Multi-agent LLMs with Offline Reinforcement Learning for Hierarchical Multi-turn Decision-making

Sangeun Park[†] **and Minhae Kwon**[†*]
[†]Department of Intelligent Semiconductors
[*]School of Electronic Engineering
Soongsil University, Seoul, South Korea
sangeunpark@soongsil.ac.kr, minhae@ssu.ac.kr

Abstract

The goal of intelligent agents is to support complex, long-horizon decision-making, yet current large language models (LLMs) struggle with multi-turn interactions. We introduce Multi^2, a hierarchical decision-making framework that operationalizes dual-process theory by separating System 1, a planner that generates long-term goals using supervised fine-tuning (SFT), from System 2, an executor that learns sequential decision-making via offline reinforcement learning (RL). Both systems share the same LLM backbone but specialize through distinct low-rank adaptation (LoRA) modules. This design enables sample-efficient training without online interaction and robust multi-turn reasoning through hierarchical decomposition. Experiments show that Multi^2 achieves 17.5% higher performance and 14.2% higher success rate than the strongest baseline. These results highlight the novelty of Multi^2 as the first framework to combine multi-agent LLMs with offline RL, providing a principled path toward scalable, multi-turn intelligent agents.

1 Introduction

What we ultimately seek in artificial intelligence is not short-term problem solving, but the multi-turn capacity to set goals, plan, and act [11]. According to dual-process theory in cognitive science, human reasoning involves a deliberative process (System 2) for structured, long-term reasoning and a reactive process (System 1) that produces context responses [6, 18, 37, 40, 49]. This interaction offers a useful analogy for designing intelligent agents, where distinct roles are divided and coordinated within an internally multi-agent-like architecture. Recent studies show that incorporating hierarchical decision-making significantly improves both efficiency and performance in long-horizon tasks [12, 19, 23]. Nevertheless, current LLMs remain limited in their ability to exhibit the sophisticated reasoning required for complex, multi-turn interactions [51, 5, 10, 45].

To develop effective decision-making strategies, recent studies often employ SFT to train agents to imitate expert demonstrations [22]. However, SFT alone cannot optimize for long-term outcomes. RL has emerged as a complementary approach, enabling LLM-based agents to leverage their broad knowledge for sequential, complex decision-making [16, 34, 35, 41]. Offline RL, in particular, learns policies from fixed datasets without online interaction, improving sample efficiency and generalization [8, 20]. By combining RL with LLMs, agents can exploit their broad knowledge while being trained for multi-turn interaction, making this approach well-suited for complex tasks [28, 33, 36, 39].

Building on these insights, we develop Multi^2, an LLM-based agent framework designed to handle multi-turn interactions. Our approach employs offline RL to enable precise decision-making and

[*]Corresponding author: M. Kwon

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Efficient Reasoning.

adopt a hierarchical structure that allows each system to specialize. In our multi-agent design, `System 1` receives the task and plans high-level goals, while `System 2` generates detailed actions to accomplish them. To facilitate efficient model updates, both agents share a common base model and fine-tune the LoRA [13] architectures. Our contributions are summarized as follows.

- We propose `Multi`^{2}, a hierarchical LLM agent that separates planning and execution into two specialized roles—System 1 for high-level goal planning and System 2 for low-level action reasoning—enabling structured multi-turn decision-making.
- System 2 is fine-tuned via offline RL, enabling stable, sample-efficient policy learning without online exploration.
- Each system is equipped with distinct LoRA modules, allowing efficient role-specific adaptation within a shared backbone while minimizing computational overhead.
- Empirically, `Multi`^{2} achieves up to 17.5% higher performance and 14.2% higher success rate than the strongest baselines on ScienceWorld, validating the effectiveness of combining hierarchical planning with offline RL.

2 Problem Setups

2.1 Problem Setup for LLM Agents

Inspired by dual-system theory, we refer to the agents as `System 1` for heuristic-driven decisions and `System 2` for sophisticated decision-making. Each agent is modeled as a distinct POMDP tailored to its role. At the beginning of the task, both agents share the same state $\mathbf{s}_t$ and the current observation $\mathbf{o}_t$. `System 1` generates step-by-step actions $\mathbf{a}_{1,t}$ corresponding to high-level goals $g_{t:H}$ of each overall task $K_{n\in N}$, thereby establishing the high-level decision for that task. Herein, the subscript $t:H$ indicates the index range from t to H. Each action from `System 1` is interpreted as a high-level goal $g_{h\in H}$ for `System 2`. In turn, `System 2` generates a single-step action $\mathbf{a}_{2,t}$ to accomplish these high-level goals, integrating the high-level goal g_h into its observations. The reward function $\mathcal{R}_t$ is defined only for `System 2`, which receives dense rewards for accomplishing the high-level goal g_h assigned by `System 1`. Notations are summarized in Appendix B.

2.2 Background: Reinforcement Learning

We formulate the considered system as a partially observable Markov decision process (POMDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \Omega, \mathcal{R}, \gamma \rangle$, where $\mathbf{s}_t \in \mathcal{S}$ is a state, $\mathbf{a}_t \in \mathcal{A}$ is an action, $\mathbf{o}_t \in \mathcal{O}$ is an observation. The state transition probability is given by $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the observation probability by $\Omega : \mathcal{S} \to \mathcal{O}$, the reward function by $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and $\gamma \in [0, 1)$ denotes a temporal discounted factor.

The objective of RL is to learn a policy that maximizes the expected cumulative return $\mathcal{J}_\pi = \mathbb{E}_\pi \left[\sum_t \gamma^t r_t\right]$, where $r_t = \mathcal{R}(\mathbf{o}_t, \mathbf{a}_{i,t}, \mathbf{o}_{t+1})$. In this context, the actor–critic structure provides a general RL framework. The actor selects an action $\mathbf{a}_{i,t}$ based on the concatenated observation $\mathbf{o}_t$, according to the policy $\pi_\theta\left(\mathbf{a}_{i,t}|\mathbf{o}_t\right)$. The critic evaluates these action by estimating the value function $Q_\omega\left(\mathbf{o}_t, \mathbf{a}_{i,t}\right) = \mathbb{E}_\pi\Big[\sum_t \gamma^t r_t|\mathbf{o}_t, \mathbf{a}_{i,t}\Big]$ and the value function $V_\psi\left(\mathbf{o}_t\right) = \mathbb{E}_\pi\Big[Q_\omega\left(\mathbf{o}_t, \mathbf{a}_{i,t}\right)\Big]$ under the policy π_θ. The advantage function $A_\theta\left(\mathbf{o}_t, \mathbf{a}_{i,t}\right) = Q_\omega\left(\mathbf{o}_t, \mathbf{a}_{i,t}\right) - V_\psi\left(\mathbf{o}_t\right)$ measures the relative benefit of an action.

3 The `Multi`^{2} Framework for Hierarchical Multi-turn Decision-making

This study introduces an LLM-based agent designed to incorporate hierarchical decision-making for multi-turn tasks. We propose the `Multi`^{2} framework, illustrated in Figure 1. The framework comprises an offline RL method, which develops an LLM-based agent to facilitate hierarchical decision-making in multi-turn environments. Specifically, `System 1` receives the overall task K_n as input and generates a sequence of high-level goals $g_{t:H} = \pi_\phi\left(\cdot|\mathbf{o}_t; K_n\right)$. Each high-level goal g_h is treated as a sub-task for `System 2`, which executes actions to accomplish each goal g_h. At each timestep, the high-level goal is concatenated with current observation $\mathbf{o}_t$ and provided as

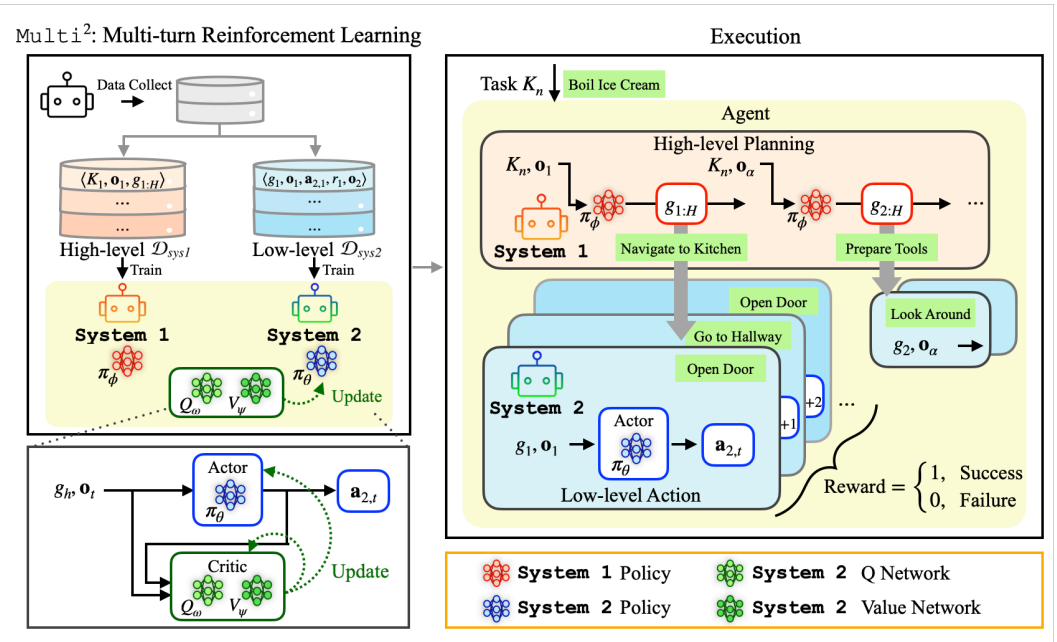

Figure 1: An illustrative diagram of the proposed framework.

input to System 2, which selects the corresponding action $\mathbf{a}_{2,t} = \pi_\theta(\cdot|\mathbf{o}_t; g_h)$.[2] Both agents are equipped with LoRA adapters, and System 2 leverages an actor–critic model to facilitate sophisticated decision-making.

3.1 Hierarchical Dataset Construction

For training the LLM-based agents, we prepare role-specific datasets. Based on the input–output requirements of each agent, the overall dataset is partitioned into $\mathcal{D}_{sys1}$ and $\mathcal{D}_{sys2}$. The dataset for System 1 is defined as follows.

$$\mathcal{D}_{sys1} = \{(K_n, \mathbf{o}_t, g_{t:H})\}_{n=1}^{|\mathcal{D}_{sys1}|} \tag{1}$$

In (1), K_n indicates the task description provided in natural language, $\mathbf{o}_t$ denotes the partial observation of the environment at the initial timestep, and $g_{t:H}$ represents the sequence of high-level goals.

The dataset for System 2 is defined as follows.

$$\mathcal{D}_{sys2} = \left\{ \left(g_h, \left\{ \xi_h^{(i)} \right\}_{i=1}^{M} \right) \right\}_{h=1}^{|\mathcal{D}_{sys2}|}, \quad \xi_h^{(i)} = \left(\mathbf{o}_t^{(h,i)}, \mathbf{a}_{2,t}^{(h,i)}, r_t^{(h,i)}, \mathbf{o}_{t+1}^{(h,i)} \right) \tag{2}$$

In (2), g_h represents the high-level goal provided by System 1, $\mathbf{a}_{2,t}$ denotes the action taken by the agent at each timestep, r_t is the reward received, and $\mathbf{o}_{t+1}$ is the subsequent observation. Each goal g_h is associated with M trajectories, and there are $|\mathcal{D}_{sys2}|$ goals in total. Since each dataset corresponds to distinct levels of action, this formulation enables the agent to perform hierarchical decision-making.

3.2 Model Architecture with with Low-Rank Adaptation

We adopt the LoRA architecture for parameter-efficient fine-tuning, which is particularly useful when full fine-tuning is computationally expensive or impractical. In our framework, System 1 and System 2 are assigned distinct LoRA matrices, as detailed in Appendix C.

[2] In the policies, ϕ and θ denote the model parameters of System 1 and System 2, respectively.

3.3 Multi^2: System 1 and System 2 Training

System 1 Training with SFT System 1 handles high-level planning and receives only sparse feedback indicating the success or failure of each task K_n. We train System 1 using SFT to preserve task-level consistency and stable goal planning. The policy π_ϕ is fine-tuned with a behavior cloning objective on the high-level dataset $\mathcal{D}_{sys1}$.

$$\mathcal{L}_\phi = -\mathbb{E}_{(K_n, \mathbf{o}_t, g_{t:H}) \sim \mathcal{D}_{sys1}} \Big(\log \pi_\phi(g_{t:H} | \mathbf{o}_t; K_n) \Big) \tag{3}$$

In (3), the loss $\mathcal{L}_\phi$ represents a standard cross-entropy loss that encourages the model to imitate expert demonstrations contained in $\mathcal{D}_{sys1}$.

System 2 Training with Offline RL System 2 executes detailed actions $\mathbf{a}_{2,t}$ based on interaction trajectories to accomplish the high-level goals g_h. To this end, we employ IQL-based offline RL [20], to train the policy π_θ using the Q-function Q_ω and the value function V_ψ. First, the Q-function Q_ω is trained by minimizing the TD error, as follows.

$$\mathcal{L}_\omega = \mathbb{E}_{(g_h, \mathbf{o}_t, \mathbf{a}_{2,t}, r_t, \mathbf{o}_{t+1}) \sim \mathcal{D}_{sys2}} \Big(r_t + \gamma V_\psi \left(\mathbf{o}_{t+1}; g_h \right) - Q_\omega \left(\mathbf{o}_t, \mathbf{a}_{2,t}; g_h \right) \Big)^2 \tag{4}$$

In (4), the term $r_t + \gamma V_\psi \left(\mathbf{o}_{t+1}; g_h \right)$ serves as the TD target, and $Q_\omega \left(\mathbf{o}_t, \mathbf{a}_{2,t}; g_h \right)$ is updated to minimize the difference from this target. Both the Q-function Q_ω and the value function V_ψ are trained, where the Q-function predicts the expected return given an action $\mathbf{a}_{2,t}$, whereas the value function estimates the expected return from an observation $\mathbf{o}_t$. It helps stabilize training and mitigate out-of-distribution issues by preventing the evaluation of actions absent from the dataset. This value function V_ψ is updated using the following loss function.

$$\mathcal{L}_\psi = \mathbb{E}_{(g_h, \mathbf{o}_t, \mathbf{a}_{2,t}) \sim \mathcal{D}_{sys2}} \Big(L_\tau^2 \big(A_\theta \left(\mathbf{o}_t, \mathbf{a}_{2,t}; g_h \right) \big) \Big), \text{where } L_\tau^2(u) = |\tau - \mathbb{1}\{u < 0\}| \tag{5}$$

In (5), $L_\tau^2(u)$ denotes the expectile regression loss, and $\mathbb{1}(\cdot)$ represents the indicator function that returns 1 if $Q_\omega \left(\mathbf{o}_t, \mathbf{a}_{2,t}; g_h \right) < V_\psi \left(\mathbf{o}_t; g_h \right)$ and 0 otherwise.

The actor is updated using both the Q-function Q_ω and the value function V_ψ, which is as follows.

$$\mathcal{L}_\theta = -\mathbb{E}_{(g_h, \mathbf{o}_t, \mathbf{a}_{2,t}) \sim \mathcal{D}_{sys2}} \Big[\exp \big(\beta A_\theta(\mathbf{o}_t, \mathbf{a}_{2,t}; g_h) \big) \times \log \pi_\theta(\mathbf{a}_{2,t} | \mathbf{o}_t; g_h) \Big] \tag{6}$$

In (6), $\log \pi_\theta(\mathbf{a}_{2,t} | \mathbf{o}_t; g_h)$ encourages the policy to reproduce the action distribution observed in $\mathbf{o}_t$. The scaling factor $\beta A_\theta(\mathbf{o}_t, \mathbf{a}_{2,t}; g_h)$ modulates the strength of imitation, weighting actions according to their estimated advantage. Based on these loss functions, both the actor and critic are updated within an offline RL framework.

The overall procedure of the offline RL-based fine-tuning phase is summarized in Appendix D.

4 Experiments

This section presents the performance analysis of the proposed Multi^2 framework. We compare it with baselines and provide ablation studies in ScienceWorld benchmark [46]. Experimental settings are described in Appendix E, Appendix F, and Appendix G.

4.1 Performance Analysis

In ScienceWorld benchmark, we evaluate performance across four representative topics, (1) Electricity, (2) Measurement, (3) Identification in Biology, and (4) Classification. Within each topic, we further assess effectiveness on several tasks. Table 1 presents model performance (Per) and success rate (SR), evaluated over 10 random seeds. Model performance is computed as the reward associated with achieving high-level goals, while SR indicates whether the model achieves complete task success. Additional experimental results are described in Appendix H and Appendix I.

The proposed framework consistently outperforms all strong baselines across every base model. Prompt-based methods such as ReAct and Reflexion exhibit limited performance, indicating that prompting alone is insufficient for complex decision-making. In addition, SwiftSage performs

Table 1: Performance comparison with baselines. The best score is highlighted in cyan.

Base Model	Methods	(1)		(2)		(3)		(4)		Average	
		Per [%]	SR [%]	Per [%]	SR [%]	Per [%]	SR [%]	Per [%]	SR [%]	Per [%]	SR [%]
Qwen-2.5-3B	`ReAct` [50]	21.70	0.00	8.00	0.00	21.70	10.00	15.90	10.00	16.83	5.00
	`Reflexion` [38]	20.70	0.00	4.60	0.00	52.50	20.00	18.40	0.00	24.05	5.00
	`SwiftSage` [26]	3.20	0.00	2.90	0.00	25.00	0.00	16.60	0.00	11.93	0.00
	`Glider` [14]	36.90	20.00	39.60	0.00	66.50	30.00	38.50	0.00	45.38	12.50
	`Multi`^{2}	57.00	40.00	57.40	20.00	86.60	70.00	47.50	10.00	62.13	35.00

weakest on certain tasks, as it only updates the Swift module and cannot effectively leverage larger models. Its performance drops sharply in (1) Electricity and (2) Measurement. `Glider` shows relatively higher performance, but its reliance on prompt-level role separation limits overall performance. These results confirm that `Multi`^{2} is well-structured and superior for multi-turn tasks.

4.2 Ablation Studies

To assess the contribution of the core components of the proposed framework, we compare the framework's performance. In Figure 2, the proposed model consistently outperforms both variants (w/o RL and Single) across all base models. In particular, Single in Gemma and w/o RL in Qwen exhibit a success rate of 0%. Interestingly, the contribution of each component varies across base models. With the Gemma base model, hierarchical decision-making plays a more critical role, whereas for the Llama and Qwen base models, offline RL fine-tuning yields greater performance improvements. Nevertheless, these results demonstrate that the combination of offline RL fine-tuning and hierarchical decision-making substantially enhances decision precision and robustness in multi-turn tasks.

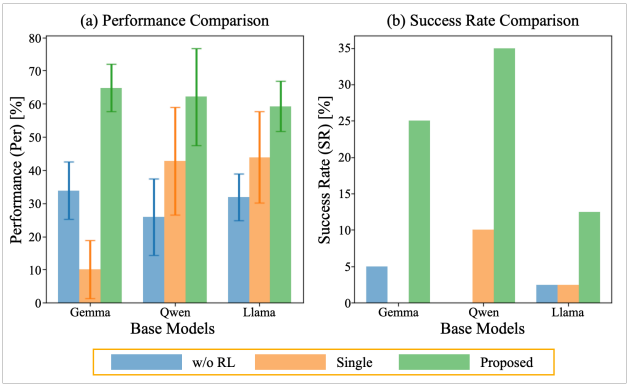

Figure 2: Ablation studies. (a) Performance comparison and (b) Success rate.

5 Discussion

Conclusion In this paper, we propose the `Multi`^{2} framework, which employs hierarchical decision-making to tackle multi-turn tasks. `Multi`^{2} allows an LLM-based agent to separate high-level long-term planning from low-level actions. Specifically, we design `System 1` and `System 2` mechanisms, where `System 1` generates high-level goals for task completion using SFT, while `System 2` executes detailed actions to achieve these high-level goals. We fine-tune the model using offline RL, enabling it to make decisions optimized for sequential tasks. Experimental results show that `Multi`^{2} consistently outperforms strong baselines across diverse domains. Ablation studies confirm the contribution of each module, with consistent performance improvements.

Broader Impacts The proposed `Multi`^{2} framework provides a foundation for developing intelligent LLM agents capable of solving problems that involve complex, multi-turn interactions. By enabling agents to make hierarchical decisions, this approach has the potential to be applied to real-world domains such as autonomous driving, defense systems, and predictive modeling for future events.

Limitations While our work demonstrates promising results, it has several limitations. In the current framework, we adopt the same base model for both `System 1` and `System 2` due to resource constraints. Additionally, our framework relies on hierarchical datasets to train each agent. To construct these datasets, we use the GPT-4 model to separate and organize data into hierarchical structures. For broader usability, we plan to automate this dataset construction process in future work.

References

[1] H. Bai, Y. Zhou, M. Cemri, J. Pan, A. Suhr, S. Levine, and A. Kumar. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

[2] P. Ball, L. Smith, I. Kostrikov, and S. Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, 2023.

[3] Y. Cao, H. Zhao, Y. Cheng, T. Shu, Y. Chen, G. Liu, G. Liang, J. Zhao, J. Yan, and Y. Li. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):9737–9757, 2025.

[4] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P.-Y. Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. In *International Conference on Machine Learning*, 2023.

[5] H. Dong, K. Duan, and C. Zhang. Enhancing decision-making of large language models via actor-critic. In *International Conference on Machine Learning*, 2025.

[6] J. Evans. In two minds: Dual-process accounts of reasoning. *Trends in cognitive sciences*, 7(10):454–459, 2003.

[7] P. Feng, Y. He, G. Huang, Y. Lin, H. Zhang, Y. Zhang, and H. Li. AGILE: A novel reinforcement learning framework of LLM agents. In *Advances in Neural Information Processing Systems*, 2024.

[8] S. Fujimoto and S. Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

[9] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

[10] H. Furuta, K. Lee, O. Nachum, Y. Matsuo, A. Faust, S. Gu, and I. Gur. Multimodal web navigation with instruction-finetuned foundation models. In *International Conference on Learning Representations*, 2024.

[11] D. Glória-Silva, R. Ferreira, D. Tavares, D. Semedo, and J. Magalhaes. Plan-grounded large language models for dual goal conversational settings. In *European Chapter of the Association for Computational Linguistics*, pages 1271–1292, 2024.

[12] B. Han, J. Kim, and J. Jang. A dual process VLA: Efficient robotic manipulation leveraging VLM. *arXiv preprint arXiv:2410.15549*, 2024.

[13] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[14] Z. Hu, W. Liu, X. Qu, X. Yue, C. Chen, Z. Wang, and Y. Cheng. Divide and conquer: Grounding LLMs as efficient decision-making agents via offline hierarchical reinforcement learning. In *International Conference on Machine Learning*, 2025.

[15] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, D. Kalashnikov, S. Levine, Y. Lu, C. Parada, K. Rao, P. Sermanet, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, M. Yan, N. Brown, M. Ahn, O. Cortes, N. Sievers, C. Tan, S. Xu, D. Reyes, J. Rettinghouse, J. Quiambao, P. Pastor, L. Luu, K. Lee, Y. Kuang, S. Jesmonth, N. Joshi, K. Jeffrey, J. Ruano, J. Hsu, K. Gopalakrishnan, B. David, A. Zeng, and K. Fu. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2023.

[16] K. Ji, J. Chen, A. Gao, W. Xie, X. Wan, and B. Wang. Unlocking LLMs' self-improvement capacity with autonomous learning for domain adaptation. In *Findings of the Association for Computational Linguistics*, 2025.

[17] C. Jia, Z. Li, P. Wang, Y. Li, Z. Hou, Y. Dong, and Y. Yu. Controlling large language model with latent action. In *International Conference on Machine Learning*, 2025.

[18] D. Kahneman. *Thinking, fast and slow*. macmillan, 2011.

[19] S. Khorasani, S. Salehkaleybar, N. Kiyavash, and M. Grossglauser. Hierarchical reinforcement learning with targeted causal interventions. In *International Conference on Machine Learning*, 2025.

[20] I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit Q-learning. In *International Conference on Learning Representations*, 2022.

[21] D. Lee and M. Kwon. Episodic future thinking mechanism for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

[22] J. Li, S. Zeng, H. Wai, C. Li, A. Garcia, and M. Hong. Getting more juice out of the SFT data: Reward learning from human demonstration improves SFT for LLM alignment. In *Advances in Neural Information Processing Systems*, 2024.

[23] S. Li. Deep reinforcement learning with hierarchical structures. In *International Joint Conference on Artificial Intelligence*, 2021.

[24] Y. Li, Z. Liu, and E. Xing. Data mixing optimization for supervised fine-tuning of large language models. In *International Conference on Machine Learning*, 2025.

[25] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[26] B. Lin, Y. Fu, K. Yang, F. Brahman, S. Huang, C. Bhagavatula, P. Ammanabrolu, Y. Choi, and X. Ren. SwiftSage: A generative agent with fast and slow thinking for complex interactive tasks. In *Advances in Neural Information Processing Systems*, 2023.

[27] Z. Lin, Y. Tang, X. Yao, D. Yin, Z. Hu, Y. Sun, and K. Chang. QLASS: Boosting language agent inference via Q-guided stepwise search. In *International Conference on Machine Learning*, 2025.

[28] Z. Liu, H. Hu, S. Zhang, H. Guo, S. Ke, B. Liu, and Z. Wang. Reason for future, act for now: A principled framework for autonomous LLM agents with provable sample efficiency. In *International Conference on Machine Learning*, 2024.

[29] W. Lu, R. Luu, and M. Buehler. Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities. *npj Computational Materials*, 11(1):84, 2025.

[30] T. Luong, X. Zhang, Z. Jie, P. Sun, X. Jin, and H. Li. ReFT: Reasoning with reinforced fine-tuning. In *Meeting of the Association for Computational Linguistics*, 2024.

[31] H. Ma, T. Hu, Z. Pu, B. Liu, X. Ai, Y. Liang, and M. Chen. Coevolving with the other you: Fine-tuning LLM with sequential cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

[32] Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, 2024.

[33] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

[34] A. Pareja, N. Nayak, H. Wang, K. Killamsetty, S. Sudalairaj, W. Zhao, S. Han, A. Bhandwaldar, G. Xu, K. Xu, L. Han, L. Inglis, and A. Srivastava. Unveiling the secret recipe: A guide for supervised fine-tuning small LLMs. In *International Conference on Learning Representations*, 2025.

[35] S. Quan. Automatically generating numerous context-driven SFT data for LLMs across diverse granularity. In *AAAI Conference on Artificial Intelligence*, 2025.

[36] R. Rafailov, A. Sharma, E. Mitchell, C. Manning, S. Ermon, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

[37] C. Shek and P. Tokekar. Option discovery using LLM-guided semantic hierarchical reinforcement learning. *arXiv preprint arXiv:2503.19007*, 2025.

[38] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

[39] C. Snell, I. Kostrikov, Y. Su, S. Yang, and S. Levine. Offline RL for natural language generation with implicit language Q learning. In *International Conference on Learning Representations*, 2023.

[40] H. Song, D. Qu, Y. Yao, Q. Chen, Q. Lv, Y. Tang, M. Shi, G. Ren, M. Yao, B. Zhao, D. Wang, and X. Li. Hume: Introducing system-2 thinking in visual-language-action model. *arXiv preprint arXiv:2505.21432*, 2025.

[41] A. Szot, M. Schwarzer, H. Agrawal, B. Mazoure, R. Metcalf, W. Talbott, N. Mackraz, R. Hjelm, and A. Toshev. Large language models as generalizable policies for embodied tasks. In *International Conference on Learning Representations*, 2024.

[42] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone. Deep reinforcement learning for robotics: A survey of real-world successes. In *AAAI Conference on Artificial Intelligence*, 2025.

[43] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[44] A. Wagenmaker and A. Pacchiano. Leveraging offline data in online reinforcement learning. In *International Conference on Machine Learning*, 2023.

[45] H. Wang, T. Liu, R. Li, M. Cheng, T. Zhao, and J. Gao. RoseLoRA: Row and column-wise sparse low-rank adaptation of pre-trained language model for knowledge editing and fine-tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2024.

[46] R. Wang, P. Jansen, M.-A. Côté, and P. Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of Empirical Methods in Natural Language Processing*, 2022.

[47] C. Wei, Y. Hong, and C. Lu. Online reinforcement learning in stochastic games. In *Advances in Neural Information Processing Systems*, 2017.

[48] Q. . A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

[49] R. Yang, J. Shi, M. Su, and D. Zhou. Leveraging imitation learning and LLMs for efficient hierarchical reinforcement learning. *https://openreview.net/forum?id=6y00rooi7i*, 2025.

[50] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

[51] L. Zhang, Y. Cao, Y. Xie, F. Fang, and Y. Li. Dynamic evaluation with cognitive reasoning for multi-turn safety of large language models. In *Association for Computational Linguistics*, pages 19588–19608, 2025.

[52] Y. Zhou, A. Zanette, J. Pan, S. Levine, and A. Kumar. Archer: Training language model agents via hierarchical multi-turn RL. *arXiv preprint arXiv:2402.19446*, 2024.

Appendix: Multi-agent LLMs with Offline Reinforcement Learning for Hierarchical Multi-turn Decision-making

Contents

A Related Works

Large Language Model for Decision-making With the rapid advances of LLMs, many studies have applied them to agent-based applications. Early research has primarily explored prompt-based frameworks, such as reasoning and acting approaches [50], which mainly exploit general knowledge. However, these approaches often fail to enable agents to recognize and correct mistakes, motivating the development of LLMs with meta-cognitive capabilities for self-reflection [38]. Despite these advances, performance remains limited because such agents lack long-term planning for complex decision-making tasks, underscoring the need for fine-tuning the LLM's weights [5, 10, 45].

A common approach to fine-tune an LLM-based agent is SFT [22], but it often overfits to the specific domain due to heavy reliance on the training dataset [16, 24, 29, 41]. To overcome this issue, recent studies have explored RL-based fine-tuning, which enables long-term planning and optimizes actions through to reward signals [17, 27, 31]. Specifically, [31] proposes a multi-agent RL approach where a single LLM is duplicated and collaboratively fine-tuned, while [27] estimates stepwise Q-values via exploration trees to guide better decisions. In this study, we propose an RL-based framework for fine-tuning an LLM-based agent to improve decision-making. The framework adopts an actor-critic structure to achieve both stable learning and efficient policy improvement.

Offline Reinforcement Learning To develop decision-making strategies, RL is an effective approach for an agent [7, 3]. The objective of RL is to construct policies that guide an agent in selecting optimal actions through interactions with its environment. Early RL approaches primarily focused on online RL, which learns decision-making policies via direct interaction with the environment [25, 9]. While this approach enables agents to adapt to dynamic environments and gradually acquire optimal behaviors, it suffers from significant sample inefficiency and raises safety concerns due to real-time exploration [47, 2, 44].

To address these limitations, offline RL has emerged as a suitable alternative, as it learns policies from pre-collected datasets without requiring online interaction [8, 20]. Offline RL is particularly advantageous in risky or impractical environments, offering improved sample efficiency while mitigating safety concerns. This approach has been successfully applied across various domains, such as autonomous driving and robotics [21, 42]. More recently, RL has also been leveraged to fine-tune LLMs [28]. Building on this direction, we propose an LLM-based agent trained with an offline RL-based approach, enabling efficient fine-tuning using pre-collected datasets.

Hierarchical Reinforcement Learning Agents for Multi-turn Tasks Fine-tuning with RL enhances performance by enabling agents to select actions with future outcomes in mind [30, 4, 15, 36, 39, 33]. Nonetheless, this approach still faces the challenge of sparse rewards, as feedback is often only available upon successful task completion. To address this, hierarchical decision-making is often employed in complex or multi-turn environments [1, 52, 14].

In this framework, a high-level policy decomposes tasks into high-level goals, while a low-level policy executes detailed actions [40, 37, 49]. For instance, [52] proposes an actor-critic RL framework that trains the high-level and the low-level policies to optimize token-level behaviors. [14] introduces a hierarchical RL framework in which the high-level policy decomposes tasks into sequential subgoals and a low-level controller executes them via RL. This structure provides intermediate rewards, improving credit assignment in sparse-reward settings and enabling more effective management of long-term dependencies [19, 12, 23].

Building on these insights, we propose an LLM fine-tuning approach that integrates RL with hierarchical decision-making. In this framework, the high-level agent formulates long-term strategies to accomplish the task, whereas the low-level agent focuses on executing specific actions. Moreover, the LoRA framework is employed for each agent to enable parameter-efficient fine-tuning.

B Notation Table

B.1 Notation of Reinforcement Learning

Notation	Description	Notation	Description
$\mathcal{S}$	state space	$\mathbf{s}$	state
$\mathcal{A}$	action space	$\mathbf{a}$	action
$\mathcal{O}$	observation space	$\mathbf{o}$	observation
$\mathcal{T}$	state transition probability	Ω	observation probability
$\mathcal{R}$	reward function	R	reward
γ	temporal discounted factor	β	scaling factor
π_ϕ	`System 1` policy	π_θ	`System 2` policy
$\hat{\pi}_\phi$	`System 1` SFT-trained policy	$\hat{\pi}_\theta$	`System 2` SFT-trained policy
Q_ω	`System 2` Q network	V_ψ	`System 2` value network
$\mathcal{J}$	objective function	A_θ	advantage function

B.2 Notation of Problem Setups

Notation	Meaning	Notation	Meaning
i	agent index	t	timesteps
K	textual task	g	high-level goal
N	the number of tasks	H	the number of high-level goals
n	n-th task	h	h-th goal
$\mathcal{D}_{sys1}$	`System 1` SFT dataset	$\mathcal{D}_{sys2}$	`System 2` SFT dataset
M	the number of trajectories	–	–

C LoRA Architecture

We introduce the LoRA architecture for parameter-efficient fine-tuning. LoRA is particularly useful in environments where full fine-tuning is computationally expensive or impractical. `System 1` and `System 2` are assigned distinct LoRA matrices that are selectively activated, as illustrated in Figure 3. `System 2` adopts an actor-critic architecture, where the actor and critic are each equipped with their own LoRA matrices. Given an input, the framework determines whether `System 1` or `System 2` is activated. When the task description K_n is provided, `System 1` is triggered to generate sub-tasks, whereas an observation $\mathbf{o}_t$ containing a sub-task g_h activates `System 2`.

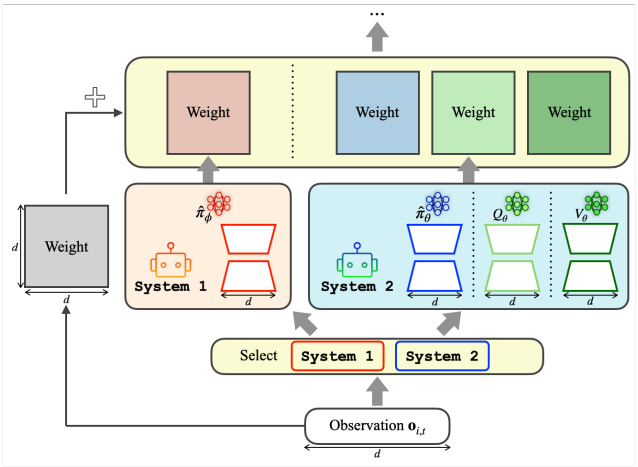

Figure 3: Illustration of the proposed LoRA-based framework.

The selected LoRA matrix processes the input in parallel with the frozen original weights, and their outputs are subsequently combined. In the final transformer block, the critic incorporates an additional multi-layer perceptron that produces a scalar value to estimate the value of the actor's action.

D Algorithm Pseudocodes

D.1 `System 1`: Supervised Fine-tuning

Algorithm 1 Supervised Fine-tuning for `System 1`

Require: High-level dataset $\mathcal{D}_{sys1}$, Training epochs EP_1
Initialize `System 1` SFT policy π_ϕ
for $i = 1, \text{EP}_1$ **do**
  Sample $\langle K_n, \mathbf{o}_t, g_{1:H} \rangle \sim \mathcal{D}_{sys1}$
  Calculate loss using (3)
  Back-propagate and update π_ϕ `//Update System 1`
end for

D.2 `System 2`: Offline Reinforcement Learning

Algorithm 2 Offline Reinforcement Learning for `System 2`

Require: SFT-trained policy $\hat{\pi}_\theta$, Low-level dataset $\mathcal{D}_{sys2}$, Training epochs EP_2
Initialize: Q-function Q_ω, Value-function V_ψ
Initialize policy network $\pi_\theta \leftarrow \hat{\pi}_\theta$
for $j = 1, \text{EP}_2$ **do**
  Sample $\langle g_h, \mathbf{o}_t, \mathbf{a}_{2,t}, r_t, \mathbf{o}_{t+1} \rangle \sim \mathcal{D}_{sys2}$
  Calculate TD loss using (4)
  Calculate Value loss using (5)
  Calculate Actor loss using (6)
  Back-propagate and update V_ψ
  Back-propagate and update Q_ω
  Back-propagate and update π_θ `//Update System 2`
end for

E Experiment setups

E.1 Experiment settings

Benchmark We evaluate our framework on the ScienceWorld [46] benchmark, which requires agents to accomplish objectives derived from the standardized elementary science curriculum. The benchmark comprises 30 tasks across 10 topics. ScienceWorld inherently demands long-term decision-making and thus requires multi-turn interactions. For example, to accomplish a task like "boil the water", an agent must sequentially plan and execute multiple sub-tasks.

Models Our framework is built on the open-source Gemma-1.1 2B [43], Llama-3.2 3B [32], and Qwen-2.5 3B [48] language models, which serve as the shared base model for both `System 1` and `System 2`. Each system applies LoRA to the value projection within the multi-query attention mechanism. We perform short SFT fine-tuning for both `System 1` and `System 2` to establish a well-structured output format. We train our model using an A100 GPU (24 CORE, 192GB RAM).

Dataset Construction Using the GPT-4 model, we construct hierarchical datasets by redefining the human-annotated trajectories provided in each benchmark. Specifically, the trajectories are decomposed into high-level goals with their corresponding reward assignments. The in-context learning prompt is as follows.

```
Based on the attached JSON files, create a dataset in the same
format as the ScienceWorld benchmark. \n (Data)
```

The model outputs as follows.

```
{
  "Task Description": "Your task is to boil ice cream. For compounds without a boiling point,
combusting the substance is also acceptable. First, focus on the substance. Then, take actions that
will cause it to change its state of matter.",
  "High-level Goal": [
    "Navigate to the kitchen",
    "Collect tools: thermometer and metal pot",
        …
  ],
  "Action": [
    "open door to kitchen",
    "go to kitchen",
    "look around",
    …
    "wait"
  ],
  "Obs": [
    "The door is now open.",
    "You move to the kitchen.",
    "This room is called the kitchen.",
    …
    "You decide to wait for 1 iterations."
  ],
  "Next Obs": [
    "The door is now open.",
    "You move to the kitchen.",
    "This room is called the kitchen. In it, you see:\n\tthe agent\n\ta substance called air\n\ta
cupboard (closed)\n\ta freezer (closed)\n\ta stove (off)\n\ta thermometer (10°C)\n\ta glass jar
(containing salt)",
    …
    "You decide to wait for 1 iterations."
  ],
  "reward": [
    0.0, 0.0, 0.0, …, 0.0
  ],
  "score": [
    0.0, 0.0, 0.0, …, 1.0
  ],
  "done": [
    false, false, false, …, true
  ]
}
```

In ScienceWorld [46], reward values are normalized to the range $[0, 1]$ and specified for each task.

For `System 1`, the input is combined with the following explanation prompt.

> You are a high-level planner. Based on the state (task description and current observation), please generate a clear and simple high-level goal. \n Task Description: \n Obs: \n

For `System 2`, we use the following explanation prompt.

> You are a low-level action executor. Based on the current high-level goal and observation, please generate an executable action and determine whether the high-level goal has been completed (true/false). \n High-level goal: \n Obs: \n

E.2 Baselines

We compare `Multi`^{2} with the following baselines, all of which adopt base models as their backbone.

- `ReAct` [50]: It enhances reasoning and action capabilities through chain-of-thought prompting. Unlike pure chain-of-thought methods, `ReAct` allows agents to interact dynamically with the environment.
- `Reflexion` [38]: It relies on self-reflection, where at the end of each episode, the agent receives feedback, generates a reflection, and stores it in long-term memory to guide future decisions.
- `SwiftSage` [26]: It employs dual process by combining the Swift module for fast, intuitive reasoning with the Sage module for deliberate, reflective reasoning. In contrast to our framework, the Swift module is trained solely via behavior cloning, while the Sage module remains unfine-tuned.
- `Glider` [14]: It adopts a hierarchical structure consisting of a high-level planner and a low-level controller. Fine-tuning is achieved through RL-based parameter updates. Unlike `Multi`^{2}, it employs a single LoRA adapter and relies on prompt-based role separation for specialization.
- `Multi`^{2} (Proposed): It integrates hierarchical decision-making, where `System 1` handles high-level planning and `System 2` executes detailed actions in multi-turn tasks. It further employs SFT to train `System 1` and the offline RL to fine-tune `System 2`, which distinguishes roles through separate LoRA adapters.

We compare the performance across four topics ((1) Electricity, (2) Measurement, (3) Identification in Biology, and (4) Classification), and further evaluate its effectiveness on individual tasks within each topic. All results are averaged over 10 random seeds.

E.3 Ablation Studies

- **w/o RL:** Both `System 1` and `System 2` are trained solely via SFT on the pre-collected dataset, without applying any offline RL. This variant evaluates the effect of hierarchical structure when RL is absent.
- **Single:** The hierarchical design is removed, and a single unified agent directly maps task descriptions to low-level actions. The model is first format-tuned via SFT and then fine-tuned with offline RL, allowing us to isolate the impact of hierarchical decomposition.
- **Proposed:** The complete `Multi`^{2} framework, where `System 1` performs high-level goal planning and `System 2` executes detailed actions. `System 1` is trained with SFT, and `System 2` is additionally optimized using offline RL.

F Hyperparameters

Hyperparameter	SFT	Offline RL
Discount factor γ	0.99	0.99
Polyak update ratio τ	–	0.2
Epoch EP (`System 1 / System 2`)	2 / 2	3 / 5
Maximum new token	32	32
Optimizer	AdamW	AdamW
Batch size	32	32
Micro batch size	–	4
`System 1` Actor learning rate	5×10^{-5}	1×10^{-5}
`System 2` Actor learning rate	1×10^{-4}	1×10^{-6}
Critic learning rate	–	1×10^{-6}
Target update frequency	–	2
Temperature	0.7	0.7
Q network hidden node	–	$[4096, 4096]$
Value network hidden node	–	$[4096, 4096]$
LoRA low rank	16	16
LoRA alpha	32	32
LoRA dropout	0.05	0.05

G System Specification

CPU	24core
GPU	A100
RAM	192GB
SSD	2TB

H Additional Experimental Results

H.1 Performance Comparison

Table 2: Performance comparison with baselines using Llama base model. The best score is highlighted in cyan.

Base models	Methods	(1)		(2)		(3)		(4)		Average	
		Per [%]	SR [%]	Per [%]	SR [%]	Per [%]	SR [%]	Per [%]	SR [%]	Per [%]	SR [%]
Gemma-1.1 2B	`ReAct` [50]	18.10	0.00	0.80	0.00	17.50	0.00	14.20	0.00	12.65	0.00
	`Reflexion` [38]	17.10	0.00	7.10	0.00	0.00	0.00	22.00	0.00	11.55	0.00
	`SwiftSage` [26]	5.30	0.00	2.50	0.00	12.50	0.00	15.00	0.00	8.83	0.00
	`Glider` [14]	49.70	20.00	43.70	0.00	48.20	10.00	54.20	30.00	48.95	15.00
	`Multi`^{2}	76.00	40.00	59.20	10.00	58.30	20.00	65.89	30.00	64.85	25.00
Llama-3.2 3B	`ReAct` [50]	15.60	0.00	1.30	0.00	0.00	0.00	26.11	0.00	10.75	0.00
	`Reflexion` [38]	17.80	0.00	9.20	0.00	25.60	0.00	22.70	0.00	18.83	0.00
	`SwiftSage` [26]	3.20	0.00	3.20	0.00	5.00	0.00	24.50	0.00	8.98	0.00
	`Glider` [14]	36.20	0.00	31.70	0.00	49.40	10.00	40.00	0.00	39.33	2.50
	`Multi`^{2}	52.40	0.00	52.00	10.00	70.00	40.00	62.77	0.00	59.29	12.50

H.2 Ablation Studies

Table 3: Ablation studies in Gemma base model. The best score is highlighted in cyan.

Method	RL	Dual	(1)		(2)		(3)		(4)		Average	
			Perform	SR	Perform	SR	Perform	SR	Perform	SR	Perform	SR
w/o RL	✗	✓	19.70%	0%	40.00%	0%	41.60%	10%	34.10%	10%	33.85%	5.00%
Single	✓	✗	2.70%	0%	1.70%	0%	23.30%	0%	12.40%	0%	10.03%	0%
Proposed	✓	✓	76.00%	40%	59.20%	10%	58.30%	20%	65.89%	30%	64.85%	25.00%

Table 4: Ablation studies in Qwen base model. The best score is highlighted in cyan.

Method	RL	Dual	(1)		(2)		(3)		(4)		Average	
			Perform	SR	Perform	SR	Perform	SR	Perform	SR	Perform	SR
w/o RL	✗	✓	39.60%	0%	34.36%	0%	19.10%	0%	10.70%	0%	25.94%	0.00%
Single	✓	✗	47.00%	0%	19.60%	0%	64.90%	40%	39.90%	0%	42.85%	10.00%
Proposed	✓	✓	57.00%	40%	57.40%	20%	86.60%	70%	47.50%	10%	62.13%	35.00%

Table 5: Ablation studies in Llama base model. The best score is highlighted in cyan.

Method	RL	Dual	(1)		(2)		(3)		(4)		Average	
			Perform	SR	Perform	SR	Perform	SR	Perform	SR	Perform	SR
w/o RL	✗	✓	29.30%	0%	30.90%	0%	43.30%	10%	24.20%	0%	31.92%	2.50%
Single	✓	✗	40.10%	0%	23.10%	0%	58.30%	10%	54.00%	0%	43.88%	2.50%
Proposed	✓	✓	52.40%	0%	52.00%	10%	70.00%	40%	62.77%	0%	59.29%	12.50%

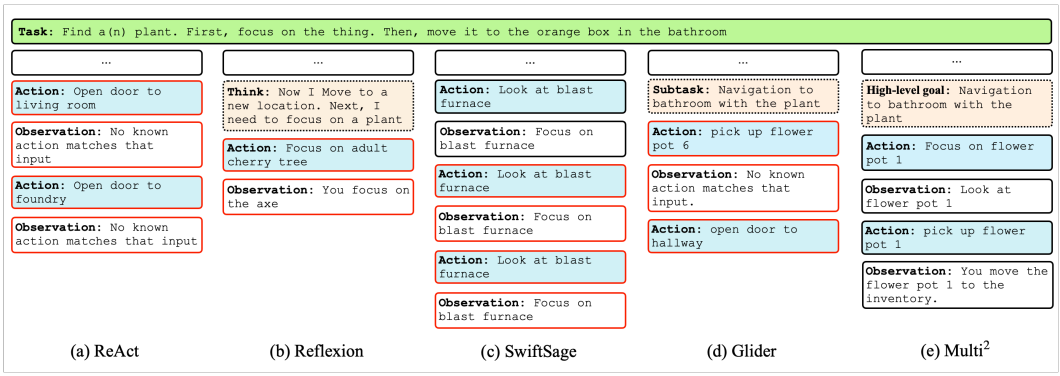

Figure 4: Failure cases of baselines evaluated on the `Find-a-plant` task. The red-outlined boxes highlight the failure factors.

I Qualitative Error Analysis

Figure 4 illustrates representative failure cases of baseline models, where red-outlined boxes indicate critical errors. The baselines mainly suffer from two failure modes: repeating identical actions without progress and generating invalid or inconsistent actions. In Figures 4(a-c), agents loop over the same actions or misidentify target objects, while Figure 4(d) shows an invalid action outside the permissible range, leading to task failure. These patterns reveal the weakness of prompt-based or non-hierarchical agents in maintaining coherent reasoning across multi-turn tasks. In contrast, `Multi`^{2} mitigates such errors through hierarchical goal decomposition and RL-based fine-tuning, producing more stable and goal-consistent behavior. Additional results for each model are provided in the following subsections.

I.1 ReAct

Task: Find a(n) plant. First, focus on the thing. Then, move it to the purple box in the living room.

Observation: This room is called the bathroom. In it, you see: the agent a substance called air a bathtub, which is turned off. In the bathtub is: nothing. a drawing a glass cup (containing nothing) a sink, which is turned off. In the sink is: nothing. a toilet. In the toilet is: A drain, which is open, a substance called water. You also see: A door to the kitchen (that is open)

Think: To solve the task, I need to Move to a new location and Move to the location of the answer box

Action: Open door to kitchen

Observation: The door is already open

Action: Go to kitchen

Observation: You move to the kitchen

Think: Now I Move to a new location and Move to the location of the answer box. Next, I need to focus on a non-living thing

Action: Focus on cupboard

Observation: You focus on the cupboard

Think: Now I focus on a non-living thing. Next, I need to move living thing to answer box

Action: Move cupboard to sink

Observation: You move the cupboard to the sink

Think: Now I move living thing to answer box. Next, I need to

Action: Open door to living room

Observation: No known action matches that input

Action: Open door to foundry

Observation: No known action matches that input

Figure 5: Test case result of ReAct, evaluated on the Find-non-living-thing task.

I.2 Reflextion

Task: Find a(n) plant. First, focus on the thing. Then, move it to the purple box in the living room.

Observation: This room is called the bathroom. In it, you see: the agent a substance called air a bathtub, which is turned off. In the bathtub is: nothing. a drawing a glass cup (containing nothing) a sink, which is turned off. In the sink is: nothing. a toilet. In the toilet is: A drain, which is open, a substance called water. You also see: A door to the kitchen (that is open)

Think: To solve the task, I need to Move to a new location and Move to the location of the answer box

Action: Open door to kitchen

Observation: The door is already open

Action: Go to kitchen

Observation: You move to the kitchen

Think: Now I Move to a new location. Next, I need to focus on a plant

Action: Focus on adult cherry tree

Observation: You focus on the axe

Figure 6: Test case result of Reflexion, evaluated on the Find-a-plant task.

I.3 SwiftSage

Task: Find a(n) plant. First, focus on the thing. Then, move it to the yellow box in the living room

Observation: This room is called the foundry. In it, you see:
the agent, a substance called air, a blast furnace, which is turned off. The blast furnace door is closed. a sink, which is turned off. In the sink is: nothing. a table. On the table is: nothing. You also see: A door to the outside (that is open)

Mode: **fast system**

Action: Wait

Observation: This room is called the foundry. In it, you see:
the agent, a substance called air, a blast furnace, which is turned off. The blast furnace door is closed. a sink, which is turned off. In the sink is: nothing. a table. On the table is: nothing. You also see: A door to the outside (that is open)

Action: Look around

Observation: This room is called the foundry. In it, you see:
the agent, a substance called air, a blast furnace, which is turned off. The blast furnace door is closed. a sink, which is turned off. In the sink is: nothing. a table. On the table is: nothing. You also see: A door to the outside (that is open)

Mode: **slow system**

Action: Look at blast furnace

Observation: Focus on blast furnace

Action: Look at blast furnace

Observation: Focus on blast furnace

Action: Look at blast furnace

Figure 7: Test case result of SwiftSage, evaluated on the Find-a-plant task.

I.4 Glider

Task: Find a(n) plant. First, focus on the thing. Then, move it to the orange box in the bathroom

Observation: This room is called the art studio. In it, you see: the agent, a substance called air, a large cupboard. The large cupboard door is closed. a table. On the table is: a bowl (containing nothing). a wood cup (containing red paint), a wood cup (containing yellow paint), a wood cup (containing blue paint) You also see: A door to the hallway (that is closed)

Subtask: Find a plant and focus it

Action: Open door to hallway

Observation: The door is now open

Action: Go to hallway

Observation: You move to the hallway

Action: Open door to greenhouse

Observation: The door is now open

Action: Go to greenhouse

Observation: You move to the greenhouse

Action: look around

Observation: This room is called the greenhouse. In it, you see: the agent, a substance called air, a bee hive. The bee hive door is closed. a flower pot 1 (containing a banana tree in the adult stage, soil), a flower pot 2 (containing nothing), a flower pot 4 (containing a pea plant in the adult stage with a tall height, soil), a flower pot 8 (containing a avocado tree in the adult stage, soil), a flower pot 9 (containing nothing), a jug (containing nothing), a sink, which is turned off. In the sink is: nothing. You also see: A door to the hallway (that is open), A door to the outside (that is closed)

Action: focus on adult banana tree

Observation: You focus on the banana tree.

Subtask: Navigation to bathroom with the plant

Action: pick up flower pot 6

Observation: No known action matches that input.

Action: open door to hallway

Figure 8: Test case result of Glider, evaluated on the Find-a-plant task.

I.5 Topic: Electricity

Task: Determine if unknown substance Z is electrically conductive. The unknown substance Z is located around the workshop. First, focus on the unknown substance Z. If it is electrically conductive, place it in the orange box. If it is electrically nonconductive, place it in the yellow box.

Observation: This room is called the living room. You can see: the agent, a substance called air, a book shelf (containing A book (Pride and Prejudice) titled Pride and Prejudice by Jane Austen), a chair. On the chair is: nothing. A couch. On the couch is: a white pillow. a finger painting, a table. On the table is: nothing.
You also see: A door to the hallway (that is closed).

High-level goals: 'Navigation to workshop', 'Find the unknown substance and focus it', 'Determine if unknown substance Z is electrically conductive'

High-level goal: Navigation to workshop.

Action: Open door to hallway.

Observation: The door is now open.

Action: Go to hallway.

Observation: You move to the hallway.

Action: Open door to workshop.

Observation: The door is now open.

Action: Go to workshop.

Observation: You move to the workshop.

High-level goal: Find the unknown substance and focus it.

Action: Pick up unknown substance.

Observation: You move the unknown substance Z to the inventory.

Action: Focus on unknown substance.

Observation: You focus on the unknown substance Z.

High-level goal: Determine if unknown substance Z is electrically conductive.

Action: Look around.

...

Figure 9: Test case result of the trained model, evaluated on the Test-conductivity-of-unknown-substances task.

I.6 Topic: Measurement

Task: Measure the temperature of wire, which is located around the bathroom.

Observation: This room is called the greenhouse. You can see: the agent, a substance called air, a bee hive, a jug (containing nothing), a shovel, and a sink, which is turned off. In the sink is: nothing. The bee hive door is closed.
You also see: A door to the hallway (that is closed), A door to the outside (that is closed).

High-level goals: 'Navigation to kitchen', 'Find the thermometer and focus it', 'Navigation to bathroom', 'Find the wire and focus it', 'Navigation to living room', 'Measure if the temperature of wire, which is located around the bathroom', 'Move wire to blue box'

High-level goal: Navigation to kitchen.

Action: Open door to hallway.

Observation: The door is now open.

Action: Go to hallway.

Observation: You move to the hallway.

Action: Open door to kitchen.

Observation: The door is now open.

Action: Go to kitchen.

Observation: You move to the kitchen.

High-level goal: Find the thermometer and focus it.

Action: Pick up thermometer.

Observation: You move the thermometer to the inventory.

Action: Focus on thermometer in inventory.

Observation: You focus on the thermometer.

High-level goal: Navigation to bathroom.

Action: Open door to kitchen.

Observation: The door is already open.

...

Figure 10: Test case result of the trained model, evaluated on the Use-thermometer task.

I.7 Topic: Identification of Biology

Task: Find the animal with the shortest life span. The animals are in the 'outside' location. Focus on the animal with the shortest life span.

Observation: This room is called the greenhouse. You can see: the agent, a substance called air a bee hive. The bee hive door is closed. a jug (containing nothing), a sink, which is turned off. In the sink is: nothing. You also see: A door to the hallway (that is closed). A door to the outside (that is closed).

High-level goals: 'Navigation to outside', 'Find the shortest life span animal and focus it'

High-level goal: Navigation to outside.

Action: Open door to outside.

Observation: The door is now open.

Action: Go to outside.

Observation: You move to the outside.

High-level goal: Find the shortest life span animal and focus it.

Action: Focus on baby baby mouse.

Observation: You focus on the baby mouse.

Action: Wait1.

Figure 11: Test case result of the trained model, evaluated on the Lifespan-shortest-lived task.

I.8 Topic: Classification

Task: Find a(n) living thing. First, focus on the thing. Then, move it to the yellow box in the living room.

Observation: This room is called the foundry. You can see: the agent, a substance called air, a blast furnace, which is turned off. The blast furnace door is closed. a sink, which is turned off. In the sink is: nothing. a table. On the table is: nothing.
You also see: A door to the outside (that is closed).

High-level goals: 'Find a living thing and focus it', 'Navigation to workshop with the living thing'

High-level goal: Find a living thing and focus it.

Action: Open door to outside.

Observation: The door is now open.

Action: Go to outside.

Observation: You move to the outside.

Action: Open door to greenhouse.

Observation: The door is now open.

Action: Go to greenhouse.

Observation: You move to the greenhouse.

Action: Look around.

Observation: This room is called the greenhouse. You can see: the agent, a substance called air, a bee hive. The bee hive door is closed. A flower pot 1 (containing a apple tree in the adult stage, soil), a flower pot 3 (containing nothing), a flower pot 4 (containing a banana tree in the adult stage, soil), a flower pot 5 (containing nothing), a flower pot 7 (containing a peach tree in the adult stage, soil), a jug (containing nothing), a sink, which is turned off. In the sink is: nothing.
You also see: A door to the hallway (that is closed)
A door to the outside (that is open)

Action: Focus on adult banana tree.

Observation: You focus on the banana tree

Action: Pick up the flower pot.

...

Figure 12: Test case result of the trained model, evaluated on the Find-living-thing task.