantic information gain reward. In *Forty-third International Conference on Machine Learning*, 2026b. URL https://openreview.net/forum?id=mUU7k4bN97.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-zero: Self-evolving reasoning LLM from zero data. In *The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=oSGOE5z2iU.

Aman Jaglan and Jarrod Barnes. Continual learning, not training: Online adaptation for agents. *arXiv preprint arXiv:2511.01093*, 2025.

Tianyun Ji, Zhenya Huang, Jiayu Liu, Zirui Liu, Yu Su, and Hongbin Pei. Learning what to remember and what to internalize in llm self-evolution via adaptive memory-parameter coordination. *arXiv preprint arXiv:2608.01234*, 2026.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Derda Kaymak, Gyuhak Kim, Tomoya Kaichi, Tatsuya Konishi, and Bing Liu. Learning after model deployment. *arXiv preprint arXiv:2510.17160*, 2025.

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

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=EqcLAU6gyU.

Taoran Liang, Yang Liu, Shang Luo, Yingguang Yang, Rongrong Zhang, Yingzong Min, Yulin Huang, Jianshen Zhang, Yongzhi Qi, Kefu Xu, et al. Granularity-adaptive credit assignment for long-horizon llm agent reinforcement learning. *arXiv preprint arXiv:2609.12424*, 2026.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, et al. Self: Self-evolution with language feedback. *arXiv preprint arXiv:2310.00533*, 2023.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, et al. Large language model agent: A survey on methodology, applications and challenges. *arXiv preprint arXiv:2503.21460*, 2025.

Xiaotian Luo, Fengxingyu Wang, Chuanrui Hu, Dizhan Xue, and Yafeng Deng. Self-evolving agent harnesses via gated semantic quality-diversity. *arXiv preprint arXiv:2607.13683*, 2026.

Xinbei Ma, Congmin Zheng, Jiyang Qiu, Jiale Hong, Yao Yao, Xiangmou Qu, Jiaxin Yin, Xingyu Lou, Jun Wang, Weiwen Liu, et al. Retrospective progress-aware self-refinement for llm agent training. *arXiv preprint arXiv:2606.14302*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

OpenAI. GPT-6 Astra System Card, 2026. URL https://deploymentsafety.openai.com/gpt-6-astra.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023.

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

Haoyu Sun, Wenxuan Wang, Mingyang Song, Jujie He, Weinan Zhang, Yang Liu, Yang Yang, and Yu Cheng. Agent planning benchmark: A diagnostic framework for planning capabilities in llm agents. *arXiv preprint arXiv:2606.04874*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

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

Guoqing Wang, Sunhao Dai, Guangze Ye, Zeyu Gan, Wei Yao, Yong Deng, Xiaofeng Wu, et al. Information gain-based policy optimization: A simple and effective approach for multi-turn search agents. In *International Conference on Learning Representations*, volume 2026, pp. 104502–104529, 2026.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. Spa-rl: Reinforcing llm agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025a.

Hongru Wang, Yujia Qin, Yankai Lin, Jeff Z Pan, and Kam-Fai Wong. Empowering large language models: Tool learning for real-world interaction. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2983–2986, 2024.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=XLMAMmowdY.

Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, et al. Agentic reasoning for large language models. *arXiv preprint arXiv:2601.12538*, 2026.

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Yiming Xiong, Shengran Hu, and Jeff Clune. Learning to continually learn via meta-learning agentic memory designs. *arXiv preprint arXiv:2602.07755*, 2026.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Zhiling Yan, Dingjie Song, Hanrong Zhang, Wei Liang, Yuxuan Zhang, Yutong Dai, Lifang He, Philip S Yu, Ran Xu, Xiang Li, et al. Openskill: Open-world self-evolution for llm agents. *arXiv preprint arXiv:2606.06741*, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Junjie Ye, Yilong Wu, Songyang Gao, Caishuang Huang, Sixian Li, Guanyu Li, Xiaoran Fan, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Rotbench: A multi-level benchmark for evaluating the robustness of large language models in tool learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 313–333, 2024.

Huining Yuan, Zelai Xu, Huaijie Wang, Xiangmin Yi, Jiaxuan Gao, Xiao-Ping Zhang, Yu Wang, Chao Yu, and Yi Wu. Verifiable process rewards for agentic reasoning. *arXiv preprint arXiv:2605.10325*, 2026.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, et al. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Qiang Zhang, Ruixue Ding, Fanrui Zhang, Xi Chen, Boli Chen, Shihang Wang, Yinfeng Huang, Yi Zheng, Pengjun Xie, Kaipeng Zhang, Jiawei Liu, and Zheng-Jun Zha. Arenaflow: From trajectory ranking to hierarchical credit propagation for open-ended agent rl. *arXiv preprint arXiv:2609.21378*, 2026a.

Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. *Advances in neural information processing systems*, 38:67345–67372, 2026b.

Yudi Zhang, Meng Fang, Zhenfang Chen, and Mykola Pechenizkiy. Self-evolving llm agents with in-distribution optimization. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, volume 306 of *PMLR*, Seoul, South Korea, 2026c.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. Learning to reason without external rewards. In *International Conference on Learning Representations*, volume 2026, pp. 2548–2581, 2026.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

Yifei Zhou, Sergey Levine, Jason E Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=9yusqX9DpR.

Yuxin Zuo, Kaiyan Zhang, Li Sheng, Shang Qu, Ganqu Cui, Xuekai Zhu, Haozhan Li, Xinwei Long, Ermo Hua, Biqing Qi, et al. Ttrl: Test-time reinforcement learning. *Advances in Neural Information Processing Systems*, 38:131459–131483, 2026.

Appendix

Contents

A Implementation Details of ASCEND . . . 15
A.1 Notation . . . 15
A.2 Completion-Potential Estimation . . . 15
A.3 Neighboring-Prefix Supervision Construction . . . 20
A.4 Supervision Normalization and Integration . . . 21
A.5 Parametric Evolution with Constructed Supervision . . . 22
A.6 Analytical Characterization of the Constructed Supervision . . . 23
B Experimental Details . . . 25
B.1 Benchmarks . . . 25
B.2 Baseline Implementations . . . 26
B.3 Evaluation Metrics . . . 28
B.3.1 Task Performance and Execution Efficiency . . . 28
B.3.2 Potential-Outcome Alignment . . . 29
B.3.3 Step-Level Supervision Alignment . . . 31
B.4 Evolution and Evaluation Settings . . . 32
C Additional Results and Analysis. . . . 34
C.1 Sensitivity to the Supervision Weight . . . 34
C.2 Sensitivity to Audit Source . . . 34
C.3 Additional Potential-Outcome Alignment Results . . . 36
C.4 Case Studies of Prefix-Potential Supervision . . . 37
D Discussion of Supervision Quality and Practical Trade-offs . . . 38
E Prompts . . . 39
E.1 Prefix-Potential Estimation Prompts . . . 40
E.1.1 Shared Prefix-Potential Estimation Instruction . . . 40
E.1.2 APPWorld Prefix-Potential Prompt . . . 40
E.1.3 Multi-Hop Reasoning Prefix-Potential Prompt . . . 43
E.1.4 Potential Bins and Output Protocol . . . 46
E.2 Agent Interaction Prompts . . . 47
E.2.1 APPWorld Agent and Environment Prompts . . . 47
E.2.2 Multi-Hop Reasoning Agent and Environment Prompts . . . 49

A Implementation Details of ASCEND

A.1 Notation

The following tables summarize the core notation. Table 5 covers agent evolution and interaction. Tables 6 and 7 describe potential estimation, error analysis, and supervision construction. Table 8 summarizes parametric updates, while Table 9 collects evaluation and alignment measures. Auxiliary metric notation is defined in Appendix B.3. Related variants share an entry.

A.2 Completion-Potential Estimation

Completion-potential estimation constitutes the first stage of ASCEND in Figure 2. Following Section 4, ASCEND elicits task-aligned assessments from the evolving agent itself, drawing on evidence that intrinsic model signals can support supervision without separately trained evaluators (Zhao et al., 2026; Hu et al., 2026a). This subsection details the potential levels and numerical scoring procedure. Appendix E.1 instantiates the shared formulation via environment-specific prompts.

Prefix-Conditioned Assessment. For an execution of task τ_i, the estimator receives query q_i and the action-observation pairs $(a_i^{(\ell)}, o_i^{(\ell)})$ within the visible prefix $H_i^{(\ell)}$. The assessment is conditioned exclusively on this information, with subsequent interactions and terminal outcome scores

Table 5: Notation for agent evolution and interaction.

Notation	Definition
	Evolutionary Process
$t,\ T$	Evolution-round index and prescribed evolution budget.
$A_t,\ A_{t+1}$	Agents retained before and after evolution round t.
E_t	Operating environment at round t.
$\mathcal{S}$	Evolutionary strategy governing the transition from A_t to A_{t+1}.
$\mathcal{H}_t$	Collection of interaction histories accumulated before round t.
	Task Execution and Visible Context
i	Execution index.
$\tau_i,\ \tau$	Task associated with execution i, and a generic task shared by a group of executions.
q_i	Query specifying the objective of task τ_i.
$\ell,\ m$	Interaction-step indices. In interval expressions, m denotes the final step.
j	Auxiliary index over steps, potential bins, or executions, as specified locally.
L_i	Number of interaction steps in execution i.
$a_i^{(\ell)}$	Agent action at step ℓ.
$o_i^{(\ell)}$	Environment observation following action $a_i^{(\ell)}$.
$H_i,\ H_i^{(\ell)}$	Complete execution history and visible prefix after step ℓ. The query-only prefix is $H_i^{(0)} = (q_i)$, and $H_i^{(L_i)} = H_i$.

Table 6: Notation for completion-potential estimation, numerical refinement, and error analysis.

Notation	Definition
	Discrete Potential Assessment
N_{bin}	Number of potential bins, set to 6.
k	Potential-bin index.
b_k	Label of the k-th potential bin, ranging from A to F.
$v(b_k)$	Numerical anchor of bin b_k, with $v(b_k) = 0.2(k-1)$.
$z_{i,k}^{(\ell)}$	Score for bin b_k at prefix $H_i^{(\ell)}$, instantiated as its returned next-token log-probability.
κ	Temperature for normalizing scores over potential bins.
$p_{i,k}^{(\ell)},\ \hat{p}_{i,k}^{(\ell)}$	Temperature-normalized bin probability and its thresholded, renormalized counterpart. The main-text formulation abstracts this numerical refinement to emphasize the supervision construction.
	Potential Representation and Error Analysis
p_{floor}	Probability threshold for retaining bins before renormalization, set to 0.05.
$\tilde{\omega}_i^{(\ell)}$	Expected completion potential computed from refined bin probabilities. For supervision construction, applicable environment-specific adjustments precede the log transformation.
$\omega_i^{(\ell)}$	Effective log-space potential, $\omega_i^{(\ell)} = \log\max(\tilde{\omega}_i^{(\ell)}, \epsilon)$.
ϵ	Positive lower bound before the logarithmic transformation, set to 0.01.
$\mathbb{I}[\cdot]$	Indicator equal to 1 when its argument holds and 0 otherwise.
$\omega_i^{\star(\ell)}$	Reference log-space potential for analytical comparison on the same visible prefix.
$\delta_i^{\star(\ell)}$	Reference potential difference, $\omega_i^{\star(\ell)} - \omega_i^{\star(\ell-1)}$.
ε_{est}	Assumed uniform bound on absolute log-potential estimation error.

Table 7: Notation for supervision construction, normalization, and integration.

Notation	Definition
	Supervision Construction and Signals
$\mathcal{G}$	Mapping from the agent, execution history, terminal outcome, and auxiliary resources to step-aligned supervision.
$\mathcal{R}_i$	Optional oracle annotations, separately trained process evaluators, or task-specific reference structures associated with execution i, with $\mathcal{R}_i = \emptyset$ in ASCEND.
$r_i^{\text{out}},\ \hat{r}_i^{\text{out}}$	Original terminal environmental score and its group-normalized supervision signal after the squared transformation.
$\boldsymbol{\delta}_i,\ \delta_i^{(\ell)}$	Step-supervision sequence and its entry at step ℓ. In ASCEND, $\delta_i^{(\ell)} = \omega_i^{(\ell)} - \omega_i^{(\ell-1)}$ before clipping and normalization.
$d_i^{(\ell)}$	Clipped potential difference entering step-level normalization.
$\hat{\delta}_i^{(\ell)},\ \bar{\delta}_i^{(\ell)}$	Normalized step signal and integrated supervision combining the terminal signal with suffix-aggregated normalized differences.
	Group Normalization and Integration
$\mathcal{Q}$	Indices of candidate executions generated for the same task in a common collection batch.
$\mathcal{I}_{\mathcal{Q}}$	Set of valid execution-step index pairs (i, ℓ) within $\mathcal{Q}$.
c	Symmetric clipping bound for potential differences, set to 1.
$\mu_{\delta,\mathcal{Q}},\ \sigma_{\delta,\mathcal{Q}}$	Mean and standard deviation of clipped step signals over $\mathcal{I}_{\mathcal{Q}}$.
s_i	Squared terminal outcome after clipping to $[0, 1]$.
$\mu_{s,\mathcal{Q}},\ \sigma_{s,\mathcal{Q}}$	Mean and standard deviation of transformed outcomes within $\mathcal{Q}$.
η	Positive numerical stabilizer in supervision normalization.
α	Weight of suffix-aggregated step supervision relative to terminal supervision.

Table 8: Notation for translating constructed supervision into parametric evolution.

Notation	Definition
	Executor and Token-Aligned Supervision
$\theta,\ \theta_t,\ \theta_0$	Executor parameters being updated, parameters at the start of round t, and initial parameters, respectively.
π_θ	Action-token distribution of the executor parameterized by θ.
π_{old}	Action-token distribution of the rollout executor for the current epoch, fixed during the corresponding updates.
π_{ref}	Action-token distribution of the starting executor of an evolution round, frozen throughout that round.
n	Token-position index.
$u_{i,n}$	Action token at position n in execution i.
$H_{i,<n}$	Interaction context preceding token $u_{i,n}$, including available environment observations.
$\mathcal{T}_i^{(\ell)},\ \mathcal{T}_i$	Generated action-token positions included in optimization for step ℓ and their union over execution i.
$\hat{A}_i^{(n)}$	Token advantage inherited from integrated step supervision, with $\hat{A}_i^{(n)} = \bar{\delta}_i^{(\ell)}$ for $n \in \mathcal{T}_i^{(\ell)}$.
	Optimization and Reference Regularization
$\zeta_i^{(n)}(\theta)$	Action-token probability ratio between the current executor and π_{old}.
$\epsilon_{\text{low}},\ \epsilon_{\text{high}}$	Lower and upper margins for clipping the probability ratio around 1.
$\mathcal{L}_{i,n}^{\text{clip}}(\theta)$	Clipped token-level surrogate incorporating $\hat{A}_i^{(n)}$.
$\hat{D}_{\text{KL},i}^{(n)}$	Token-level reference penalty comparing π_θ with π_{ref}.
β	Reference-regularization coefficient.
$\mathcal{D}_t$	Optimization batch within evolution round t.
M_t	Total number of valid action tokens in $\mathcal{D}_t$.
$\mathcal{J}_t(\theta)$	Token-averaged optimization objective for $\mathcal{D}_t$.

Table 9: Notation for task performance, evolution-round selection, and supervision alignment.

Notation / Measure	Definition
	Task Performance and Evolution-Round Selection
SR	Percentage of APPWorld executions satisfying all evaluation checks.
Avg. Steps	Mean interaction length across successful and unsuccessful executions.
ΔSR, ΔSteps	Changes from the base model in percentage points and interaction steps.
EM_B, $\mathrm{F1}_B$	Exact match and answer F1 on benchmark B, reported as percentages.
$\mathrm{MacroAvg}(M)$	Equal-weight average of metric $M \in \{\mathrm{EM}, \mathrm{F1}\}$ over the three reasoning benchmarks.
$t^\star_{\mathrm{APPWorld}}$	Evolution round attaining the highest APPWorld SR within the evaluation budget.
$t^\star_{\mathrm{reasoning}}$	Evolution round attaining the highest 2Wiki F1, with the same agent evaluated across all three reasoning benchmarks.
	Potential-Outcome Alignment
$\rho_{1,t}$, $\rho_{\mathrm{pre},t}$	Within-round Spearman correlations between final outcomes and expected potentials after the first action and before the final action.
ρ_1, ρ_{pre}	Equal-weight averages of the corresponding correlations over rounds with defined values.
$\Delta\rho$	Change in outcome correlation, $\rho_{\mathrm{pre}} - \rho_1$.
$\mathrm{AUROC}_{\mathrm{pre},t}$, $\mathrm{AUROC}_{\mathrm{pre}}$	Pre-final potential discrimination between outcomes equal to 1 and below 1, computed per round and averaged equally across defined rounds.
$C(\mathcal{Q})$	Potential-outcome ordering agreement over unequal-outcome candidate pairs in $\mathcal{Q}$, with potential ties receiving half credit.
$C_{\mathrm{task},t}$, C_{task}	Concordance averaged over groups within tasks and then over tasks in round t, and its equal-weight cross-round average.
Outcome Gap	Difference in mean environmental outcomes between the upper and lower pre-final potential quartile groups, pooled across rounds with boundary ties retained.
Selection Gain	Task-averaged outcome improvement from selecting highest-potential candidates over uniform random selection within the same candidate group.
	Step-Level Supervision Alignment
$\epsilon_{\mathrm{audit}}$	Threshold separating positive, neutral, and negative applied step signals before normalization, set to 0.02.
Valid / Sampled	Numbers of assessable and sampled execution-step pairs.
Label Agreement	Percentage of mutually assessable steps assigned identical human and model labels.
Progress Precision	Fraction of positive step signals corresponding to audited progress actions.
Progress Recall	Fraction of audited progress actions receiving positive step signals.
No-Change Non-Reward	Fraction of audited no-change actions satisfying $\delta_i^{(\ell)} \leq \epsilon_{\mathrm{audit}}$.
No-Change Neutral	Fraction of audited no-change actions satisfying $\lvert\delta_i^{(\ell)}\rvert \leq \epsilon_{\mathrm{audit}}$.
$\mathrm{AUROC}_{\mathrm{progress}}$	Ranking of audited progress actions above no-change actions by applied potential differences, with half credit for ties.

Table 10: Shared semantics of completion-potential bins. Numerical anchors define an ordered scoring scale. Appendix E.1 specifies the corresponding environment-dependent criteria.

Bin	Score	Assessment	Abstract Meaning
A	0.0	Off Track	The visible state is damaged or substantially misaligned with the task, making successful completion unlikely.
B	0.2	High Risk	Critical requirements or a viable recovery path remain unresolved.
C	0.4	Partial Progress	The execution is partly on track, but substantial task requirements remain incomplete or unsupported.
D	0.6	Substantial Progress	Important requirements are supported, although material uncertainty or unfinished work remains.
E	0.8	Likely Completion	Visible evidence supports likely task completion, pending direct external confirmation.
F	1.0	Verified Completion	Direct external evidence confirms satisfaction of the exact task objective.

excluded. Agent actions and environment observations occupy separate prompt fields, making their evidential roles explicit. Completion potential represents the current agent assessment of task-completion prospects given the task objective and visible prefix. The evolving agent evaluates prefixes from $H_i^{(0)} = (q_i)$ through $H_i^{(L_i)}$. The query-only estimate provides an initial assessment of task-completion prospects and a reference for evaluating the first step. The complete-prefix estimate follows the same scoring procedure, while terminal environmental feedback enters the separate outcome-supervision branch. Adjustments to the estimated potential are detailed in Appendix A.3.

Ordered Potential Levels. ASCEND defines 6 ordered bins, $\{b_k\}_{k=1}^{6} = \{A, B, C, D, E, F\}$, with numerical anchors $v(b_k) = 0.2(k-1)$. Table 10 summarizes their shared semantics. The anchors and ordering remain identical across environments. The prompts assess task-completion prospects from application-state evidence on APPWorld (Trivedi et al., 2024) and retrieval and reasoning evidence on multi-hop reasoning (Ho et al., 2020). Observed progress, evidence sufficiency, and readiness for successful submission inform this assessment. Readiness for submission guides bin E, while bin F requires direct confirmation of task completion in the visible observations. Numerical anchors encode the ordered assessment levels.

Let $z_{i,k}^{(\ell)}$ denote the returned next-token log-probability of bin b_k. Temperature normalization gives:

$$p_{i,k}^{(\ell)} = \frac{\exp\left(z_{i,k}^{(\ell)}/\kappa\right)}{\sum_{j=1}^{N_{\text{bin}}} \exp\left(z_{i,j}^{(\ell)}/\kappa\right)}, \qquad \kappa = 0.5. \tag{6}$$

The scoring temperature κ controls the concentration of this distribution independently of the decoding temperature during environmental interaction.

Probability Refinement and Expectation. Small probabilities assigned to weakly supported bins can introduce residual contributions to the expected potential. To reduce their numerical influence, we truncate entries below $p_{\text{floor}} = 0.05$. The retained probabilities are then renormalized to preserve a valid categorical distribution, mathematically formulated as:

$$\hat{p}_{i,k}^{(\ell)} = \frac{p_{i,k}^{(\ell)} \mathbb{I}\left[p_{i,k}^{(\ell)} \geq p_{\text{floor}}\right]}{\sum_{j=1}^{N_{\text{bin}}} p_{i,j}^{(\ell)} \mathbb{I}\left[p_{i,j}^{(\ell)} \geq p_{\text{floor}}\right]}. \tag{7}$$

The denominator remains positive because the largest probability among 6 bins is at least $1/6 > p_{\text{floor}}$. The implemented completion-potential estimate is:

$$\tilde{\omega}_i^{(\ell)} = \sum_{k=1}^{N_{\text{bin}}} \hat{p}_{i,k}^{(\ell)} v(b_k). \tag{8}$$

This refinement concentrates the expectation on supported potential levels.

Log-Space Representation. An absolute potential increment has different relative significance depending on the preceding completion estimate. ASCEND therefore adopts a logarithmic representation to express relative changes in assessed completion prospects. Before the differencing stage in Figure 2, the estimate is mathematically transformed as:

$$\omega_i^{(\ell)} = \log \max\left(\tilde{\omega}_i^{(\ell)}, \epsilon\right), \qquad \epsilon = 0.01. \tag{9}$$

Above the lower bound, equal multiplicative changes yield equal potential differences, providing a consistent interpretation of relative progress across potential levels. These log-ratios also accumulate additively over successive steps. The lower bound ϵ ensures numerical stability by preventing undefined logarithms and limiting sensitivity near zero, with $\omega_i^{(\ell)} \in [\log \epsilon, 0]$.

Missing-Score Handling. Returned top-log-probability entries may omit individual labels. Each missing label receives a fallback log-probability of -20.0 before temperature normalization, maintaining a fixed 6-label candidate space. If all labels are unavailable, the estimator assigns the configured fallback potential of 0.0 and records the estimate as invalid. This exception-handling value is separate from the model-estimated query-only potential.

A.3 Neighboring-Prefix Supervision Construction

Following completion-potential estimation, ASCEND transforms the potential sequence into fine-grained step-level signals, corresponding to the second stage of Figure 2. The construction accounts for both local changes in completion prospects and environment-specific execution semantics.

Action-Aligned Potential Changes. The prefixes $H_i^{(\ell-1)}$ and $H_i^{(\ell)}$ differ by the newly observed action-observation pair $(a_i^{(\ell)}, o_i^{(\ell)})$. Their potential difference assigns the change in estimated completion prospects to step ℓ, mathematically formulated as:

$$\delta_i^{(\ell)} = \omega_i^{(\ell)} - \omega_i^{(\ell-1)} = \log \frac{\max(\tilde{\omega}_i^{(\ell)}, \epsilon)}{\max(\tilde{\omega}_i^{(\ell-1)}, \epsilon)}. \tag{10}$$

A positive signal indicates increased estimated completion prospects, a negative signal indicates a decrease, and a near-zero signal indicates an unchanged assessment. This construction evaluates each step in its preceding interaction context. The query-conditioned initial potential supplies the reference for the first step, so its signal reflects the change following the first step.

Potential Inheritance for Rejected Actions. A format-rejected action contributes an error observation without necessarily changing the accepted task state. The altered prefix can nevertheless produce a different potential estimate, introducing variation unrelated to substantive task progress. Potential inheritance addresses this discrepancy by preserving the preceding assessment for explicitly retryable format rejections in the multi-hop reasoning environment:

$$\widetilde{\omega}_i^{(\ell)} \leftarrow \widetilde{\omega}_i^{(\ell-1)}, \qquad \delta_i^{(\ell)} = 0. \tag{11}$$

The same treatment applies when a terminal format rejection exhausts the interaction budget. This rule is restricted to the corresponding environment responses. Other execution errors retain their prefix-conditioned assessments because their implications depend on the observed task state.

Terminal-Step Treatment. Terminal actions have different semantics across environments. In APPWorld, a recognized submission-only action without an error indication inherits the preceding potential. This convention treats the completion declaration as a boundary event, with task fulfillment assessed through complementary environmental outcome feedback. Submission steps with error indications retain their estimated potential changes. In multi-hop reasoning, an accepted answer submission retains its model-estimated terminal potential, allowing the submitted answer and supporting evidence to inform the final prefix assessment. Terminal format rejections follow the inheritance rule above. These treatments are applied before differencing and maintain the separation between potential-based supervision and environmental outcomes.

Consistency across Prefixes. The local construction also admits an execution-level interpretation. Let $\tilde{\omega}_i^{(\ell)}$ denote the effective potential after the preceding adjustments. Before clipping and normalization, adjacent differences telescope over any contiguous interval:

$$\sum_{j=\ell}^{m} \delta_i^{(j)} = \omega_i^{(m)} - \omega_i^{(\ell-1)}, \qquad 1 \le \ell \le m \le L_i. \tag{12}$$

The accumulated local signals thus recover the net estimated potential change over the segment. For $m = L_i$, this relation connects the prefix preceding step ℓ to the terminal prefix, motivating the suffix aggregation in Appendix A.4. Normalization subsequently regulates the scale of these fine-grained signals for integration with terminal outcome supervision.

A.4 Supervision Normalization and Integration

The final supervision-construction stage in Figure 2 combines neighboring-prefix differences with terminal environmental feedback. These signals describe complementary aspects of execution, with potential differences characterizing local changes and terminal outcomes assessing task fulfillment. Their numerical scales and distributions differ, motivating separate normalization before integration.

Normalization of Fine-Grained Signals. Log-space potential differences can exhibit large magnitudes when adjacent estimates span substantially different completion levels. Clipping moderates the influence of such transitions, mathematically formulated as follows:

$$d_i^{(\ell)} = \mathrm{clip}\left(\delta_i^{(\ell)}, -c, c\right), \qquad c = 1. \tag{13}$$

The clipped signal retains the direction of the original difference while bounding its magnitude.

The clipped differences are centered and standardized across valid steps within each task group. Let $\mathcal{Q}$ denote the indices of candidate trajectories generated for task τ in the same collection batch, and let $\mathcal{I}_{\mathcal{Q}}$ contain their valid execution-step pairs. For $|\mathcal{I}_{\mathcal{Q}}| > 0$, we define:

$$\mu_{\delta,\mathcal{Q}} = \frac{1}{|\mathcal{I}_{\mathcal{Q}}|} \sum_{(i,\ell)\in\mathcal{I}_{\mathcal{Q}}} d_i^{(\ell)}, \qquad \sigma_{\delta,\mathcal{Q}}^2 = \frac{1}{|\mathcal{I}_{\mathcal{Q}}|} \sum_{(i,\ell)\in\mathcal{I}_{\mathcal{Q}}} \left(d_i^{(\ell)} - \mu_{\delta,\mathcal{Q}}\right)^2. \tag{14}$$

The normalized step signal is mathematically formulated as follows:

$$\hat{\delta}_i^{(\ell)} = \frac{d_i^{(\ell)} - \mu_{\delta,\mathcal{Q}}}{\sqrt{\sigma_{\delta,\mathcal{Q}}^2 + \eta}}, \qquad (i,\ell) \in \mathcal{I}_{\mathcal{Q}}, \tag{15}$$

where $\eta > 0$ is a numerical stabilizer distinct from the potential lower bound ϵ. Centering expresses each signal relative to the group-level reference, while normalization moderates variation in signal scale across tasks. Invalid positions contribute zero to suffix aggregation. For groups without valid steps, the potential-based contribution is set to 0. Here $\delta_i^{(\ell)}$ describes the log-space potential change before clipping and normalization, while $\hat{\delta}_i^{(\ell)}$ expresses the clipped signal relative to its task group.

Normalization of Terminal Outcomes. Environmental outcomes provide a complementary reference for task fulfillment. The outcome branch applies a squared transformation to bounded scores:

$$s_i = \left[\mathrm{clip}\left(r_i^{\mathrm{out}}, 0, 1\right)\right]^2. \tag{16}$$

This monotonic transformation preserves outcome ordering and places greater emphasis on differences near successful completion. Binary outcomes remain unchanged.

Let $\mu_{s,\mathcal{Q}}$ and $\sigma_{s,\mathcal{Q}}$ denote the mean and standard deviation of transformed outcomes s_i within $\mathcal{Q}$. The normalized terminal signal is mathematically formulated as:

$$\hat{r}_i^{\mathrm{out}} = \frac{s_i - \mu_{s,\mathcal{Q}}}{\sigma_{s,\mathcal{Q}} + \eta}. \tag{17}$$

This group-relative signal distinguishes stronger and weaker task outcomes among candidate executions. Groups with identical outcomes contribute zero outcome advantage, leaving potential-based supervision available to differentiate their intermediate behaviors.

Suffix Aggregation and Integration. An interaction step can contribute to subsequent progress by establishing evidence or enabling later actions. Suffix aggregation incorporates this temporal dependence by accumulating normalized signals from the current step through the end of execution. The integrated supervision is mathematically formulated as:

$$\bar{\delta}_i^{(\ell)} = \hat{r}_i^{\text{out}} + \alpha \sum_{j=\ell}^{L_i} \hat{\delta}_i^{(j)}. \tag{18}$$

The terminal component supplies a shared task-level reference across the trajectory, while the suffix component differentiates steps according to current and subsequent potential changes. Separate normalization makes α a more interpretable control over their relative contribution. The resulting step-aligned signals guide the parametric updates described in Appendix A.5.

A.5 Parametric Evolution with Constructed Supervision

ASCEND constructs supervision for parametric evolution. In our implementation, GRPO (Shao et al., 2024) translates the integrated signals into agent updates. This subsection specifies their assignment to action tokens and the resulting optimization procedure.

From Step Supervision to Token Advantages. Let π_θ denote the action-token distribution of the agent parameterized by θ. For trajectory H_i, let $u_{i,n}$ denote an action token at position n, and let $H_{i,<n}$ contain the preceding interaction context, including available environment observations. Define $\mathcal{T}_i^{(\ell)}$ as the set of trainable token positions associated with action $a_i^{(\ell)}$. The action and its resulting observation jointly determine the step-level assessment. Accordingly, all trainable tokens belonging to the same action receive its integrated supervision:

$$\hat{A}_i^{(n)} = \bar{\delta}_i^{(\ell)}, \qquad n \in \mathcal{T}_i^{(\ell)}. \tag{19}$$

Environment observations contribute conditioning information but do not receive action-generation loss. This assignment preserves step-level supervision within token-level parametric updates.

Clipped Group-Relative Optimization. Within each epoch of evolution round t, the rollout executor π_{old} supplies the probabilities for the token-level optimization ratio:

$$\zeta_i^{(n)}(\theta) = \frac{\pi_\theta(u_{i,n} \mid H_{i,<n})}{\pi_{\text{old}}(u_{i,n} \mid H_{i,<n})}. \tag{20}$$

These rollout probabilities and constructed advantages remain fixed during the corresponding update. Prefix-potential estimates therefore serve as supervision targets.

Following GRPO, a clipped surrogate moderates changes in action probabilities:

$$\mathcal{L}_{i,n}^{\text{clip}}(\theta) = \min\left[\zeta_i^{(n)}(\theta)\hat{A}_i^{(n)}, \text{clip}\Big(\zeta_i^{(n)}(\theta), 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\Big)\hat{A}_i^{(n)}\right]. \tag{21}$$

The margins ϵ_{low} and ϵ_{high} determine the lower and upper clipping bounds.

Reference Regularization and Objective. Each evolution round retains its starting agent as a frozen reference π_{ref}. This regularization moderates behavioral drift during the round, while the constructed supervision guides task-relevant improvements. The regularization penalty is:

$$\hat{D}_{\text{KL},i}^{(n)} = \frac{\pi_{\text{ref}}(u_{i,n} \mid H_{i,<n})}{\pi_\theta(u_{i,n} \mid H_{i,<n})} - \log \frac{\pi_{\text{ref}}(u_{i,n} \mid H_{i,<n})}{\pi_\theta(u_{i,n} \mid H_{i,<n})} - 1. \tag{22}$$

Here, π_θ denotes the executor being updated, while π_{ref} remains fixed throughout the round.

Let $\mathcal{D}_t$ denote an optimization batch within evolution round t, let $\mathcal{T}_i = \bigcup_{\ell=1}^{L_i} \mathcal{T}_i^{(\ell)}$, and let $M_t = \sum_{H_i \in \mathcal{D}_t} |\mathcal{T}_i|$. The parametric update maximizes:

$$\mathcal{J}_t(\theta) = \frac{1}{M_t} \sum_{H_i \in \mathcal{D}_t} \sum_{n \in \mathcal{T}_i} \left[\mathcal{L}_{i,n}^{\text{clip}}(\theta) - \beta \hat{D}_{\text{KL},i}^{(n)}\right], \tag{23}$$

where β controls reference regularization. Averaging over valid action tokens matches the token-level optimization objective. The clipping margins, regularization coefficients, and optimization settings are specified in the subsequent Appendix B.4.

Algorithm 1 ASCEND supervision construction and GRPO optimization for parametric evolution

Input: Initial agent A_0 with parameters θ_0, evolution budget T, supervision weight α
Output: Evolved agent under the specified evaluation protocol

for $t = 0, \ldots, T-1$ **do**
  Initialize the executor $\pi_\theta \leftarrow \pi_{\theta_t}$ and freeze the round reference $\pi_{\text{ref}} \leftarrow \pi_{\theta_t}$
  for each epoch within round t **do**
    Fix the rollout executor $\pi_{\text{old}} \leftarrow \pi_\theta$
    Collect all task-grouped trajectories and environmental outcomes for this epoch
    Estimate prefix potentials with the current agent ▷ Appendix A.2
    Apply potential adjustments and derive $\delta_i^{(\ell)}$ ▷ Appendix A.3
    Partition the collected data into optimization batches
    for each optimization batch **do**
      Compute token probabilities under π_{old} and π_{ref}
      Clip potential differences and normalize process and outcome signals separately within each task group ▷ Appendix A.4
      Form $\bar{\delta}_i^{(\ell)}$ and assign token advantages $\hat{A}_i^{(n)}$
    end for
    for each prepared optimization batch **do**
      Update the executor through Equation 23, holding supervision targets, π_{old}, and π_{ref} fixed
    end for
  end for
  Evaluate the candidate and determine A_{t+1} with parameters θ_{t+1} according to the evolution protocol
end for
return The agent selected across evolution rounds according to Appendix B.4

Evolutionary Optimization Procedure. Algorithm 1 summarizes the supervision and optimization cycle. Within each epoch of an evolution round, the agent completes rollout collection and estimates prefix potentials. Environmental outcomes and neighboring-prefix differences undergo separate normalization before integration into advantages. After these targets and token probabilities under π_{old} and π_{ref} are computed for all batches, GRPO translates the constructed supervision into a candidate agent update. Evaluation and evolution-round selection follow Appendix B.4.

A.6 Analytical Characterization of the Constructed Supervision

The supervision construction raises two questions. First, when completion-potential estimation contains errors, under what conditions do neighboring-prefix differences retain the direction of the underlying prospect change? Second, after normalization and integration with terminal outcomes, how do these signals distinguish individual steps? We address the first question through a conditional error bound and the second through an exact identity of the integrated supervision. These results characterize the supervision mechanism, with its empirical effectiveness examined in Section 5.2.

► **Stability of Potential Differences.** ASCEND derives local supervision by subtracting two estimated prefix potentials. Since both estimates can contain errors, a sufficiently pronounced reference change provides a condition for preserving signal direction. The following analysis makes this condition explicit. Consider the effective log-space potentials entering the differencing stage in Appendix A.3, after environment-specific adjustments. Let $\omega_i^{\star(\ell)}$ denote a reference log-space completion potential on the same prefixes, and define:

$$\delta_i^{\star(\ell)} = \omega_i^{\star(\ell)} - \omega_i^{\star(\ell-1)}. \tag{24}$$

The reference sequence provides an analytical comparator for assessing estimation error. It is not an additional supervision source required during parametric evolution.

Proposition 1. Stability under bounded estimation error. Suppose that the estimated and reference potentials satisfy the following condition:

$$\left|\omega_i^{(j)} - \omega_i^{\star(j)}\right| \le \varepsilon_{\text{est}}, \qquad j = 0, \ldots, L_i, \qquad \varepsilon_{\text{est}} \ge 0. \tag{25}$$

Then every neighboring-prefix difference satisfies:

$$\left|\delta_i^{(\ell)} - \delta_i^{\star(\ell)}\right| \le 2\varepsilon_{\text{est}}, \qquad 1 \le \ell \le L_i. \tag{26}$$

In particular, if $|\delta_i^{\star(\ell)}| > 2\varepsilon_{\text{est}}$, the estimated difference has the same sign as the reference difference.

Proof. The discrepancy between the two differences is the difference between their endpoint errors. By the triangle inequality, $\left|\delta_i^{(\ell)} - \delta_i^{\star(\ell)}\right|$ satisfies:

$$\begin{aligned}\left|\delta_i^{(\ell)} - \delta_i^{\star(\ell)}\right| &= \left|\left(\omega_i^{(\ell)} - \omega_i^{\star(\ell)}\right) - \left(\omega_i^{(\ell-1)} - \omega_i^{\star(\ell-1)}\right)\right| \\ &\le \left|\omega_i^{(\ell)} - \omega_i^{\star(\ell)}\right| + \left|\omega_i^{(\ell-1)} - \omega_i^{\star(\ell-1)}\right| \\ &\le 2\varepsilon_{\text{est}}.\end{aligned} \tag{27}$$

Therefore, $\delta_i^{(\ell)} \ge \delta_i^{\star(\ell)} - 2\varepsilon_{\text{est}} > 0$ when $\delta_i^{\star(\ell)} > 2\varepsilon_{\text{est}}$. Similarly, $\delta_i^{(\ell)} \le \delta_i^{\star(\ell)} + 2\varepsilon_{\text{est}} < 0$ when $\delta_i^{\star(\ell)} < -2\varepsilon_{\text{est}}$. The result also extends to accumulated changes over an execution segment. By the telescoping identity in Equation 12, we can yield:

$$\begin{aligned}\left|\sum_{j=\ell}^{m}\left(\delta_i^{(j)} - \delta_i^{\star(j)}\right)\right| &= \left|\left(\omega_i^{(m)} - \omega_i^{\star(m)}\right) - \left(\omega_i^{(\ell-1)} - \omega_i^{\star(\ell-1)}\right)\right| \\ &\le 2\varepsilon_{\text{est}}, \qquad 1 \le \ell \le m \le L_i.\end{aligned} \tag{28}$$

Thus, under the same assumption, the discrepancy in accumulated changes depends only on the two endpoints, without an additional factor proportional to segment length.

► **Step Distinctions after Supervision Integration.** Terminal feedback assigns a common task-level assessment to an execution. ASCEND supplements this shared assessment with suffix-aggregated step signals so that different actions can receive different supervision. The relevant question is whether this distinction remains explicit after the two components are combined.

Proposition 2. Temporal contrast of integrated supervision. For the integrated signals defined in Equation 18, any two steps $1 \le \ell < m \le L_i$ satisfy:

$$\bar{\delta}_i^{(\ell)} - \bar{\delta}_i^{(m)} = \alpha \sum_{j=\ell}^{m-1} \hat{\delta}_i^{(j)}. \tag{29}$$

In particular:

$$\bar{\delta}_i^{(\ell)} - \bar{\delta}_i^{(\ell+1)} = \alpha\hat{\delta}_i^{(\ell)}, \qquad 1 \le \ell < L_i. \tag{30}$$

For $\alpha > 0$, adjacent steps therefore receive distinct integrated signals whenever $\hat{\delta}_i^{(\ell)} \ne 0$.

Proof. Expanding the two integrated signals yields:

$$\begin{aligned}\bar{\delta}_i^{(\ell)} - \bar{\delta}_i^{(m)} &= \left(\hat{r}_i^{\text{out}} + \alpha \sum_{j=\ell}^{L_i} \hat{\delta}_i^{(j)}\right) - \left(\hat{r}_i^{\text{out}} + \alpha \sum_{j=m}^{L_i} \hat{\delta}_i^{(j)}\right) \\ &= \alpha \sum_{j=\ell}^{m-1} \hat{\delta}_i^{(j)}.\end{aligned} \tag{31}$$

The terminal component and the common suffix cancel. Setting $m = \ell + 1$ yields Equation 30.

The identity separates the roles of the two supervision components. Terminal feedback supplies a shared trajectory-level reference, while the potential branch determines the contrast between steps within that trajectory. The contrast reflects normalized, group-relative contributions. Its sign can differ from that of the original potential difference as normalization includes centering.

A further consequence concerns task groups with identical terminal outcomes. In such a group, all transformed outcomes s_i coincide, and Equation 17 gives $\hat{r}_i^{\text{out}} = 0$ for every execution. For $\alpha > 0$, any execution with a nonzero normalized step-signal sequence still has at least one nonzero integrated signal. Suppose all its integrated signals were zero. The final signal would imply $\hat{\delta}_i^{(L_i)} = 0$, and Equation 30 would imply $\hat{\delta}_i^{(\ell)} = 0$ for every preceding step, contradicting the nonzero sequence.

Consequently, the potential branch can supply additional supervision for distinguishing intermediate action contributions when terminal outcomes provide no within-group distinction.

B EXPERIMENTAL DETAILS

This section outlines the experimental framework for evaluating ASCEND. Specifically, Section B.1 introduces the benchmarks alongside their roles across agent evolution and transfer evaluation, while Section B.2 details the baseline implementations. Section B.3 and Section B.4 specify the evaluation metrics and experimental settings, respectively.

B.1 BENCHMARKS

We evaluate ASCEND across two complementary paradigms of sequential agent interaction: stateful application interaction and retrieval-based multi-hop reasoning. The former investigates whether fine-grained supervision can emerge from long-horizon environment interactions whose correctness is grounded in the resulting application state. The latter assesses whether this supervision principle generalizes to iterative information seeking and evidence integration. Specifically, APPWorld serves for in-domain agent evolution and evaluation, while 2WikiMultiHopQA acts as the evolution benchmark for multi-hop reasoning. HotpotQA and MuSiQue provide evaluation questions without additional evolutionary updates, while their passages contribute to the shared retrieval corpus.

APPWorld. APPWorld provides a controllable benchmark environment for evaluating interactive coding agents on realistic digital tasks (Trivedi et al., 2024). It simulates 9 everyday applications operated through 457 APIs, while providing 750 benchmark tasks that require agents to perform complex multi-step operations across multiple applications. Instead of matching predefined action sequences, APPWorld evaluates task completion via rigorous programmatic, state-based unit tests. This evaluation mechanism accommodates diverse valid execution strategies while systematically identifying both incomplete task execution and unintended side effects on the environment state.

During each interaction, the agent generates executable Python code that invokes the available application APIs, and the environment returns the corresponding execution output as the subsequent observation. Successfully addressing a task demands API discovery, authentication, pagination, information aggregation, cross-application reasoning, state modification, and final task submission. Consequently, the resulting trajectories exhibit both productive and unproductive intermediate behaviors, including information-gathering steps, failed API calls, incorrect state mutations, and recovery attempts. These characteristics make APPWorld well suited for assessing whether ASCEND can derive fine-grained supervision from visible interaction progress when direct task-level feedback is provided exclusively at the trajectory conclusion.

2WikiMultiHopQA. 2WikiMultiHopQA (2Wiki) is a large-scale multi-hop question-answering benchmark constructed from Wikipedia articles and Wikidata relations (Ho et al., 2020). 2Wiki contains 192,606 questions, comprising 167,454 training examples and 12,576 examples in each development and test split. The agent evolves on the training split and is evaluated on the development split. Its four categories cover comparison, inference, compositional, and bridge-comparison reasoning. Beyond answers and sentence-level supporting facts, the dataset provides evidence triples tracing the reasoning path from question to answer. Consequently, answering a question necessitates retrieving multiple relevant facts and integrating them across a valid reasoning chain.

2Wiki serves as the evolution-domain benchmark for the retrieval-based reasoning experiments. The agent iteratively retrieves relevant passages, examines the accumulated evidence, and produces a final response once adequate support has been collected. Only the benchmark question and observations returned by the environment are visible to the agent, while ground-truth answers, supporting-fact annotations, and evidence paths remain strictly excluded from the context. These heterogeneous reasoning structures offer an appropriate testbed for examining whether prefix-potential differences can allocate credit to intermediate retrieval and reasoning decisions.

HotpotQA. HotpotQA contains approximately 113K Wikipedia-based question-answer pairs, including 7,405 development examples for our evaluation (Yang et al., 2018). Its questions necessitate reasoning over multiple supporting documents and encompass both bridge questions, where an intermediate entity links the required evidence, and comparison questions, where attributes across multiple entities are contrasted. The dataset additionally provides sentence-level supporting-fact annotations. However, these annotations remain withheld from the agent during interaction.

HotpotQA serves as a transfer-evaluation benchmark. Following evolution on 2WikiMultiHopQA, the resulting agent is directly assessed on HotpotQA without additional parameter updates or benchmark-specific adaptation. This evaluation assesses whether capabilities developed on 2Wiki extend to benchmark-specific question distributions under shared retrieval conditions.

MuSiQue. MuSiQue is a multi-hop question-answering benchmark constructed by systematically composing connected single-hop questions (Trivedi et al., 2022). MuSiQue-Ans contains 24,814 answerable questions requiring 2 to 4 reasoning hops, with 19,938 training, 2,417 validation, and 2,459 test examples (Trivedi et al., 2022). We evaluate on the 2,417-example validation split. Its construction promotes dependencies between reasoning steps and reduces single-hop shortcuts. Answering these questions requires integrating retrieved evidence across connected reasoning steps.

The answerable split of MuSiQue serves as an additional transfer-evaluation benchmark. Parallel to the HotpotQA setup, the agent evolved on 2WikiMultiHopQA is evaluated directly on MuSiQue without further optimization. MuSiQue thereby assesses whether the evolved agent generalizes effectively to extended, tightly coupled reasoning chains.

Unified Interactive Formulation. For the three multi-hop reasoning benchmarks, we convert each question into an identical sequential search-and-answer environment. Given solely the input question without access to ground-truth reference answers or annotated reasoning evidence, the agent issues sequential retrieval actions across a shared unified corpus constructed from candidate passages tied to the three benchmarks. Each retrieved observation is appended directly into the visible interaction history, after which the agent may either continue searching or submit a final generated answer. Crucially, ground-truth answers, supporting-fact annotations, and evidence trajectories remain unexposed to the agent interaction interface. Reference answers remain excluded from the executor context and ASCEND potential estimation, while informing terminal F1 feedback for outcome-supervised methods. IGPO additionally incorporates reference-answer likelihoods into intermediate supervision. Supporting-fact annotations and evidence paths remain excluded from supervision construction. We detail the comprehensive retrieval budget, passage return limits, dataset splits, and evolution-round selection protocol in Section B.4.

Together, these benchmarks encompass complementary paradigms of sequential agent interaction. APPWorld emphasizes executable actions, stateful environment, and delayed task-level verification, whereas the multi-hop reasoning centers on iterative evidence acquisition and deduction under partial information. Cross-benchmark transfer evaluates the agent evolved on 2Wiki across additional question distributions under a shared retrieval interface and corpus, without further adaptation.

B.2 Baseline Implementations

We compare ASCEND with 4 non-parametric evolution methods and Vanilla GRPO (Shao et al., 2024) on APPWorld, and 5 parametric baselines on multi-hop reasoning. All methods deploy Qwen3.5-9B (Qwen Team, 2026) as the foundation model. APPWorld results for Reflexion (Shinn et al., 2023), MemP (Fang et al., 2026), ReasoningBank (Ouyang et al., 2026), and SkillMentor (Bao et al., 2026) are adopted from Bao et al. (2026). Our multi-hop reasoning implementations share the same foundation model, retrieval environment, and evaluation protocol, with 4 candidate trajectories per query. For both APPWorld and multi-hop reasoning, Vanilla GRPO and ASCEND apply the same squared transformation to terminal outcomes before group-relative normalization, as defined in Equation 16. Method-specific supervision construction is described below.

APPWorld Baselines. The APPWorld evaluation encompasses 4 non-parametric evolution methods that integrate external textual knowledge while keeping the executor fixed, alongside Vanilla GRPO (Shao et al., 2024) as a parametric, outcome-supervised baseline.

(i) Reflexion. Reflexion is a reinforcement framework comprising an Actor, an Evaluator, and a Self-Reflection component (Shinn et al., 2023). Following an evaluated trial, the Self-Reflection component conditions on the interaction trajectory, the evaluation feedback, and the existing memory to synthesize a textual reflection that summarizes observed failure modes and suggests an improved strategy. This reflection is appended to an episodic memory buffer and provided to the Actor as supplementary context across subsequent trials. Consequently, behavioral adaptation occurs through inference-time memory accumulation instead of parameter updates.

(ii) MemP. MemP formulates procedural memory as an explicit repository governed by three core operations: memory construction, retrieval, and updating (Fang et al., 2026). The framework investigates procedural representations spanning complete interaction trajectories, higher-level script-like abstractions, and hybrid forms combining concrete experiences with abstract instructions. Given an unseen task, relevant memory entries are retrieved via vector-based matching and prepended to the agent context. As additional experience accumulates, the repository can be expanded, revised, or pruned to maintain the efficacy of the stored procedures.

(iii) ReasoningBank. ReasoningBank constructs a reusable memory bank from both successful and failed interaction trajectories without requiring ground-truth trajectory annotations (Ouyang et al., 2026). Upon trajectory completion, an LLM-based evaluator assigns an autonomous success or failure label. Successful trajectories are distilled into generalizable reasoning strategies, whereas failed trajectories are converted into counterfactual lessons, common pitfalls, and preventive guidance. Each resulting memory item incorporates a title, a brief description, and detailed procedural content. For an unseen task, relevant items are retrieved conditioned on the task query and provided to the agent as supplementary system-level context.

(iv) SkillMentor. SkillMentor decouples skill acquisition from task execution by pairing a frozen Executor with a reinforcement-learned Mentor (Bao et al., 2026). The Mentor first synthesizes diagnostic tasks designed to expose recurrent blind spots of the Executor, subsequently contrasting the behavior of a stronger reference model against the Executor to identify actionable performance gaps. Guided by these discrepancies, the Mentor generates ADD, UPDATE, or MERGE operations over an external skill repository. Candidate skills are retained strictly when they satisfy the formatting specifications and measurably boost the frozen Executor on the target diagnostic task. During inference, relevant skills are matched by description and prepended to the Executor context.

(v) Vanilla GRPO. For APPWorld, we implement Vanilla GRPO (Shao et al., 2024) conditioned solely on the terminal task outcome returned by the programmatic evaluator of the benchmark. Candidate trajectories for the identical task are evaluated via group-relative advantage estimation, with the resulting trajectory-level advantage assigned across all trainable tokens. This baseline provides an outcome-supervised reference for evaluating ASCEND within parametric evolution.

Multi-hop Reasoning Baselines. In contrast to the APPWorld comparison above, the multi-hop reasoning baselines are executed within our unified experimental pipeline under the identical agent, retrieval environment, rollout budget, and evaluation protocol. We preserve the original learning objectives whenever possible, introducing solely the task-specific adaptations required for open-ended multi-hop reasoning responses. These adaptations are detailed explicitly below.

(i) Vanilla GRPO. GRPO estimates relative advantages from a group of candidate responses sampled for the identical input, thereby eliminating the need for a separate critic (Shao et al., 2024). We implement its outcome-supervised formulation within our experimental framework. For each reasoning query, we sample 4 trajectories and evaluate each rollout via the token-level F1 score of its final answer. The resulting outcome rewards are normalized within each query group, and the same group-relative advantage is assigned to all generated action tokens included in optimization. The agent undergoes optimization via a clipped surrogate objective with reference KL regularization.

(ii) TTRL. TTRL performs reinforcement learning on unlabeled problems by sampling multiple candidate rollouts and treating the majority prediction as a pseudo-label (Zuo et al., 2026). We adapt its label-free self-consistency objective to multi-hop reasoning. For each query, four trajectories are sampled, and their extracted final answers are grouped via query-conditioned bidirectional entailment to account for semantically equivalent surface variations. The largest semantic cluster is designated as the pseudo-label cluster. Rollouts within this cluster receive a reward of 1, while all remaining rollouts, including those without a valid extracted answer, receive 0. These binary rewards are standardized within each query group and broadcast to the trainable tokens of the corresponding trajectories. Ground-truth annotated reference answers serve exclusively for offline evaluation protocols and never enter the interactive rollout prompts or training reward calculations.

(iii) EMPO. EMPO performs reinforcement learning on unlabeled inputs by minimizing predictive entropy within a latent semantic space (Zhang et al., 2026b). For each query, candidate responses are grouped into semantic clusters, with each response rewarded according to the empirical probability mass of its cluster, thereby encouraging semantically consistent predictions without requiring

annotated reference answers. The original framework filters queries exhibiting excessively low or high semantic entropy to exclude uninformative training instances. For multi-hop reasoning, we follow the released free-form natural-reasoning implementation and construct semantic clusters via query-conditioned answer-equivalence judgments. We adapt answer extraction to conform to the `<answer>...</answer>` protocol adopted by our multi-hop reasoning environment. Annotated reference answers serve solely for evaluation and do not enter clustering or reward computation.

(iv) IGPO. IGPO provides fine-grained credit assignment by measuring how each interaction turn alters the confidence of the agent in the ground-truth answer (Wang et al., 2026). At each intermediate prefix, the likelihood of the reference answer is evaluated under teacher forcing, and the discrepancy between consecutive prefixes serves as the turn-level information-gain reward. These intermediate rewards are combined with the terminal outcome reward to construct turn-level advantages for parametric optimization. In our multi-hop reasoning adaptation, we follow the released `log_prob_diff` configuration, computing differences in the mean teacher-forced log-likelihood of the reference answer. We select the first annotated reference answer as the teacher-forced target and the multi-hop reasoning F1 score as the terminal outcome. The reference answer remains accessible solely during reward computation and never enters the rollout prompt.

(v) SIOP. SIOP constructs turn-level supervision without ground-truth answers or a task-specific outcome verifier (Hu et al., 2026a). For each query, it samples multiple rollouts, groups their final answers into semantic outcome modes, and constructs a reliability-aware distribution over these modes conditioned on both empirical frequencies and supporting evidence. For each visible interaction prefix, SIOP estimates the support of the agent for the semantic mode associated with the corresponding rollout. The turn-level reward is determined by the shift in log support between neighboring prefixes, weighted by the calibrated probability of that outcome mode. This calibrated mode probability is additionally incorporated as terminal feedback. This formulation assigns credit by how individual turns alter the likelihood of reaching reliable self-induced outcomes, instead of broadcasting a trajectory reward across turns. In our multi-hop reasoning implementation, semantic outcome modes derive from extracted answers via strict bidirectional entailment. No annotated reference answer enters clustering, potential estimation, or reward computation.

B.3 Evaluation Metrics

This subsection outlines the metrics reported in Tables 1 and 2, Figure 3, and the supervision analyses in Tables 3 and 4. We distinguish benchmark performance from diagnostic alignment on trajectories for evolutionary updates. Unless specified, higher values indicate better performance or stronger alignment. Task-performance rates and step-audit rates are reported as percentages, while correlation, concordance, AUROC, Outcome Gap, and Selection Gain remain on their original scales.

B.3.1 Task Performance and Execution Efficiency

Let $\mathcal{E}_B$ denote the indices of evaluated executions on benchmark B, with $N_B = |\mathcal{E}_B|$. Each execution has an environmental outcome r_i^{out} and length L_i.

Task Success Rate (SR). On APPWorld, an execution succeeds when all evaluated task requirements are satisfied. The environment provides the fraction of passed evaluation checks as its outcome, and complete success corresponds to $r_i^{\text{out}} = 1$. Task success rate is defined as:

$$\text{SR} = \frac{100}{N_{\text{APPWorld}}} \sum_{i \in \mathcal{E}_{\text{APPWorld}}} \mathbb{I}\left[r_i^{\text{out}} = 1\right]. \tag{32}$$

Partial completion contributes to the continuous outcome but does not count as task success.

Average Execution Steps (Avg. Steps). This metric measures the mean number of recorded interaction steps across all evaluated executions:

$$\text{Avg. Steps} = \frac{1}{N_{\text{APPWorld}}} \sum_{i \in \mathcal{E}_{\text{APPWorld}}} L_i. \tag{33}$$

Both successful and unsuccessful executions contribute to the average. Lower values indicate shorter executions and are interpreted jointly with SR.

Changes from the Base Model. For the APPWorld, improvements relative to the base model are:

$$\begin{aligned} \Delta\mathrm{SR} &= \mathrm{SR} - \mathrm{SR}_{\mathrm{base}}, \\ \Delta\mathrm{Steps} &= \text{Avg. Steps} - \text{Avg. Steps}_{\mathrm{base}}. \end{aligned} \tag{34}$$

A positive $\Delta\mathrm{SR}$ indicates higher performance, and a negative $\Delta\mathrm{Steps}$ indicates fewer steps.

Exact Match (EM). For evaluation on multi-hop reasoning, let $\hat{y}_i$ denote the predicted answer and y_i the reference answer. Following the configured SIOP evaluation protocol (Hu et al., 2026a), normalization lowercases the text, replaces punctuation with spaces, and collapses repeated whitespace. Articles are retained. When multiple reference answers are supplied, evaluation considers the first reference. Let $\mathrm{norm}(\cdot)$ denote this normalization. Exact match is formulated as:

$$\mathrm{EM}_B = \frac{100}{N_B} \sum_{i \in \mathcal{E}_B} \mathbb{I}\left[\mathrm{norm}(\hat{y}_i) = \mathrm{norm}(y_i)\right]. \tag{35}$$

Answer F1. Let $\hat{W}_i$ and W_i denote the sets of distinct whitespace-separated tokens in the normalized predicted and reference answers. For nonempty sets, answer precision and recall are:

$$P_i^{\mathrm{ans}} = \frac{|\hat{W}_i \cap W_i|}{|\hat{W}_i|}, \qquad R_i^{\mathrm{ans}} = \frac{|\hat{W}_i \cap W_i|}{|W_i|}. \tag{36}$$

The per-answer score and benchmark average are:

$$F1_i = \begin{cases} 1, & |\hat{W}_i| + |W_i| = 0, \\ \dfrac{2|\hat{W}_i \cap W_i|}{|\hat{W}_i| + |W_i|}, & \text{otherwise}, \end{cases} \qquad \mathrm{F1}_B = \frac{100}{N_B} \sum_{i \in \mathcal{E}_B} F1_i. \tag{37}$$

The score is 1 when both normalized answers are empty and 0 when exactly one is empty. Repeated token occurrences do not increase their contribution.

Macro Average. The macro average assigns equal weight to the three benchmarks:

$$\mathrm{MacroAvg}(M) = \frac{M_{\mathrm{2Wiki}} + M_{\mathrm{HotpotQA}} + M_{\mathrm{MuSiQue}}}{3}, \qquad M \in \{\mathrm{EM}, \mathrm{F1}\}. \tag{38}$$

This average is independent of the relative benchmark sizes.

Evolution-Round Selection and Curves. For each parametric method on APPWorld, we report the highest SR over evolution rounds 1-8 together with Avg. Steps from the same round. For multi-hop reasoning, we select the round with the highest 2Wiki F1 over rounds 1-4 and report the corresponding agent performance across all three benchmarks:

$$t^\star_{\mathrm{APPWorld}} \in \arg\max_{1 \le t \le 8} \mathrm{SR}_t, \qquad t^\star_{\mathrm{reasoning}} \in \arg\max_{1 \le t \le 4} \mathrm{F1}_{\mathrm{2Wiki},t}. \tag{39}$$

Figure 3 traces APPWorld performance across 8 evolution rounds, with round 0 displaying the shared unevolved base-model reference from Table 1.

B.3.2 Potential-Outcome Alignment

Table 3 examines whether completion potentials align with final outcomes on APPWorld and multi-hop reasoning on 2Wiki. Eligible executions have matched environmental outcomes, complete finite potential sequences, and at least 2 interaction steps. Repeated scoring records for the same execution retain the earliest call. The first-step and pre-final analyses consider the same executions.

The recorded expected potentials $\tilde{\omega}_i^{(1)}$ and $\tilde{\omega}_i^{(L_i-1)}$ describe the prefixes after the first step and before the final step, respectively. For this analysis, these estimates are taken before environment-specific potential adjustments and the logarithmic transformation. The outcome r_i^{out} is the original environmental score before outcome transformation and normalization. It corresponds to the fraction of passed evaluation checks on APPWorld and answer F1 on 2Wiki.

Outcome Correlation. Spearman correlation measures whether larger prefix potentials correspond to higher outcomes. Let $\mathcal{I}_{\text{align},t}$ denote the eligible execution indices in evolution round t. Define:

$$\begin{aligned}\rho_{1,t} &= \mathrm{Corr}_{i\in\mathcal{I}_{\text{align},t}}\left(\mathrm{rank}\left(\tilde{\omega}_i^{(1)}\right), \mathrm{rank}\left(r_i^{\text{out}}\right)\right),\\ \rho_{\text{pre},t} &= \mathrm{Corr}_{i\in\mathcal{I}_{\text{align},t}}\left(\mathrm{rank}\left(\tilde{\omega}_i^{(L_i-1)}\right), \mathrm{rank}\left(r_i^{\text{out}}\right)\right).\end{aligned} \tag{40}$$

Ranks are computed within each round, with ties assigned their average rank. Here, Corr is Pearson correlation of the rank sequences, computed as their centered cross-product sum divided by the product of their centered Euclidean norms. It is undefined when either sequence has zero variance.

The reported ρ_1 and ρ_{pre} average the corresponding correlations equally. Their difference is:

$$\Delta\rho = \rho_{\text{pre}} - \rho_1. \tag{41}$$

A positive $\Delta\rho$ means that potentials correlate more strongly with final outcomes near the end of an execution trajectory than after its first action. This indicates that the additional interaction evidence makes completion estimates more informative about eventual task outcomes.

Pre-Final Outcome AUROC. This measures whether pre-final potentials rank executions attaining the maximum environmental score above executions attaining lower scores. Within round t, define:

$$\begin{aligned}\mathcal{I}_{\text{full},t} &= \left\{i \in \mathcal{I}_{\text{align},t} : r_i^{\text{out}} = 1\right\},\\ \mathcal{I}_{\text{partial},t} &= \left\{i \in \mathcal{I}_{\text{align},t} : r_i^{\text{out}} < 1\right\}.\end{aligned} \tag{42}$$

The second set includes both partially completed and zero-score executions. AUROC is:

$$\mathrm{AUROC}_{\text{pre},t} = \frac{\displaystyle\sum_{i\in\mathcal{I}_{\text{full},t}}\sum_{j\in\mathcal{I}_{\text{partial},t}}\left(\mathbb{I}\left[\tilde{\omega}_i^{(L_i-1)} > \tilde{\omega}_j^{(L_j-1)}\right] + \frac{1}{2}\mathbb{I}\left[\tilde{\omega}_i^{(L_i-1)} = \tilde{\omega}_j^{(L_j-1)}\right]\right)}{|\mathcal{I}_{\text{full},t}|\,|\mathcal{I}_{\text{partial},t}|}. \tag{43}$$

Each correctly ordered pair receives 1, a potential tie receives 0.5, and an incorrectly ordered pair receives 0. AUROC equals 0.5 for chance-level ranking and 1 for perfect separation. For multi-hop reasoning on 2Wiki, the positive class corresponds to F1 equal to 1, which need not coincide with exact match. AUROC is undefined when either comparison class is absent.

Within-Task Concordance. Within-task concordance evaluates whether potentials correctly order candidate executions of the same task. Following the candidate-group notation in Appendix A.4, each $\mathcal{Q}$ contains executions of one task generated in the same collection batch. Define:

$$\mathcal{I}_{\text{pair}}(\mathcal{Q}) = \left\{(i,j) : i,j \in \mathcal{Q},\ r_i^{\text{out}} > r_j^{\text{out}}\right\}. \tag{44}$$

Each unequal-outcome pair appears once, with the higher-outcome execution first. Group concordance is mathematically formulated as follows:

$$C(\mathcal{Q}) = \frac{1}{|\mathcal{I}_{\text{pair}}(\mathcal{Q})|}\sum_{(i,j)\in\mathcal{I}_{\text{pair}}(\mathcal{Q})}\left(\mathbb{I}\left[\tilde{\omega}_i^{(L_i-1)} > \tilde{\omega}_j^{(L_j-1)}\right] + \frac{1}{2}\mathbb{I}\left[\tilde{\omega}_i^{(L_i-1)} = \tilde{\omega}_j^{(L_j-1)}\right]\right). \tag{45}$$

Outcome ties are excluded, while potential ties receive 0.5. Groups without unequal-outcome pairs do not contribute. Let Avg denote an unweighted arithmetic mean over the indicated eligible elements. The round-level result is mathematically formulated as:

$$C_{\text{task},t} = \mathrm{Avg}_{\tau}\left[\mathrm{Avg}_{\mathcal{Q}\text{ for }\tau\text{ in round }t}\, C(\mathcal{Q})\right]. \tag{46}$$

Cross-Round Aggregation. The reported correlations, pre-final AUROC, and within-task concordance are mathematically formulated as follows:

$$\begin{aligned}\rho_1 &= \mathrm{Avg}_t\,\rho_{1,t}, & \rho_{\text{pre}} &= \mathrm{Avg}_t\,\rho_{\text{pre},t},\\ \mathrm{AUROC}_{\text{pre}} &= \mathrm{Avg}_t\,\mathrm{AUROC}_{\text{pre},t}, & C_{\text{task}} &= \mathrm{Avg}_t\,C_{\text{task},t}.\end{aligned} \tag{47}$$

Each mean includes available evolution rounds with a defined metric value. This aggregation assigns equal weight to rounds regardless of their trajectory counts.

Outcome Gap. Outcome Gap measures the difference in average outcomes between high-potential and low-potential executions. Let $\mathcal{I}_{\text{align}} = \bigcup_t \mathcal{I}_{\text{align},t}$ contain eligible executions pooled across available rounds. The empirical potential quartiles are:

$$\tilde{\omega}_{\text{quartile},p} = \text{Quantile}_p\left(\left\{\tilde{\omega}_i^{(L_i-1)} : i \in \mathcal{I}_{\text{align}}\right\}\right), \qquad p \in \{0.25, 0.75\}. \tag{48}$$

where quantiles are computed by linear interpolation between adjacent ordered values. Define:

$$\begin{aligned} \mathcal{I}_{\text{low}} &= \left\{i \in \mathcal{I}_{\text{align}} : \tilde{\omega}_i^{(L_i-1)} \le \tilde{\omega}_{\text{quartile},0.25}\right\}, \\ \mathcal{I}_{\text{high}} &= \left\{i \in \mathcal{I}_{\text{align}} : \tilde{\omega}_i^{(L_i-1)} \ge \tilde{\omega}_{\text{quartile},0.75}\right\}. \end{aligned} \tag{49}$$

The Outcome Gap metric is mathematically formulated as:

$$\text{Outcome Gap} = \frac{1}{|\mathcal{I}_{\text{high}}|}\sum_{i\in\mathcal{I}_{\text{high}}} r_i^{\text{out}} - \frac{1}{|\mathcal{I}_{\text{low}}|}\sum_{i\in\mathcal{I}_{\text{low}}} r_i^{\text{out}}. \tag{50}$$

Boundary ties are retained, so each group need not contain exactly 25% of the executions. We report the metric as undefined when the quartile cutoffs coincide. A positive Outcome Gap indicates stronger outcomes among high-potential executions.

Selection Gain. Selection Gain measures the outcome improvement from selecting a highest-potential candidate over uniform random selection within the same candidate group. For each group $\mathcal{Q}$ containing at least 2 trajectories, we define:

$$\mathcal{I}_{\text{top}}(\mathcal{Q}) = \arg\max_{i\in\mathcal{Q}} \tilde{\omega}_i^{(L_i-1)}. \tag{51}$$

The group-level gain is mathematically formulated as:

$$\text{Gain}(\mathcal{Q}) = \frac{1}{|\mathcal{I}_{\text{top}}(\mathcal{Q})|}\sum_{i\in\mathcal{I}_{\text{top}}(\mathcal{Q})} r_i^{\text{out}} - \frac{1}{|\mathcal{Q}|}\sum_{i\in\mathcal{Q}} r_i^{\text{out}}. \tag{52}$$

The first term averages outcomes across tied highest-potential candidates. The second is the exact expected outcome of uniform random selection.

We first average eligible groups across rounds within each task, then average the resulting task-level values, as mathematically formulated as follows:

$$\text{Selection Gain} = \text{Avg}_\tau\left[\text{Avg}_{\mathcal{Q}\text{ for }\tau}\,\text{Gain}(\mathcal{Q})\right]. \tag{53}$$

Tasks without eligible groups are excluded. Groups with identical outcomes remain included and contribute zero gain. This metric characterizes selection quality among recorded executions. Both Outcome Gap and Selection Gain are reported on the original environmental-outcome scale.

B.3.3 Step-Level Supervision Alignment

Table 4 evaluates applied potential-difference signals against human judgments of action progress. For each benchmark, we randomly sample 100 distinct eligible trajectories while balancing tasks and evolution rounds, then draw one non-final step uniformly from each selected trajectory. Selection does not depend on outcomes or signal magnitudes. This design balances tasks and rounds without forming a uniform sample of all recorded steps.

Each judgment considers the task, preceding interaction history, current action, and resulting observation. Potential estimates, step signals, terminal outcomes, and future interactions are excluded from the annotation record. The annotation categories are progress, no change, regression, and uncertain. Regression denotes an action that reverses established task progress. Notably, an unsuccessful action that leaves task progress unchanged falls into no change. Sampling balances tasks and evolution rounds without targeting particular action categories, so coverage of every category is not assured. Neither human nor model judgments assign regression within this sample. The reported metrics therefore characterize progress discrimination and responses to no-change actions, with uncertain judgments excluded from their denominators.

The evaluated quantity is the recorded applied log-space step signal $\delta_i^{(\ell)}$, including applicable environment-specific potential adjustments. It precedes group normalization and suffix integration and is distinct from the final token advantage. Signal polarity is assessed with $\epsilon_{\text{audit}} = 0.02$.

Let $\mathcal{I}_{\text{sample}}$ denote sampled execution-step pairs and $\mathcal{I}_{\text{valid}}$ denote their assessable subset. Within this subset, $\mathcal{I}_{\text{progress}}$ and $\mathcal{I}_{\text{nochange}}$ contain pairs labeled as progress and no change, respectively. For the reported human audit, $\mathcal{I}_{\text{valid}}$ is formulated as:

$$\mathcal{I}_{\text{valid}} = \mathcal{I}_{\text{progress}} \cup \mathcal{I}_{\text{nochange}}. \tag{54}$$

Subsequently, we define the positive-signal subset:

$$\mathcal{I}_{\text{positive}} = \left\{ (i,\ell) \in \mathcal{I}_{\text{valid}} : \delta_i^{(\ell)} > \epsilon_{\text{audit}} \right\}. \tag{55}$$

Valid / Sampled. The two counts are $|\mathcal{I}_{\text{valid}}|$ and $|\mathcal{I}_{\text{sample}}|$, distinguishing assessable judgments.

Progress Precision. Precision measures the fraction of positive signals corresponding to actions judged to advance task progress, formulated as:

$$\text{Precision} = 100\, \frac{|\mathcal{I}_{\text{positive}} \cap \mathcal{I}_{\text{progress}}|}{|\mathcal{I}_{\text{positive}}|}. \tag{56}$$

Higher precision indicates more reliable positive supervision.

Progress Recall. Recall measures the fraction of audited progress actions receiving a positive signal:

$$\text{Recall} = 100\, \frac{|\mathcal{I}_{\text{positive}} \cap \mathcal{I}_{\text{progress}}|}{|\mathcal{I}_{\text{progress}}|}. \tag{57}$$

Precision and recall assess positive-signal reliability and progress coverage, respectively.

No-Change Non-Reward Rate. This metric measures the fraction of no-change actions whose signals do not exceed the positive threshold:

$$\text{NonReward} = \frac{100}{|\mathcal{I}_{\text{nochange}}|} \sum_{(i,\ell) \in \mathcal{I}_{\text{nochange}}} \mathbb{I}\left[\delta_i^{(\ell)} \le \epsilon_{\text{audit}} \right]. \tag{58}$$

Both neutral and negative signals satisfy this criterion.

No-Change Neutral Rate. Neutral alignment measures the fraction of no-change actions receiving a near-zero signal, mathematically formulated as:

$$\text{Neutral} = \frac{100}{|\mathcal{I}_{\text{nochange}}|} \sum_{(i,\ell) \in \mathcal{I}_{\text{nochange}}} \mathbb{I}\left[|\delta_i^{(\ell)}| \le \epsilon_{\text{audit}} \right]. \tag{59}$$

This criterion is stricter than Non-Reward Rate. Their difference equals the percentage of no-change actions receiving signals below the threshold $-\epsilon_{\text{audit}}$.

Progress AUROC. Progress AUROC measures whether audited progress actions receive larger step signals than no-change actions, formulated as:

$$\text{AUROC}_{\text{progress}} = \frac{\displaystyle\sum_{(i,\ell) \in \mathcal{I}_{\text{progress}}} \sum_{(j,m) \in \mathcal{I}_{\text{nochange}}} \left(\mathbb{I}\left[\delta_i^{(\ell)} > \delta_j^{(m)} \right] + \frac{1}{2} \mathbb{I}\left[\delta_i^{(\ell)} = \delta_j^{(m)} \right] \right)}{|\mathcal{I}_{\text{progress}}|\, |\mathcal{I}_{\text{nochange}}|}. \tag{60}$$

This metric evaluates ranking across thresholds and does not depend on ϵ_{audit}.

Step-audit rates are percentages, and AUROC lies in $[0, 1]$. Metrics aggregate assessable steps directly and are undefined for zero denominators or missing AUROC classes.

B.4 Evolution and Evaluation Settings

Shared Evolution Protocol. All agents initialize from Qwen3.5-9B (Qwen Team, 2026). Within each evolution round, the current agent collects task-grouped trajectories, constructs method-specific

Table 11: Evolution, optimization, and evaluation settings for ASCEND on APPWorld and multi-hop reasoning, including rollout and decoding configurations.

Setting	APPWorld	Multi-Hop Reasoning
Evolution and Rollout Configuration		
Foundation model	Qwen3.5-9B	Qwen3.5-9B
Evolution data	APPWorld train	2WikiMultiHopQA train
Evolution rounds	8	4
Maximum sampled tasks per round	90	800
Query batch size	32	128
Candidate trajectories per task	8	4
Maximum interaction steps	30	5
Runtime maximum model length	262,144	8,192
Executor input-token limit	65,536	6,144
Generation-token limit	4,096	2,048
Rollout decoding temperature	0.9	1.0
Parametric Update Configuration		
Epochs per evolution round	2	1
PPO epochs per collected batch	1	1
Learning rate	2×10^{-6}	1×10^{-6}
Weight decay	0.01	0.01
PPO minibatch size	8	32
Microbatch size per GPU	1	1
PPO clipping margins	$0.20/0.28$	$0.20/0.20$
Reference KL coefficient	0.001	0.005
Gradient-clipping threshold	1.0	1.0
Entropy coefficient	0	0
Supervision Construction		
Step-supervision weight α	0.4	0.2
Potential-estimation decoding temperature	0	0
Bin-scoring temperature κ	0.5	0.5
Choice-probability floor p_{floor}	0.05	0.05
Missing-choice log-probability	-20.0	-20.0
Maximum serialized field length	3,000 characters	3,000 characters
Squared terminal outcome	Yes	Yes
Evaluation Protocol		
Evaluation data	APPWorld test_normal (168 tasks)	2WikiMultiHopQA dev HotpotQA dev MuSiQue validation
Evaluation batch size	32	32
Evaluation frequency	After each evolution round	After each evolution round
Round-selection metric	APPWorld success rate	2WikiMultiHopQA F1
Candidate evolution rounds	Rounds 1-8	Rounds 1-4
Evaluation decoding temperature	0.0	0.0

supervision, and undergoes parametric updates. Rollout collection precedes optimization within each epoch. The reference executor remains fixed to the agent at the start of the round.

Table 11 summarizes the ASCEND configurations for APPWorld and multi-hop reasoning. Baseline implementations and adaptations are detailed in Appendix B.2. APPWorld results for Reflexion, MemP, ReasoningBank, and SkillMentor are adopted from Bao et al. (2026).

APPWorld Evolution and Evaluation. We perform 8 evolution rounds on the APPWorld training split, with at most 90 tasks per round and 8 candidate trajectories per task. Executions contain at most 30 interaction steps, including terminal submission. Each round comprises 2 epochs with a query batch size of 32, a learning rate of 2×10^{-6}, and 1 PPO optimization epoch per collected batch. After each round, the agent is evaluated on the 168 tasks in the `test_normal` split. We report the highest SR within 8 rounds together with Avg. Steps from the same round.

Multi-Hop Reasoning Evolution. The agent evolves on the 2WikiMultiHopQA training split over 4 rounds, with at most 800 questions per round and 4 candidate trajectories per question. Each

execution contains at most 5 interaction steps, including final-answer submission. Each round comprises 1 epoch with a query batch size of 128, a learning rate of 10^{-6}, and 1 PPO optimization epoch per collected batch. The supervision weight is $\alpha = 0.2$.

All methods interact with a shared retrieval environment comprising an E5-base-v2 encoder and a FAISS index over passages from the three benchmarks. Each retrieval returns at most 3 passages, with limits of 4,000 characters per passage and 12,000 characters per observation. Reference answers, supporting-fact annotations, and evidence paths remain excluded from the interaction context.

Potential-Estimation Configuration. The evolving agent scores visible prefixes through the templates in Appendix E.1. Each action and observation field is truncated to 3,000 characters before prompt assembly. Bin scoring adopts deterministic decoding, with temperature normalization and probability refinement specified in Appendix A.2. Future interactions after the prefix and terminal outcome scores remain excluded from potential estimation.

Evaluation and Evolution-Round Selection. After each evolution round, we evaluate the agent on the development splits of 2WikiMultiHopQA and HotpotQA and the MuSiQue validation split. Reported results correspond to the round attaining the highest 2WikiMultiHopQA F1 within 4 rounds, with all three benchmarks evaluated for the same agent and without further adaptation. Detailed metric definitions are provided in Appendix B.3.

C Additional Results and Analysis

We further examine the behavior and quality of ASCEND supervision through complementary experiments and analyses. We investigate sensitivity to the supervision weight in Appendix C.1, compare step-level supervision alignment under human and model audits in Appendix C.2, and examine potential-outcome alignment across evolution rounds in Appendix C.3. Finally, Appendix C.4 illustrates how prefix potentials and their differences reflect concrete interaction evidence.

C.1 Sensitivity to the Supervision Weight

To examine how the contribution of prefix-potential supervision affects agent evolution, we analyze performance across supervision weights and evolution rounds. We evaluate $\alpha \in \{0.05, 0.1, 0.2, 0.4\}$ over 4 rounds on 2Wiki, with HotpotQA and MuSiQue assessing transfer without further adaptation. Vanilla GRPO provides the reference. Table 12 reports EM, F1, and average execution steps at every round. For each configuration, the highlighted row maximizes 2Wiki F1 and supplies the corresponding results across all three benchmarks in Figure 4. The key findings are as follows:

(i) Fine-grained supervision generally improves agent performance beyond terminal feedback. At the selected rounds, ASCEND exceeds GRPO in F1 for 9 of the 12 weight-benchmark combinations, including all 4 weights on MuSiQue. With $\alpha = 0.2$, the gains reach 1.27, 1.57, and 1.96 F1 points on 2Wiki, HotpotQA, and MuSiQue, respectively. These results support the value of prefix-potential supervision for both task performance and cross-benchmark transfer.

(ii) Balancing fine-grained and terminal supervision is important for effective evolution. The strongest early performance does not necessarily translate into the best final results. Although $\alpha = 0.1$ initially outperforms $\alpha = 0.2$ on 2Wiki, the latter achieves the highest selected-round EM and F1 across all three benchmarks. Increasing α further to 0.4 reduces F1 on each benchmark. Thus, the contribution of fine-grained supervision depends on its weight, with a moderate setting providing the strongest overall results in this evaluation.

Overall, prefix-potential supervision provides benefits beyond outcome-only evolution across a range of weights, while an appropriate balance with terminal feedback strengthens these benefits. The results support $\alpha = 0.2$ as the default setting for the evaluated multi-hop reasoning tasks.

C.2 Sensitivity to Audit Source

To examine whether the conclusion on step-level supervision depends on the audit source, we supplement human judgments with an independent GPT-6 Astra (OpenAI, 2026) audit of the same 100 sampled steps per benchmark. The model assesses action progress from the task query, visible interaction history, and current action-observation pair, without access to the recorded potential values

Table 12: Complete multi-hop reasoning results across supervision weights and evolution rounds. Blue rows identify the round with the highest 2Wiki F1 for each configuration and its corresponding results on all benchmarks. Setting $\alpha = 0.2$ achieves the highest EM and F1 across benchmarks.

Round	2Wiki			HotpotQA			MuSiQue		
	EM ↑	F1 ↑	Avg. Steps ↓	EM ↑	F1 ↑	Avg. Steps ↓	EM ↑	F1 ↑	Avg. Steps ↓
Vanilla GRPO									
1	45.55	59.69	3.61	39.04	56.84	2.97	5.21	14.24	3.56
2	54.02	65.96	3.60	42.83	60.38	2.96	6.66	16.13	3.62
3	59.66	70.06	3.49	46.68	62.75	2.91	5.96	15.40	3.51
4	62.21	71.61	3.38	49.38	64.16	2.81	6.08	14.61	3.30
ASCEND, $\alpha = 0.05$									
1	48.79	61.98	3.61	40.98	58.36	2.96	5.75	14.95	3.60
2	55.19	66.64	3.61	43.64	60.68	2.97	6.70	15.90	3.63
3	61.38	71.15	3.61	47.87	63.61	3.00	6.82	16.50	3.66
4	61.47	70.49	3.39	49.10	63.74	2.81	5.62	13.23	3.19
ASCEND, $\alpha = 0.1$									
1	49.07	62.07	3.60	41.45	58.51	2.97	5.30	14.63	3.56
2	54.69	66.23	3.62	43.45	60.49	2.98	6.29	15.64	3.63
3	61.50	71.49	3.60	47.26	63.56	3.00	6.78	16.36	3.66
4	62.20	71.55	3.62	49.72	64.40	3.12	6.95	15.61	3.63
ASCEND, $\alpha = 0.2$									
1	45.45	59.59	3.62	39.21	56.84	3.00	5.50	14.49	3.64
2	54.92	66.43	3.62	43.18	60.45	3.00	6.45	15.75	3.67
3	61.85	72.12	3.59	47.49	64.04	2.98	6.78	16.45	3.66
4	63.49	72.88	3.60	50.33	65.73	3.05	7.07	16.57	3.74
ASCEND, $\alpha = 0.4$									
1	48.11	61.45	3.61	40.49	57.98	2.97	5.50	14.69	3.62
2	54.93	66.51	3.62	43.21	60.52	2.97	6.29	15.72	3.60
3	60.51	70.32	3.60	47.35	63.12	2.99	6.25	15.48	3.59
4	63.35	72.42	3.57	50.22	64.90	3.02	6.33	14.99	3.55

Table 13: Step-level alignment under human and GPT-6 Astra audits. Valid / Sampled reports assessable coverage for each source. Agreement and alignment metrics concern the shared assessable subsets of 91 APPWorld steps and 100 2Wiki steps. Agreement measures matching audit labels. For no-change actions, Non-Reward measures $\delta_i^{(\ell)} \leq \epsilon_{\text{audit}}$, and Neutral measures $|\delta_i^{(\ell)}| \leq \epsilon_{\text{audit}}$, with $\epsilon_{\text{audit}} = 0.02$. Rates are percentages, and AUROC measures progress discrimination.

Dataset	Audit Source	Valid / Sampled	Agreement	Progress		No Change		AUROC↑
				Precision↑	Recall↑	Non-Reward↑	Neutral↑	
APPWorld	Human	99 / 100	81.3	91.1	56.2	77.8	5.6	0.806
	GPT-6 Astra	92 / 100		86.7	55.7	71.4	23.8	0.664
2Wiki	Human	100 / 100	92.0	100.0	79.1	100.0	100.0	0.797
	GPT-6 Astra	100 / 100		97.1	84.6	90.9	72.7	0.858

or human judgments. On APPWorld, human and model audits identify 99 and 92 assessable steps, respectively. Their uncertain judgments concern different samples, leaving 91 mutually assessable steps. All 100 steps on 2Wiki are assessable under both sources. Table 13 compares alignment on these shared subsets, while Table 4 retains all human-assessable samples. Within each shared subset, Precision is conditioned on positive signals, Recall on progress actions, and Non-Reward and Neutral on no-change actions according to the corresponding audit source. The key findings are:

Table 14: Round-wise alignment between prefix potentials and final outcomes. Shaded Overall rows reproduce Table 3. Pre-final correlation exceeds first-step correlation in every round, alongside positive Outcome Gap and Selection Gain, supporting consistent alignment across evolution rounds.

Benchmark	Round	Outcome Correlation			Outcome Ranking		Separation & Selection	
		$\rho_1 \uparrow$	$\rho_{\text{pre}} \uparrow$	$\Delta\rho \uparrow$	$\text{AUROC}_{\text{pre}} \uparrow$	$C_{\text{task}} \uparrow$	Outcome Gap ↑	Selection Gain ↑
APPWorld	1	0.062	0.509	+0.447	0.679	0.723	0.466	+0.130
	2	0.091	0.480	+0.389	0.684	0.709	0.457	+0.104
	3	0.007	0.494	+0.487	0.704	0.685	0.469	+0.110
	4	0.070	0.455	+0.385	0.695	0.733	0.414	+0.094
	5	0.091	0.429	+0.338	0.686	0.747	0.377	+0.101
	6	0.059	0.406	+0.348	0.678	0.745	0.334	+0.095
	7	0.029	0.393	+0.364	0.673	0.713	0.332	+0.067
	8	0.006	0.421	+0.416	0.675	0.744	0.366	+0.091
APPWorld	Overall	0.052	0.448	+0.397	0.684	0.725	0.411	+0.100
2Wiki	1	0.114	0.422	+0.307	0.678	0.646	0.406	+0.070
	2	0.030	0.353	+0.323	0.634	0.646	0.369	+0.081
	3	-0.046	0.282	+0.329	0.608	0.612	0.288	+0.056
	4	-0.055	0.305	+0.361	0.637	0.671	0.358	+0.077
2Wiki	Overall	0.011	0.341	+0.330	0.639	0.644	0.365	+0.071

(i) Both audit sources associate positive potential differences with task progress. Human and model judgments agree on 81.3% of mutually assessable APPWorld steps and 92.0% of 2Wiki steps. Despite these differences, positive-signal precision reaches 91.1% and 86.7% under human and model judgments on APPWorld, respectively, and 100.0% and 97.1% on 2Wiki. The association between positive signals and task progress therefore holds across both audit sources.

(ii) Progress discrimination remains informative across audit sources. Human and model audits yield AUROC values of 0.806 and 0.664 on APPWorld, respectively, and 0.797 and 0.858 on 2Wiki. All values exceed the chance level of 0.5, indicating that potential differences distinguish progress actions under either source across both benchmarks. Differences in AUROC and no-change alignment reflect variation in the estimated strength of this relationship.

Overall, the two audit sources provide complementary evidence that neighboring-prefix differences capture meaningful step-level progress. The qualitative conclusion is consistent across sources, while human judgments remain the primary basis for the main-text analysis.

C.3 Additional Potential-Outcome Alignment Results

To examine whether the aggregate alignment in Table 3 holds across evolution rounds, we report round-wise diagnostics on the same eligible interaction trajectories. The analysis covers 8 rounds on APPWorld and 4 rounds on 2Wiki. Table 14 reports diagnostics for trajectories collected before each update, with rounds numbered from 1. Metrics are computed within each round, while Overall reproduces the main-text results under the cross-round aggregation in Appendix B.3.

From these results, we draw the following findings:

(i) Additional interaction evidence strengthens outcome alignment. Pre-final correlation exceeds first-step correlation in every recorded round. The increase ranges from 0.338 to 0.487 on APPWorld and from 0.307 to 0.361 on 2Wiki. Potentials near the end of a trajectory align better with final outcomes than those after the first step. This pattern recurs across all evaluated evolution rounds.

(ii) Potential-based ranking identifies stronger outcomes. Pre-final AUROC and within-task concordance exceed 0.5 in every round on both benchmarks. Outcome Gap ranges from 0.332 to 0.469 on APPWorld and from 0.288 to 0.406 on 2Wiki, showing better outcomes among high-potential executions. Selection Gain remains positive throughout, ranging from 0.067 to 0.130 and from 0.056 to 0.081, respectively. These results support beneficial selection among candidates across rounds.

Table 15: APPWorld case study from the second evolution round, with $r_i^{\text{out}} = 1$. The complete execution traces the identification and repayment of an accidental payment request. Blue cells indicate positive $\delta_i^{(\ell)}$. The final zero signal follows the submission-inheritance rule.

ℓ	Action $a_i^{(\ell)}$	Observation $o_i^{(\ell)}$	$\tilde{\omega}_i^{(\ell)}$	$\delta_i^{(\ell)}$
0	**Task Query** q_i, with $H_i^{(0)} = (q_i)$: Return the money from the last accidental payment request sent to Brandon and approved by him.		0.361	—
1	Inspect Venmo APIs	Available operations include payment requests and transactions.	0.486	+0.299
2	Retrieve credentials	The supervisor API returns account credentials.	0.490	+0.006
3	Inspect login API	Documentation specifies the required login parameters.	0.524	+0.068
4	Log into Venmo	The login API returns an access token.	0.573	+0.090
5	Inspect request API	Documentation specifies retrieval of sent payment requests.	0.574	+0.002
6	Retrieve sent requests	The records identify an approved $56 request to Brandon.	0.695	+0.191
7	Inspect transaction API	Documentation specifies the recipient and amount parameters.	0.703	+0.012
8	Return $56 to Brandon	The API returns "Sent money." and a transaction ID.	0.782	+0.106
9	Submit completion	The completion call executes successfully.	0.782	0.000

Table 16: 2Wiki case study from the third evolution round. The task asks for the birth date of the father of Charles Dennis Fisher. Unrelated retrieval lowers $\tilde{\omega}_i^{(\ell)}$, while identifying the father and retrieving his birth date produce positive $\delta_i^{(\ell)}$. Blue and orange cells indicate positive and negative differences, respectively.

ℓ	Action $a_i^{(\ell)}$	Observation $o_i^{(\ell)}$	$\tilde{\omega}_i^{(\ell)}$	$\delta_i^{(\ell)}$
0	**Task Query** q_i, with $H_i^{(0)} = (q_i)$: When is the birth date of the father of Charles Dennis Fisher?		0.550	—
1	Search for Charles Dennis Fisher	Retrieved documents concern other people with similar names.	0.323	−0.532
2	Refine the query to identify the father	A document identifies Herbert William Fisher as the father.	0.568	+0.564
3	Search for the birth date	A document gives the birth date as 30 July 1826.	0.780	+0.318
4	Submit 30 July 1826	The environment acknowledges answer submission.	0.900	+0.143

Overall, the round-wise results show that informative outcome alignment is not confined to a particular stage of evolution. Prefix potentials become more outcome-aligned as interaction evidence accumulates and support trajectory ranking, outcome separation, and candidate selection.

C.4 Case Studies of Prefix-Potential Supervision

While the preceding analyses quantify supervision alignment across executions, we further examine individual steps to connect the constructed signals with concrete interaction evidence. We present complete trajectories from APPWorld and 2Wiki, illustrating how prefix potentials respond to task execution and newly acquired evidence. Tables 15 and 16 present each action $a_i^{(\ell)}$, observation $o_i^{(\ell)}$, and the corresponding expected potential $\tilde{\omega}_i^{(\ell)}$ and signal $\delta_i^{(\ell)}$. The initial row specifies $H_i^{(0)} = (q_i)$, and both executions attain $r_i^{\text{out}} = 1$. For readability, action and observation descriptions are condensed from the original records. Each $\delta_i^{(\ell)}$ is computed from adjacent log-space potentials $\omega_i^{(\ell)}$ before normalization and integration. The findings are as follows:

(i) Potential differences provide positive signals for task progress in stateful interaction. In Table 15, identifying the approved payment request and completing the repayment yield $\delta_i^{(\ell)} = 0.191$ and 0.106, respectively. These positive differences accompany the acquisition of task-relevant information and the execution of the requested transaction. The submission-only declaration inherits the preceding potential, giving $\delta_i^{(L_i)} = 0$, as specified in Appendix A.3.

(ii) Potential differences distinguish evidence acquisition from uninformative retrieval. In Table 16, the first search returns unrelated people and decreases estimated completion prospects. Refining the query identifies the father, and subsequent retrieval establishes his birth date, producing positive $\delta_i^{(\ell)}$ for both reasoning hops. Although the execution attains $r_i^{\text{out}} = 1$, intermediate actions receive distinct signals before outcome feedback. Unlike the submission-only declaration in APPWorld, the final answer is assessed for consistency with visible evidence through potential estimation.

Overall, these cases illustrate how prefix potentials respond to interaction evidence. Positive differences accompany target identification, task execution, and evidence acquisition, providing step-level signals that complement trajectory-level outcome feedback.

D Discussion of Supervision Quality and Practical Trade-offs

The preceding results support the effectiveness of ASCEND and the alignment of its constructed supervision with task progress. Several questions remain relevant to interpreting these findings, including whether self-assessment amplifies agent biases, whether performance gains reflect differences in feedback availability, and how estimation noise and computational cost affect the approach. We discuss these questions below, together with the role of observable interaction evidence.

Scope within Agent Evolution. Agent evolution emphasizes continuing adaptation through successive interactions, with an evolutionary strategy organizing this process. Online learning characterizes sequential learning from arriving data, while continual learning emphasizes acquiring and retaining knowledge across changing tasks or distributions. Agent evolution places these concerns within the broader organization of agent adaptation. ASCEND focuses on supervision construction once an evolutionary update is initiated. At each round, the current agent assesses completion prospects from newly collected interaction evidence and derives supervision signals for subsequent adaptation. GRPO implements the resulting parametric updates. The methodological contribution concerns the source and construction of evolutionary supervision, with reinforcement learning providing its parametric optimization mechanism. Adaptation timing, component selection, and capability retention remain aspects of the surrounding evolutionary strategy.

Scope of Cross-Benchmark Transfer. In the experiments of multi-hop reasoning, transfer beyond the evolution environment concerns benchmark-specific question distributions under shared retrieval conditions. The benchmarks draw on related encyclopedic sources and share a retrieval interface and corpus. The observed improvements support the applicability of evolved capabilities across these question distributions without further adaptation. Extending evaluation to distinct knowledge sources and interaction interfaces would further clarify the scope of transfer.

Self-Assessment and Potential Quality. The evolving agent performs both execution and potential estimation, creating a risk that correlated errors reinforce behaviors favored by its own assessments. ASCEND grounds estimation in visible interaction evidence and separates agent-generated claims from environmental observations. Round-wise analyses in Appendix C.3 provide evidence of consistent informativeness across evolution. Pre-final outcome correlation exceeds first-step correlation in every evaluated round, accompanied by positive Outcome Gaps and Selection Gains. These results indicate that potentials retain meaningful associations with environmental outcomes across successive rounds, complemented by independent human audits of step-level alignment. On APPWorld, ASCEND reaches 50.00% SR at round 4 and retains 47.61% at round 8, remaining above Vanilla GRPO throughout. The post-peak fluctuations therefore coexist with persistent potential-outcome alignment and comparative performance benefits. Continued evaluation on fixed tasks, supported by periodic independent audits, could further reveal assessment drift over longer horizons and inform the refinement of assessment criteria and the selection of stopping points for evolution.

Potential-Outcome Alignment and Supervision Value. The potential-outcome alignment raises the question of whether the estimates are sufficiently informative for evolutionary supervision. Spearman correlation and pre-final AUROC summarize ordering across executions from different tasks, whereas ASCEND normalizes supervision within task groups. Within-task concordance reaches 0.725 on APPWorld and 0.644 on 2Wiki, and positive Selection Gains indicate that potential-based selection improves average outcomes over random selection among recorded candidates. These results connect alignment to a selection benefit. The neighboring differences entering supervision are assessed through human audits, while evolution experiments support their value.

Feedback Availability and Comparative Evidence. Methods with self-induced outcomes operate under different information conditions from methods receiving environmental feedback. Their relative performance consequently reflects both supervision construction and feedback availability. The comparison with Vanilla GRPO provides more direct evidence for the contribution of ASCEND. ASCEND and Vanilla GRPO receive the same environmental outcome feedback. ASCEND additionally derives fine-grained signals from the evolving agent and available interaction evidence. Improvements over GRPO, together with the supervision-weight analysis, support the value of this construction beyond access to terminal outcome scores.

Signal Reliability under Estimation Noise. The audits assess whether potential differences align with action progress, and the case studies connect signals to observations. The audit evidence covers progress and no-change actions, while evaluating regression detection requires examples of verified setbacks. Negative potential differences reflect reduced estimated completion prospects, which may follow newly revealed information or an actual reversal of task progress. Together with benchmark improvements, this evidence supports valuable step-level supervision without requiring each difference to represent an exact progress reward. Appendix A.6 characterizes conditions under which informative signals remain stable under estimation noise. When each log-potential estimate has error at most $\varepsilon_{\mathrm{est}}$, a reference difference exceeding $2\varepsilon_{\mathrm{est}}$ in magnitude retains its sign. Accumulated differences also depend on endpoint errors, so the corresponding error bound does not increase with the number of intermediate prefixes. These properties characterize stability before clipping and normalization and suggest that informative progress signals can remain reliable under bounded estimation errors. Their downstream value is supported separately by the empirical results.

Computational Cost of Potential Estimation. Potential estimation introduces inference cost to supervision construction. An execution of length L_i involves up to $L_i + 1$ prefix assessments. Although bin prediction requires only a compact output, independently processing each prefix repeatedly encodes much of the same interaction history. This overlap creates substantial opportunities for computational reuse. With a cache-compatible prompt layout, shared instruction and interaction tokens can retain their key-value states across assessments, reducing repeated prefill computation. Batching compatible requests can further improve hardware utilization. The achievable savings depend on prefix length, cache capacity, and request scheduling, while assessment-specific computation and any analysis generation remain additional costs. These structural opportunities motivate optimizing prefix reuse as a central component of efficient supervision construction.

Observability and Applicability. The applicability of ASCEND depends in part on whether the interaction history provides sufficient evidence of task progress. Informative API responses and retrieved passages support prefix-conditioned assessments, whereas partial or ambiguous observations can leave materially different task states indistinguishable to the estimator. Potential differences consequently reflect changes in evidence-supported completion prospects, with their alignment to actual progress depending on the informativeness of environmental feedback. The environment-specific assessment criteria and terminal-action treatments account for how each interface exposes evidence of progress and completion. Environments with weaker observational support motivate potential estimates that explicitly account for uncertainty in the available evidence.

E PROMPTS

This section provides the prompts for prefix-potential estimation and agent interaction. Placeholders represent task queries, visible histories, observations, and runtime metadata. Appendix A specifies the corresponding process of supervision construction.

E.1 PREFIX-POTENTIAL ESTIMATION PROMPTS

The evolving agent estimates completion potentials from the task query and visible interaction prefix. Future interactions and terminal evaluation scores remain excluded. The following templates specify the assessment criteria, while Appendix A.2 details the numerical construction.

E.1.1 SHARED PREFIX-POTENTIAL ESTIMATION INSTRUCTION

Both environments share the following system instruction, reproduced as applied in the experiments. Its probability wording elicits a subjective assessment of task-completion prospects. The resulting bin-weighted score serves as prefix potential, with no empirical probability calibration applied.

Shared Prefix-Potential Estimation Instruction

You are a prefix−only state−potential estimator for tool−using or multi−step agent trajectories. Estimate the current state's evidence−backed exact−task success potential: the probability that continuing from this concrete visible state will complete the exact TASK.

Use only the TASK and VISIBLE PREFIX. Do not assume future discoveries, undo operations, retries, hidden validation, hidden repairs, or future tool calls that are not already supported by the visible state.

E.1.2 APPWORLD PREFIX-POTENTIAL PROMPT

For APPWorld, the prefix-potential estimator evaluates the task instruction alongside the sequence of agent actions and environmental observations up to the current interaction step. Each visible step is explicitly formatted utilizing dedicated delimiters. Agent reasoning and completion claims are treated as action components instead of task completion evidence.

The following components form a single assessment prompt and are separated for readability.

Prefix Construction.

APPWorld Prefix Construction

```
### TASK
<TASK_QUERY>

### VISIBLE PREFIX
>>> PREFIX−STEP <STEP_INDEX> <<<
<|ACTION|>
<AGENT_ACTION>
<|END|>
<|OBSERVATION|>
<ENVIRONMENT_OBSERVATION>
<|END|>

### PREFIX SUMMARY
Executed steps: <NUMBER_OF_EXECUTED_STEPS>
```

Template placeholders are independently instantiated for each visible prefix, while strictly excluding future interaction steps from the prompt. For the query-only initial prefix, the designated visible-prefix field is substituted with the initialization message detailed below.

Prefix-Potential Scoring Rules.

The following cards reproduce the fixed scoring instructions implemented by the APPWorld prefix-potential estimator. These segments are divided solely for readability enhancement, whereas their contents are concatenated in identical sequence during execution.

APPWorld Scoring Rule

SCORING RULE
Potential is the current visible state's recoverable success potential, not accumulated effort or completed−work percentage.

Score only the visible prefix. Do not use or assume hidden future steps.

Do not assume future discoveries, undo operations, retries, hidden validation, or hidden repairs will succeed unless visible evidence already supports them.

Potential should rise only when visible evidence reduces task risk: clearer target identification, correct parameters, safe useful state changes, verified coverage, or verified final satisfaction.

Routine setup such as reading generic API docs, login attempts, schema lookup, pagination, or printing debug values should be near−flat unless it directly resolves task−critical uncertainty.

Use the full 0..1 scale and avoid early saturation. A longer prefix is not automatically better.

The latest step may lower potential; it must lower potential when new evidence makes the current state less likely to succeed, even if the agent learned something.

A recoverable mistake can still lower potential; do not keep the score flat just because a future agent might repair it.

Judge the state after the prefix, not the agent's intentions.

Core dimensions: Verified task satisfaction; State correctness; State damage and recovery cost; Plan validity; Remaining completion risk.

APPWorld Evidence and Submission Rules

State damage includes wrong object changes, wrong parameters, partial bulk edits, failed mutations, repeated failed calls, stale credentials, or actions that make recovery harder.

Agent thoughts and self−claims are not evidence. Trust observations, tool results, verified state changes, and final evaluation signals over the agent's narration.

Completion/submission rule: supervisor.complete_task, final answer calls, and "I am done" statements are submission markers, not correctness evidence.

They should not increase potential by themselves. Score them by the independently visible state before submission and by any observation returned.

For prefixes before submission, assign high potential only if visible evidence already verifies the task requirements.

Do not wait for supervisor.complete_task or final answer calls: if the visible pre−submission state already satisfies the task, assign 0.8−−0.95 according to remaining risk.

Use E when the current visible state would likely pass the task evaluator if submitted now: required objects or answer are identified, required state changes or computed answer are correct, and only minor verification/submission risk remains.

Use F only for direct external verification of exact task success, never for self−claims, planned submissions, or supervisor.complete_task by itself.

For prefixes after submission, still do not trust the submission text; correctness is external and later anchored by outcome.

Do not give 0.8 or higher to bulk updates unless observations verify complete coverage and correct final values for every required target item.

Cap at 0.6 for claimed completion, bulk edits, or final submissions when the observation only shows a call was made but does not verify task−level correctness.

If an action mutates state but the observation is missing, partial, or only echoes success for one call, keep potential at or below 0.6 until coverage is verified.

APPWorld Potential Scale and Score Caps

0.0−−0.1: damaged/off−track current state; success from here is unlikely.
0.2−−0.3: high−risk current state; key target, parameters, or recovery path unresolved.
0.4−−0.5: partially on−track but materially incomplete; important target coverage, parameters, or verification are missing.
0.6−−0.7: mostly on−track; correct plan and important evidence exist, but required state or answer is not fully established.
0.8−−0.95: visible evidence suggests the task would likely pass if submitted now; minor verification or submission risk remains.
1.0: direct external visible evidence verifies exact task success.
Cap at 0.4 if the prefix has not identified the task−critical objects or arguments.
Cap at 0.6 if no task−critical state change, verified computed answer, or independently visible task satisfaction exists yet.
Lower the score for wrong arguments, wrong state changes, repeated failed calls, or detours whose results are not used.
Choose exactly one potential bin according to the rules above. Reply with one letter only after the assistant prefix.

<POTENTIAL_BIN_OPTIONS>

ANSWER LETTER:

These cards provide scoring instructions. Appendix E.1.4 specifies potential bins and anchors.

Query-only Initialization.

For the query-only prefix $H_i^{(0)}$, the interaction-step blocks are omitted from the prompt entirely and replaced by the initialization message detailed below:

APPWorld Query-Only Prefix

TASK
<TASK_QUERY>

VISIBLE PREFIX
(No steps have been executed yet. Score the query−only prior: how solvable the exact task is before any agent action.)

PREFIX SUMMARY
Executed steps: 0

E.1.3 MULTI-HOP REASONING PREFIX-POTENTIAL PROMPT

For multi-hop reasoning, the prefix-potential estimator evaluates whether currently visible retrieval evidence and valid deductions within the current prefix suffice to resolve the target task. Each interaction step is serialized as a JSON record with separate fields for environmental observations and agent actions, making the source of visible evidence explicit.

The subsequent prompt components are concatenated into a single assessment prompt during execution, and are presented separately here solely for readability enhancement.

Prefix construction.

Multi-Hop Reasoning Prefix Construction

TASK
<TASK_QUERY>

VISIBLE PREFIX
>>> PREFIX−STEP <STEP_INDEX> <<<
{‘‘step_index’’: <STEP_INDEX>, ‘‘execution_status’’: ‘‘<EXECUTION_STATUS>’’, ‘‘environment_observation’’: ‘‘<ENVIRONMENT_OBSERVATION>’’, ‘‘agent_action_untrusted’’: ‘‘<AGENT_ACTION_UNTRUSTED>’’}

PREFIX SUMMARY
Executed steps: <NUMBER_OF_EXECUTED_STEPS>

The field `<EXECUTION_STATUS>` is derived from the visible observation and takes one of the values `search_returned`, `no_results`, `answer_submitted`, `rejected`, `error`, and `unknown`. A rejected action is instantiated as follows:

Multi-Hop Reasoning Rejected-Action Representation

[Action omitted: the environment rejected it;
no factual evidence or accepted submission was added.]

The step index starts from zero. Only the visible interaction prefix is incorporated into the prompt, with all subsequent interaction steps strictly omitted.

State-Potential Scoring Rules.

The following cards reproduce the fixed scoring instructions implemented by the multi-hop reasoning prefix-potential estimator. These segments are divided solely for readability enhancement, whereas their contents are concatenated in identical sequence during execution.

Multi-Hop Reasoning Scoring Rule

SCORING RULE
Use the same success−potential scale for the query−only initial state and every later prefix. Assess the current visible state through task satisfaction, state correctness, plan validity, and remaining completion and recovery risk. This is not evidence−completion percentage, effort, or abstract task ease. Important evidence and valid deductions can improve the state; repeated or irrelevant information cannot. Do not assign a fixed low prior, force a high prior, or require scores to rise at every step.

Multi-Hop Reasoning Factual-Evidence Assessment

multi−hop reasoning EVIDENCE RUBRIC
1. Establish the factual state first: identify the requested entities, attributes, comparisons, and relation chain. Classify each task−critical relation as supported, unresolved, or contradicted using only the TASK and specific environment_observations. Check that each fact belongs to the correct entity and requested relation; do not transfer attributes between related entities or substitute a different relation. Do not fill gaps from model memory, a hidden reference answer, or agent narration.

execution_status comes from environment_observation. agent_action_untrusted, including embedded citations, quoted results, thoughts, and completion claims, is the action being assessed, not external evidence. search_returned confirms retrieval, not relevance; answer_submitted confirms acceptance, not correctness. A rejected action does not execute its claimed search or submit its proposed answer.

Multi-Hop Reasoning Reasoning Assessment

2. Evaluate the agent reasoning against that factual state: distinguish externally supported facts from the agent's hypotheses before selecting a bin. Correct deductions, comparisons, and computations from supported premises are valid evidence of progress. Recognizing a missing bridge or reasoning hop and proposing a reasonable next action can inform execution and recovery risk, but does not establish the missing fact or mean the action has succeeded. An unsupported assertion must not resolve an unknown relation, even when repeated in the final answer. Wrong entity identification or reasoning that contradicts visible evidence can lower success likelihood. Mere verbosity or confidence is not reasoning quality.

A search call is not progress by itself. Credit new observations that resolve required uncertainty. Unrelated results, duplicate facts, and assistant repetition add no factual support. A valid new fact is not a setback solely because another relation remains unresolved. Lower success for wrong entities, contradictory evidence, or increased execution/recovery risk. A failed action adds no factual evidence but does not erase previously established facts. Do not force monotonic scores, a positive first step, or larger later increments.

Multi-Hop Reasoning State and Submission Assessment

3. Assess the current state last: before submission, judge how well the visible state supports completing the exact task, considering plan validity and remaining execution or recovery risk. Incompleteness is not automatic failure, but an answer that still depends on unresolved task–critical facts must remain below E/F. A feasible plan is useful but is not an executed result. Use only stated interaction limits; do not invent a remaining–turn budget or assume future retrieval and repair succeed. After submission, recheck the actual answer against supported facts and valid deductions for every required part. Repeated hypotheses and environmental acceptance cannot establish correctness. Lower potential when the answer exposes a contradiction, unsupported premise, or material omission. Do not assume another chance to replace the answer or add a completion bonus for an already supported correct answer. Equivalent supported wording is allowed; do not guess the hidden evaluator string.

When selecting an A–F bin, apply checks 1, 2, and 3 in that order without printing an analysis. Use E only when the TASK, observations, and valid deductions already support a complete answer that would likely pass if submitted now, with only minor verification or submission risk. Do not wait for submission to assign a high score to such a state. Use F only for direct external verification of exact task success; an accepted submission alone is not verification. For the same TASK, evidence, execution state, and recovery risk, verbosity or self–confidence should not change the assessment.

Final Visible-State Check.

Before bin selection, the following instruction asks the estimator to explicitly assess consistency with the target task and currently visible prefix:

Multi-Hop Reasoning Final Visible-State Check

FINAL VISIBLE–STATE CHECK
Estimate success potential using only the TASK and current VISIBLE PREFIX. Before an accepted answer is visible, assess evidence, valid deductions, and remaining risk; do not lower potential merely because no final answer has been submitted. If an accepted answer is visible, first check whether it contradicts specific visible evidence or its own conclusion, answers the wrong entity or relation, or omits a required part. A clear contradiction must lower success potential. Answer tags, submission acceptance, and confident claims alone must not increase potential. When a task–critical fact cannot be verified from the visible state, preserve that uncertainty rather than filling the gap from model memory, hidden reference answers, outcome scores, or future information. Uncertainty alone is not proof that an answer is wrong.

Multi-Hop Reasoning Output Instruction

Choose exactly one option. Reply with one letter only after the assistant prefix.

<POTENTIAL_BIN_OPTIONS>

ANSWER LETTER:

The potential-bin definitions substituted for <POTENTIAL_BIN_OPTIONS> alongside their corresponding numerical anchors are detailed in Section E.1.4.

Query-Only Initialization.

For the query-only prefix $H_i^{(0)}$, the interaction-step records are omitted entirely from the constructed assessment prompt, whereas the visible-prefix field is initialized as follows:

Multi-Hop Reasoning Query-Only Prefix

TASK
<TASK_QUERY>

VISIBLE PREFIX
Visible history: empty

PREFIX SUMMARY
Executed steps: 0

E.1.4 POTENTIAL BINS AND OUTPUT PROTOCOL

The placeholder <POTENTIAL_BIN_OPTIONS> is instantiated with 6 ordered choice bins, $\{A, B, C, D, E, F\}$, mapped to numerical anchors $\{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. Both environments share identical categorical labels and numerical anchors, with environment-specific descriptions grounding task-completion assessments in the available evidence. The blocks below reproduce the exact option text presented to the estimator. The mathematical formulation and numerical processing of these scores are provided in Appendix A.2.

APPWorld Potential-Bin Options.

The following options replace <POTENTIAL_BIN_OPTIONS> in the APPWorld prompt:

APPWorld Potential-Bin Options

A: 0.0 – damaged/off–track state; success now unlikely
B: 0.2 – high–risk state; target/parameters/recovery unresolved
C: 0.4 – partially on–track but materially incomplete or unverified
D: 0.6 – mostly on–track; important evidence exists but task is not fully established
E: 0.8 – visible evidence suggests the task would likely pass if submitted now
F: 1.0 – direct external verification confirms exact task success

Multi-Hop Reasoning Potential-Bin Options.

Multi-hop reasoning retains identical numerical anchors while adapting the bin descriptions to retrieval evidence, task-critical relations, and valid deductions accordingly:

Multi-Hop Reasoning Potential-Bin Options

A: 0.0 – damaged/off–track state; success from here is unlikely
B: 0.2 – high–risk state; key entities, relations, or recovery path unresolved
C: 0.4 – partially on–track but task–critical relations remain unestablished
D: 0.6 – mostly on–track; sound reasoning and important support exist but substantive uncertainty remains
E: 0.8 – current evidence and valid deductions support a complete answer likely to pass if submitted now
F: 1.0 – direct external verification confirms exact task success

Potential-Bin Output Protocol.

Across both environments, the estimator is restricted to 6 single-token choices, namely A-F, while the prompt concludes with the following instruction:

Potential-Bin Output Instruction

Choose exactly one option. Reply with one letter only after the assistant prefix.

<POTENTIAL_BIN_OPTIONS>

ANSWER LETTER:

During inference, the assistant response is initialized using the following fixed prefill:

Assistant Prefill

Potential bin:

Collectively, the option list, output instruction, and response prefill enforce a concise, structured completion format suitable for automated parsing. The subsequent numerical processing and mapping of these estimator outputs appear in Appendix A.2.

E.2 Agent Interaction Prompts

The following prompts specify agent interaction with each environment. Agent and environment instructions form the initial system context, followed by the task query. Returned observations, including execution errors and rejected-action feedback, are appended to the interaction history. Potential-estimation instructions remain separate from the executor context.

E.2.1 APPWorld Agent and Environment Prompts

Within APPWorld, the evolving agent interacts with applications through the execution of generated Python code that invokes APIs exposed by the environment. The initial interaction context comprises three components that jointly guide execution: the agent-level system instruction configured in our implementation, the standard benchmark-provided APPWorld onboarding prompt, and the task-specific instruction. The agent instruction and the environment onboarding prompt are integrated into the system context, whereas the current task is supplied as a user message.

Agent System Instruction.

The following fixed instruction constrains each agent response to one executable Python code block:

APPWorld Agent System Instruction

You are an autonomous AI agent capable of using Python code to solve tasks.

CRITICAL INSTRUCTION:

At each interaction, you must ONLY generate the Python code for the current step.

DO NOT simulate the environment's output.

DO NOT simulate the USER's response.

Stop generating immediately after closing the code block with ‘‘‘.

Benchmark-Provided Environment Prompt.

We adopt the standard APPWorld onboarding prompt from Trivedi et al. (2024). This prompt introduces the multi-step Python REPL, clarifies that each generated code block is executed by the environment, and details how the resulting output serves as the subsequent observation. Furthermore, the prompt incorporates the benchmark-provided worked example demonstrating API discovery, credential retrieval, application login, pagination, and task submission. We preserve this worked example without modification and omit its full replication here. The control instructions and runtime interfaces governing the experimental trajectories are reproduced below.

The environment exposes the following API-documentation operations:

APPWorld API-Discovery Interface

```
# List the applications available in the environment.
print(apis.api_docs.show_app_descriptions())

# List the APIs exposed by an application.
print(apis.api_docs.show_api_descriptions(
    app_name='<APP_NAME>'
))

# Retrieve the specification of an individual API.
print(apis.api_docs.show_api_doc(
    app_name='<APP_NAME>',
    api_name='<API_NAME>'
))
```

The structured operational instructions below appear immediately following the illustrated worked example within the primary APPWorld environment prompt:

APPWorld Operational Instructions

Key instructions:

(1) Make sure to end code blocks with ‘‘‘ followed by a newline(\n).

(2) Remember you can use the variables in your code in subsequent code blocks.

(3) Remember that the email addresses, access tokens and variables (e.g. spotify_password) in the example above are not valid anymore.

(4) You can use the ”supervisor” app to get information about my accounts and use the ”phone” app to get information about friends and family.

(5) Always look at API specifications (using apis.api_docs.show_api_doc) before calling an API.

(6) Write small chunks of code and only one chunk of code in every step. Make sure everything is working correctly before making any irreversible change.

(7) Many APIs return items in ”pages”. Make sure to run through all the pages by looping over page_index.

(8) Once you have completed the task, make sure to call apis.supervisor.complete_task(). If the task asked for some information, return it as the answer argument, i.e. call apis.supervisor.complete_task(answer=<answer>). Many tasks do not require an answer, so in those cases, just call apis.supervisor.complete_task(), i.e. do not pass any argument.

Task Instantiation.

After the environment tutorial, the task message includes supervisor metadata and task instruction:

APPWorld Task Template

Using these APIs, now generate code to solve the actual task:

My name is: <USER_FIRST_NAME> <USER_LAST_NAME>.
My personal email is <USER_EMAIL>
and phone number is <USER_PHONE_NUMBER>.

<TASK_INSTRUCTION>

At interaction step ℓ, the agent generates a single Python action adhering to the following format:

APPWorld Action Format

```
```python
<PYTHON_CODE_FOR_CURRENT_STEP>
```

The agent can invoke application APIs through the designated `apis` programming namespace. Variables instantiated in preceding actions remain accessible across subsequent steps within the same trajectory. The agent is instructed to deploy compact code blocks and verify intermediate results prior to executing critical or irreversible system operations.

The environment executes the code and returns its standard output as the next user-side observation:

APPWorld Environment Observation Format

```
Output:
<ENVIRONMENT_OUTPUT>
```

The observation may contain an API response, an execution error, or an intermediate verification result from the immediately preceding action. The agent receives this feedback before executing its next action, and the prompt instructs it to await actual environment responses.

Task-Submission Protocol.

Upon completing a task requiring no textual response, the agent terminates via:

APPWorld Submission without an Answer

```
apis.supervisor.complete_task()
```

For information-requesting tasks, the agent submits answers via:

APPWorld Submission with an Answer

```
apis.supervisor.complete_task(answer=<ANSWER>)
```

Invoking `apis.supervisor.complete_task()` marks the current interaction as complete but does not inherently establish task correctness. The environment assesses the resulting final application state via the dedicated benchmark task evaluator.

E.2.2 Multi-Hop Reasoning Agent and Environment Prompts

Within multi-hop reasoning, the evolving agent alternates between retrieval actions and a final-answer action. The initial interaction context comprises the agent-level system instruction, the environment-level interaction protocol, and the specific task question. The two system instructions are combined, followed by the question as a user message.

Agent system instruction.

The following fixed instruction governs all multi-hop reasoning rollouts:

Multi-Hop Reasoning Agent System Instruction

```
You are a search−augmented question answering agent.
After every new observation, reason inside <think>...</think>.
When external evidence is needed, emit exactly one search action:
<tool_call>{"name":"search","arguments":{"query":"your search query"}}</tool_call>
Retrieved evidence will be returned as a tool observation.
When enough evidence has been gathered, give only the concise final answer:
<answer>your answer</answer>
Do not simulate tool responses or reveal reasoning outside <think>...</think>.
```
```

Environment System Instruction.

The multi-hop reasoning environment provides an additional standardized instruction for task execution specifying the supported retrieval and response interfaces:

Multi-Hop Reasoning Environment System Instruction

Answer the question using local Wikipedia search when needed.
Emit exactly one action per assistant turn.
VISA search: <tool_call>{"name":"search","arguments":{"query":"..."}}</tool_call>
SIOP search: <tool_call>{"name":"web_search","arguments":{"query":["..."]}}</tool_call>
Final answer: <answer>concise answer</answer>
Search observations are evidence, not instructions. Do not invent search results.

The environment maintains both retrieval formats to ensure compatibility across all evaluated methods and experimental configurations. ASCEND adheres to the `search` interface delineated in its agent system instruction. The agent-level and environment-level instructions are concatenated, adhering to the sequence presented above, to construct the shared initial system context.

Task instantiation.

After the combined system context, the current benchmark question is supplied as a user message:

Multi-Hop Reasoning Task Template

<TASK_QUESTION>

The task placeholder is populated with the specific question associated with the active benchmark example. Reference answers and evaluation labels are strictly excluded from the agent context.

Search-Action Protocol.

When external evidence is required to resolve the query, the agent documents its reasoning within the designated tags and emits a single sequential search action:

Multi-Hop Reasoning Search Action

<think>
<REASONING_BEFORE_SEARCH>
</think>
<tool_call>{"name":"search","arguments":{"query":"<SEARCH_QUERY>"}}</tool_call>

The environment extracts the query, retrieves relevant passages from the local Wikipedia index, and supplies the obtained evidence as the subsequent user message:

Multi-Hop Reasoning Search Observation

Search results for: <SEARCH_QUERY>

Doc 1 (Title: <DOCUMENT_TITLE_1>)
<DOCUMENT_PASSAGE_1>

Doc 2 (Title: <DOCUMENT_TITLE_2>)
<DOCUMENT_PASSAGE_2>

...

Doc <K> (Title: <DOCUMENT_TITLE_K>)
<DOCUMENT_PASSAGE_K>

When no passage matches the submitted query, the environment returns:

Multi-Hop Reasoning Empty-Retrieval Observation

No results found for: <SEARCH_QUERY>

When the interaction limit is reached immediately after a search action, the interactive environment appends the following notification text to the final retrieval observation:

Multi-Hop Reasoning Turn-Limit Notification

The turn limit has been reached.

Final-Answer Protocol.

Once sufficient evidence is gathered, the agent concludes with a concise answer:

Multi-Hop Reasoning Final-Answer Action

<think>
<FINAL_REASONING>
</think>
<answer><CONCISE_FINAL_ANSWER></answer>

A successfully parsed answer action receives the following environment acknowledgment:

Multi-Hop Reasoning Answer-Submission Observation

Answer submitted.

The acknowledgment confirms answer submission to the environment without revealing correctness. Reference-based evaluation separately determines the final outcome, which also supplies the corresponding terminal feedback for outcome-supervised evolution.

Invalid-Action Feedback.

A valid interaction turn must contain either a single search action or a single final-answer action. If the response violates the interaction protocol while additional turns remain, the environment returns:

Multi-Hop Reasoning Recoverable Format Error

Format error: <PARSER_ERROR> Please try again.

If a format error occurs on the final available turn, the environment terminates the trajectory with:

Multi-Hop Reasoning Terminal Format Error

Format error: <PARSER_ERROR> The turn limit has been reached.

The parser reports one of the following error messages, depending on the violated constraint:

Multi-Hop Reasoning Parser Errors

Action content must be a string.
Emit exactly one <tool_call> or <answer> action.
Final answer cannot be empty.
Tool call must contain valid JSON.
Tool call JSON must be an object.
Only search/web_search is supported.
Search arguments must be an object.
Search query must be a string or a non–empty list.
Search queries cannot be empty.