VTD: DISTILLING SYMBOLIC TASK STRUCTURE FOR LONG-HORIZON AGENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Solving long-horizon tasks requires agents to make a sequence of interdependent decisions before receiving a reliable outcome signal. On-policy distillation supplements this sparse feedback with token-level guidance from a stronger teacher on trajectories generated by the student agent. However, the teacher conditions each prediction on the student's interaction history, so earlier mistakes can distort all subsequent guidance. Our analysis further shows that even teachers that can solve a task on their own increasingly favor the failing student's actions when conditioned on this history. We therefore propose Verifiable Task Decomposition (VTD), which separates task decomposition from trajectory execution using a symbolic progress graph constructed by the teacher before interaction. During rollout, the environment turns completed graph milestones into verified progress that complements the terminal outcome in policy optimization. Graph distillation transfers this decomposition capability to the student, allowing the student to generate progress graphs without teacher access at test time. Experiments on ALFWorld, WebShop, and TextCraft show that VTD accelerates policy learning and outperforms the strongest Qwen2.5-7B baseline by 1.8, 3.4, and 8.0 points, respectively. VTD also exhibits strong structural generalization beyond the training distribution, reaching 92.8% success on held-out ALFWorld task families compared with 68.0% for the strongest baseline and extending to deeper TextCraft recipe hierarchies.

1 INTRODUCTION

Long-horizon agentic tasks require an agent to complete a sequence of interdependent actions (Shridhar et al., 2021; Yao et al., 2022; Prasad et al., 2024; Zhou et al., 2024; Luo et al., 2025; Zhou et al., 2025; He et al., 2026; Liu et al., 2026; Xi et al., 2026a;b; Zhang et al., 2026). The environment evaluates task success only after the interaction ends (Chen et al., 2025; Wang et al., 2025; 2026d). Outcome-based policy optimization methods such as Group Relative Policy Optimization (GRPO) use the final task outcome as the reward for the entire trajectory (Shao et al., 2024; Chen et al., 2025; Wang et al., 2025). A failed trajectory that completes several necessary steps therefore receives the same reward as one that makes no progress. Successful trajectories are rare early in training, so most sampled trajectories receive the same zero reward (Wang et al., 2026d).

On-policy distillation (OPD) offers a natural way to supplement sparse outcome feedback with dense supervision from a stronger teacher (Gu et al., 2024; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). OPD trains the student to match the teacher's next-token distribution on trajectories sampled from the student policy (Agarwal et al., 2024; Hübotter et al., 2026; Zhao et al., 2026). This token-level objective allows failed trajectories to contribute to training even when they receive zero outcome reward (Zhao et al., 2026; Lu et al., 2026). However, in a multi-turn task, each teacher prediction is conditioned on the interaction history generated by the student, so an early mistake affects all subsequent guidance (Wang et al., 2026b; Li et al., 2026; Zhu et al., 2026; Zhong et al., 2026; Lu et al., 2026). Our analysis in Section 3 shows that even teachers that can solve a task on their own increasingly favor the failing student's actions when conditioned on the student's interaction history. The central problem is therefore not the amount of teacher supervision, but its dependence on the student's interaction history.

Recent work addresses history-dependent supervision through temporal curricula and turn-level teacher intervention (Wang et al., 2026b; Li et al., 2026). Other methods stabilize the distillation

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.

objective or allocate supervision selectively across tokens and turns (Zhu et al., 2026; Lu et al., 2026; Zhong et al., 2026). CCOPD (Lin et al., 2026) instead conditions the teacher on a canonical context. Across these designs, teacher supervision still depends on the actions that the agent produces during task execution. This dependence motivates our central research question (RQ):

> ***RQ.*** *Can a teacher supervise a long-horizon agent without observing the agent's trajectory?*

To answer this question, we propose Verifiable Task Decomposition (VTD), which separates task decomposition from trajectory execution. Before interaction, the teacher decomposes the task into a symbolic progress graph. Milestones form the graph's nodes, with each node specifying a concrete condition for task progress that the environment can verify. Dependency edges capture prerequisite relationships among milestones and organize the milestones into the task's multi-step structure. For each task, the resulting graph provides a common definition of progress across student trajectories.

During interaction, the environment checks the graph milestones against the current environment state and records those completed by the student's trajectory. VTD uses the resulting verified progress together with the terminal outcome during policy optimization. Failed trajectories that complete more milestones therefore receive a stronger learning signal than trajectories that make less progress. Graph distillation turns the teacher's task decomposition into a student capability, enabling the student to construct and use symbolic progress graphs without access to the teacher at test time.

We evaluate VTD on ALFWorld, WebShop, and TextCraft with Qwen2.5-1.5B and Qwen2.5-7B models (Qwen Team, 2024). With Qwen2.5-7B, VTD reaches 99.2% success on ALFWorld, 86.7% on WebShop, and 97.0% on TextCraft, outperforming the strongest baselines by 1.8, 3.4, and 8.0 points, respectively. On TextCraft, where each target item requires a hierarchy of intermediate recipes, VTD also outperforms the strongest Qwen2.5-1.5B baseline by 25.2 points. In controlled ablations, Qwen2.5-1.5B policies learn substantially faster with verified progress than with terminal outcomes alone on ALFWorld and WebShop. Graph distillation successfully transfers symbolic task structure from the teacher to the student. On ALFWorld, this transfer allows the Qwen2.5-1.5B student to construct structurally accepted graphs for every task, reaching 97.0% verified progress and raising task success by 8.9 points. Notably, VTD exhibits strong structural generalization beyond the training distribution, reaching 92.8% success on held-out ALFWorld task families compared with 68.0% for the strongest baseline and extending to deeper TextCraft recipe hierarchies. Our contributions are:

- We identify a failure mode of teacher supervision in long-horizon OPD, in which dense feedback progressively loses its corrective value when conditioned on a failing student's interaction history. We establish a new supervision principle that shifts the object of knowledge transfer from trajectory behavior to symbolic task structure.
- We propose VTD, in which the teacher decomposes the task into a symbolic progress graph before interaction. Environment verification turns milestone completion into verified progress for policy optimization, while graph distillation makes task decomposition a student capability.
- We demonstrate that VTD outperforms the strongest Qwen2.5-7B baselines by 1.8, 3.4, and 8.0 points on ALFWorld, WebShop, and TextCraft, respectively. VTD also exhibits strong structural generalization, reaching 92.8% success on held-out ALFWorld task families compared with 68.0% for the strongest baseline and extending to deeper TextCraft recipe hierarchies.

2 Related Work

On-policy distillation for language agents. On-policy distillation (OPD) uses teacher token probabilities to supervise sequences sampled from the current student policy (Gu et al., 2024; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025). Self-distillation extends this paradigm by conditioning the teacher branch on environment feedback (Hübotter et al., 2026), demonstrations (Shenfeld et al., 2026), or reference solutions (Zhao et al., 2026), turning training-only information into targets for the student policy. For multi-turn agents, TCOD (Wang et al., 2026b) and Guided-OPD (Li et al., 2026) control how much of each rollout is generated by the student through temporal curricula and teacher intervention. Zhu et al. (2026) analyze failure modes in the distillation objective and evaluate several fixes, while SDAR (Lu et al., 2026) and SOD (Zhong et al., 2026) allocate supervision across tokens and turns. CCOPD (Lin et al., 2026) instead evaluates the student's answer prefixes under a canonical teacher context. Despite these variations, existing methods still distill teacher behavior on

the outputs that the student generates. VTD instead distills task structure, transferring the teacher's symbolic decomposition of the task as a progress graph before any trajectory is generated.

Task decomposition and structured agent learning. Language agents organize long-horizon execution through plans and reusable skills (Ichter et al., 2023; Huang et al., 2023; Yuan et al., 2023; Wang et al., 2024). HiPER (Peng et al., 2026) organizes interaction through dynamic subgoals, BEACON (Wang et al., 2026d) rewards intermediate milestones, and StraTA (Xue et al., 2026) generates a global strategy before acting. Self-Guide (Wang et al., 2026c) produces verbal guidance and internal rewards during interaction. A parallel line represents temporally extended objectives with temporal logic and reward machines, using symbolic states and transitions to define task progress and compositional structure (Toro Icarte et al., 2018; Camacho et al., 2019; Toro Icarte et al., 2022; Vaezipoor et al., 2021; Nangue Tasse et al., 2024; Yalcinkaya et al., 2024). Recent neurosymbolic methods construct executable task representations from language (Ma et al., 2024; Xie et al., 2024; Creus Castanyer et al., 2026) or learn them from interaction data (Li et al., 2025; Kheirandish et al., 2025). Language-level decomposition leaves task progress in free-form text, whereas milestone rewards and formal specifications keep executable progress semantics outside the policy. VTD bridges this divide by training the language policy to construct verifiable task structure.

3 Problem Statement

In this section, we first formalize outcome-based policy optimization and on-policy distillation (OPD) for long-horizon agents. We then reveal a characteristic failure mode of such multi-turn teacher supervision, in which dense feedback loses its corrective value as the interaction proceeds.

Outcome-based agent learning. We consider an agent policy π_θ that receives a task instruction x and interacts with an environment for T turns. At turn t, the agent receives an observation o_t and samples an action a_t from the policy conditioned on its interaction history:

$$h_t = (x, o_0, a_0, o_1, \ldots, a_{t-1}, o_t), \qquad a_t \sim \pi_\theta(\cdot \mid h_t). \tag{1}$$

The environment executes a_t and returns o_{t+1}, which becomes part of the interaction history for the next turn. The interaction produces a trajectory $\tau = (x, o_0, a_0, o_1, \ldots, a_{T-1}, o_T)$, and the environment returns a terminal success indicator $y(\tau) \in \{0, 1\}$. Outcome-based methods such as GRPO use this terminal outcome to optimize the policy (Shao et al., 2024; Chen et al., 2025; Wang et al., 2025). We represent the underlying outcome objective as

$$\mathcal{J}_{\text{out}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[y(\tau)\right]. \tag{2}$$

All failed trajectories therefore have the same outcome regardless of how much progress they make.

On-policy distillation. OPD (Gu et al., 2024; Agarwal et al., 2024; Lu & Thinking Machines Lab, 2025) supplements this outcome-based objective with token-level guidance from a teacher policy π_ϕ on trajectories generated by the student policy π_θ. To define this guidance, we write each action as an autoregressive token sequence $a_t = (a_{t,1}, \ldots, a_{t,L_t})$ of length L_t, where $c_{t,j} = (h_t, a_{t,<j})$ denotes the prediction context of token $a_{t,j}$. At each context, OPD uses the teacher distribution $\pi_\phi(\cdot \mid c_{t,j})$ as the target for the student distribution $\pi_\theta(\cdot \mid c_{t,j})$. One representative objective minimizes the reverse Kullback–Leibler (KL) divergence (Gu et al., 2024; Agarwal et al., 2024):

$$\mathcal{L}_{\text{OPD}}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\frac{1}{N_\tau}\sum_{t=0}^{T-1}\sum_{j=1}^{L_t} D_{\text{KL}}\left(\pi_\theta(\cdot \mid c_{t,j}) \,\|\, \pi_\phi(\cdot \mid c_{t,j})\right)\right]. \tag{3}$$

Here $N_\tau = \sum_{t=0}^{T-1} L_t$ is the number of action tokens in the trajectory. In a multi-turn task, however, every context $c_{t,j}$ contains the interaction history h_t. Because this history contains the student's preceding actions and the resulting observations, even a teacher with fixed parameters produces later predictions that depend on the student's earlier decisions.

Teacher feedback becomes less corrective along failed trajectories. To determine whether a capable teacher can still provide corrective guidance on a failed student history, we identify 128 ALFWorld (Shridhar et al., 2021) tasks that the student fails but an adapted 1.5B teacher and a

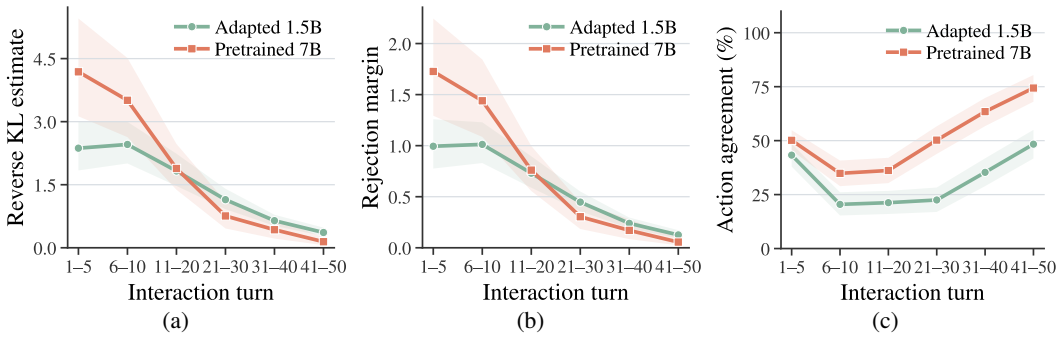

Figure 1: **Teacher–student divergence and action agreement along failed ALFWorld trajectories.** We evaluate a Qwen2.5-1.5B-Instruct teacher adapted with GRPO (Shao et al., 2024) and a pretrained Qwen2.5-7B-Instruct teacher at every prefix of failed 50-turn trajectories of the base Qwen2.5-1.5B-Instruct student (Qwen Team, 2024) on 128 tasks that both teachers solve independently. Panel (a) shows the reverse KL estimate on action tokens sampled by the student. Panel (b) shows the logit margin between the teacher's preferred token and the token selected by the student. Panel (c) shows whether the teacher and student assign the highest score to the same complete admissible action. Lines show task means, and shaded bands show 95% task-bootstrap confidence intervals.

pretrained 7B teacher solve independently. For each task, we hold the student's 50-turn failed trajectory fixed and evaluate both teachers at every turn using the same history prefix and set of admissible actions. We measure how strongly each teacher disagrees with the student's actions at the level of action tokens and complete admissible actions (Appendix A). Figures 1a and 1b show that this token-level disagreement declines substantially as the interaction proceeds. In turns 41–50, the reverse KL estimate falls to 0.36 for the adapted teacher and 0.14 for the pretrained teacher, while their rejection margins fall to 0.13 and 0.06. Since the reverse KL estimate evaluates the OPD objective in Eq. (3) on these tokens, this decline directly weakens the learning signal that OPD provides in later turns. Figure 1c shows that action agreement between the student and each teacher drops to a minimum in turns 6–20 before rising to 48.4% and 74.4%, respectively, in turns 41–50. Since every underlying student trajectory remains unsuccessful, the late agreement cannot be explained by the student acquiring the teachers' task-solving ability. Conditioning on the failed history therefore progressively removes the contrast that makes teacher feedback corrective.

4 Methodology

In this section, we introduce VTD, a framework for learning long-horizon agent policies with symbolic task structure, as illustrated in Figure 2. First, we formalize the teacher's task decomposition as a symbolic progress graph constructed before interaction. Next, we define environment-verified progress from the graph milestones completed along each trajectory. Finally, we formulate complementary objectives that optimize the policy with verified progress and transfer the task decomposition capability to the student through graph distillation.

4.1 Constructing Symbolic Progress Graphs

Supervision constructed before interaction must describe task progress for any trajectory the student may generate. To turn this description into a learning signal, the environment must determine how much of the task each trajectory completes. VTD therefore uses a small predicate vocabulary $\mathcal{P}$, defined once for each environment, to express verifiable conditions of task progress. Each predicate specifies an argument schema and a deterministic verification rule. Instantiating a predicate with arguments from the task instruction produces a milestone. For example, ACQUIRE(`apple`) is satisfied when the agent holds an apple. The same predicate definitions apply across tasks and trajectories, giving the teacher a fixed symbolic language for task decomposition.

Individual milestones identify conditions of progress, but a task decomposition must also specify how those conditions depend on one another. Before interaction begins, the teacher π_ϕ constructs a symbolic progress graph $G_x = (V_x, E_x)$ from the task instruction x and predicate vocabulary $\mathcal{P}$. Each node $v \in V_x$ is a milestone that instantiates one predicate in $\mathcal{P}$ with arguments from x. An

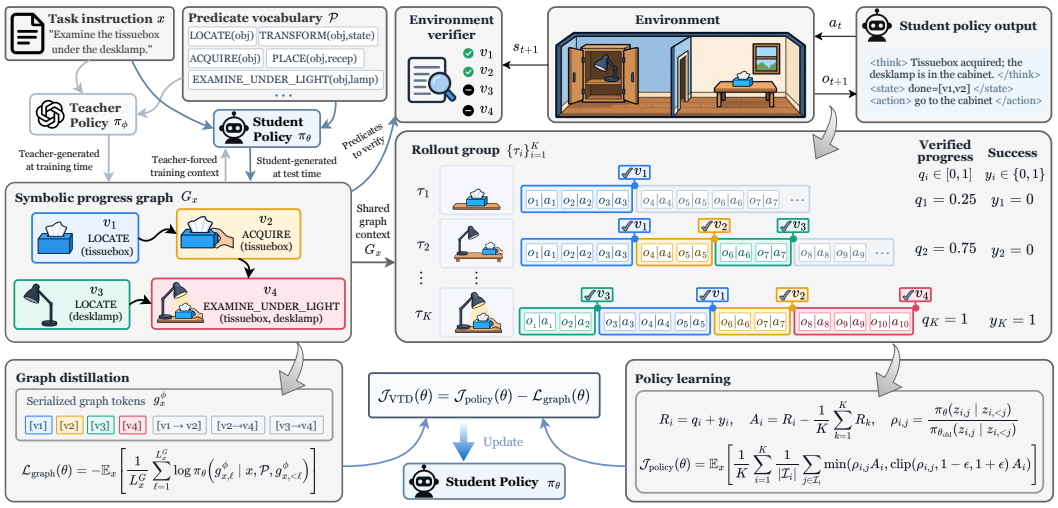

Figure 2: **Overview of VTD.** Given a task instruction and predicate vocabulary, the teacher constructs the symbolic progress graph used during training, while the student generates its own graph without teacher access at test time. The rollout group shares the graph, and the environment verifier records completed milestones to compute verified progress q_i alongside task success y_i. Graph distillation on graph tokens and group-relative policy learning on interaction tokens jointly update the student.

edge $(u, v) \in E_x$ indicates that milestone u is a prerequisite for milestone v. Together, the nodes and edges give the student a structured description of what constitutes progress and how these conditions are related. The resulting G_x is the only teacher output used for x, so VTD requires neither the teacher's token distributions nor any teacher query during interaction. Every student rollout receives this graph before its first environment action, so all trajectories share the same milestone definitions and dependency structure. The graph-construction prompts are provided in Appendix E.

4.2 Verifying Progress during Interaction

The symbolic progress graph defines what constitutes progress, but policy learning requires determining which milestones the student's trajectory actually satisfies. At each turn, an environment-specific verifier evaluates the predicate and arguments of every graph node against the current environment state. Let $s_{i,t}$ denote the environment state at turn t of rollout τ_i, with $s_{i,0}$ denoting the initial state. We write $\text{Verify}(v, s_{i,t}) = 1$ when node v is satisfied in $s_{i,t}$, and 0 otherwise. VTD records a milestone when its predicate first becomes true:

$$M_{i,t} = \{v \in V_x : \exists\, 0 \le t' \le t,\ \text{Verify}(v, s_{i,t'}) = 1\}, \tag{4}$$

where $M_{i,t}$ is the set of milestones verified by turn t. Once a milestone enters $M_{i,t}$, it remains completed even if its underlying condition changes later in the rollout. Appendix B lists the graph schema, predicate vocabularies, and verification conditions.

During interaction, the milestones and dependency edges of the graph remain in the student's context for action selection. Before each environment action, the student outputs three structured blocks:

`<think>...</think><state>done=[` $v_1, \ldots, v_k$ `]</state><action>...</action>`

The `<think>` block contains the student's reasoning, the `<state>` block lists the graph-node identifiers that the student believes are complete before acting, and the `<action>` block contains the selected environment action. The state report remains in the interaction history, allowing subsequent decisions to condition on the student's current estimate of task progress. The environment verifier computes $M_{i,t}$ independently of these reports, and its results never enter the student's context.

For a rollout ending at turn T_i, VTD summarizes milestone achievement as verified progress:

$$q_i = \frac{|M_{i,T_i}|}{|V_x|} \in [0, 1]. \tag{5}$$

Normalizing by $|V_x|$ places graphs of different sizes on a common scale. By accumulating verified milestones, q_i distinguishes unsuccessful trajectories that the terminal outcome treats identically.

4.3 Policy Optimization and Graph Distillation

Verified progress records how far a trajectory advances, while terminal success records whether the trajectory completes the task. For each instruction x, the rollout policy $\pi_{\theta_{\text{old}}}$ samples K trajectories $\{\tau_i\}_{i=1}^{K}$ under the shared graph G_x. We define the trajectory score and group-relative advantage as

$$R_i = y(\tau_i) + q_i, \qquad A_i = R_i - \frac{1}{K}\sum_{k=1}^{K} R_k. \tag{6}$$

We omit standard-deviation normalization because it can disproportionately amplify gradients from low-variance rollout groups (Liu et al., 2025; Feng et al., 2025). When every trajectory in the group has $y(\tau_i) = 0$, variation in q_i still produces relative advantages among trajectories that verify different numbers of milestones. Let $z_{i,j}$ denote the token at position j in the serialized context of rollout τ_i, let $\mathbf{z}_{i,<j}$ denote its complete preceding prefix, and let $\rho_{i,j} = \pi_\theta(z_{i,j} \mid \mathbf{z}_{i,<j}) / \pi_{\theta_{\text{old}}}(z_{i,j} \mid \mathbf{z}_{i,<j})$ denote the corresponding importance ratio. We use $\mathcal{I}_i$ to index the positions generated by the student after graph construction. We then maximize the clipped surrogate objective (Schulman et al., 2017):

$$\mathcal{J}_{\text{policy}}(\theta) = \mathbb{E}_x \left[\frac{1}{K}\sum_{i=1}^{K} \frac{1}{|\mathcal{I}_i|} \sum_{j \in \mathcal{I}_i} \min(\rho_{i,j} A_i, \text{clip}(\rho_{i,j}, 1-\epsilon, 1+\epsilon)\, A_i) \right], \tag{7}$$

where ϵ is the clipping threshold, and the inner average over $\mathcal{I}_i$ gives every rollout equal weight regardless of response length (Shao et al., 2024). The implementation further applies dual clipping (Ye et al., 2020) and a reference-policy KL penalty to the tokens in $\mathcal{I}_i$ (Appendix C).

Verified progress provides supervision for trajectory execution under G_x. Graph distillation complements this signal by making the construction of symbolic progress graphs a capability of the student policy. Let $\mathbf{g}_x^\phi = (g_{x,1}^\phi, \ldots, g_{x,L_x^G}^\phi)$ denote the length-L_x^G token sequence representing the graph constructed by the teacher. Averaged over training instructions, graph distillation minimizes

$$\mathcal{L}_{\text{graph}}(\theta) = -\mathbb{E}_x \left[\frac{1}{L_x^G} \sum_{\ell=1}^{L_x^G} \log \pi_\theta\left(g_{x,\ell}^\phi \mid x, \mathcal{P}, \mathbf{g}_{x,<\ell}^\phi\right) \right]. \tag{8}$$

The graph loss is evaluated only on the teacher-forced graph tokens, whereas $\mathcal{J}_{\text{policy}}$ is evaluated only on tokens generated during interaction. These disjoint token scopes preserve the separation between task decomposition and trajectory execution within a single policy. The combined objective is

$$\mathcal{J}_{\text{VTD}}(\theta) = \mathcal{J}_{\text{policy}}(\theta) - \mathcal{L}_{\text{graph}}(\theta). \tag{9}$$

At test time, the student generates its own graph and executes the task without teacher access.

5 Experiments

In this section, we compare VTD with existing agentic reinforcement learning (RL) methods on three benchmarks and two model scales. We further analyze how environment-verified progress shapes policy learning and how graph distillation transfers the task decomposition capability to the student.

5.1 Experimental Setup

Models and benchmarks. We use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct (Qwen Team, 2024) as the base models for policy learning. We evaluate VTD on the widely used long-horizon benchmarks ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and TextCraft (Prasad et al., 2024). ALFWorld contains 6 families of household tasks that require a sequence of grounded interactions to complete. WebShop requires an agent to navigate a simulated shopping website and purchase a product that satisfies the constraints in a natural-language instruction. TextCraft requires an agent to craft a target item from Minecraft recipes by collecting raw materials and crafting the intermediate items that each recipe requires. Together, these benchmarks test the same framework across embodied interaction, web navigation, and compositional crafting.

Table 1: **Overall performance on ALFWorld, WebShop, and TextCraft.** We report task success rate (%) for Qwen2.5-Instruct at the 1.5B and 7B scales, with ALFWorld results by task family. Baseline results marked with † are obtained with the public implementations, and the others are reported by prior work (Section 5.1). The † entries and the VTD entries show the mean and standard deviation over 3 random seeds. Best and second-best results at each scale are bolded and underlined.

Method	ALFWorld							WebShop	TextCraft
	Pick	Look	Clean	Heat	Cool	Pick2	All	Success	Success
Qwen2.5-1.5B-Instruct									
GRPO	85.3 ± 1.5	53.7 ± 8.0	84.5 ± 6.8	78.2 ± 7.9	59.7 ± 5.0	53.5 ± 5.6	72.8 ± 3.6	56.8 ± 3.8	43.8 ± 2.2
GiGPO	94.4 ± 5.9	67.5 ± 4.6	94.8 ± 3.8	<u>94.4</u> ± 7.8	79.8 ± 4.7	76.4 ± 5.4	86.7 ± 1.7	67.4 ± 4.5	<u>58.3</u> ± 7.8
BEACON	**100.0**	<u>88.2</u>	86.7	**100.0**	78.9	**92.9**	91.4	<u>75.6</u>	15.2 ± 4.6†
HiPER	<u>98.9</u> ± 1.9	**91.7** ± 14.4	<u>97.5</u> ± 4.3	90.9 ± 4.7	**96.7** ± 2.8	<u>91.3</u> ± 9.6	**95.3** ± 1.4	71.4 ± 9.0	0.0 ± 0.0†
SDAR†	77.4 ± 6.6	66.7 ± 11.5	80.6 ± 2.9	65.7 ± 1.7	81.3 ± 2.3	66.7 ± 3.6	75.0 ± 2.8	60.9 ± 3.6	54.4 ± 2.2
VTD (Ours)	**100.0** ± 0.0	82.2 ± 1.9	**100.0** ± 0.0	93.0 ± 0.3	<u>84.5</u> ± 2.1	84.1 ± 2.8	<u>92.2</u> ± 0.8	**79.4** ± 0.5	**83.5** ± 1.3
Qwen2.5-7B-Instruct									
GRPO	90.8 ± 5.1	66.1 ± 6.7	89.3 ± 5.4	74.7 ± 6.9	72.5 ± 5.4	64.7 ± 7.3	77.6 ± 5.2	66.1 ± 3.7	76.5 ± 3.1
GiGPO	<u>97.7</u> ± 1.6	82.7 ± 7.9	<u>98.8</u> ± 1.6	83.7 ± 7.2	89.3 ± 8.2	79.2 ± 6.6	90.8 ± 1.3	75.2 ± 3.8	87.0 ± 2.4
BEACON	**100.0**	81.8	96.3	92.9	<u>94.7</u>	90.0	94.5	79.7	41.4 ± 1.9†
HiPER	**100.0** ± 0.0	<u>84.8</u> ± 2.6	**100.0** ± 0.0	<u>96.3</u> ± 6.4	**100.0** ± 0.0	**95.5** ± 4.4	<u>97.4</u> ± 1.6	<u>83.3</u> ± 0.9	<u>89.0</u> ± 2.6†
SDAR	94.7	75.0	**100.0**	86.7	68.2	78.9	85.9	82.8	69.6 ± 2.2†
VTD (Ours)	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	<u>95.2</u> ± 0.2	**99.2** ± 0.0	**86.7** ± 2.1	**97.0** ± 1.9

Baselines. To make our performance comparison informative and comprehensive, we include existing state-of-the-art (SoTA) baselines spanning the main paradigms of long-horizon agent learning. GRPO (Shao et al., 2024) represents outcome-based policy optimization, and GiGPO (Feng et al., 2025) extends this paradigm with step-level credit assignment. BEACON (Wang et al., 2026d) and HiPER (Peng et al., 2026) represent structured credit assignment through milestone rewards and hierarchical subgoals, respectively. SDAR (Lu et al., 2026) represents on-policy distillation through self-distillation, because teacher-based OPD requires teacher token distributions that the black-box teacher of VTD does not provide. On ALFWorld and WebShop, we use the GRPO results reported by GiGPO and the published results of the remaining methods. Because SDAR does not report Qwen2.5-1.5B results, we train it at this scale using its public implementation. On TextCraft, we use the GRPO and GiGPO results reported by BiPACE (Wang et al., 2026a) and train BEACON, HiPER, and SDAR with their public implementations under the same evaluation protocol as VTD.

Training and evaluation. We use GPT-5.5 with medium reasoning effort as the teacher. We set the rollout group size to 8 and the training batch size to 16 prompts for ALFWorld and TextCraft and 32 prompts for WebShop. Unless stated otherwise, we train each policy for 200 training steps using AdamW (Loshchilov & Hutter, 2019) with a constant learning rate of 10^{-6} on ALFWorld and WebShop and 2×10^{-6} on TextCraft. The rollout temperature is set to 1.0 during training and 0.4 during evaluation. We use the same VTD objective and coefficients across environments and model scales. We report the mean and standard deviation over 3 random seeds. See Appendix C for details.

5.2 Experimental Results

Overall performance. Table 1 compares VTD with existing SoTA agentic RL baselines. With Qwen2.5-7B-Instruct, VTD achieves 99.2% success on ALFWorld, 86.7% on WebShop, and 97.0% on TextCraft, outperforming the strongest baseline by 1.8, 3.4, and 8.0 points, respectively. At the 1.5B scale, VTD reaches 92.2% success on ALFWorld and leads all compared methods on WebShop and TextCraft with 79.4% and 83.5%. The largest ALFWorld advantage at the 7B scale appears on the Look task, where VTD reaches 100.0% success compared with 84.8% for the strongest baseline. A Look task requires locating the target object and a light source and examining the object under the activated light, so success depends on several interdependent conditions over multiple turns. Similarly, WebShop requires the agent to track multiple product constraints across search results and product pages before completing a purchase. TextCraft further requires the agent to follow a hierarchy of recipes, in which every intermediate item must be crafted from its own ingredients before the target

Table 2: **Task structure generalizes beyond the training distribution.** All methods use Qwen2.5-7B-Instruct and train for 150 steps on the 4 task families in the left block, and training contains no CLEAN tasks or HEAT tasks. We report task success rate (%) on the complete 274-task evaluation set, which contains 177 tasks from the training families and 97 held-out tasks. Best and second-best results are bolded and underlined.

Method	Training families					Held-out families			All tasks
	Pick	Pick2	Look	Cool	All	Clean	Heat	All	Overall
GRPO	86.4	65.9	58.1	69.6	72.3	51.7	38.5	46.4	63.1
GiGPO	<u>93.2</u>	<u>92.7</u>	80.6	<u>87.0</u>	<u>89.3</u>	<u>72.4</u>	43.6	60.8	79.2
BEACON	86.4	78.0	<u>90.3</u>	**93.5**	87.0	67.2	69.2	<u>68.0</u>	<u>80.3</u>
HiPER	89.8	82.9	71.0	80.4	82.5	6.9	**97.4**	43.3	68.6
SDAR	86.4	46.3	80.6	65.2	70.6	55.2	56.4	55.7	65.3
VTD (Ours)	**100.0**	**100.0**	**100.0**	**93.5**	**98.3**	**91.4**	<u>94.9</u>	**92.8**	**96.4**

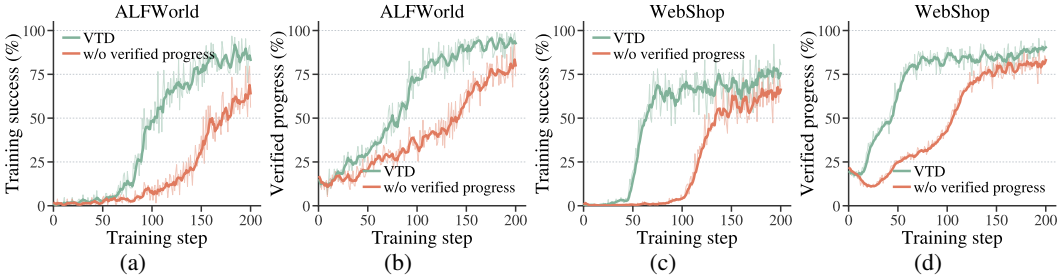

Figure 3: **Verified progress accelerates agent learning.** We compare Qwen2.5-1.5B-Instruct VTD with a matched variant trained without using verified progress. Panels (a) and (b) show task success and verified progress on ALFWorld, while panels (c) and (d) show the corresponding curves on WebShop. Pale lines show per-step batch means, and dark lines show 5-step trailing averages.

item can be crafted. The symbolic progress graph lays out this hierarchy before the first action, and VTD achieves its largest margins on TextCraft, surpassing the strongest baseline by 25.2 points at the 1.5B scale. Among the baselines, SDAR, which applies on-policy self-distillation along the student's own trajectories, trails VTD on every benchmark at both scales, in line with the history dependence analyzed in Section 3. HiPER fails to learn TextCraft at the 1.5B scale because its policy almost never samples a successful trajectory, so its hierarchical advantages receive no task reward and the policy settles into repeating a single action that the environment rejects. BEACON improves early in training, but its milestone rewards produce large advantages that gradually lengthen its outputs. Many turns then reach the 512-token limit per turn before completing an action, and BEACON's success declines thereafter. Taken together, these results establish symbolic task decomposition as an effective basis for learning across different forms of multi-stage interaction.

Task structure generalizes beyond the training distribution. A learned task decomposition should remain useful beyond the task families that supply policy trajectories. We train all Qwen2.5-7B-Instruct policies for 150 steps on the 4 ALFWorld task families in the left block of Table 2, excluding all CLEAN and HEAT tasks, and evaluate every method on the complete 274-task set. VTD retains 98.3% success on the training families and reaches 92.8% on the held-out families, outperforming BEACON, the strongest baseline at 68.0%, by 24.8 points. The family-level results distinguish broad generalization from family-specific behavior. VTD reaches 91.4% on CLEAN tasks and 94.9% on HEAT tasks, whereas HiPER reaches 97.4% on HEAT tasks but only 6.9% on CLEAN tasks despite achieving 82.5% on the training families. Both held-out families involve object states and interactions that no training task requires, such as cleaning an object in a sink basin or heating it in a microwave. VTD constructs a symbolic progress graph from each new instruction, so the policy pursues the required object state as an explicit milestone without having practiced the interaction that produces this state. The learned task structure also extends to deeper hierarchies. On TextCraft, a

Qwen2.5-1.5B VTD policy trained only on items with 2 levels of crafting reaches 37.9% success on 3-level items, compared with 21.8% for the strongest baseline trained on all levels (Appendix D).

Verified progress accelerates agent learning. To isolate the role of verified progress, we compare VTD with a matched variant that does not use verified progress to guide policy learning. Figures 3a and 3c show that VTD learns successful task execution substantially earlier on ALFWorld and WebShop, whereas the matched variant remains near zero for a much longer portion of training. Figures 3b and 3d explain this advantage. VTD completes a larger fraction of the progress graph before the corresponding rise in task success, so it first learns to satisfy intermediate task requirements and then converts this progress into successful execution. Table 3 shows that this learning advantage persists at test time, where VTD reaches 92.2% success compared with 66.4% on ALFWorld and 79.4% compared with 62.2% on WebShop, with higher verified progress on both benchmarks.

Table 3: **Evaluation of policies trained with and without verified progress.** The w/o progress variant uses task success alone as the trajectory score. We evaluate Qwen2.5-1.5B-Instruct after 200 training steps and report success rate and verified progress (%) as mean ± standard deviation over 3 random seeds. Each policy generates its own graph without teacher access.

Benchmark	Variant	Success	Verified progress
ALFWorld	VTD	**92.2** ± 0.8	**97.0** ± 0.5
	w/o progress	66.4 ± 2.8	78.7 ± 1.8
WebShop	VTD	**79.4** ± 0.5	**90.9** ± 0.1
	w/o progress	62.2 ± 1.2	81.2 ± 0.8

Graph distillation internalizes task structure. During training, the teacher supplies the symbolic progress graph, whereas at test time the student must construct the graph before acting. We test whether the student acquires this capability by comparing VTD with a matched variant trained without graph distillation. Figure 4a shows that the graph-generation negative log-likelihood (NLL) rapidly decreases under VTD but rises without graph distillation. The opposing trends show that acting under a teacher graph does not by itself teach the student to generate one. Figure 4b reveals the difference between trajectory execution and task decomposition. Without graph distillation, the student still solves 83.3% of the tasks, yet none of its outputs forms a structurally accepted graph (Appendix B) and no graph milestones can be verified. VTD produces structurally accepted graphs for 100.0% of the tasks, reaches 97.0% verified progress, and raises task success to 92.2%. Graph distillation therefore makes the teacher's task decomposition a student capability.

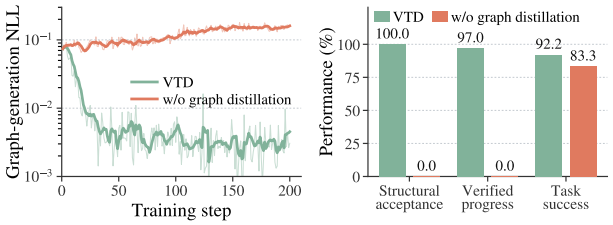

Figure 4: **Graph distillation transfers task decomposition to the student.** We compare Qwen2.5-1.5B-Instruct VTD with a matched variant trained without graph distillation on ALFWorld. Panel (a) shows the graph-generation NLL during training, with per-step values in pale lines and 5-step trailing averages in dark lines. Panel (b) reports structural graph acceptance, verified progress, and task success under student-generated graphs.

6 Conclusion

In this work, we introduced VTD, a framework in which the teacher decomposes a task into a symbolic progress graph before interaction and the student learns task execution through policy optimization. With Qwen2.5-7B, VTD outperforms the strongest baselines on ALFWorld, WebShop, and TextCraft. Our results show that environment-verified progress turns partial task completion into an effective signal that accelerates policy learning. Graph distillation enables the student to internalize the teacher's knowledge as a symbolic representation of task progress. With this internalized structure, VTD exhibits strong structural generalization beyond the training distribution. We acknowledge that the current formulation relies on a predicate vocabulary that the environment can verify and constructs the task decomposition before interaction. A natural direction for future work is to enable VTD to learn such predicates from interaction and to revise symbolic progress graphs as new task information emerges during execution.

AI USE STATEMENT

In this work, we used generative AI tools to improve the wording and readability of the manuscript. All scientific work was conducted and verified by the authors, who reviewed all AI-assisted edits and take responsibility for the final content.

REPRODUCIBILITY STATEMENT

Section 4 defines VTD and its learning objectives, and Section 5.1 describes the experimental protocol. The appendix provides the implementation details required to reproduce the results. Appendix B specifies the graph schema, predicate vocabularies, and environment verification conditions. Appendix C reports the optimization settings, including dual clipping and the reference-policy KL penalty, together with the compute and evaluation details. Appendix E provides the graph-construction prompts. We will release the code upon acceptance.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5be69a584901a26c521c2b51e40a4c20-Paper-Conference.pdf.

Alberto Camacho, Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. LTL and beyond: Formal languages for reward function specification in reinforcement learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 6065–6073. International Joint Conferences on Artificial Intelligence Organization, July 2019. URL https://doi.org/10.24963/ijcai.2019/840.

Kevin Chen, Marco Cusumano-Towner, Brody Huval, Aleksei Petrenko, Jackson Hamburger, Vladlen Koltun, and Philipp Krähenbühl. Reinforcement learning for long-horizon interactive LLM agents. *arXiv preprint arXiv:2502.01600*, 2025. URL https://arxiv.org/abs/2502.01600.

Roger Creus Castanyer, Faisal Mohamed, Pablo Samuel Castro, Cyrus Neary, and Glen Berseth. ARM-FM: Automated reward machines via foundation models for compositional reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 64154–64191, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/68f24fad8f460d4b7bbac402c252ebbb-Paper-Conference.pdf.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 46375–46408. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/420c9f777c0b4f78d515e53cf74d58b2-Paper-Conference.pdf.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 32694–32717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/8ac015d409635f196f9e3e9dcfb9a94e-Paper-Conference.pdf.

Shuo He, Lang Feng, Qi Wei, Xin Cheng, Lei Feng, and Bo An. Hierarchy-of-groups policy optimization for long-horizon agentic tasks. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 27572–27593, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/2e7eb932f93d5db3b61c7a001e72fd0f-Paper-Conference.pdf.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner Monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2023. URL https://proceedings.mlr.press/v205/huang23c.html.

Jonas Hübotter, Frederike Lübeck, Lejs Deen Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=QkfkxyRizZ.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as I can, not as I say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Alireza Kheirandish, Duo Xu, and Faramarz Fekri. LLM-augmented symbolic RL with landmark-based task decomposition. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, April 2025. URL https://doi.org/10.1109/ICASSP49660.2025.10889431.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Andrew Li, Toryn Klassen, Andrew Wang, Parand A. Alamdari, and Sheila McIlraith. Ground-compose-reinforce: Grounding language in agentic behaviours using limited data. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 160838–160874. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/eba22e306ead56b4ef108a615c6dad4a-Paper-Conference.pdf.

Gengsheng Li, Mao Zheng, Mingyang Song, Ruiqi Liu, Tianyu Yang, Jie Sun, Qiyong Zhong, Haiyun Guo, Junfeng Fang, Dan Zhang, and Jinqiao Wang. On-policy distillation with curriculum turn-level guidance for multi-turn agents. *arXiv preprint arXiv:2606.15912*, 2026. URL https://arxiv.org/abs/2606.15912.

Zizhuo Lin, Quanling Liu, Jinsheng Quan, Chao Zhang, Yifan Zhu, Xing Shi, Jingtao Xu, Zhihui Li, and Yawei Luo. Same evidence, different answers: Canonical-context on-policy distillation for multi-turn language models. *arXiv preprint arXiv:2605.30251*, 2026. URL https://arxiv.org/abs/2605.30251.

Xiaoqian Liu, Ke Wang, Yuchuan Wu, Fei Huang, Yongbin Li, Jianbin Jiao, and Junge Zhang. Agentic reinforcement learning with implicit step rewards. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 129271–129291, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d1e6739f319be4ba8e748cc05f23bba1-Paper-Conference.pdf.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. URL https://thinkingmachines.ai/blog/on-policy-distillation.

Zhengxi Lu, Zhiyuan Yao, Zhuowen Han, Zi-Han Wang, Jinyang Wu, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Self-distilled agentic reinforcement learning. *arXiv preprint arXiv:2605.15155*, 2026. URL https://arxiv.org/abs/2605.15155.

Xufang Luo, Yuge Zhang, Zhiyuan He, Zilong Wang, Siyun Zhao, Dongsheng Li, Luna K. Qiu, and Yuqing Yang. Agent Lightning: Train ANY AI agents with reinforcement learning. *arXiv preprint arXiv:2508.03680*, 2025. URL https://arxiv.org/abs/2508.03680.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 26516–26560, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/70c26937fbf3d4600b69a129031b66ec-Paper-Conference.pdf.

Geraud Nangue Tasse, Devon Jarvis, Steven James, and Benjamin Rosman. Skill machines: Temporal logic skill composition in reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 36937–36956, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9ee3a664ccfeabc0da16ac6f1f1cfe59-Paper-Conference.pdf.

Jiangweizhi Peng, Yuanxin Liu, Ruida Zhou, Charles Fleming, Zhaoran Wang, Alfredo Garcia, and Mingyi Hong. HiPER: Hierarchical reinforcement learning with explicit credit assignment for large language model agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=RJWQcgd3hX.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4226–4252, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-naacl.264/.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. URL https://arxiv.org/abs/2402.03300.

Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=qA6FgH0nnZ.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. URL https://doi.org/10.1145/3689031.3696075.

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

Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2107–2116. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/icarte18a.html.

Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, and Sheila A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, January 2022. ISSN 1076-9757. URL https://doi.org/10.1613/jair.1.12440.

Pashootan Vaezipoor, Andrew C Li, Rodrigo A. Toro Icarte, and Sheila A. McIlraith. LTL2Action: Generalizing LTL instructions for multi-task RL. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10497–10508. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/vaezipoor21a.html.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Hanyang Wang, Weijieying Ren, Yuxiang Zhang, Ding Cao, Zhizhao Zeng, Ke Zeng, and Tianxiang Zhao. BiPACE: Bisimulation-guided policy optimization with action counterfactual estimation for LLM agents. *arXiv preprint arXiv:2606.25556*, 2026a. URL https://arxiv.org/abs/2606.25556.

Jiaqi Wang, Wenhao Zhang, Weijie Shi, Yaliang Li, and James Cheng. Exploring temporal curriculum in on-policy distillation for multi-turn autonomous agents. In *Third Conference on Language Modeling*, 2026b. URL https://openreview.net/forum?id=KH9eJRz8WR.

Xinyu Wang, Hanwei Wu, Jingwei Song, Shuyuan Zhang, Jiayi Zhang, Fanqi Kong, Tung Sum Thomas Kwok, Xiao-Wen Chang, Yuyu Luo, Chenglin Wu, and Bang Liu. Co-evolution of policy and internal reward for language agents. *arXiv preprint arXiv:2604.03098*, 2026c. URL https://arxiv.org/abs/2604.03098.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: Understanding self-evolution in LLM agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025. URL https://arxiv.org/abs/2504.20073.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Milestone-guided policy learning for long-horizon language agents. In *Forty-third International Conference on Machine Learning*, 2026d. URL https://openreview.net/forum?id=Ga3AR4EF6R.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: An open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 12685–12727, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1571ce1ff735be4551db50887043726e-Paper-Conference.pdf.

Zhiheng Xi, Chenyang Liao, Guanyu Li, Zhihao Zhang, Wenxiang Chen, Binghai Wang, Senjie Jin, Yuhao Zhou, Jian Guan, Wei Wu, Tao Ji, Tao Gui, Qi Zhang, and Xuanjing Huang. AgentPRM: Process reward models for LLM agents via step-wise promise and progress. In *Proceedings of the ACM Web Conference 2026*, pp. 4184–4195. ACM, April 2026b. URL https://doi.org/10.1145/3774904.3792551.

Tianbao Xie, Siheng Zhao, Chen Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Reward shaping with language models for reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35663–35699, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/9941833e8327910ef25daeb9005e4748-Paper-Conference.pdf.

Xiangyuan Xue, Yifan Zhou, ZiDong Wang, Shengji Tang, Philip Torr, Wanli Ouyang, Lei Bai, and Zhenfei Yin. StraTA: Incentivizing agentic reinforcement learning with strategic trajectory abstraction. In *Workshop on Failure Modes of Agentic AI at ICML 2026*, 2026. URL https://openreview.net/forum?id=XD3e8vjKWz.

Beyazit Yalcinkaya, Niklas Lauffer, Marcell Vazquez-Chanlatte, and Sanjit Seshia. Compositional automata embeddings for goal-conditioned reinforcement learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 72933–72963. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/858fc542b70d3b39067f7d3b1cd77635-Paper-Conference.pdf.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20744–20757. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf.

Deheng Ye, Zhao Liu, Mingfei Sun, Bei Shi, Peilin Zhao, Hao Wu, Hongsheng Yu, Shaojie Yang, Xipeng Wu, Qingwei Guo, Qiaobo Chen, Yinyuting Yin, Hao Zhang, Tengfei Shi, Liang Wang, Qiang Fu, Wei Yang, and Lanxiao Huang. Mastering complex control in MOBA games with deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04): 6672–6679, April 2020. URL https://doi.org/10.1609/aaai.v34i04.6144.

Haoqi Yuan, Chi Zhang, Hongcheng Wang, Feiyang Xie, Penglin Cai, Hao Dong, and Zongqing Lu. Skill reinforcement learning and planning for open-world long-horizon tasks. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023. URL https://openreview.net/forum?id=tTZVrnr45N.

Zijing Zhang, Ziyang Chen, Mingxiao Li, Zhaopeng Tu, and Xiaolong Li. RLVMR: Reinforcement learning with verifiable meta-reasoning rewards for robust long-horizon agents. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 114676–114700, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/baa41b7368408670c6a14e06a04420d9-Paper-Conference.pdf.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Jpxfof0EaS.

Qiyong Zhong, Mao Zheng, Mingyang Song, Xin Lin, Jie Sun, Houcheng Jiang, Xiang Wang, and Junfeng Fang. SOD: Step-wise on-policy distillation for small language model agents. *arXiv preprint arXiv:2605.07725*, 2026. URL https://arxiv.org/abs/2605.07725.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.),

Proceedings of the 41st International Conference on Machine Learning, volume 235 of *Proceedings of Machine Learning Research*, pp. 62178–62209. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24t.html.

Yifei Zhou, Song Jiang, Yuandong Tian, Jason Weston, Sergey Levine, Sainbayar Sukhbaatar, and Xian Li. SWEET-RL: Training multi-turn LLM agents on collaborative reasoning tasks. *arXiv preprint arXiv:2503.15478*, 2025. URL https://arxiv.org/abs/2503.15478.

Siqi Zhu, Xuyan Ye, Hongyu Lu, Weiye Shi, and Ge Liu. The many faces of on-policy distillation: Pitfalls, mechanisms, and fixes. *arXiv preprint arXiv:2605.11182*, 2026. URL https://arxiv.org/abs/2605.11182.

A TEACHER FEEDBACK ANALYSIS

This section defines the three measurements in Figure 1. For each of the 128 tasks, we hold the student's failed trajectory fixed and evaluate both teachers on every prefix of this trajectory with the same interaction history h_t and the same set of admissible actions. Following Section 3, $a_t = (a_{t,1}, \ldots, a_{t,L_t})$ denotes the action generated by the student at turn t, and $c_{t,j} = (h_t, a_{t,<j})$ denotes the context of the j-th token of a_t.

The reverse KL estimate at turn t evaluates the objective in Eq. (3) on the student's action tokens,

$$\widehat{D}_t = \frac{1}{L_t} \sum_{j=1}^{L_t} \Big[\log \pi_\theta(a_{t,j} \mid c_{t,j}) - \log \pi_\phi(a_{t,j} \mid c_{t,j}) \Big],$$

which is a Monte Carlo estimate of the reverse KL divergence along the student's own samples. The student trajectories are sampled at the evaluation temperature of 0.4, and we compute both distributions at this temperature. The rejection margin at turn t averages the gap between the largest teacher logit and the teacher logit of the student's token over the same tokens,

$$m_t = \frac{1}{L_t} \sum_{j=1}^{L_t} \Big[\max_w \ell_\phi(w \mid c_{t,j}) - \ell_\phi(a_{t,j} \mid c_{t,j}) \Big],$$

where ℓ_ϕ denotes the logits of the teacher. The margin is zero when the teacher prefers the student's token and grows as the teacher more strongly prefers another token. Complete-action agreement compares the complete actions that the two models prefer. Each model scores every admissible action by the length-normalized log-likelihood of that action given h_t, and agreement is 1 when both models assign the highest score to the same action.

For each teacher, we average every measurement over the turns in each interaction range of Figure 1 and then over tasks, so that every task contributes equally. The shaded bands are 95% confidence intervals from 10,000 task-level bootstrap resamples. For both teachers and all three measurements, the 95% bootstrap interval of the paired difference between turns 1–10 and 41–50 excludes zero.

B GRAPH SCHEMA AND VERIFICATION

The symbolic progress graph uses the following serialized schema:

Symbolic progress graph schema

```
{
  "milestones": [
    {
      "id": "<node_id>",
      "predicate": "<predicate>",
      "arguments": {"<argument>": "<value>"}
    },
    ...
  ],
  "dependencies": [
    ["<prerequisite_id>", "<dependent_id>"],
    ...
  ]
}
```

Each entry in `milestones` defines one graph node, and each pair in `dependencies` connects a prerequisite milestone to a dependent milestone. A generated graph is structurally accepted when identifiers are valid and unique, predicates belong to the environment vocabulary, every dependency references a defined milestone, and all terminal milestones use valid completion predicates (`PLACE` or `EXAMINE_UNDER_LIGHT` in ALFWorld, `PURCHASE` in WebShop, and `GOAL` in TextCraft).

At reset and after every environment action, the verifier evaluates each milestone using the corresponding condition in Table 4. Once verified, a milestone remains recorded for the rest of the trajectory. For example, a PICK2 task contains two `LOCATE–ACQUIRE–PLACE` chains, and the second milestone

Table 4: Predicate vocabularies and verification conditions for ALFWorld, WebShop, and TextCraft.

Environment	Predicate	Arguments	Verification condition
ALFWorld	LOCATE	obj	An instance of obj is visible at the agent's current location.
	ACQUIRE	obj	The agent holds an instance of obj.
	TRANSFORM	obj, state	An acquired instance of obj has reached the required clean, hot, or cool state.
	PLACE	obj, recep	An acquired instance of obj is in or on recep and satisfies the required transformation.
	EXAMINE_UNDER_LIGHT	obj, lamp	The agent holds obj at the location of an activated lamp.
WebShop	FIND	type, price_max	The current search results contain a product of the requested type within the price limit, where a limit of 0 means no limit.
	INSPECT	attr	The opened product satisfies the requested attribute.
	CONFIGURE	dim, value	The requested option value has been selected.
	PURCHASE	–	The environment confirms a successful purchase of an item satisfying the task.
TextCraft	COLLECT	item, n	Successful get actions have obtained at least n units of the raw item.
	CRAFT	item, n	Successful crafts have produced at least n units of the intermediate item.
	OBTAIN_TAG	tag, n	One concrete member of the ingredient family tag has reached n units.
	GOAL	item	Crafting the target item triggers task success.

of each type is verified only when a different object instance satisfies it. The verifier state therefore includes the object instances bound to earlier milestones and, in TextCraft, the cumulative counts of successful get and craft actions. Dependency edges describe the intended progress order for action selection and belong to the structure that graph distillation transfers to the student. Milestone verification depends only on whether the corresponding condition is satisfied, so a trajectory that completes milestones in a different valid order still receives credit for them.

C Training and Evaluation Details

Training. We implement VTD with veRL (Sheng et al., 2025) and vLLM (Kwon et al., 2023). We use Qwen2.5-1.5B-Instruct and Qwen2.5-7B-Instruct as the student policies and GPT-5.5 with medium reasoning effort to construct the training graphs. ALFWorld, WebShop, and TextCraft contain 3,553, 5,400, and 318 unique training tasks, respectively. TextCraft follows the split of BiPACE (Wang et al., 2026a), which reserves 79 separate target items requiring 2 to 4 levels of crafting for evaluation. We use training batch sizes of 16 prompts for ALFWorld and TextCraft and 32 prompts for WebShop. Each instruction produces 8 trajectories that share one teacher graph, keeping the milestone definitions fixed within the group-relative comparison. Table 5 gives the optimization configuration shared across environments and model scales, except that TextCraft uses a learning rate of 2×10^{-6}.

The reference-policy KL term regularizes the interaction tokens toward the frozen initial policy, and dual clipping lower-bounds the surrogate at 3 times the advantage for tokens with negative advantages. Teacher-forced graph tokens are excluded from both terms. We mean-center trajectory scores without standard-deviation normalization to avoid amplifying low-variance rollout groups. We average token-level surrogate terms within each trajectory before averaging across trajectories, so every rollout receives the same weight regardless of response length. The generalization experiment

Table 5: Optimization hyperparameters shared across the main VTD experiments.

Hyperparameter	Value
Optimizer	AdamW
Learning rate	10^{-6} (2×10^{-6} for TextCraft)
Policy epochs per step	1
Clipping threshold	0.2
Dual-clip coefficient	3.0
Reference-policy KL coefficient	10^{-3}
Outcome coefficient	1
Verified-progress coefficient	1
Graph-distillation coefficient	1
Rollout temperature	1.0

Table 6: Wall-clock time of 200 VTD training steps on one node with 8 NVIDIA H100 80GB GPUs.

Model	ALFWorld	WebShop	TextCraft
Qwen2.5-1.5B	8 h 26 min	3 h 29 min	3 h 13 min
Qwen2.5-7B	6 h 47 min	4 h 56 min	5 h 7 min

trains both objectives on 2,444 ALFWorld tasks from the PICK, PICK2, LOOK, and COOL families for 150 training steps.

Compute. Table 6 reports the wall-clock time of VTD training. On ALFWorld, Qwen2.5-1.5B takes longer than Qwen2.5-7B because its early rollouts often reach the limit of 50 actions.

Evaluation. Following the evaluation protocol of GiGPO (Feng et al., 2025), we evaluate ALFWorld and WebShop on 128 tasks with a sampling temperature of 0.4. For each benchmark, we sample three 128-task subsets with different random seeds, from the 140-task seen validation split of ALFWorld and from the WebShop test split. For TextCraft, we evaluate all 79 held-out target items with three sampling seeds at the same temperature. We allow at most 50 environment actions in ALFWorld, 15 in WebShop, and 40 in TextCraft. Before its first action, the VTD policy generates a symbolic progress graph and then completes the task with this graph in context, without access to the teacher. When the generated graph is not structurally accepted, no milestone is verified, and the policy still completes the episode. We report task success rate from the environment's binary completion signal. For the main and controlled evaluations, we report the mean and standard deviation over the 3 random seeds. The task-family generalization results use the complete 274-task ALFWorld evaluation set, comprising 177 tasks from the 4 training families and 97 held-out CLEAN and HEAT tasks.

D GENERALIZATION TO DEEPER TASK STRUCTURES

To test whether the learned task structure extends beyond the dependency depth seen during training, we group the 79 held-out TextCraft target items by the number of crafting levels they require. We train a separate VTD policy only on the 219 training items that require 2 levels and keep all other settings of Section 5.1. Following the depth groups of BiPACE (Wang et al., 2026a), Table 7 leaves out the 2 items that require 4 levels. BiPACE reports the peak validation success rate of GRPO and GiGPO separately for each depth, so these per-depth results need not average to the overall results in Table 1. VTD trained on 2-level items outperforms every baseline on 3-level items at both scales, even though every baseline trains on 3-level items. Training on all levels further raises VTD to 65.2% and 97.0% on 3-level items with Qwen2.5-1.5B and Qwen2.5-7B, respectively.

E GRAPH CONSTRUCTION PROMPTS

The following templates define the graph-construction interface for ALFWorld, WebShop, and TextCraft, with `{task_instruction}` replaced by the current task. The teacher uses this interface to construct training graphs, and the student uses the same graph format at evaluation.

Table 7: **VTD extends task structure to deeper recipe hierarchies.** Success rate (%) on TextCraft evaluation items grouped by crafting depth, with 55 items requiring 2 levels and 22 items requiring 3 levels. VTD (2-level training) trains only on items with 2 levels, whereas all other methods train on items of all levels. GRPO and GiGPO results are reported by BiPACE (Wang et al., 2026a), and † marks results obtained with public implementations. The † entries and the VTD entries show the mean and standard deviation over 3 random seeds. Best and second-best results at each scale are bolded and underlined.

Method	Qwen2.5-1.5B-Instruct		Qwen2.5-7B-Instruct	
	2 levels	3 levels	2 levels	3 levels
GRPO	59.5 ± 1.9	20.6 ± 4.2	90.3 ± 0.8	55.5 ± 10.0
GiGPO	76.7 ± 5.4	21.8 ± 9.6	94.0 ± 1.6	75.0 ± 3.5
BEACON†	21.2 ± 6.9	1.5 ± 2.6	53.9 ± 2.1	13.6 ± 7.9
HiPER†	0.0 ± 0.0	0.0 ± 0.0	97.6 ± 1.0	75.8 ± 6.9
SDAR†	71.5 ± 2.1	16.7 ± 2.6	89.1 ± 1.8	27.3 ± 4.5
VTD (Ours)	**93.9** ± 1.0	**65.2** ± 5.2	**98.8** ± 1.0	**97.0** ± 2.6
VTD (2-level training)	<u>90.3</u> ± 2.8	<u>37.9</u> ± 6.9	<u>98.2</u> ± 1.8	<u>78.8</u> ± 5.2

ALFWorld graph-construction instruction

```
## Before interaction: construct the symbolic progress graph
Decompose the task into a symbolic progress graph containing the minimal set of verifiable
milestones required for task completion and the dependencies among them. The graph will
guide the agent during interaction, and its milestone IDs will be used in the agent's
progress reports.

## Milestone predicates (use only these)
Each milestone specifies a condition that the environment can verify; name objects,
receptacles and lamps by their exact lowercase class as written in the task.

- **LOCATE(obj)**: you have reached the target `obj` and can see it (you are at the open
receptacle or the surface it sits on). The "found and can see the object" state.
- **ACQUIRE(obj)**: a target `obj` is in your hand. The "picked it up" state.
- **TRANSFORM(obj, state)**: the held `obj` has reached `state`, one of `clean` | `hot` |
`cool`. The object's required end-condition when the task involves cleaning, heating or
cooling it.
- **PLACE(obj, recep)**: a target `obj` is in or on receptacle `recep`. The finishing
state of a put task. If the task needs several copies of the object, include one full
LOCATE + ACQUIRE + PLACE milestone chain per copy, each with its own IDs; the copies
complete cumulatively (the first copy's milestones finish first).
- **EXAMINE_UNDER_LIGHT(obj, lamp)**: you are holding `obj` at the location of `lamp`
after that lamp has been used. The finishing state of a look/examine task.

## Output format
Output one JSON object and nothing else (no prose, no code fence): a `milestones` list
(each item has `id`, `predicate`, `arguments`) and a `dependencies` list of
`[prerequisite_id, dependent_id]` pairs.
- Give each milestone a short, unique `id` matching `[a-z][a-z0-9_]*`; the agent will
reuse these exact IDs in the cumulative `<state> done=[...] </state>` report during
interaction.
- Use `dependencies` to encode prerequisite relationships (e.g. the agent must acquire an
object before placing it).

## Example: a different task (cool a mug and put it on the diningtable)
{"milestones": [{"id": "locate", "predicate": "LOCATE", "arguments": {"obj": "mug"}},
                {"id": "acquire", "predicate": "ACQUIRE", "arguments": {"obj": "mug"}},
                {"id": "transform", "predicate": "TRANSFORM", "arguments": {"obj": "mug",
"state": "cool"}},
                {"id": "place", "predicate": "PLACE", "arguments": {"obj": "mug", "recep":
"diningtable"}}],
 "dependencies": [["locate", "acquire"], ["acquire", "transform"], ["transform", "place"]]}

## Task
{task_instruction}
Now output the symbolic progress graph as JSON.
```

WebShop graph-construction instruction

```
## Before interaction: construct the symbolic progress graph
Decompose the task into a symbolic progress graph containing the verifiable milestones
required to purchase an item that satisfies the instruction and the dependencies among
them. The graph will guide the agent during interaction, and its milestone IDs will be
used in the agent's progress reports.

## Milestone predicates (use only these)
Each milestone specifies a condition that the environment can verify from the store state.

- **FIND(type, price_max)**: the results list a candidate whose kind is `type` and whose
price is at most `price_max`. The "the search turned up the right kind of item, in budget"
state.
- **INSPECT(attr)**: available evidence shows that the viewed item has property `attr`.
One INSPECT per requested property; it records evidence when available rather than
requiring every text tab to be opened.
- **CONFIGURE(dim, value)**: you have selected the option `value` for choice `dim` (e.g.
color, size). One CONFIGURE per option the task asks for.
- **PURCHASE**: you buy an item that is the right type, has every required property, has
every required option selected, and is within budget.

### How to fill the arguments
The task describes ONE product: its product kind, requested properties, menu options, and
possibly a price cap. Separate them before writing the milestones:
- `type` = the shortest noun phrase that still names the product category as written (e.g.
"men's henley", "tv stand", "mattress"). Keep audience/category words such as men's /
women's / kids, and keep compound or coordinated category names together. If removing a
descriptive modifier still leaves the product category, that modifier is an attribute; do
not replace the category with a nearby property or accessory just because it sounds more
specific.
- `attr` = every distinct requested property outside the option and price clauses, one
INSPECT each. Copy its wording without paraphrasing or dropping it because it overlaps the
type. A comma that separates listed properties creates separate INSPECT nodes. A connector
such as `with`, `for`, or `to` may introduce a new property but is not part of its value:
"recycled fill for outdoor use" gives `recycled fill` and `outdoor use`, not one merged
attribute.
- `dim` / `value` = every menu option written as `<dim>: <value>` with a colon (e.g.
`color: navy`, `size: large`, `item shape: runner`), one CONFIGURE each for ANY dimension.
Copy the complete value verbatim, including punctuation; options are never INSPECT
attributes.
- `price_max` = the numeric cap in "price lower than X". Use `0` when the task has no
price clause.

## Output format
Output one JSON object and nothing else (no prose, no code fence): a `milestones` list
(each item has `id`, `predicate`, `arguments`) and a `dependencies` list of
`[prerequisite_id, dependent_id]` pairs. Give each milestone a short unique `id` matching
`[a-z][a-z0-9_]*`; the agent will reuse these exact IDs in `<state>` during interaction.
Use `dependencies` to encode the intended progress order: find an item before inspecting
or configuring it, then purchase after completing those milestones.

## Example: a different task ("stain resistant, machine washable patio chair cushions with
recycled fill for outdoor use with pattern: botanical, and size: 18 x 18, and price lower
than 45.00 dollars")
{"milestones": [{"id": "find", "predicate": "FIND", "arguments": {"type": "patio chair
cushions", "price_max": 45.0}},
                {"id": "attr_stain", "predicate": "INSPECT", "arguments": {"attr": "stain
resistant"}},
                {"id": "attr_washable", "predicate": "INSPECT", "arguments": {"attr":
"machine washable"}},
                {"id": "attr_fill", "predicate": "INSPECT", "arguments": {"attr":
"recycled fill"}},
                {"id": "attr_use", "predicate": "INSPECT", "arguments": {"attr": "outdoor
use"}},
                {"id": "pattern", "predicate": "CONFIGURE", "arguments": {"dim":
"pattern", "value": "botanical"}},
                {"id": "size", "predicate": "CONFIGURE", "arguments": {"dim": "size",
"value": "18 x 18"}},
                {"id": "purchase", "predicate": "PURCHASE", "arguments": {}}],
 "dependencies": [["find", "attr_stain"], ["find", "attr_washable"], ["find",
"attr_fill"], ["find", "attr_use"], ["find", "pattern"], ["find", "size"],
                  ["attr_stain", "purchase"], ["attr_washable", "purchase"], ["attr_fill",
"purchase"], ["attr_use", "purchase"], ["pattern", "purchase"], ["size", "purchase"]]}
(The example only shows the format and how to split type / attributes / options / price.)

## Task
{task_instruction}
```

```
Now output the symbolic progress graph as JSON.
```

TextCraft graph-construction instruction

```
## Before interaction: construct the symbolic progress graph
Construct a symbolic progress graph for the task below. The graph will guide the agent
during interaction, and its milestone IDs will be used in the agent's progress reports.

## Milestone predicates (use only these)
Use item and tag names exactly as displayed in the task, in lowercase without the
`minecraft:` prefix.

- **COLLECT(item, n)**: successful `get` actions cumulatively obtain at least `n` of a
concrete raw item that never appears as the output of a displayed recipe.
- **CRAFT(item, n)**: successful displayed recipes cumulatively produce at least `n` of a
concrete non-goal output; `n` counts produced items, not actions.
- **OBTAIN_TAG(tag, n)**: one listed concrete member of a family reaches `n`; use this
predicate only when that exact family is a heading under `Generic ingredient members`.
Every other ingredient name is a concrete item, even if it is plural or sounds generic.
Different members cannot be added together.
- **GOAL(item)**: a legal craft of the exact target triggers task success.

## Graph rules
- Starting at the goal, recursively select one displayed recipe for each crafted item.
Ignore distractors; include one milestone for every collected raw item and crafted
non-goal intermediate on this route, connect it only to its selected recipe's direct
inputs, and represent the target only with GOAL.
- Represent each resource once and aggregate demand shared by downstream recipes. For
demand `d` and recipe yield `o`, plan `ceil(d/o)` batches and set `n` to the total amount
those batches obtain.
- For an input listed under `Generic ingredient members`, choose one concrete member route
and represent its resulting quantity with OBTAIN_TAG. Connect the consumer to that tag
milestone and the tag milestone to the selected route's direct inputs.

## Output format
Output one JSON object and nothing else: a `milestones` list whose items contain exactly
`id`, `predicate`, and `arguments`, and a `dependencies` list of `[prerequisite_id,
dependent_id]` pairs forming a DAG with one GOAL sink. Give every milestone a short unique
`id` matching `[a-z][a-z0-9_]*`; the agent will reuse these exact IDs in `<state>
done=[...] </state>` during interaction. Use positive JSON integers for every `n`.

## Example: a different, fictional task
Suppose the task provides recipes that make four cedar planks from one cedar log, two iron
brackets from four iron ingots, and one supply cart from six generic planks, one bracket,
and three more iron ingots. Its symbolic progress graph is:
{"milestones": [{"id": "logs", "predicate": "COLLECT", "arguments": {"item": "cedar logs",
"n": 2}},
                {"id": "planks", "predicate": "OBTAIN_TAG", "arguments": {"tag": "planks",
"n": 8}},
                {"id": "iron", "predicate": "COLLECT", "arguments": {"item": "iron ingot",
"n": 7}},
                {"id": "brackets", "predicate": "CRAFT", "arguments": {"item": "iron
bracket", "n": 2}},
                {"id": "finish", "predicate": "GOAL", "arguments": {"item": "supply
cart"}}],
 "dependencies": [["logs", "planks"], ["iron", "brackets"], ["planks", "finish"], ["iron",
"finish"], ["brackets", "finish"]]}
Its items and recipes are not part of the task below.

## Task
{task_instruction}

Now output the symbolic progress graph as JSON.
```