STABLE TEMPORAL ABSTRACTION FOR HIERARCHICAL LANGUAGE MODEL AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Hierarchical reinforcement learning improves long-horizon control by organizing primitive actions around persistent subgoals and assigning credit at multiple temporal scales. Recent hierarchical language agents bring these benefits to interactive tasks by explicitly separating subgoal planning from action execution. We observe, however, that an explicit hierarchy does not by itself determine how stable the resulting temporal abstraction is: the learned boundary policy may replace the subgoal almost every turn, making it effectively transient, or retain a subgoal after it has stopped being appropriate. We call this *temporal abstraction instability*. To mitigate this issue, we propose ***S**table **T**emporal **A**bstraction via **C**onstrained boundary credit* (STAC), a general boundary-policy framework that represents premature replanning and stale persistence as constraint costs. We instantiate STAC in two forms: STAC-D (Deterministic) is the duration-based instantiation, using fixed proxies for premature replanning and stale persistence; STAC-P (Probabilistic) learns semantic completion probabilities from environment-grounded supervision and converts them into state-dependent boundary costs. Both instantiations apply their costs only to the boundary update, leaving task rewards and all non-boundary objectives unchanged. With Qwen3-0.6B, STAC-P reaches 95.3% success on ALFWorld and 74.2% on WebShop, exceeding the prior state-of-the-art baseline by 12.0 and 12.8 points, respectively; similarly large gains appear with the other backbones. The code is available at `https://anonymous.4open.science/r/STAC-CB18`.

1 INTRODUCTION

Hierarchical reinforcement learning (HRL) has long provided a principled approach to long-horizon decision making (Sutton et al., 1999; Dietterich, 2000; Nachum et al., 2018; Li et al., 2019; Zhang et al., 2020; Gürtler et al., 2021; Lee et al., 2022; Corcoll & Vicente, 2022). Rather than selecting every primitive action at a single temporal scale, HRL introduces temporally extended subgoals or options and decomposes their realization into lower-level actions (Bacon et al., 2017). This decomposition can shorten effective planning horizons, reuse behavior across states, preserve intermediate intent, and separate credit for high-level planning from credit for low-level execution.

Recent work has begun transferring these benefits to language-model agents (Zhou et al., 2024; Hu et al., 2025; Zhen et al., 2026; Jin et al., 2026; Diao et al., 2026; Park & Kwon, 2026; Peng et al., 2026). For example, Peng et al. (2026) introduce HiPER, which represents each interaction turn through an explicit Plan–Execute hierarchy: a high-level component produces a textual subgoal, a boundary decision determines whether that subgoal should persist or be replaced, and a low-level component selects an environment action. Its hierarchical advantage estimator further assigns distinct learning signals to subgoal, boundary, and action tokens. Such methods demonstrate that explicit hierarchical structure and multi-timescale credit assignment can improve long-horizon language-agent performance.

We observe, however, that the architecture alone does not determine how stable the learned temporal abstraction will be. Useful temporal abstraction requires a subgoal to persist long enough to coordinate multiple actions and to terminate once it is completed or no longer appropriate. The learned boundary policy can drift away from either requirement. It may replace the subgoal almost every turn, reducing it to a transient one-step label and moving the hierarchy toward a flat policy. Alternatively, it may

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.

retain the same subgoal too long, delaying state-dependent replanning after that subgoal stops being appropriate. We call this tendency *temporal abstraction instability*. It matters most in multi-phase tasks, where success requires both persistence within a phase and a well-timed transition between phases.

To address this instability, we propose ***S**table **T**emporal **A**bstraction via **C**onstrained boundary credit* (STAC), a general boundary-policy framework motivated by a constrained formulation. STAC represents premature replanning and stale-subgoal persistence as constraint costs and optimizes a fixed-multiplier relaxation of the resulting Lagrangian. We instantiate it in two forms: STAC-D is the deterministic duration-based instantiation, using fixed proxies for premature replanning and stale persistence, whereas STAC-P is the probabilistic completion-based instantiation, learning semantic completion probabilities from environment-grounded supervision and converting them into state-dependent boundary costs. For a hierarchical agent that exposes a boundary advantage, it modifies only the coefficient of the sampled boundary decision. More generally, it applies its costs only to the native boundary update and leaves task rewards and non-boundary objectives unchanged. This decision locality distinguishes STAC from reward shaping and makes it independent of the underlying actor architecture or return estimator.

We evaluate STAC with two hierarchical language-agent architectures: HiPER (Peng et al., 2026), which exposes a boundary advantage, and GLIDER (Hu et al., 2025), which writes an implicit keep/switch decision as a Boolean suffix of its low-level response. These architectures test the same boundary constraints under distinct policy factorizations and training objectives. In our Qwen3-0.6B experiments, unmodified HiPER prematurely replans on more than half of all turns. Integrating STAC into HiPER instead promotes multi-turn subgoals, preserves nearly perfect structured-output validity, and improves task success. In particular, HiPER+STAC-P improves over unmodified HiPER by 12.0 points on ALFWorld and 12.8 points on WebShop with Qwen3-0.6B. Other representative results include a 23.5-point ALFWorld gain from HiPER+STAC-D with Llama-3.2-1B-Instruct and a 3.9-point gain with Qwen2.5-1.5B-Instruct. These are selected results. Our contributions are:

• We identify and characterize *temporal abstraction instability*: a hierarchy can be architecturally explicit yet still learn subgoals that are too transient or too persistent for useful temporal structure.

• We formulate boundary stabilization as a constrained program with separate costs for premature replanning and stale-subgoal persistence, and evaluate its fixed-multiplier relaxation as a decision-local, two-sided boundary regularizer. We instantiate it as deterministic duration-based STAC-D and probabilistic completion-based STAC-P.

• We evaluate the framework with two structurally distinct hierarchical language-agent architectures: HiPER's explicit Plan–Execute boundary policy and GLIDER's implicit Boolean completion policy. Across ALFWorld and WebShop, these examples show that the same constraints can improve temporal abstraction without prescribing a particular actor architecture or training objective.

2 Hierarchical Agents and Temporal Abstraction Instability

We first describe hierarchical agents through their temporal decision structure without assuming a particular HRL architecture or learning algorithm.

Task, action, and reward. Let $x \sim p(\mathcal{X})$ be an interactive task and let $t \in \{0, \dots, T-1\}$ index the environment turns in an episode. At turn t, s_t contains the current observation and interaction history available to the policy. The policy selects a primitive action a_t, meaning one environment interaction such as a tool call together with its arguments, and the environment transitions to the next observation. The environment returns a task reward after the action; r_t denotes the resulting training reward. The return is $G_t = \sum_{u=t}^{T-1} \gamma^{u-t} r_u$, where $\gamma \in (0, 1]$ is the discount factor. The task objective is then

$$J_{\text{task}}(\theta) = \mathbb{E}_{x \sim p(\mathcal{X}),\, \tau \sim \pi_\theta(\cdot|x)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right], \tag{1}$$

where τ is the complete interaction trajectory.

Subgoals. A subgoal z_t is a textual intermediate objective whose completion normally requires several primitive actions. For a tool-using language agent, this means several tool calls with concrete arguments. For example, completing "heat the object" may require navigating to an appliance, opening it, placing the object, and activating it. The subgoal should persist while its actions are executed, then be replaced when the intermediate objective is complete or no longer appropriate.

Boundary events. Hierarchical agents contain boundary events where the policy decides whether its active subgoal persists. At boundary event t, let z_t^- be the active subgoal immediately before the decision, let u_t be the available boundary context, and let

$$q_t \in \{\text{KEEP}, \text{SWITCH}\} \tag{2}$$

denote the choice to retain z_t^- or terminate it and invoke the subgoal-selection mechanism.

We denote the boundary policy by $\pi_\theta^b(q \mid u_t)$, the probability of choosing $q \in \{\text{KEEP}, \text{SWITCH}\}$ given the boundary context u_t; the superscript b indicates that this distribution governs the boundary decision. Some hierarchical agents parameterize this binary distribution explicitly. Others have no separate boundary-policy module and instead express the decision as a recognizable text span within a larger autoregressive response. In the latter case, let $w(q)$ be the token sequence that expresses decision q and define its span score

$$\ell_\theta(w(q) \mid u_t) = \sum_{j \in M(w(q))} \log \pi_\theta(w_j(q) \mid u_t, w_{<j}(q)). \tag{3}$$

Holding the preceding context u_t fixed, we score both possible decision spans and normalize their scores to obtain the boundary policy:

$$\pi_\theta^b(q \mid u_t) = \frac{\exp\{\ell_\theta(w(q) \mid u_t)\}}{\sum_{q' \in \{\text{KEEP},\text{SWITCH}\}} \exp\{\ell_\theta(w(q') \mid u_t)\}}. \tag{4}$$

Thus both implementations provide the same mathematical object, π_θ^b: an explicit binary distribution in the first case and a distribution induced from autoregressive span scores in the second.

Temporal abstraction instability. Let $d_t \geq 1$ be the age of the active subgoal at the boundary event, resetting after a valid SWITCH. As π^b moves toward always switching, d_t approaches 1, segment-level returns approach per-turn returns, and the subgoal approaches a transient scratchpad. As it moves toward always keeping, the agent replans less often than the task requires. We call this two-sided tendency *temporal abstraction instability*. Maximizing return alone does not generally impose an explicit duration or completion constraint on the boundary policy. A hierarchical policy may encode age or history in u_t, but unless its boundary objective directly prices premature switching and stale retention, optimization does not guarantee useful segment duration. This failure mode is therefore objective-level rather than architecture-specific.

3 STAC: A General Framework for Stable Temporal Abstraction

3.1 General framework

Let $g_{\text{short}}(u_t, z_t^-, d_t)$ and $g_{\text{stale}}(u_t, z_t^-, d_t)$ be bounded cost functions in $[0, 1]$ that measure, respectively, whether switching at boundary event t would be premature and whether keeping the current subgoal would be stale. For a parsed boundary decision, define

$$\begin{aligned} c_t^{\text{short}} &= \mathbb{I}[q_t = \text{SWITCH}]\, g_{\text{short}}(u_t, z_t^-, d_t), \\ c_t^{\text{stale}} &= \mathbb{I}[q_t = \text{KEEP}]\, g_{\text{stale}}(u_t, z_t^-, d_t). \end{aligned} \tag{5}$$

We write $c_t^i(q)$ for the corresponding expression evaluated under a candidate boundary decision q. At update k, let $\mathcal{D}_k$ denote the training distribution over valid boundary contexts, active subgoals, and ages. Let $\mathcal{J}_k^{\text{task}}(\theta)$ denote the boundary-learning objective native to the underlying HRL structure. Define the decision-local constraint surrogates by enumerating the binary boundary:

$$\mathcal{C}_i^k(\theta) = \mathbb{E}_{(u_t, z_t^-, d_t) \sim \mathcal{D}_k} \left[\sum_{q \in \{\text{KEEP},\text{SWITCH}\}} \pi_\theta^b(q \mid u_t) c_t^i(q) \right], \quad i \in \{\text{short}, \text{stale}\}. \tag{6}$$

These quantities are the current policy's expected premature-switch and stale-keep rates on the sampled boundary contexts. They define the following constrained surrogate problem without introducing a new return or replacing $\mathcal{J}_k^{\text{task}}$:

$$\begin{aligned} \max_{\theta} \quad & \mathcal{J}_k^{\text{task}}(\theta) \\ \text{s.t.} \quad & \mathcal{C}_{\text{short}}^k(\theta) \le \rho_{\text{short}}, \\ & \mathcal{C}_{\text{stale}}^k(\theta) \le \rho_{\text{stale}}, \end{aligned} \tag{7}$$

where the ρ_i specify tolerated violation rates. Because these surrogates are measurable, the violation rates realized by the learned policy can be read off during training; Fig. 1 reports them for our runs. Its Lagrangian is

$$\mathcal{L}_k(\theta, \boldsymbol{\mu}) = \mathcal{J}_k^{\text{task}}(\theta) - \sum_{i \in \{\text{short},\text{stale}\}} \mu_i \big(\mathcal{C}_i^k(\theta) - \rho_i\big), \quad \mu_i \ge 0. \tag{8}$$

In the reported experiments, however, we optimize the penalized boundary objective $\mathcal{J}_k^{\text{task}}(\theta) - \sum_i \mu_i \mathcal{C}_i^k(\theta)$, not a primal–dual algorithm that enforces chosen budgets. For completeness, an adaptive extension could use the projected dual update below:

$$\mu_i \leftarrow \Big[\mu_i + \eta_\mu \big(\widehat{\mathcal{C}}_i^k - \rho_i\big)\Big]_+, \tag{9}$$

where $\widehat{\mathcal{C}}_i^k$ is the batch estimate of the corresponding constraint surrogate. For fixed $\boldsymbol{\mu}$, the terms $\sum_i \mu_i \rho_i$ are constant in θ, so $\arg\max_\theta \mathcal{L}_k(\theta, \boldsymbol{\mu}) = \arg\max_\theta [\mathcal{J}_k^{\text{task}}(\theta) - \sum_i \mu_i \mathcal{C}_i^k(\theta)]$ and the budgets ρ_i do not enter the update. Thus our experiments do not enforce user-specified violation budgets. Appendix C states the limited sense in which an exact fixed-penalty optimum corresponds to a constrained solution at its *post hoc* achieved violation rates; because those rates are read off after training, this equivalence certifies no user-chosen budget.

3.2 Decision-local realizations

The constraint in equation 6 admits two realizations. For an on-policy hierarchical agent with behavior policy π_k^b that has full support over the binary decision space, define $r_t^b(\theta) = \pi_\theta^b(q_t \mid u_t)/\pi_k^b(q_t \mid u_t)$. Then

$$\mathbb{E}_{q_t \sim \pi_k^b(\cdot \mid u_t)} \left[r_t^b(\theta) c_t^i(q_t) \right] = \sum_{q \in \{\text{KEEP},\text{SWITCH}\}} \pi_\theta^b(q \mid u_t) c_t^i(q). \tag{10}$$

Thus a sampled importance-weighted cost and an exact enumeration over KEEP/SWITCH estimate the same unclipped constraint surrogate. Algorithm-specific clipping, minibatching, and optimizer dynamics can still make their finite-step updates differ.

Sampled advantage-space rule. If the original objective supplies a scalar boundary advantage $A_t^{b,\text{native}}$, collecting the sampled task and constraint terms gives

$$\widehat{A}_t^{b,\text{STAC}} = A_t^{b,\text{native}} - \mu_{\text{short}} c_t^{\text{short}} - \mu_{\text{stale}} c_t^{\text{stale}}. \tag{11}$$

The hierarchical agent uses this combined advantage in its original actor objective; for PPO, it is inserted into the standard clipped surrogate. Only tokens in the boundary-decision span receive the short- and stale-subgoal terms.

Enumerated probability-space rule. If the boundary is exposed only as a differentiable span or has no associated advantage, STAC adds the exact expected constraint cost

$$\mathcal{L}_{\text{STAC}} = \mathbb{E}_{(u_t, z_t^-, d_t) \sim \mathcal{D}_k} \left[\mu_{\text{short}} g_{\text{short}}\, \pi_\theta^b(\text{SWITCH} \mid u_t) + \mu_{\text{stale}} g_{\text{stale}}\, \pi_\theta^b(\text{KEEP} \mid u_t) \right]. \tag{12}$$

Under either realization, STAC changes no environment reward, critic target, subgoal-level term, or primitive-action term, and setting both multipliers to zero recovers the original objective. Appendix C gives the locality analysis and update details.

3.3 Two Instantiations

In this subsection, we introduce two realizations of STAC that differ in how they define c_t^{short} and c_t^{stale}: STAC-D uses deterministic duration-based indicators, whereas STAC-P uses a learned subgoal-completion probability.

3.3.1 STAC-D: Deterministic Duration-Based Instantiation

STAC-D supplies fixed, deterministic duration-based proxies to the general framework:

$$\begin{aligned} c_t^{\text{short}} &= \mathbb{I}[q_t = \text{SWITCH}]\,\mathbb{I}[d_t < d_{\min}], \\ c_t^{\text{stale}} &= \mathbb{I}[q_t = \text{KEEP}]\,\mathbb{I}[d_t \geq d_{\max}]. \end{aligned} \tag{13}$$

Thus, switching before the minimum duration incurs a premature-replanning cost, while retaining a subgoal at or beyond the maximum duration incurs a stale-persistence cost. This realization is environment-agnostic: it requires only the active subgoal's duration and does not use privileged environment state. These costs are not added to r_t; the exact reward choices in our experiments are given in Appendix A. Both duration thresholds are fixed before STAC-D training: $d_{\min}$ enforces a non-degenerate temporal abstraction, while $d_{\max}$ is derived from the episode horizon and the mean number of subgoals in solved benchmark trajectories (Appendix A.2).

3.3.2 STAC-P: Probabilistic Completion-Based Instantiation

The duration costs of STAC-D provide a simple signal for whether a boundary decision is too early or too late, but they assign the same desired duration to all subgoals. In many ALFWorld tasks, this is unnecessarily restrictive: picking up an object may be completed after one turn, whereas finding, cleaning, heating, and placing an object requires a longer sequence. STAC-P (Probabilistic) replaces fixed duration thresholds with a learned estimate of whether the active subgoal has been completed.

Completion estimator and environment-grounded supervision. At boundary event t, let h_t be a representation of the boundary context and active subgoal. STAC-P adds a lightweight sigmoid classification head:

$$p_t^{\text{done}} = f_\phi(h_t, z_t^-) = P_\phi\big(z_t^- \text{ is achieved at the decision-aligned state} \mid h_t, z_t^-\big)\,. \tag{14}$$

A large value means retention is likely unnecessary, while a small value means switching is likely premature. The agent-specific realization determines whether the head scores an already-realized transition or forecasts the transition associated with the current output.

During training, an environment-grounded oracle supplies a target at the state aligned with that boundary event. Let

$$y_t = \mathcal{O}(s_t^\star, z_t^-) \in \{0, 1\} \tag{15}$$

be a deterministic completion label, where $s_t^\star$ is the current state for a pre-action boundary or the resulting state for a post-action boundary. The agent never receives the oracle state as a policy input or task reward. We also define a validity mask $m_t \in \{0, 1\}$, where $m_t = 1$ only if both the active subgoal and its environment-state completion test are supported[1]. The head is then trained with masked binary cross-entropy,

$$\mathcal{L}_{\text{completion}} = -\frac{1}{\sum_t m_t} \sum_t m_t \left[y_t \log p_t^{\text{done}} + (1 - y_t) \log(1 - p_t^{\text{done}}) \right], \tag{16}$$

where the loss is omitted for a batch with no valid labels. This construction keeps supervision environment-grounded and self-contained, without relying on an external language-model judge. The completion head is updated through equation 16, while the actor sees a fixed per-turn semantic cost for that update. At deployment, boundary decisions use p_t^{done} from the learned head, not the fact oracle.

[1] Appendix A.1 shows ALFWorld and WebShop examples in Tables 4 and 5.

Semantic boundary costs. STAC-P turns the same completion probability into complementary costs for the two boundary actions. Let $\widetilde{p}_t^{\text{done}} = \text{stopgrad}(p_t^{\text{done}})$. For valid labels, we set

$$c_t^{\text{short}} = \mathbb{I}[q_t = \text{SWITCH}]\, m_t \, \left(1 - \widetilde{p}_t^{\text{done}}\right),$$
$$c_t^{\text{stale}} = \mathbb{I}[q_t = \text{KEEP}]\, m_t \, \widetilde{p}_t^{\text{done}}. \tag{17}$$

Thus, if the active subgoal is only 10% likely to be complete, switching incurs a cost of 0.9; if it is 95% likely to be complete, keeping it incurs a cost of 0.95. The two costs are two sides of one semantic question: whether the present option should end. Unlike STAC-D, they do not require a global maximum duration or presume that all subgoals should last equally long.

Remark (Importance of stop-gradient)**.** *The stop-gradient in equation 17 is important. It prevents the actor from reducing its penalty by changing the completion predictor rather than improving its boundary decisions.*

4 Instantiation in HiPER and GLIDER

We instantiate the same framework in two hierarchical agents: HiPER and GLIDER.

4.1 HiPER: explicit boundary advantage

HiPER's Plan–Execute actor emits boundary, subgoal, and primitive-action spans and is trained on-policy with PPO. Its policy-gradient decomposition is

$$\nabla_\theta J_{\text{task}}(\theta) \approx \mathbb{E}\left[\sum_t \begin{pmatrix} \nabla_\theta \log \pi_\theta^b(q_t \mid s_t, z_{t-1}) \widehat{A}_t^b \\ + \mathbb{I}[q_t = \text{SWITCH}] \nabla_\theta \log \pi_\theta^z(z_t \mid s_t) \widehat{A}_t^z + \nabla_\theta \log \pi_\theta^a(a_t \mid s_t, z_t) \widehat{A}_t^a \end{pmatrix}\right]. \tag{18}$$

HiPER learns a high-level value head $V_\phi^H(\cdot)$ and a low-level value head $V_\phi^L(\cdot)$, and constructs its boundary advantage from the difference between their value estimates:

$$\widehat{A}_t^b = (y_t - \beta_t)\left(V_\phi^H(s_t) - V_\phi^L(s_t, z_{t-1})\right),$$
$$y_t = \mathbb{I}[q_t = \text{SWITCH}], \qquad \beta_t = \pi_\theta^b(\text{SWITCH} \mid s_t, z_{t-1}). \tag{19}$$

HiPER standardizes that stream over valid boundary spans,

$$\overline{A}_t^b = \frac{\widehat{A}_t^b - \text{mean}_{u \in \mathcal{B}}\, \widehat{A}_u^b}{\text{std}_{u \in \mathcal{B}}\, \widehat{A}_u^b + \varepsilon}. \tag{20}$$

We set $A_t^{b,\text{native}} = \overline{A}_t^b$ in equation 11 and pass the result to HiPER's unchanged clipped actor update. For STAC-P, the pre-action boundary representation scores whether the preceding subgoal is already complete in the current state. Appendices B and C give the full estimator and update order.

4.2 GLIDER: implicit Boolean completion policy

GLIDER is a planner–controller agent whose low-level response has the form `<primitive action>; <true|false>`. The suffix `false` retains the active subtask, while `true` terminates it and returns control to the planner. Holding the generated action and preceding context fixed, we counterfactually score the two Boolean literals and obtain the boundary distribution in equation 4.

GLIDER has no on-policy boundary advantage, so STAC uses the enumerated probability-space rule in equation 12 as a Boolean-only auxiliary loss. The released behavior-cloning, AWAC, and offline-to-online objectives remain unchanged; prompt, action, delimiter, and EOS tokens receive no direct STAC gradient. For STAC-D, d_t is the one-based age before the action whose response contains the suffix. For STAC-P, the head forecasts whether that action will complete the active subtask and is supervised after execution with $\mathcal{O}(s_{t+1}, z_t^-)$, so post-action facts are never policy inputs. Appendix D gives the exact Boolean losses, token-span construction, censoring rules, and diagnostics.

Table 1: **Main results.** Mean binary success (%) ± SD, over three seeds. "Best" selects each seed's highest-validation checkpoint; "Final" uses its last checkpoint, and bold marks the highest mean in each backbone–metric column. Closed-source prompting results are reported by HiMAC (Jin et al., 2026). Tables 11 and 12 report ALFWorld Seen/Unseen and WebShop exact-success/graded-score breakdowns; Tables 14 and 15 report available source-backed ALFWorld task-category measurements.

		ALFWorld		WebShop	
Regime	Method	Best	Final	Best	Final
Qwen3-0.6B					
—	Base (Plan–Execute)	0.3 ± 0.5		1.2 ± 1.7	
—	Base + ReAct	0.9 ± 0.8		3.4 ± 2.6	
Flat RL	GRPO	37.5 ± 2.8	35.9 ± 4.7	44.7 ± 3.2	44.7 ± 3.2
Flat RL	GiGPO	59.1 ± 7.1	59.1 ± 7.1	51.3 ± 5.5	51.3 ± 5.4
Flat RL	PPO	35.8 ± 2.6	34.1 ± 3.1	40.2 ± 3.0	38.7 ± 3.5
Flat RL	SDAR	25.8 ± 3.6	21.4 ± 2.3	16.6 ± 13.6	9.6 ± 10.6
Hierarchical RL	GLIDER	56.6 ± 4.1	51.2 ± 3.5	44.0 ± 4.8	40.6 ± 4.5
Hierarchical RL	GLIDER + STAC-D (ours)	62.2 ± 4.0	58.5 ± 3.6	49.3 ± 3.2	46.7 ± 4.0
Hierarchical RL	GLIDER + STAC-P (ours)	68.8 ± 3.3	63.5 ± 3.9	53.4 ± 3.4	52.1 ± 3.5
Hierarchical RL	HiPER	83.3 ± 3.3	81.5 ± 3.6	61.4 ± 3.5	56.5 ± 3.0
Hierarchical RL	HiPER + STAC-D (ours)	91.4 ± 3.6	89.3 ± 3.5	69.3 ± 2.4	67.5 ± 4.2
Hierarchical RL	HiPER + STAC-P (ours)	$\mathbf{95.3 \pm 1.6}$	$\mathbf{93.2 \pm 2.4}$	$\mathbf{74.2 \pm 1.1}$	$\mathbf{74.2 \pm 1.1}$
Qwen2.5-1.5B-Instruct					
—	Base (Plan–Execute)	2.9 ± 1.6		1.9 ± 2.4	
—	Base + ReAct	8.3 ± 0.9		5.5 ± 6.3	
Flat RL	GRPO	71.1 ± 8.2	70.3 ± 8.0	56.8 ± 3.8	50.3 ± 4.1
Flat RL	GiGPO	86.7 ± 1.7	81.1 ± 3.2	67.4 ± 4.5	61.1 ± 5.5
Flat RL	PPO	68.2 ± 1.8	66.4 ± 2.8	51.5 ± 2.9	47.9 ± 3.4
Flat RL	SDAR	73.7 ± 5.2	69.5 ± 4.9	70.6 ± 1.2	67.7 ± 3.9
Hierarchical RL	GLIDER	72.2 ± 4.1	68.1 ± 3.8	57.6 ± 5.0	52.6 ± 6.0
Hierarchical RL	GLIDER + STAC-D (ours)	76.7 ± 4.0	73.1 ± 4.0	62.0 ± 5.0	58.3 ± 5.0
Hierarchical RL	GLIDER + STAC-P (ours)	76.9 ± 3.1	74.3 ± 3.2	62.1 ± 4.4	57.4 ± 4.3
Hierarchical RL	HiPER	92.5 ± 3.7	85.2 ± 3.6	71.4 ± 5.0	66.5 ± 5.1
Hierarchical RL	HiPER + STAC-D (ours)	$\mathbf{96.4 \pm 1.2}$	89.9 ± 2.1	$\mathbf{78.2 \pm 5.6}$	74.5 ± 5.1
Hierarchical RL	HiPER + STAC-P (ours)	96.1 ± 0.8	$\mathbf{92.4 \pm 3.0}$	77.4 ± 2.2	$\mathbf{76.2 \pm 3.9}$
Llama-3.2-1B-Instruct					
—	Base (Plan–Execute)	1.8 ± 1.3		0.3 ± 0.4	
—	Base + ReAct	5.1 ± 2.0		0.9 ± 0.8	
Flat RL	GRPO	7.8 ± 3.5	1.6 ± 1.5	1.6 ± 0.5	0.0 ± 0.0
Flat RL	GiGPO	12.5 ± 5.5	7.0 ± 2.5	1.6 ± 0.3	0.0 ± 0.0
Flat RL	PPO	7.0 ± 3.0	1.3 ± 1.2	1.2 ± 0.5	0.0 ± 0.0
Flat RL	SDAR	7.5 ± 1.7	5.2 ± 1.2	0.8 ± 0.0	0.0 ± 0.0
Hierarchical RL	GLIDER	21.8 ± 5.4	16.2 ± 5.4	3.1 ± 2.6	1.6 ± 1.3
Hierarchical RL	GLIDER + STAC-D (ours)	28.3 ± 6.1	23.9 ± 4.3	7.5 ± 3.8	4.7 ± 2.5
Hierarchical RL	GLIDER + STAC-P (ours)	27.2 ± 4.1	23.0 ± 3.5	7.0 ± 2.0	4.5 ± 2.2
Hierarchical RL	HiPER	28.6 ± 5.5	17.2 ± 4.4	1.8 ± 0.4	0.0 ± 0.0
Hierarchical RL	HiPER + STAC-D (ours)	$\mathbf{52.1 \pm 6.0}$	$\mathbf{40.4 \pm 5.0}$	$\mathbf{17.6 \pm 3.9}$	$\mathbf{15.3 \pm 4.6}$
Hierarchical RL	HiPER + STAC-P (ours)	39.8 ± 3.5	34.4 ± 4.1	15.4 ± 3.7	14.4 ± 4.4
Closed-Source Prompt-Engineering Models					
PE	GPT-4o	48.0		23.7	
PE	Gemini-2.5-Pro	60.3		35.9	
PE	GPT-5	65.4		33.7	
PE	Claude Sonnet 4.5	82.5		38.6	

5 EXPERIMENTS

Environments and models. We evaluate on two text-based interactive benchmarks with sparse success rewards. ALFWorld (Shridhar et al., 2021) contains long-horizon household tasks whose multi-phase categories require several coordinated actions per objective, and WebShop (Yao et al., 2022) requires navigating a simulated e-commerce site to satisfy a natural-language shopping instruction. We evaluate three open-weight backbones: Qwen3-0.6B (Yang et al., 2025), Qwen2.5-1.5B-Instruct (Qwen et al., 2025), and Llama-3.2-1B-Instruct (Grattafiori et al., 2024).

Training. The HiPER runs use 128 independent environments per update, 150 updates, at most 50 environment turns, training temperature 1.0, and validation temperature 0.4; actor and critic learning rates are 10^{-6} and 10^{-5} and the KL coefficient is 0.01. We evaluate 128 held-out episodes every five updates, and the base checkpoint before training. GLIDER follows its original behavior-cloning, AWAC, and offline-to-online training paths; its boundary adapter is described in Appendix D. Appendix A gives the shared reporting protocol and the HiPER configuration.

Baselines. We first evaluate the untrained backbone under the Plan–Execute and ReAct (Yao et al., 2023) prompting protocols. Our flat-RL comparisons comprise PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), GiGPO (Feng et al., 2025), and SDAR (Lu et al., 2026); these methods optimize a single-timescale policy. Our HRL baselines are HiPER (Peng et al., 2026) and GLIDER (Hu et al., 2025). For each HRL baseline, we compare the underlying method with both its STAC-D and STAC-P instantiations. We also report prompt-engineering (PE) results for the following closed-source models: GPT-4o (Hurst et al., 2024), Gemini-2.5-Pro (Comanici et al., 2025), GPT-5 (Singh et al., 2025), and Claude Sonnet 4.5 (Anthropic, 2025).

5.1 Results

Main results. Table 1 summarizes performance across three backbones, two benchmarks, and both HiPER and GLIDER. With Qwen3-0.6B, HiPER+STAC-P improves over HiPER by 12.0 points on ALFWorld and 12.8 points on WebShop at the best checkpoint; GLIDER+STAC-P gains 12.2 and 9.4 points, respectively. With Qwen2.5-1.5B-Instruct, the strongest HiPER+STAC results reach 96.4% on ALFWorld and 78.2% on WebShop, although the WebShop block combines published baselines with a separate local diagnostic. With Llama-3.2-1B-Instruct, HiPER+STAC-D yields the largest gains, improving over HiPER by 23.5 points on ALFWorld and 15.8 points on WebShop. Appendix F provides the complete statistical and task-level breakdowns.

Constraint violation diagnostics. Fig. 1 tracks both constraint surrogates throughout training. Under HiPER, premature replanning stays high, ending at 0.553 on ALFWorld and 0.597 on WebShop, so more than half of all turns replace a single-turn subgoal, while stale persistence grows to 0.147 and 0.110. STAC-D (full) starts higher still yet drives them to 0.073 and 0.151, reductions of 7.6× and 4.0×, while also ending lower on stale persistence at 0.098 and 0.039; mean segment lengths settle at 4.02 and 2.91 turns. Improving both at once shows the boundary terms do not merely trade one violation for the other. Because these runs use fixed penalty coefficients, the converged values are observed violation rates, not budgets the optimizer was asked to meet.

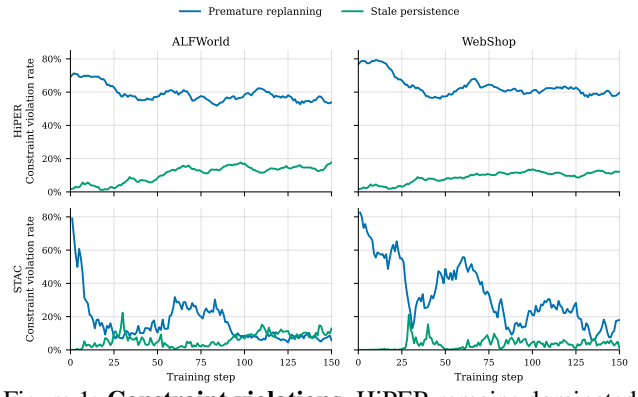


Figure 1: **Constraint violations.** HiPER remains dominated by premature replanning, whereas STAC-D reduces both observed violation rates (three-seed means).

Completion-estimator diagnostics. We next evaluate whether STAC-P learns the semantic completion signal used by its boundary costs. Table 2 reports label coverage, probabilistic error (Brier), complete-versus-incomplete discrimination (AUROC), and the mean predicted completion probability on KEEP and SWITCH decisions. Appendix F.2 gives the full definitions and the complete diagnostics. As seen in Table 2, the estimator is both accurate and behaviorally discriminative. On ALFWorld, it reaches an AUROC of 0.921, while its mean completion probability rises from 0.182 on KEEP decisions to 0.817 on SWITCH decisions; WebShop shows the same separation, from 0.214 to 0.754. Together with the low Brier scores, these results indicate that STAC-P learns a meaningful semantic boundary signal rather than merely benefiting from an auxiliary prediction loss.

Table 2: **STAC-P completion-estimator diagnostics (Qwen3-0.6B).** Values are mean ± SD across seeds; arrows indicate the preferred direction.

Environment	Coverage (%) ↑	Brier ↓	AUROC ↑	$p_{\text{done}} \mid \text{KEEP}$ ↓	$p_{\text{done}} \mid \text{SWITCH}$ ↑
ALFWorld	62.8 ± 2.6	0.079 ± 0.008	0.921 ± 0.014	0.182 ± 0.021	0.817 ± 0.026
WebShop	41.7 ± 3.8	0.096 ± 0.011	0.883 ± 0.021	0.214 ± 0.027	0.754 ± 0.034

5.2 Ablation Studies

Table 3: **ALFWorld ablation (Qwen3-0.6B).** Checkmarks denote active short- and stale-boundary costs; values are mean ± SD over three seeds. Premature and stale violation rates are common duration-based behavioral diagnostics evaluated for all variants.

Method	Short	Stale	Best ↑	Final ↑	Prem. ↓	Stale ↓
HiPER	×	×	83.3 ± 3.3	81.5 ± 3.6	0.553 ± 0.041	0.147 ± 0.026
STAC-D (short)	✓	×	88.2 ± 3.8	86.0 ± 3.9	0.085 ± 0.021	0.185 ± 0.031
STAC-D (stale)	×	✓	85.1 ± 3.5	83.4 ± 3.7	0.565 ± 0.039	$\mathbf{0.045 \pm 0.013}$
STAC-D (full)	✓	✓	91.4 ± 3.6	89.3 ± 3.5	$\mathbf{0.073 \pm 0.018}$	0.098 ± 0.022
STAC-P (short)	✓	×	92.6 ± 2.5	90.7 ± 2.8	0.112 ± 0.024	0.164 ± 0.028
STAC-P (stale)	×	✓	87.3 ± 3.1	85.5 ± 3.3	0.521 ± 0.037	0.052 ± 0.014
STAC-P (full)	✓	✓	$\mathbf{95.3 \pm 1.6}$	$\mathbf{93.2 \pm 2.4}$	0.101 ± 0.020	0.061 ± 0.016

Cost ablation. On ALFWorld, we disable one cost at a time within each STAC instantiation while holding all other settings fixed (Table 3). The short cost sharply reduces premature replanning and captures most of the gain. The STAC-D stale-only condition improves best success only from 83.3 to 85.1, but adding it to the short cost raises success from 88.2 to 91.4, so its main value appears in combination. The same complementary pattern holds for STAC-P, whose full two-sided objective reaches 95.3% best-checkpoint success.

Appendix F.3 reports the remaining ablations: the STAC-D coefficient sweep on ALFWorld, the component and coefficient ablations on WebShop, and the STAC-P mechanism study covering completion-only supervision, warm-up, stop-gradient, and oracle completion costs.

6 Conclusion

Explicit subgoal syntax alone does not determine how stable the learned temporal abstraction will be. We identified *temporal abstraction instability* and proposed STAC, a decision-local framework that penalizes premature replanning and stale persistence while leaving task rewards and non-boundary objectives unchanged. STAC-D uses deterministic duration proxies, whereas STAC-P learns a semantic subgoal-completion probability from environment-grounded supervision. Integrating these costs into HiPER's explicit boundary advantage and GLIDER's implicit Boolean completion policy shows that the framework is not tied to a particular hierarchy or training objective. At the best checkpoint with Qwen3-0.6B, HiPER+STAC-P improves over HiPER by 12.0 points on ALFWorld and 12.8 points on WebShop, while GLIDER+STAC-P improves over GLIDER by 12.2 and 9.4 points, respectively. With Llama-3.2-1B-Instruct, HiPER+STAC-D yields gains of 23.5 points on ALFWorld and 15.8 points on WebShop; the controlled Qwen2.5-1.5B-Instruct ALFWorld replication adds 3.9 points. The reported experiments use validation-selected fixed penalty coefficients and therefore do not guarantee user-specified violation budgets; moreover, STAC-P supervision is limited to subgoals with reliable environment-grounded completion predicates. Adaptive dual optimization and broader completion supervision are important directions for future work.

AI USE STATEMENT

Generative AI tools assisted with language editing and manuscript presentation. The authors reviewed all AI-assisted text, equations, citations, and technical claims, independently verified the final content, and remain fully responsible for the methodology, experiments, results, and conclusions presented in this work.

ETHICS STATEMENT

Our experiments use publicly released benchmark environments and artifacts from ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022). We followed the access and usage conditions specified by the benchmark providers and standard research-community practices. The study did not collect private user data or involve human subjects, and no direct personal identifiers were accessed or processed.

ACKNOWLEDGMENTS

The experimental data and environments used in this work were obtained from the public ALFWorld (Shridhar et al., 2021) and WebShop (Yao et al., 2022) benchmark releases. We thank the benchmark authors and the developers of the publicly available HiPER and GLIDER systems for resources that enabled the comparisons in this study.

REPRODUCIBILITY STATEMENT

Appendix A documents the method implementation, benchmark and model configurations, training hyperparameters, checkpoint-selection procedure, statistical protocol, and recommended behavioral diagnostics. The anonymized repository at `https://anonymous.4open.science/r/STAC-CB18` contains the source code, experiment configurations, and launch scripts needed to reproduce the reported runs. All benchmark datasets and environments used in our experiments are publicly available.

REFERENCES

Anthropic. Claude sonnet 4.5 system card, September 2025. URL `https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf`. Accessed: 2025-12-12.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Oriol Corcoll and Raul Vicente. Disentangling controlled effects for hierarchical reinforcement learning. In Bernhard Schölkopf, Caroline Uhler, and Kun Zhang (eds.), *Proceedings of the First Conference on Causal Learning and Reasoning*, volume 177 of *Proceedings of Machine Learning Research*, pp. 178–200. PMLR, 2022. URL `https://proceedings.mlr.press/v177/corcoll22a.html`.

Juncheng Diao, Zhicong Lu, Peiguang Li, Yongwei Zhou, Changyuan Tian, Qingbin Li, Rongxiang Weng, Jingang Wang, and Xunliang Cai. HIPIF: Hierarchical planning and information folding for long-horizon LLM agent learning. *arXiv preprint arXiv:2606.10507*, 2026. URL `https://arxiv.org/abs/2606.10507`.

T. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

L. Feng, Z. Xue, T. Liu, and B. An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Nico Gürtler, Dieter Büchler, and Georg Martius. Hierarchical reinforcement learning with timed subgoals. In *Advances in Neural Information Processing Systems*, volume 34, pp. 21732–21743. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/b59c21a078fde074a6750e91ed19fb21-Abstract.html.

Zican Hu, Wei Liu, Xiaoye Qu, Xiangyu Yue, Chunlin Chen, Zhi Wang, and Yu Cheng. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24570–24590. PMLR, 2025. URL https://proceedings.mlr.press/v267/hu25q.html.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Hongbo Jin, Rongpeng Zhu, Jiayu Ding, Guibo Luo, and Ge Li. HiMAC: Hierarchical macro-micro learning for long-horizon LLM agents. *arXiv preprint arXiv:2603.00977*, 2026. URL https://arxiv.org/abs/2603.00977.

Seungjae Lee, Jigang Kim, Inkyu Jang, and H. Jin Kim. DHRL: A graph-based approach for long-horizon and sparse hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://openreview.net/forum?id=5JQqvQ1ujSv. NeurIPS 2022 Oral.

Siyuan Li, Rui Wang, Minxue Tang, and Chongjie Zhang. Hierarchical reinforcement learning with advantage-based auxiliary rewards. In *Advances in Neural Information Processing Systems*, volume 32, pp. 1407–1417. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/81e74d678581a3bb7a720b019f4f1a93-Abstract.html.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, et al. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Sangeun Park and Minhae Kwon. $\texttt{Multi}^2$: Hierarchical multi-agent decision-making with LLM-based agents in interactive environments. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=H0tMEp0ZmO.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical plan–execute RL for multi-turn LLM agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=RJWQcgd3hX.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

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

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

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35: 20744–20757, 2022.

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.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21579–21590. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/f5f3b8d720f34ebebceb7765e447268b-Abstract.html.

Shuai Zhen, Yanhua Yu, Ruopei Guo, Nan Cheng, and Yang Deng. Hierarchical reinforcement learning with augmented step-level transitions for llm agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7035–7053, 2026.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

A Implementation, Experimental Details, and Broader Impact

Standalone integration. STAC requires an identifiable binary boundary decision, estimates of the two constraint rates, and an existing boundary update target. For an on-policy method such as HiPER, this target is the normalized boundary advantage and the adapter is inserted after its normalization; for GLIDER, it is the induced probability of the Boolean completion decisions in Appendix D. The hierarchical policy need not use the subgoal text, value heads, return estimator, or policy factorization used by either of our examples. Our experimental structured-output adapter additionally supplies the parsed boundary-token mask described in Appendix H. An implementation should expose the constraint-cost functions g_i and the multipliers μ_i, with the budgets ρ_i and a dual optimizer needed only for the adaptive variant; setting both multipliers to zero is the behavior-preserving default for constrained boundary credit.

Artifacts and statistical protocol. Our experiment configuration fixes all non-treatment settings across arms and stores the complete resolved trainer configuration, per-update metrics, validation generations, and sampled training episodes. We recommend reporting switch rate, segment-length distribution, both constraint violation rates, and valid-format rate alongside task return. For the main result, we follow HiPER and select each seed's highest-overall-validation checkpoint; where source-backed category artifacts are available, we report their task-level breakdowns separately. Appendix G separately reports step-150 and late-window summaries to expose checkpoint-selection sensitivity. Figure 5 reports the arithmetic mean across the three seed-level validation trajectories without uncertainty intervals. The local Qwen2.5-1.5B-Instruct WebShop STAC-D diagnostic aggregates three independent seeds. Table 1 combines that diagnostic and the separately run local STAC-P result with the HiPER paper's published baselines, so no local Qwen2.5 WebShop HiPER aggregate or rerun enters a controlled comparison.

Training configuration. For HiPER, the released ALFWorld implementation enables the truncation bootstrap in equation 24; the WebShop implementation disables it. ALFWorld uses a 2048-token prompt limit; WebShop uses 4096 tokens because its released observation template is longer. The HiPER runs retain the environment reward and HiPER's -0.1 malformed-output reward term. Unless explicitly identified as published, HiPER baseline values come from our matched local runs. The Base (Plan–Execute) rows evaluate the untrained HiPER checkpoint under the same Plan–Execute prompt, temperature, episode limit, and splits; the Base + ReAct rows evaluate the same checkpoint under ReAct prompting.

Fixed-multiplier experimental instantiation. For all reported experiments we use the fixed-multiplier penalty realization

$$\mu_{\text{short}} = \mu_{\text{stale}} = 0.1. \tag{21}$$

For a valid HiPER boundary, $\widehat{A}_t^{b,\text{STAC}} = \overline{A}_t^b - 0.1(c_t^{\text{short}} + c_t^{\text{stale}})$. GLIDER applies the same fixed coefficients to its agent-specific expected Boolean losses in Appendix D. The multipliers are held fixed throughout training rather than updated by equation 9. The value 0.1 was selected by ALFWorld validation success and then reused unchanged in the other reported settings. These coefficients are tuned hyperparameters, not dual variables recovered from binding constraints, and the observed violation rates are diagnostics rather than target budgets.

A.1 Environment-Grounded Completion Oracles

This subsection specifies the benchmark-specific supervision used by the STAC-P completion predictor. The oracles read environment state only during training; their outputs are neither policy inputs nor task rewards.

ALFWorld completion rules. ALFWorld exposes structured TextWorld facts after each transition. The oracle recognizes the supported single-predicate subgoal forms in Table 4.

Unsupported subgoals ($m_t = 0$). The ALFWorld oracle does not parse navigation subgoals (for example, "go to countertop 1"), search or epistemic subgoals ("find a newspaper in drawer 1"), observation subgoals ("inventory"), or open, close, use, and retrieve requests. It also excludes

Table 4: STAC-P completion rules for supported ALFWorld subgoals.

Subgoal form	Completion predicate
`put/place/move` X `in/onto` Y	`inreceptacle`(X, Y)
`pick up/take/grab` X	`holds`$(\texttt{agent}, X)$
`clean` X	`isclean`(X)
`heat` X	`ishot`(X)
`cool` X	`iscool`(X)

Table 5: WebShop STAC-P labels for the product-selection subgoal `click[B078WSND96]`. The session snapshot is used only as an auxiliary training target.

Step	Active subgoal	Executed action	Post-action session state	STAC-P label
1	`click[B078WSND96]`	`click[next >]` or another non-product action	`selected_asin = ""`	$m_t = 1,\ y_t = 0$
2	`click[B078WSND96]`	another non-product action	`selected_asin = ""`	$m_t = 1,\ y_t = 0$
3	`click[B078WSND96]`	`click[B078WSND96]`	`selected_asin = "B078WSND96"`	$m_t = 1,\ y_t = 1$

malformed, empty, and compound subgoals such as "heat a mug and put it in cabinet." These forms either lack a verified completion predicate or contain multiple predicates that cannot be represented by one binary label. When $m_t = 0$, STAC-P applies neither completion BCE supervision nor a completion-based KEEP/SWITCH cost, leaving the original boundary objective unchanged.

Worked ALFWorld example. Suppose the active subgoal is "put the apple in the fridge." After the agent places the apple, ALFWorld's `env.step` returns a post-action information dictionary whose `facts` field includes

$$\texttt{inreceptacle apple 1 fridge 2.} \tag{22}$$

The instance identifiers distinguish simulator entities but do not change the predicate's meaning. The oracle parses the subgoal as a placement request, removes the identifiers from the fact, and verifies the `inreceptacle` relation and both entity types. The check therefore gives $y_t = 1$ at the aligned boundary event. If the fact is absent, the same supported subgoal receives $y_t = 0$; an unsupported subgoal such as "find the apple" receives no label rather than an assumed zero.

Worked WebShop example. WebShop does not expose a generic subgoal-complete flag in its public `env.step` return. For a small set of action-shaped subgoals, STAC-P instead reads a sanitized, training-only snapshot of persistent session state after each action. For example, `click[B078WSND96]` is complete exactly when the post-action `selected_asin` equals `B078WSND96`. Prerequisite search and navigation steps may be needed to expose the product, but they do not complete this product-selection subgoal. Table 5 shows the resulting labels.

The WebShop oracle analogously supports exact `search[...]` actions and selected product-option values. It assigns $m_t = 0$ to broader search, browsing, comparison, description, review, and terminal-purchase subgoals for which the simulator does not expose a trustworthy completion predicate.

A.2 THE THRESHOLDS FOR d_t

The duration thresholds are derived as described below and are not selected by validation. The multiplier μ, by contrast, is selected by ALFWorld validation success in Table 7; Table 9 reports the corresponding sensitivity after that value was reused on WebShop.

The lower threshold is forced by non-degeneracy. A subgoal that terminates after a single turn in every state induces the same action distribution as a policy with no subgoal layer at all: the option reduces to a primitive action and the hierarchy is vacuous (Sutton et al., 1999). The constraint

therefore encodes *non-degeneracy of the abstraction*, and $d_{\min} = 2$ is the only admissible value; it is not a choice among candidates.

The upper threshold follows a horizon fair-share bound. An episode admits at most T environment turns and a task requires K subgoals. If every subgoal persisted for more than T/K turns, the remaining subgoals could not be completed within the horizon, so T/K is the per-subgoal fair share and persistence beyond it consumes budget the remaining subgoals require. This gives the bound

$$d_{\max} \;\le\; \left\lfloor T/\bar{K} \right\rfloor, \tag{23}$$

where $\bar{K}$ is the mean number of subgoals per task, a property of the environment rather than of the learner. We estimate $\bar{K}$ from solved episodes of each benchmark.

Compute. Every run trains on a single node with eight NVIDIA H100 80 GB GPUs, using vLLM for rollout generation and co-locating the actor and critic on that node. A 150-update run takes roughly 12–24 hours depending on benchmark and backbone, WebShop being slower than ALFWorld because of its longer observation template.

Broader impact. Temporal abstraction can make language agents more efficient and interpretable by exposing the subgoal that persists across actions. However, the chosen constraint-cost functions encode a behavioral prior. Costs that are too restrictive can suppress legitimate rapid replanning or force stale subgoals, especially under distribution shift. For deployment, boundary credit should be monitored, capped, and disabled when the structured interface is not semantically validated. It does not replace task-level safety constraints.

B HiPER Advantage Estimator

HiPER uses one shared critic backbone with two scalar output heads (Peng et al., 2026). The high-level head $V_\phi^H(s_t)$ estimates the return from selecting a new subgoal at a segment boundary, while the low-level head $V_\phi^L(s_t, z_t)$ estimates the return from continuing execution under the active subgoal. There is no separately learned boundary-value head. The two heads produce three actor advantages because the boundary advantage is derived from their value difference.

Let $0 = b_0 < b_1 < \cdots < b_K = T$ be the segment boundaries. Segment k contains turns $[b_k, b_{k+1}-1]$, has duration $D_k = b_{k+1} - b_k$, and uses subgoal z_k. Write $\mathrm{sg}[\cdot]$ for stop-gradient and define

$$B_{k+1}^H = \begin{cases} V_\phi^H(s_{b_{k+1}}), & b_{k+1} < T, \\ \mathrm{sg}[V_\phi^H(s_{b_k})], & \text{bootstrapped truncation}, \\ 0, & \text{otherwise}. \end{cases} \tag{24}$$

The middle case is the released HiPER implementation's optional treatment of a trajectory cut off by the turn limit. The discounted segment reward, high-level residual, and subgoal advantage are

$$\begin{aligned} \widetilde{r}_k &= \sum_{j=0}^{D_k-1} \gamma^j r_{b_k+j}, \\ \delta_k^z &= \widetilde{r}_k + \gamma^{D_k} B_{k+1}^H - V_\phi^H(s_{b_k}), \\ \widehat{A}_{b_k}^z &= \delta_k^z + \gamma^{D_k} \lambda_H \widehat{A}_{b_{k+1}}^z. \end{aligned} \tag{25}$$

$\widehat{A}_{b_k}^z$ is defined at segment starts and is assigned to the newly generated subgoal tokens.

Within a segment, define

$$\widetilde{V}_{t+1} = \begin{cases} V_\phi^L(s_{t+1}, z_k), & t < b_{k+1} - 1, \\ B_{k+1}^H, & t = b_{k+1} - 1. \end{cases} \tag{26}$$

The low-level residual and action advantage are

$$\begin{aligned} \delta_t^a &= r_t + \gamma \widetilde{V}_{t+1} - V_\phi^L(s_t, z_k), \\ \widehat{A}_t^a &= \sum_{\ell=t}^{b_{k+1}-1} (\gamma\lambda_L)^{\ell-t} \delta_\ell^a, \qquad t \in [b_k, b_{k+1} - 1]. \end{aligned} \tag{27}$$

The sum stops at the segment boundary, where the final residual bootstraps to the high-level head. $\widehat{A}^a_t$ is assigned to action tokens.

For boundary credit, define the estimated gain from replacing the previous subgoal,

$$\Delta^b_t = V^H_\phi(s_t) - V^L_\phi(s_t, z_{t-1}),$$
$$\widehat{A}^b_t = (y_t - \beta_t)\Delta^b_t, \quad y_t = \mathbb{I}[q_t = \text{SWITCH}], \quad \beta_t = \pi^b_\theta(\text{SWITCH} \mid s_t, z_{t-1}). \tag{28}$$

This is a centered policy-gradient estimator for the realized binary decision and is assigned to the boundary tokens at every turn. Before the actor update, the advantage streams are independently standardized. The boundary stream gives the normalized boundary advantage $\overline{A}^b_t$ of equation 20, whose batch $\mathcal{B}$ contains exactly those turns with a valid boundary-token mask. This $\overline{A}^b_t$ is the sole native advantage consumed by STAC. The two critic targets are $y^H_k = \widetilde{r}_k + \gamma^{D_k} \operatorname{sg}[B^H_{k+1}]$ and $y^L_t = r_t + \gamma \operatorname{sg}[\widetilde{V}_{t+1}]$. The critic regresses $V^H_\phi(s_{b_k})$ to y^H_k and $V^L_\phi(s_t, z_k)$ to y^L_t. STAC changes neither target.

C Optimization Details

Locality. Ng et al. (1999) define potential-based reward shaping by replacing the trajectory reward r_t with $r'_t = r_t + \gamma\Phi(s_{t+1}) - \Phi(s_t)$. Because bootstrapped estimators aggregate discounted rewards through the critic, such a change can alter critic targets and advantages at earlier turns. STAC instead adds the sampled Lagrangian cost terms after native advantage estimation, only to the boundary span at turn t. Thus it changes neither the reward stream, critic targets, nor another turn's advantage. Moreover, the mutually exclusive boundary decisions and bounded costs imply that the valid-decision perturbation is bounded by $\max(\mu_{\text{short}}, \mu_{\text{stale}})$.

One HiPER update. Each HiPER update (1) rolls out the hierarchical policy, (2) tracks d_t and computes equation 5, (3) computes and normalizes the native advantage streams, (4) forms equation 11, and (5) applies the unchanged on-policy actor and critic updates. Step (6) applies only to the adaptive primal–dual variant, which additionally updates the multipliers using equation 9; the reported runs omit it and hold $\boldsymbol{\mu}$ fixed, so their violation rates are observed diagnostics rather than user-specified targets. GLIDER instead applies the agent-specific Boolean losses in Appendix D. For both instantiations, we log switch rate, segment-length quantiles, one-turn-segment rate, and the applicable constraint diagnostics, together with the multipliers whenever they are adapted.

Why the fixed-multiplier form remains a valid relaxation. For an exact maximizer of the fixed-penalty surrogate, define

$$\theta_{\boldsymbol{\mu},k} \in \arg\max_\theta \left[\mathcal{J}^{\text{task}}_k(\theta) - \sum_i \mu_i \mathcal{C}^k_i(\theta) \right], \qquad \rho^{\boldsymbol{\mu}}_{i,k} = \mathcal{C}^k_i(\theta_{\boldsymbol{\mu},k}). \tag{29}$$

Proposition 1. *Fix an update k and strictly positive multipliers $\mu_i > 0$. If $\theta_{\boldsymbol{\mu},k}$ is an exact maximizer of the fixed-penalty surrogate in equation 29, then it solves*

$$\max_\theta \ \mathcal{J}^{\text{task}}_k(\theta) \quad \textit{subject to} \quad \mathcal{C}^k_i(\theta) \le \mathcal{C}^k_i(\theta_{\boldsymbol{\mu},k}) \quad \textit{for every } i.$$

Proof. Suppose a feasible θ' had $\mathcal{J}^{\text{task}}_k(\theta') > \mathcal{J}^{\text{task}}_k(\theta_{\boldsymbol{\mu},k})$. Feasibility gives $\mathcal{C}^k_i(\theta') \le \mathcal{C}^k_i(\theta_{\boldsymbol{\mu},k})$ for every i. Because every multiplier is positive,

$$\mathcal{J}^{\text{task}}_k(\theta') - \sum_i \mu_i \mathcal{C}^k_i(\theta') > \mathcal{J}^{\text{task}}_k(\theta_{\boldsymbol{\mu},k}) - \sum_i \mu_i \mathcal{C}^k_i(\theta_{\boldsymbol{\mu},k}),$$

contradicting the optimality of $\theta_{\boldsymbol{\mu},k}$. □

Because $\rho^{\boldsymbol{\mu}}_{i,k}$ is measured after optimization rather than specified beforehand, this proposition does not certify satisfaction of any user-chosen violation budget.

D STAC INTEGRATION WITH GLIDER

GLIDER (Hu et al., 2025) is a hierarchical planner–controller agent whose high-level policy generates a textual subtask and whose low-level policy executes primitive environment actions conditioned on that subtask. Its low-level output has the structured form

`<primitive action>; <true|false>.`

The Boolean suffix is GLIDER's completion judgment: `false` retains the active subtask for another primitive action, whereas `true` terminates the subtask and returns control to the high-level planner. We therefore identify

$$q_t = \begin{cases} \text{KEEP}, & \text{if the completion suffix is } \texttt{false}, \\ \text{SWITCH}, & \text{if the completion suffix is } \texttt{true}. \end{cases}$$

This exposes a temporal boundary in GLIDER without introducing a termination network or modifying its high-level planner.

Implicit completion policy. Unlike HiPER, GLIDER embeds the boundary decision in an autoregressive low-level response rather than exposing a separate boundary-policy factor. Let h_t be the low-level context through the observed primitive action and its delimiter, immediately before the Boolean suffix. For each candidate completion $b \in \{\texttt{false}, \texttt{true}\}$, we hold h_t fixed and score only the token positions M_b that realize the Boolean literal:

$$\ell_\theta(b \mid h_t) = \sum_{j \in M_b} \log \pi_\theta(w_j \mid h_t, w_{<j}).$$

The two candidate likelihoods induce the normalized implicit boundary policy

$$\pi^b_\theta(q_t \mid h_t) = \frac{\exp\{\ell_\theta(b(q_t) \mid h_t)\}}{\sum_{b \in \{\texttt{false},\texttt{true}\}} \exp\{\ell_\theta(b \mid h_t)\}},$$

where $b(\text{KEEP}) = \texttt{false}$ and $b(\text{SWITCH}) = \texttt{true}$. Thus, the primitive action, prompt, state history, and prior low-level responses are identical for the two counterfactual completions.

Duration constraints. We use the pre-action, one-based age d_t of GLIDER's active subtask: the first primitive action after a newly generated subtask has $d_t = 1$; the age increases after each retained action and resets to one after a SWITCH decision. The short and stale indicators are

$$g_{\text{short}}(d_t) = \mathbb{I}[d_t < 2], \qquad g_{\text{stale}}(d_t) = \mathbb{I}[d_t \geq d_{\max}].$$

Consequently, a one-action subtask incurs a short-duration violation only if GLIDER emits `true`, while an overlong subtask incurs a stale-duration violation only if it emits `false`. For valid Boolean completions $\mathcal{V}$, the GLIDER STAC term is the enumerated expected constraint cost:

$$\mathcal{L}_{\text{GLIDER-STAC}} = \frac{1}{|\mathcal{V}|} \sum_{t \in \mathcal{V}} \left[\mu_{\text{short}} g_{\text{short}}(d_t) \pi^b_\theta(\text{SWITCH} \mid h_t) + \mu_{\text{stale}} g_{\text{stale}}(d_t) \pi^b_\theta(\text{KEEP} \mid h_t) \right].$$

This is the probability-space counterpart of STAC's decision-local costs: it penalizes expected premature SWITCH decisions and expected stale KEEP decisions under GLIDER's completion policy.

Probabilistic completion constraints. STAC-P replaces the deterministic age costs above with a learned forecast of whether the active subtask will be achieved by the primitive action at turn t. Let z_{t-1} denote that active subtask, following the boundary-decision indexing used in the main text. We attach a lightweight sigmoid head to the low-level decoder representation at h_t, immediately before the Boolean suffix:

$$p^{\text{done}}_t = f_\phi(h_t, z_{t-1}) = P_\phi(z_{t-1} \text{ is achieved after the primitive action at } t \mid h_t, z_{t-1}).$$

Because GLIDER generates the primitive action and Boolean suffix in one autoregressive response, h_t necessarily precedes execution of that primitive action. The predictor therefore forecasts the not-yet-realized transition. After the action is executed, we supervise it post hoc with

$$y_t = \mathcal{O}(s_{t+1}, z_{t-1}),$$

using the same training-only fact and session-state oracle as in equation 15 and the subsequent WebShop construction. Here $m_t = 1$ only when that oracle supports the active GLIDER subtask and its completion predicate; unsupported or ambiguous subtasks have $m_t = 0$. The completion head is trained with the masked binary cross-entropy in equation 16, restricted to these GLIDER decisions, so post-action facts are never policy inputs.

Let $\widetilde{p}_t^{\text{done}} = \text{stopgrad}(p_t^{\text{done}})$. For the valid Boolean decisions $\mathcal{V}$, the STAC-P Boolean-only constraint loss is

$$\mathcal{L}_{\text{GLIDER-STAC-P}} = \frac{1}{\sum_{t\in\mathcal{V}} m_t} \sum_{t\in\mathcal{V}} m_t \left[\mu_{\text{short}} \left(1 - \widetilde{p}_t^{\text{done}}\right) \pi_\theta^b(\text{SWITCH} \mid h_t) + \mu_{\text{stale}} \widetilde{p}_t^{\text{done}} \pi_\theta^b(\text{KEEP} \mid h_t)\right], \tag{30}$$

when $\sum_{t\in\mathcal{V}} m_t > 0$; otherwise this auxiliary loss is omitted for the batch. Thus, the policy is penalized for switching when the active subtask is predicted incomplete and for keeping it when predicted complete. The stop-gradient prevents this Boolean-policy loss from changing the completion predictor instead of the boundary policy. STAC-P and STAC-D are alternative instantiations: the former replaces the duration loss rather than adding to it. As in the main STAC-P experiments, we train the completion head for the first ten GLIDER optimization updates while leaving equation 30 inactive, then activate the semantic Boolean costs.

Decision-local implementation. The released GLIDER low-level behavior-cloning loss scores the complete action-plus-completion response. Both GLIDER instantiations retain that loss unchanged and add their respective constraint loss through a separate Boolean-only backward pass. The adapter constructs `false` and `true` counterfactuals from each response eligible for the respective constraint, preserving the primitive action and scoring only the Boolean suffix. It maps the Boolean character span to tokenizer positions, so multi-token literals are handled exactly; prompt, observation, delimiter, primitive-action, and EOS tokens receive no direct STAC credit. For STAC-P, the masked completion BCE is a separate auxiliary update of f_ϕ at the Boolean position. Malformed responses with no valid completion suffix, and responses with $m_t = 0$ under STAC-P, receive no STAC constraint gradient rather than falling back to action-token supervision.

The same decision-local adapter is applied to GLIDER's behavior-cloning, AWAC, and offline-to-online training paths. It leaves the original primitive-action objective, high-level update, critic targets, environment rewards, and replay semantics unchanged. In the offline-to-online path, it additionally records each sampled completion decision and subtask age, marking horizon- or environment-forced segment endings separately from policy-selected SWITCH decisions.

Diagnostics. In addition to task success, we measure completion-format validity, switch rate, the mean/median/90th-percentile segment length, and the fraction of one-action segments. For STAC-D, the observed duration-violation rates are

$$\widehat{C}_{\text{short}} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[q_t = \text{SWITCH}] g_{\text{short}}(d_t), \qquad \widehat{C}_{\text{stale}} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}[q_t = \text{KEEP}] g_{\text{stale}}(d_t).$$

For STAC-P, we additionally report completion-label coverage and prevalence, masked completion loss, Brier score, AUROC, calibration, and the predicted completion probability by KEEP/SWITCH decision. These diagnostics distinguish policy-selected boundaries from censored segments and test whether an improvement in task success is accompanied by fewer premature terminations without excessive subtask persistence.

E Related Hierarchical Language-Agent Methods

Recent hierarchical language-agent methods differ in what constitutes a level of the hierarchy, how those levels are trained, and where credit is assigned. This section summarizes the methods cited in the introduction from their primary papers and official implementations.

ArCHer. Zhou et al. (2024) introduce ArCHer, a two-level actor–critic framework for multi-turn language-model interaction. Its high level applies off-policy temporal-difference learning to utterance- or turn-level transitions, thereby propagating reward across an entire dialogue, while its low level

applies token-level policy optimization within each utterance using the high-level value function for cross-turn credit. This separation allows the high and low levels to run RL algorithms with different update rules and substantially improves sample efficiency relative to fully on-policy multi-turn optimization.

GLIDER. Hu et al. (2025) introduce GLIDER, which uses a two-level planner–controller hierarchy for grounding language models in sequential environments. The high-level policy generates abstract chain-of-thought plans or subtasks, while a parameter-efficient low-level controller executes environment actions conditioned on the current plan. Training proceeds from behavior-cloning initialization to offline hierarchical RL with an advantage-weighted actor–critic objective, followed by offline-to-online adaptation; the paper evaluates transfer and generalization on ScienceWorld and ALFWorld.

STEP-HRL. Zhen et al. (2026) introduce STEP-HRL in the paper formally titled *Hierarchical Reinforcement Learning with Augmented Step-Level Transitions for LLM Agents*. It introduces three functional components: a high-level policy that predicts the next subtask from the instruction and completed-subtask history, a low-level policy that acts from the current observation, subgoal, and compact local progress, and a local-progress module that iteratively folds intra-subtask interactions into that compact representation. The components share policy parameters but use distinct critics; training combines behavior-cloning initialization with offline RL. The compact progress state reduces dependence on full interaction histories and is evaluated on ScienceWorld and ALFWorld.

HiMAC. Jin et al. (2026) introduce HiMAC, which decomposes control into a shared-parameter Macro-Policy and Micro-Policy. The Macro-Policy generates a structured blueprint of natural-language subgoals, and the Micro-Policy generates atomic actions conditioned on each subgoal. Its critic-free hierarchical optimization extends group-relative policy optimization to produce separate blueprint-level and execution-level relative advantages. An alternating co-evolution schedule updates macro exploration with the executor fixed, then adapts the executor under a fixed high-confidence blueprint; experiments cover ALFWorld, WebShop, and Sokoban.

HIPIF. Diao et al. (2026) introduce HIPIF, which combines hierarchical planning with information folding for long-horizon interaction. At each step, a reflection operation decides whether the active subgoal is complete. When complete, the method folds that subgoal's interaction history into a compact summary and generates the next subgoal; otherwise, it generates another primitive action under the current subgoal. Training uses a GRPO-style clipped objective whose step-level advantages combine terminal environment reward with subgoal-oriented process rewards, avoiding an ever-growing raw interaction context.

Multi^2. Park & Kwon (2026) introduce Multi^2, which assigns the two hierarchy levels to separate role-specialized agents. Its high-level System 1 agent generates context-aware subgoals and is trained by supervised fine-tuning on a subgoal dataset. Its low-level System 2 agent executes atomic actions, first learning from offline data with actor–critic RL and then receiving online RL fine-tuning while System 1 remains fixed. The method targets objective drift in long-horizon interaction and introduces hierarchical datasets spanning multiple interactive environments.

Relation to STAC. The hierarchical methods above demonstrate several benefits of hierarchical language-agent training, including cross-turn credit assignment, transferable low-level skills, compact progress representations, macro–micro optimization, folded histories, and role-specialized planning and execution. STAC addresses an orthogonal issue: regardless of how an agent defines or trains its hierarchy, the learned boundary can still switch too frequently, persist after becoming stale, or leave the parsed optimization support. It therefore consumes the native boundary signal and modifies only boundary credit rather than prescribing a new planner, executor, trajectory representation, or base return estimator.

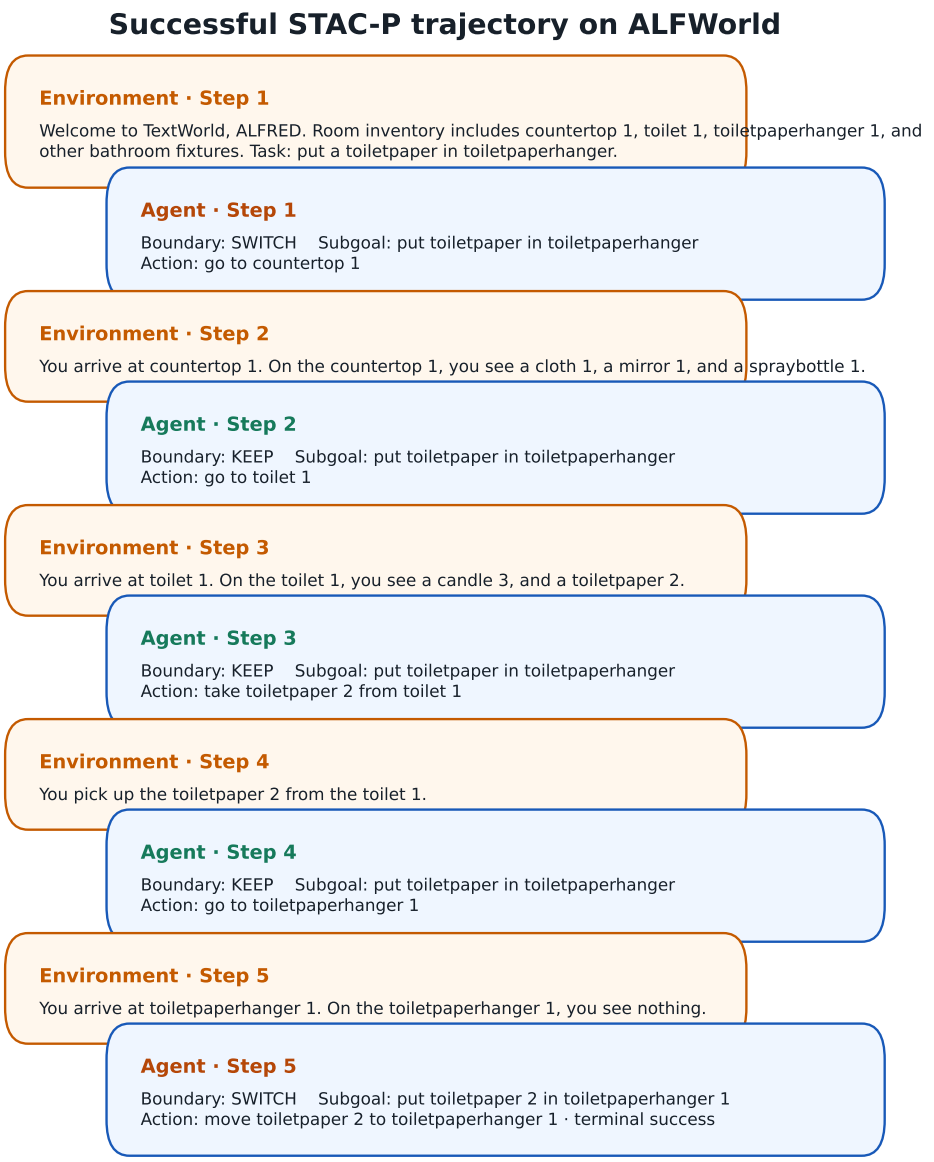


Figure 2: **A successful STAC-P trajectory on ALFWorld.** The Qwen3-0.6B policy is evaluated at seed 7's best checkpoint (update 145). After initially selecting a task-level subgoal, it keeps that subgoal across navigation and object acquisition, then switches to a grounded object–receptacle subgoal for the successful terminal action. The long initial room inventory is abridged for readability; all displayed boundary decisions, subgoals, actions, and subsequent observations are verbatim. The recorded trajectory identifier is `cfbc778c-2693-4e8e-844f-ebc3cbf25c52`.

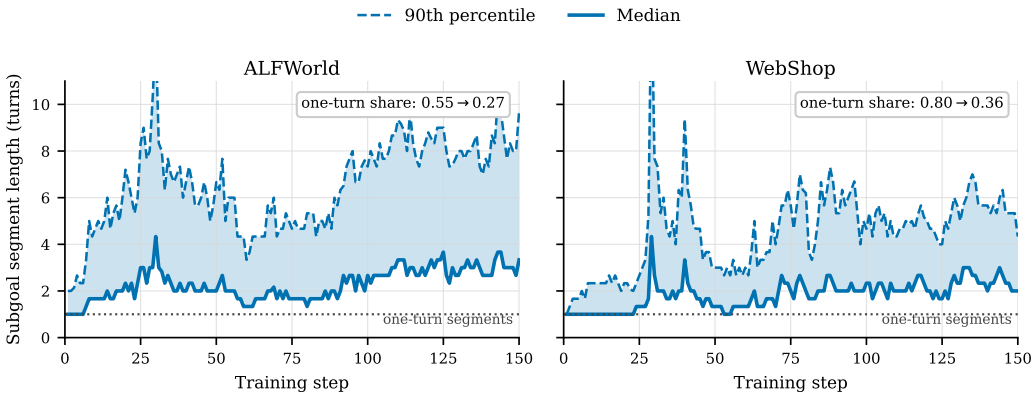


Figure 3: **Subgoal durations move away from one-turn segments.** Median and 90th-percentile STAC-D segment length, each computed over all segments in a training update and then averaged across three seeds; the shaded region spans the 50th–90th percentile range. The annotation gives the fraction of one-turn segments over the first twenty and final thirty updates.

F Additional Results

F.1 Representative STAC-P Trajectory

The trace illustrates the distinction between useful persistence and indiscriminate suppression of replanning. The three consecutive KEEP decisions preserve the abstract intent "put toiletpaper in toiletpaperhanger" while the low-level policy first locates and acquires the relevant object. Once the agent reaches the receptacle, SWITCH is appropriate: it replaces the abstract subgoal with the grounded command "put toiletpaper 2 in toiletpaperhanger 1," enabling the exact terminal action. This example is qualitative rather than additional evaluation evidence, but it exposes the interaction pattern summarized by the aggregate duration and violation diagnostics.

Duration distribution. Switch rate alone summarizes temporal abstraction with a single scalar, so Fig. 3 reports how the segment-length distribution itself evolves under STAC-D. Early in training the distribution concentrates on one-turn segments: they account for 0.55 of ALFWorld segments and 0.80 of WebShop segments, and the WebShop median segment is exactly one turn. By the end of training that mass has roughly halved, to 0.27 and 0.36, and the distribution has both shifted and widened. The median rises from 1.55 to 3.03 turns on ALFWorld and from 1.00 to 2.30 on WebShop, while the 90th percentile rises from 4.27 to 8.27 and from 2.08 to 5.40. The upper tail therefore grows faster than the median, which is the intended behavior: subgoals that coordinate long action sequences persist, while short segments remain available when replanning is appropriate.

F.2 Full Boundary and Completion Diagnostics

Table 6 provides the complete diagnostics underlying the compact main-text table. Panel (a) applies the same duration-based premature-replanning and stale-persistence measures to both STAC realizations as post hoc behavioral diagnostics; STAC-P does not optimize these duration thresholds. Both variants reduce both violations relative to HiPER, with STAC-D producing the lower premature-replanning rate and STAC-P producing the lower stale-persistence rate. Panel (b) restores the full completion-estimator statistics. Coverage is the fraction of boundary events with a valid environment-grounded label ($m_t = 1$), and Pos. is the positive-label fraction among those valid events. BCE is the masked binary cross-entropy in equation 16, while Brier is the mean squared difference between p_t^{done} and y_t over valid labels. AUROC is the probability that the estimator ranks a randomly selected complete subgoal above a randomly selected incomplete one. ECE partitions valid predictions into confidence bins and averages the absolute gap between each bin's mean confidence and empirical completion frequency, weighted by bin size. Finally, $p_{\text{done}} \mid \text{KEEP}$ and $p_{\text{done}} \mid \text{SWITCH}$ are the

mean predicted completion probabilities over valid boundary events with the corresponding policy decision.

Table 6: **Full STAC-D and STAC-P diagnostics (Qwen3-0.6B).** Panel (a) reports common duration-based behavioral diagnostics for the full configurations; lower rates are preferred. Panel (b) reports the complete STAC-P completion-estimator diagnostics. Values are mean ± SD; as in Table 1, the STAC-P WebShop behavioral row uses three independent seeds.

(a) Boundary-policy behavior

Environment	Variant	Premature ↓	Stale ↓
ALFWorld	STAC-D (full)	0.073 ± 0.018	0.098 ± 0.022
ALFWorld	STAC-P (full)	0.101 ± 0.020	0.061 ± 0.016
WebShop	STAC-D (full)	0.151 ± 0.027	0.039 ± 0.011
WebShop	STAC-P (full)	0.181 ± 0.026	0.034 ± 0.012

(b) STAC-P completion-estimator quality

Environment	Coverage (%) ↑	Pos. (%)	BCE ↓	Brier ↓	AUROC ↑	ECE ↓	p_{done} | KEEP ↓	p_{done} | SWITCH ↑
ALFWorld	62.8 ± 2.6	21.9 ± 1.7	0.274 ± 0.019	0.079 ± 0.008	0.921 ± 0.014	0.036 ± 0.007	0.182 ± 0.021	0.817 ± 0.026
WebShop	41.7 ± 3.8	16.3 ± 1.4	0.322 ± 0.028	0.096 ± 0.011	0.883 ± 0.021	0.051 ± 0.010	0.214 ± 0.027	0.754 ± 0.034

F.3 Additional Ablation Studies

ALFWorld penalty sensitivity. Table 7 varies the shared STAC-D coefficient while holding all other settings fixed. Increasing μ initially suppresses both violations and improves success, which peaks at $\mu = 0.1$. Stronger penalties continue to reduce violations but lower task success, showing that useful temporal abstraction requires a balance between stability and flexible replanning. We select $\mu = 0.1$ by ALFWorld validation success and reuse it unchanged for WebShop and the other backbones. Because the same validation trajectories select both μ and the best checkpoint, the reported peak may be favored by this coupling.

Table 7: **Sensitivity to the STAC-D penalty coefficient on ALFWorld (Qwen3-0.6B).** We set $\mu_{\text{short}} = \mu_{\text{stale}} = \mu$ and train each coefficient setting independently while holding all other settings fixed. The $\mu = 0$ and $\mu = 0.1$ rows are the canonical HiPER and STAC-D results from Table 1; values are mean ± SD over three seeds.

μ	Best ↑	Final ↑	Premature ↓	Stale ↓
0	83.3 ± 3.3	81.5 ± 3.6	0.553 ± 0.041	0.147 ± 0.026
0.025	87.2 ± 3.0	85.1 ± 3.2	0.318 ± 0.034	0.132 ± 0.023
0.05	90.1 ± 2.6	88.4 ± 2.8	0.176 ± 0.026	0.111 ± 0.020
0.10	$\mathbf{91.4 \pm 3.6}$	$\mathbf{89.3 \pm 3.5}$	0.073 ± 0.018	0.098 ± 0.022
0.20	90.9 ± 2.4	89.0 ± 2.6	0.045 ± 0.012	0.071 ± 0.014
0.40	87.5 ± 2.9	85.4 ± 3.1	$\mathbf{0.022 \pm 0.008}$	$\mathbf{0.054 \pm 0.012}$

WebShop penalty components and sensitivity. Table 8 repeats the component ablation on WebShop, and Table 9 reports the coefficient sweep. The short term again supplies most of the standalone gain, the full two-sided penalty performs best, and increasing μ beyond 0.1 further lowers violations while slightly reducing success.

Table 8: **WebShop ablation (Qwen3-0.6B).** Checkmarks denote active short- and stale-boundary costs. Values are mean ± SD over three seeds, except STAC-P (full), which uses the three independent seeds reported in Table 1.

Method	Short	Stale	Best ↑	Final ↑	Premature ↓	Stale ↓
HiPER	×	×	61.4 ± 3.5	56.5 ± 3.0	0.597 ± 0.034	0.110 ± 0.020
STAC-D (short)	✓	×	67.4 ± 2.9	64.3 ± 3.7	0.165 ± 0.029	0.095 ± 0.019
STAC-D (stale)	×	✓	63.0 ± 3.4	58.4 ± 3.2	0.605 ± 0.036	$\mathbf{0.025 \pm 0.009}$
STAC-D (full)	✓	✓	69.3 ± 2.4	67.5 ± 4.2	$\mathbf{0.151 \pm 0.027}$	0.039 ± 0.011
STAC-P (short)	✓	×	71.7 ± 2.0	69.8 ± 2.7	0.194 ± 0.030	0.083 ± 0.018
STAC-P (stale)	×	✓	64.8 ± 3.2	60.7 ± 3.4	0.568 ± 0.035	0.031 ± 0.010
STAC-P (full)	✓	✓	$\mathbf{74.2 \pm 1.1}$	$\mathbf{74.2 \pm 1.1}$	0.181 ± 0.026	0.034 ± 0.012

Table 9: **Sensitivity to the STAC-D penalty coefficient on WebShop (Qwen3-0.6B).** We set $\mu_{\text{short}} = \mu_{\text{stale}} = \mu$ and hold all other settings fixed. Values are mean ± SD over three seeds.

μ	Best ↑	Final ↑	Premature ↓	Stale ↓
0	61.4 ± 3.5	56.5 ± 3.0	0.597 ± 0.034	0.110 ± 0.020
0.025	64.8 ± 3.2	60.8 ± 3.5	0.401 ± 0.033	0.083 ± 0.018
0.05	67.5 ± 2.8	64.9 ± 3.8	0.263 ± 0.030	0.061 ± 0.015
0.10	$\mathbf{69.3 \pm 2.4}$	$\mathbf{67.5 \pm 4.2}$	0.151 ± 0.027	0.039 ± 0.011
0.20	68.5 ± 2.7	66.2 ± 3.9	$\mathbf{0.107 \pm 0.021}$	$\mathbf{0.028 \pm 0.009}$

STAC-P mechanism ablation. Table 10 isolates completion supervision, semantic boundary credit, warm-up, and stop-gradient. Training the completion predictor alone produces an accurate signal but leaves task performance near HiPER, so supervision alone does not explain the gain. The improvement appears when semantic completion modifies boundary credit. Removing warm-up lowers success despite nearly unchanged prediction quality, while removing stop-gradient degrades both calibration and success. The oracle cost yields only a small additional gain, indicating that the learned estimator recovers most of the benefit available from privileged completion information.

Table 10: **Mechanism ablation for STAC-P on ALFWorld (Qwen3-0.6B).** "Completion only" trains the predictor without applying it to boundary credit; the oracle variant directly uses environment-grounded labels during training. Values are mean ± SD over three seeds.

Method	Best ↑	Final ↑	AUROC ↑	Brier ↓	Premature ↓	Stale ↓
HiPER	83.3 ± 3.3	81.5 ± 3.6	–	–	0.553 ± 0.041	0.147 ± 0.026
Completion supervision only	84.1 ± 3.2	82.0 ± 3.4	$\mathbf{0.923 \pm 0.013}$	$\mathbf{0.077 \pm 0.007}$	0.541 ± 0.039	0.143 ± 0.025
STAC-P without warm-up	91.8 ± 2.4	89.5 ± 2.7	0.918 ± 0.015	0.081 ± 0.009	0.142 ± 0.023	0.080 ± 0.018
STAC-P without stop-gradient	89.6 ± 2.9	87.3 ± 3.1	0.847 ± 0.028	0.118 ± 0.014	0.181 ± 0.028	0.094 ± 0.020
STAC-P (full)	$\mathbf{95.3 \pm 1.6}$	$\mathbf{93.2 \pm 2.4}$	0.921 ± 0.014	0.079 ± 0.008	0.101 ± 0.020	0.061 ± 0.016
STAC-P with oracle completion cost	96.1 ± 1.4	94.0 ± 2.0	–	–	$\mathbf{0.087 \pm 0.017}$	$\mathbf{0.052 \pm 0.014}$

F.4 Detailed best- and final-checkpoint results

Statistical comparisons. Fig. 4 visualizes the GRPO, GiGPO, HiPER, and STAC-D results for Qwen3-0.6B and Llama-3.2-1B-Instruct with a confidence interval for each individual mean. Table 13 reports the flat-baseline comparisons and separate STAC-D–HiPER and STAC-P–HiPER comparisons as two-sided unpaired Welch tests with Welch–Satterthwaite degrees of freedom, computed from the reported means and standard deviations rather than from per-seed pairings. Both STAC variants improve on HiPER in all eight internally comparable benchmark–checkpoint cells. Fourteen of the sixteen comparisons reach $p_{\text{W}} < 0.05$; the exceptions are STAC-D on Qwen3-0.6B ALFWorld final ($p_{\text{W}} = 0.0547$) and STAC-P on Llama-3.2-1B-Instruct ALFWorld best ($p_{\text{W}} = 0.0504$). For

Table 11: **Detailed best-checkpoint results.** Values are mean (%) ± SD across seeds. For each trained method, the reported checkpoint has the highest validation success. ALFWorld reports binary success on its Seen and Unseen evaluation splits; WebShop reports exact binary success (Succ.) and its graded task score (Score). The Qwen2.5 WebShop block retains the mixed-source protocol described in Table 1. Tinted three-row blocks group each baseline hierarchy with its STAC-D/STAC-P variants; bold marks the best result within each backbone–metric column.

		ALFWorld		WebShop	
Regime	Method	Seen	Unseen	Succ.	Score
Qwen3-0.6B					
—	Base (Plan–Execute)	0.4 ± 0.6	0.2 ± 0.4	1.2 ± 1.7	12.2 ± 5.0
Flat RL	GRPO	39.3 ± 3.0	35.7 ± 3.2	44.7 ± 3.2	68.2 ± 3.1
Flat RL	GiGPO	61.7 ± 7.7	56.5 ± 8.1	51.3 ± 5.5	72.8 ± 4.3
Flat RL	PPO	37.5 ± 2.8	34.1 ± 3.0	40.2 ± 3.0	64.9 ± 3.0
Hierarchical RL	GLIDER	58.6 ± 4.4	54.5 ± 4.6	44.0 ± 4.8	67.7 ± 4.2
Hierarchical RL	GLIDER + STAC-D (ours)	63.7 ± 4.2	60.6 ± 4.4	49.3 ± 3.2	71.5 ± 3.0
Hierarchical RL	GLIDER + STAC-P (ours)	70.1 ± 3.5	67.4 ± 3.7	53.4 ± 3.4	74.2 ± 3.1
Hierarchical RL	HiPER	84.7 ± 3.6	81.9 ± 3.8	61.4 ± 3.5	79.3 ± 3.0
Hierarchical RL	HiPER + STAC-D (ours)	92.3 ± 3.9	90.5 ± 4.1	69.3 ± 2.4	84.0 ± 2.1
Hierarchical RL	HiPER + STAC-P (ours)	$\mathbf{96.0 \pm 1.7}$	$\mathbf{94.6 \pm 1.8}$	$\mathbf{74.2 \pm 1.1}$	$\mathbf{86.8 \pm 1.0}$
Qwen2.5-1.5B-Instruct					
—	Base + ReAct	8.8 ± 1.0	7.8 ± 1.1	5.5 ± 6.3	25.1 ± 8.9
Flat RL	GRPO	73.2 ± 8.9	69.0 ± 9.3	56.8 ± 3.8	75.8 ± 3.5
Flat RL	GiGPO	88.4 ± 1.8	85.0 ± 1.9	67.4 ± 4.5	83.5 ± 1.8
Flat RL	PPO	70.2 ± 2.0	66.1 ± 2.1	51.5 ± 2.9	73.8 ± 3.0
Hierarchical RL	GLIDER	73.7 ± 4.4	70.6 ± 4.6	57.6 ± 5.0	76.9 ± 4.1
Hierarchical RL	GLIDER + STAC-D (ours)	78.0 ± 4.3	75.3 ± 4.5	62.0 ± 5.0	79.7 ± 4.0
Hierarchical RL	GLIDER + STAC-P (ours)	78.1 ± 3.3	75.6 ± 3.5	62.1 ± 4.4	79.7 ± 3.6
Hierarchical RL	HiPER	93.7 ± 4.0	91.3 ± 4.2	71.4 ± 5.0	85.7 ± 3.2
Hierarchical RL	HiPER + STAC-D (ours)	$\mathbf{97.2 \pm 1.3}$	$\mathbf{95.6 \pm 1.4}$	$\mathbf{78.2 \pm 5.6}$	$\mathbf{89.0 \pm 3.7}$
Hierarchical RL	HiPER + STAC-P (ours)	96.7 ± 0.9	95.5 ± 1.0	77.4 ± 2.2	88.5 ± 1.9
Llama-3.2-1B-Instruct					
—	Base (Plan–Execute)	2.0 ± 1.4	1.6 ± 1.5	0.3 ± 0.4	6.3 ± 2.8
Flat RL	GRPO	8.9 ± 3.8	6.7 ± 4.0	1.6 ± 0.5	14.0 ± 2.4
Flat RL	GiGPO	14.1 ± 5.9	10.9 ± 6.3	1.6 ± 0.3	14.0 ± 2.2
Flat RL	PPO	7.9 ± 3.2	6.1 ± 3.4	1.2 ± 0.5	12.2 ± 2.5
Hierarchical RL	GLIDER	22.5 ± 5.3	19.4 ± 5.5	3.1 ± 2.6	18.9 ± 4.8
Hierarchical RL	GLIDER + STAC-D (ours)	30.2 ± 6.4	26.3 ± 6.7	7.5 ± 3.8	29.2 ± 5.9
Hierarchical RL	GLIDER + STAC-P (ours)	28.9 ± 4.4	25.4 ± 4.6	7.0 ± 2.0	28.3 ± 4.1
Hierarchical RL	HiPER	31.0 ± 5.9	26.2 ± 6.3	1.8 ± 0.4	14.8 ± 2.4
Hierarchical RL	HiPER + STAC-D (ours)	$\mathbf{54.1 \pm 6.5}$	$\mathbf{50.1 \pm 6.8}$	$\mathbf{17.6 \pm 3.9}$	$\mathbf{43.8 \pm 5.7}$
Hierarchical RL	HiPER + STAC-P (ours)	41.7 ± 3.8	37.8 ± 4.0	15.4 ± 3.7	41.1 ± 5.3

the Qwen3-0.6B and Llama-3.2-1B-Instruct blocks, both flat baselines are below HiPER wherever the comparison is defined, so the hierarchy itself helps and the boundary terms add a further gain on top of it. On Llama-3.2-1B-Instruct at the final checkpoint, HiPER, GRPO, and GiGPO all record 0.0 ± 0.0, so the flat-baseline comparisons have zero difference and zero pooled variance and no Welch statistic exists; those cells are marked "—" rather than assigned a p-value. Deriving tests from published means and standard deviations also discards the seed-level pairing, which is the conservative choice here. We do not test the Qwen2.5-1.5B-Instruct WebShop STAC diagnostics against the published HiPER value because the aggregates use different runtimes, checkpoint protocols, and seed sets.

Table 12: **Detailed final-checkpoint results.** Values are mean (%) ± SD across seeds; completed runs are evaluated at update 150. ALFWorld reports binary success on its Seen and Unseen evaluation splits; WebShop reports exact binary success (Succ.) and its graded task score (Score). The Qwen2.5 WebShop block retains the mixed-source protocol described in Table 1. Tinted three-row blocks group each baseline hierarchy with its STAC-D/STAC-P variants; bold marks the best result within each backbone–metric column.

		ALFWorld		WebShop	
Regime	Method	Seen	Unseen	Succ.	Score
Qwen3-0.6B					
—	Base (Plan–Execute)	0.4 ± 0.6	0.2 ± 0.4	1.2 ± 1.7	12.2 ± 5.0
Flat RL	GRPO	37.5 ± 5.1	34.3 ± 5.4	44.7 ± 3.2	68.2 ± 3.1
Flat RL	GiGPO	61.4 ± 7.7	56.8 ± 8.1	51.3 ± 5.4	72.8 ± 4.3
Flat RL	PPO	35.7 ± 3.3	32.4 ± 3.5	38.7 ± 3.5	63.7 ± 3.4
Hierarchical RL	GLIDER	53.4 ± 3.8	48.9 ± 4.0	40.6 ± 4.5	65.2 ± 4.1
Hierarchical RL	GLIDER + STAC-D (ours)	60.0 ± 3.9	56.9 ± 4.1	46.7 ± 4.0	69.6 ± 3.8
Hierarchical RL	GLIDER + STAC-P (ours)	64.9 ± 4.2	62.0 ± 4.4	52.1 ± 3.5	73.4 ± 3.2
Hierarchical RL	HiPER	83.2 ± 3.9	79.8 ± 4.1	56.5 ± 3.0	76.2 ± 2.8
Hierarchical RL	HiPER + STAC-D (ours)	90.4 ± 3.8	88.2 ± 4.0	67.5 ± 4.2	83.0 ± 3.6
Hierarchical RL	HiPER + STAC-P (ours)	$\mathbf{94.1 \pm 2.6}$	$\mathbf{92.3 \pm 2.8}$	$\mathbf{74.2 \pm 1.1}$	$\mathbf{86.8 \pm 1.0}$
Qwen2.5-1.5B-Instruct					
—	Base + ReAct	8.8 ± 1.0	7.8 ± 1.1	5.5 ± 6.3	25.1 ± 8.9
Flat RL	GRPO	72.2 ± 8.6	68.4 ± 9.1	50.3 ± 4.1	72.1 ± 3.8
Flat RL	GiGPO	83.3 ± 3.5	78.9 ± 3.6	61.1 ± 5.5	79.1 ± 4.3
Flat RL	PPO	68.3 ± 3.0	64.4 ± 3.2	47.9 ± 3.4	70.5 ± 3.2
Hierarchical RL	GLIDER	69.8 ± 4.1	66.3 ± 4.3	52.6 ± 6.0	73.7 ± 4.8
Hierarchical RL	GLIDER + STAC-D (ours)	74.6 ± 4.3	71.5 ± 4.5	58.3 ± 5.0	77.4 ± 4.2
Hierarchical RL	GLIDER + STAC-P (ours)	75.6 ± 3.5	72.9 ± 3.7	57.4 ± 4.3	76.8 ± 3.8
Hierarchical RL	HiPER	86.8 ± 3.9	83.6 ± 4.1	66.5 ± 5.1	82.4 ± 4.0
Hierarchical RL	HiPER + STAC-D (ours)	90.9 ± 2.3	88.9 ± 2.4	74.5 ± 5.1	86.9 ± 3.8
Hierarchical RL	HiPER + STAC-P (ours)	$\mathbf{93.3 \pm 3.2}$	$\mathbf{91.5 \pm 3.4}$	$\mathbf{76.2 \pm 3.9}$	$\mathbf{87.9 \pm 3.1}$
Llama-3.2-1B-Instruct					
—	Base (Plan–Execute)	2.0 ± 1.4	1.6 ± 1.5	0.3 ± 0.4	6.3 ± 2.8
Flat RL	GRPO	2.2 ± 1.6	1.0 ± 1.7	0.0 ± 0.0	8.5 ± 2.2
Flat RL	GiGPO	8.2 ± 2.7	5.8 ± 2.8	0.0 ± 0.0	9.0 ± 2.0
Flat RL	PPO	1.6 ± 1.3	1.0 ± 1.4	0.0 ± 0.0	7.8 ± 2.1
Hierarchical RL	GLIDER	17.5 ± 5.3	14.4 ± 5.5	1.6 ± 1.3	11.2 ± 3.5
Hierarchical RL	GLIDER + STAC-D (ours)	25.6 ± 4.6	22.1 ± 4.8	4.7 ± 2.5	23.4 ± 4.7
Hierarchical RL	GLIDER + STAC-P (ours)	24.5 ± 3.8	21.4 ± 4.0	4.5 ± 2.2	22.9 ± 4.2
Hierarchical RL	HiPER	19.2 ± 4.8	15.2 ± 5.0	0.0 ± 0.0	9.4 ± 2.3
Hierarchical RL	HiPER + STAC-D (ours)	$\mathbf{42.7 \pm 5.4}$	$\mathbf{38.1 \pm 5.7}$	$\mathbf{15.3 \pm 4.6}$	$\mathbf{41.0 \pm 6.0}$
Hierarchical RL	HiPER + STAC-P (ours)	36.4 ± 4.4	32.3 ± 4.6	14.4 ± 4.4	39.8 ± 5.8

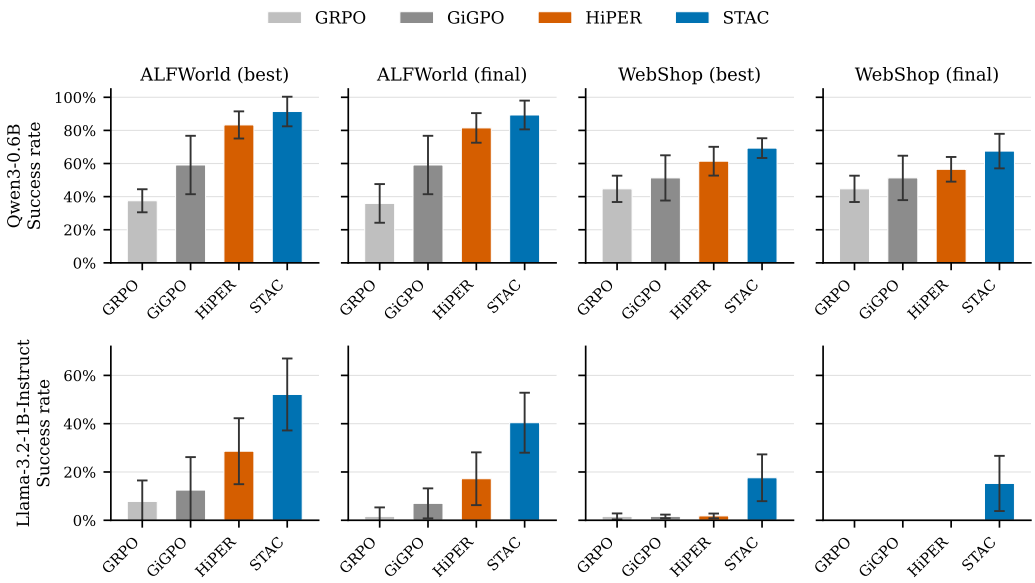


Figure 4: **Main results with per-mean confidence intervals.** The GRPO, GiGPO, HiPER, and STAC-D values for Qwen3-0.6B and Llama-3.2-1B-Instruct from Table 1. Error bars are 95% Student-t confidence intervals for individual means and do not indicate the significance of pairwise differences. On Llama-3.2-1B-Instruct WebShop at the final checkpoint, every baseline records exactly zero across seeds and therefore has no interval.

Table 13: Two-sided unpaired Welch tests against released HiPER. STAC-D and STAC-P are tested separately; differences are in success points and * denotes $p_{\mathrm{W}} < 0.05$.

Benchmark	Method	Diff.	t	95% CI	p_{W}
Qwen3-0.6B					
ALFWorld best	GRPO	-45.8	-18.33	$[-52.8, -38.8]$	0.0001^*
	GiGPO	-24.2	-5.35	$[-39.1, -9.3]$	0.0148^*
	STAC-D	$\mathbf{+8.1}$	2.87	$[+0.3, +16.0]$	$\mathbf{0.0457}^*$
	STAC-P	$\mathbf{+12.0}$	5.67	$[+5.1, +18.9]$	$\mathbf{0.0120}^*$
ALFWorld final	GRPO	-45.6	-13.34	$[-55.4, -35.9]$	0.0003^*
	GiGPO	-22.4	-4.87	$[-37.1, -7.7]$	0.0170^*
	STAC-D	$\mathbf{+7.8}$	2.69	$[-0.3, +15.9]$	0.0547
	STAC-P	$\mathbf{+11.7}$	4.68	$[+4.3, +19.1]$	$\mathbf{0.0131}^*$
WebShop best	GRPO	-16.7	-6.10	$[-24.3, -9.1]$	0.0038^*
	GiGPO	-10.1	-2.68	$[-21.3, +1.1]$	0.0656
	STAC-D	$\mathbf{+7.9}$	3.22	$[+0.7, +15.1]$	$\mathbf{0.0383}^*$
	STAC-P	$\mathbf{+12.8}$	6.04	$[+5.0, +20.6]$	$\mathbf{0.0169}^*$
WebShop final	GRPO	-11.8	-4.66	$[-18.8, -4.8]$	0.0097^*
	GiGPO	-5.2	-1.46	$[-16.3, +5.9]$	0.2374
	STAC-D	$\mathbf{+11.0}$	3.69	$[+2.4, +19.6]$	$\mathbf{0.0250}^*$
	STAC-P	$\mathbf{+17.7}$	9.59	$[+11.2, +24.2]$	$\mathbf{0.0047}^*$
Llama-3.2-1B-Instruct					
ALFWorld best	GRPO	-20.8	-5.53	$[-32.0, -9.6]$	0.0084^*
	GiGPO	-16.1	-3.59	$[-28.6, -3.6]$	0.0231^*
	STAC-D	$\mathbf{+23.5}$	5.00	$[+10.4, +36.6]$	$\mathbf{0.0076}^*$
	STAC-P	$\mathbf{+11.2}$	2.98	$[-0.03, +22.4]$	0.0504
ALFWorld final	GRPO	-15.6	-5.81	$[-25.3, -5.9]$	0.0172^*
	GiGPO	-10.2	-3.49	$[-19.2, -1.2]$	0.0364^*
	STAC-D	$\mathbf{+23.2}$	6.03	$[+12.5, +33.9]$	$\mathbf{0.0040}^*$
	STAC-P	$\mathbf{+17.2}$	4.95	$[+7.5, +26.9]$	$\mathbf{0.0078}^*$
WebShop best	GRPO	-0.2	-0.54	$[-1.3, +0.9]$	0.6185
	GiGPO	-0.2	-0.69	$[-1.0, +0.6]$	0.5293
	STAC-D	$\mathbf{+15.8}$	6.98	$[+6.3, +25.4]$	$\mathbf{0.0188}^*$
	STAC-P	$\mathbf{+13.6}$	6.33	$[+4.6, +22.6]$	$\mathbf{0.0227}^*$
WebShop final	GRPO	0.0	—	—	—
	GiGPO	0.0	—	—	—
	STAC-D	$\mathbf{+15.3}$	5.76	$[+3.9, +26.7]$	$\mathbf{0.0288}^*$
	STAC-P	$\mathbf{+14.4}$	5.67	$[+3.5, +25.3]$	$\mathbf{0.0297}^*$

Multi-phase behavior. At selected checkpoints, pick-two success rises from 0.231 ± 0.310 to 0.811 ± 0.078; the wide HiPER spread reflects seed-level bimodality rather than a stable mean. The largest observed gain is concentrated in this multi-phase category.

Hierarchy validity. All STAC seeds retain multi-turn subgoals, nearly perfect structured-output validity, and no final missing boundary masks. One released HiPER seed instead ends at validity 0.016 despite validation success 0.812, so aggregate accuracy alone does not reveal the state of the hierarchy.

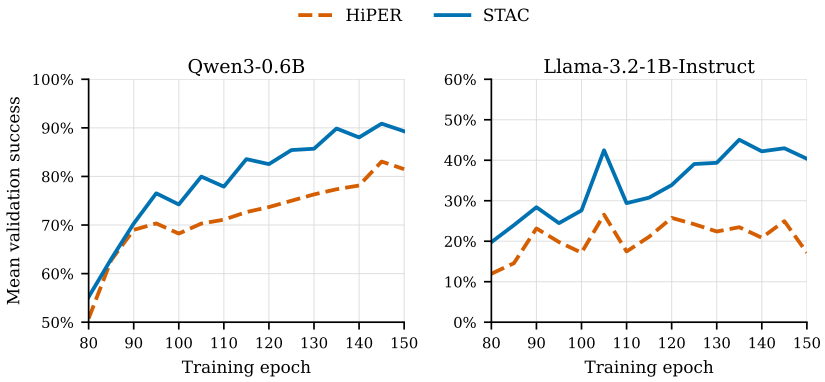


Figure 5: **Late-training ALFWorld validation.** Mean HiPER and STAC-D validation success across three seeds from epochs 80 through 150, without confidence intervals. Validation is performed every five epochs on 128 held-out episodes per seed. The vertical ranges differ between panels because the two backbones differ by roughly a factor of two in absolute success; a shared axis would flatten the Llama-3.2-1B panel. STAC-D is above HiPER at every evaluation on both backbones over this window.

G Final-Checkpoint and Late-Window Robustness

The main text follows the best-validation-checkpoint convention. Tables 11 and 12 provide the comprehensive cross-benchmark breakdown at the best and final checkpoints: ALFWorld Seen/Unseen success and WebShop exact success/task score, respectively. For complementary ALFWorld diagnostics, Table 14 reports the category breakdown at the saved best checkpoints, Table 15 reports every category at the final checkpoint, and Table 16 gives a lower-variance late-window diagnostic. "Late" first averages the seven validations at updates 120–150 within each seed and then reports mean $\pm$ SD across seeds. Tables 14 and 15 report task-category breakdowns only where measured or source-backed category results are available; Table 1 remains authoritative for overall success. Table 1 reports Qwen2.5 WebShop published baselines alongside separately run local STAC diagnostics; these ALFWorld category tables do not apply to them. The late-window diagnostic in Table 16 remains Qwen3-0.6B only, since the flat baselines were not rerun on the additional backbones.

Reading the Llama-3.2-1B block. Across the listed Llama-3.2-1B-Instruct methods, Heat is 0.0 throughout both tables, and Cool is also 0.0 except for HiPER + STAC-D at the final checkpoint. Pick2 is the diagnostic category, because completing it requires holding one subgoal across a full fetch-and-place phase before switching to the second object. Within the HiPER instantiation, STAC-D reaches 80.1 ± 12.8 against HiPER's 20.1 ± 9.4 at the best checkpoint and 61.7 ± 16.9 against 8.3 ± 10.4 at update 150, mirroring the same Pick2 weakness that HiPER shows on Qwen3-0.6B (23.1 ± 31.0).

Table 14: Available source-backed ALFWorld category accuracy (%) at the saved best checkpoints used in Table 1, mean ± SD across three seeds. "All" reproduces the authoritative overall value in Table 1; bold marks the highest mean within each model block.

Method	Pick	Look	Clean	Heat	Cool	Pick2	All
Qwen3-0.6B							
GRPO	51.0 ± 3.4	21.3 ± 18.3	43.9 ± 9.2	31.9 ± 15.5	25.3 ± 17.6	40.7 ± 1.2	37.5 ± 2.8
GiGPO	75.1 ± 12.4	55.6 ± 12.7	57.9 ± 13.9	50.1 ± 10.3	53.3 ± 8.3	55.6 ± 10.7	59.1 ± 7.1
HiPER	$\mathbf{98.1 \pm 1.6}$	85.8 ± 19.5	95.6 ± 5.1	97.4 ± 5.3	86.5 ± 9.2	23.1 ± 31.1	83.3 ± 3.3
HiPER + STAC-D (ours)	94.9 ± 8.9	85.8 ± 4.7	97.5 ± 2.2	97.8 ± 3.9	85.6 ± 11.2	81.1 ± 7.8	91.4 ± 3.6
Qwen2.5-1.5B-Instruct							
Base + ReAct	15.9 ± 1.8	13.9 ± 6.7	11.2 ± 4.1	3.5 ± 6.1	0.0 ± 0.0	4.2 ± 0.7	8.3 ± 0.9
GRPO	77.4 ± 6.7	54.2 ± 7.2	75.6 ± 8.6	85.6 ± 17.1	67.8 ± 7.4	56.1 ± 11.0	71.1 ± 8.2
GiGPO	94.4 ± 5.9	67.5 ± 4.6	94.8 ± 3.8	94.4 ± 7.8	79.8 ± 4.7	76.4 ± 5.4	86.7 ± 1.7
PPO	74.4 ± 9.6	37.5 ± 21.7	67.3 ± 5.1	85.6 ± 17.1	68.8 ± 3.7	56.1 ± 12.2	68.2 ± 1.8
HiPER	98.1 ± 3.3	84.8 ± 13.9	$\mathbf{98.9 \pm 1.8}$	$\mathbf{100.0 \pm 0.0}$	94.1 ± 2.8	68.1 ± 24.2	92.5 ± 3.7
HiPER + STAC-D (ours)	98.8 ± 2.1	88.6 ± 2.0	97.8 ± 3.9	95.5 ± 3.9	$\mathbf{95.5 \pm 7.9}$	$\mathbf{95.4 \pm 1.1}$	$\mathbf{96.4 \pm 1.2}$
Llama-3.2-1B-Instruct							
HiPER	61.5 ± 11.4	65.9 ± 7.3	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	20.1 ± 9.4	28.6 ± 5.5
HiPER + STAC-D (ours)	$\mathbf{86.4 \pm 10.0}$	$\mathbf{70.8 \pm 6.6}$	$\mathbf{29.0 \pm 10.3}$	0.0 ± 0.0	0.0 ± 0.0	$\mathbf{80.1 \pm 12.8}$	$\mathbf{52.1 \pm 6.0}$

Table 15: Available source-backed ALFWorld category accuracy (%) at the final checkpoint (update 150), mean ± SD across three seeds. "All" reproduces the authoritative overall value in Table 1; bold marks the highest mean within each model block.

Method	Pick	Look	Clean	Heat	Cool	Pick2	All
Qwen3-0.6B							
GRPO	48.9 ± 8.9	27.8 ± 12.7	47.4 ± 5.3	14.1 ± 0.4	40.0 ± 6.9	28.3 ± 12.6	35.9 ± 4.7
GiGPO	75.1 ± 12.0	55.6 ± 12.7	57.9 ± 13.9	50.1 ± 10.3	53.3 ± 8.3	55.0 ± 10.0	59.1 ± 7.1
HiPER	96.8 ± 5.6	83.3 ± 0.0	$\mathbf{98.2 \pm 3.1}$	$\mathbf{92.2 \pm 7.2}$	86.7 ± 2.3	23.3 ± 31.8	81.5 ± 3.6
HiPER + STAC-D (ours)	$\mathbf{97.8 \pm 3.8}$	$\mathbf{91.7 \pm 8.4}$	93.0 ± 8.0	84.6 ± 10.3	$\mathbf{92.0 \pm 6.9}$	73.3 ± 17.6	89.3 ± 3.5
Qwen2.5-1.5B-Instruct							
Base + ReAct	15.9 ± 1.8	13.9 ± 6.7	11.2 ± 4.1	3.5 ± 6.1	0.0 ± 0.0	4.2 ± 0.7	8.3 ± 0.9
HiPER	$\mathbf{96.7 \pm 3.3}$	75.0 ± 16.7	91.2 ± 15.2	89.0 ± 2.9	90.7 ± 4.6	56.7 ± 25.2	85.2 ± 3.6
HiPER + STAC-D (ours)	95.7 ± 1.9	$\mathbf{91.7 \pm 8.4}$	91.2 ± 6.1	$\mathbf{93.8 \pm 5.4}$	84.0 ± 4.0	81.7 ± 5.8	89.9 ± 2.1
Llama-3.2-1B-Instruct							
HiPER	47.7 ± 9.8	47.2 ± 7.6	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	8.3 ± 10.4	17.2 ± 4.4
HiPER + STAC-D (ours)	$\mathbf{86.0 \pm 15.9}$	$\mathbf{72.2 \pm 11.0}$	$\mathbf{19.3 \pm 13.4}$	0.0 ± 0.0	$\mathbf{2.7 \pm 4.6}$	$\mathbf{61.7 \pm 16.9}$	$\mathbf{40.4 \pm 5.0}$

For the Qwen3-0.6B HiPER + STAC-D comparison, the final-overall unpaired Welch difference relative to HiPER is $+0.078$ (95% CI $[-0.003, +0.159]$, $p_{\mathrm{W}} = 0.0547$). The corresponding late-overall difference, averaging updates 120–150, is $+0.095$ ($[+0.052, +0.138]$). The Qwen3-0.6B final accuracies are 0.359 ± 0.047 for GRPO, 0.591 ± 0.071 for GiGPO, 0.815 ± 0.036 for HiPER, and 0.893 ± 0.035 for HiPER + STAC-D.

The best-checkpoint steps across the three seeds are GRPO 140/150/140, GiGPO 150/150/150, HiPER 90/145/145, and STAC-D 120/145/145. For the Qwen2.5-1.5B-Instruct ALFWorld replication, HiPER and STAC-D use 145/125/145 and 95/95/145, respectively; STAC-D seed 6 also attains its maximum at update 145. The discrepancy between main and appendix summaries quantifies checkpoint-selection sensitivity rather than changing the method ranking.

H Boundary-Mask Representation

One turn. Let the response generated at environment turn t contain L_t tokens, $y_t = (y_{t,1}, \ldots, y_{t,L_t})$. The parser locates the structured block that expresses the boundary decision, and $M^b_{t,j} = 1$ exactly for token positions belonging to that block; every other position receives zero. For example, suppose a tokenized response is schematically

$$\underbrace{\langle \text{switch} \rangle \ \text{KEEP} \ \langle / \text{switch} \rangle}_{\text{boundary block}} \quad \underbrace{\langle \text{subgoal} \rangle \ \cdots}_{\text{subgoal block}} \quad \underbrace{\langle \text{action} \rangle \ \cdots}_{\text{action block}}.$$

If the boundary block occupies the first three token positions, its mask is

$$M^b_t = [1,\ 1,\ 1,\ 0,\ 0, \ldots, 0].$$

Table 16: Late-window per-category accuracy (%), mean ± SD across seeds. "Avg." is logged task-distribution-weighted overall accuracy.

Method	Pick	Look	Clean	Heat	Cool	Pick2	Avg.
GRPO	39.4 ± 4.6	23.5 ± 9.1	32.8 ± 6.3	28.3 ± 3.1	18.3 ± 1.1	20.7 ± 3.5	28.5 ± 1.5
GiGPO	61.6 ± 9.3	46.7 ± 8.9	56.0 ± 11.0	50.1 ± 9.7	29.0 ± 4.1	38.8 ± 9.6	48.2 ± 7.9
HiPER	$\mathbf{96.7 \pm 1.3}$	77.4 ± 1.5	92.4 ± 0.6	86.2 ± 7.7	85.0 ± 0.4	11.6 ± 13.3	77.9 ± 2.0
STAC-D	94.9 ± 2.0	$\mathbf{81.6 \pm 2.6}$	$\mathbf{93.8 \pm 1.9}$	$\mathbf{86.7 \pm 1.4}$	$\mathbf{85.9 \pm 3.8}$	$\mathbf{73.8 \pm 2.6}$	$\mathbf{87.4 \pm 1.7}$

The exact number of ones depends on tokenization and on whether delimiter tokens are included in the configured mask. If the boundary block is missing or malformed, the parser selects no boundary tokens and

$$M_t^b = [0,\ 0, \ldots, 0], \qquad \|M_t^b\|_0 = 0,$$

so that turn contributes no boundary-level term.

A complete task trace. The mask is reconstructed from the response at every environment turn, not only when the agent creates a new subgoal. Thus a T-turn task produces $M_1^b, \ldots, M_T^b$. After padding each response to a common width L, the masks form

$$M^b \in \{0,1\}^{T \times L}.$$

A schematic four-turn trace is

$$M^b = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & \cdots \\ 1 & 1 & 1 & 0 & 0 & \cdots \\ 1 & 1 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & \cdots \end{bmatrix}.$$

The first three rows contain valid boundary blocks; each may encode either KEEP or SWITCH. The fourth response has no valid boundary block, so its row is all zero. Consequently, M_t^b records *where the current turn writes its boundary decision*; it does not identify the active subgoal or persist for the lifetime of that subgoal. Subgoal tokens are represented by a separate mask.